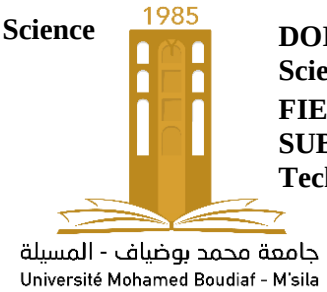
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: Mathematics and Computer Science
DEPARTEMENT: Computer Science
N°: ……



DOMAIN: Mathematics and Computer Science
FIELD: Computer Science
SUB-FIELD: Information and Communication Technologies

A Dissertation in Fulfillment
for the Requirement of the Degree of MASTER

By: Nadjmeddine BOUDJELLAL

TOPIC:

Design and Implementation of a Customer service chatbot using deep learning approach

publicly on: 25/03/2018, to the jury:

Board of Examiners

Brahimi Mahmoud	University of M'sila	Chairman
Tahar Mehenni	University of M'sila	Supervisor
Fares Mezrag	University of M'sila	Examiner

Academic year: 2017/2018

To my parents.

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human"
– Alan Turing –

Acknowledgements

I would like to thank my parents for their continuous support, my supervisors for his Guidance, unrelenting support, and helping me whenever I needed his help during my research work, also to my Friend Noureddine Bouabdallah who helped me lot's and stayed up with me for too many nights.

Contents

List of Figures

Introduction

Artificial intelligence has been one of the goals of computer science since its inception. Alan Turing presented his Turing Test in 1950, which is designed to figured out the ability of a machine to interact just like a human agent would. Since then, conversational agents have been trying to pass that test, and in 2013 a super computer managed to convince the panel of judges from the Royal Society that it was a 13-year-old-boy, marking the first time in the competition history for an AI to pass the Turing test. That conversational agent, or chatbot, was specifically tailored for the competition, and nowadays, hundreds of similar bots are available online for people to chat with. The potential of chatbots goes beyond simple conversations, they can be used as personal assistants to schedule meetings, check the weather or even suggest an outfit for the day. They can also be used by businesses to offer customer support services.

Traditionally, customer support happened over the phone, then with the advent of internet technology, email was also utilized, but in both mediums, an actual human being is responsible for answering the client's request. The problem with that is no matter how many people are working on customer support, it is never enough, especially for major brands and small startups. Customers now want more self-service options, as waiting times for support tickets are too long even for small inquiries. In order to improve accessibility, companies are redesigning the experience from human-to-human interactions into self-service models, implementing chatbots, that serve as automated customer support agents available 24/7, in messaging platforms to integrate closer to the communication services the customers are using.

Chatbots have seen a resurgence in research topics in recent years due to the progress made in deep learning techniques, which can be attributed to the massive increase in computational power, the availability of huge datasets, and the development of new and better deep learning models. These breakthroughs solved the main problems with deep learning as it required a lot of data to train the models, and the new super-fast compute units allowed for relatively fast iterations on those models, which allowed researches to produce new more efficient models, like Sequence-to-Sequence. Thus, resulting in chatbots, and other technologies like personal assistants, becoming usable in the real world, and now they are widely used and actively developed, by companies and amateurs alike.

In this first chapter we give an overview of artificial intelligence, deep learning, chatbots, Artificial Neural Networks, and the necessary information to choose the correct technique for our work.

In the second chapter we introduce several mature and field-tested techniques for building chatbots, we talk about the chatbot design pipeline, then we introduce the main models that are used to develop a chatbot and the challenges during this process.

In the last chapter we give an overview on the current customer service problem that lead to the use of chatbots, then we will share the models of our two chatbots, and discuss the obtained results that we have collected.

Chapter 1

Chatbots and Deep Learning

In this chapter we will first talk about artificial intelligence, because most of the recent chatbot are AI-based solutions, after that, we will give the definition of a chatbot then we explore customer service chatbots in detail. Because chatbots are software that interacts with users using natural language, we are going to talk about natural language processing and how NLP is used in chatbots. Next up, we need to understand the fundamental AI parts, which are Artificial Neural Networks, and be able to choose the right type of cells and neural architecture that will be suitable for our work, also we get to understand the mechanism that makes our model learn and how we can measure the learning rate.

1 Artificial intelligence

The name behind the idea of AI is John McCarthy, who began research on the subject in 1955 and assumed that each aspect of learning and other domains of intelligence can be described so precisely that they can be simulated by a machine. Even the terms ‘artificial intelligence’ and ‘intelligent human behavior’ are not clearly defined, however.[1]

Artificial intelligence describes the work processes of machines that would require intelligence if performed by humans. The term ‘artificial intelligence’ thus means ‘investigating intelligent problem-solving behavior and creating intelligent computer systems’.

There are two kinds of artificial intelligence:

- Weak artificial intelligence: The computer is merely an instrument for investigating cognitive processes – the computer simulates intelligence.
- Strong artificial intelligence: The processes in the computer are intellectual, self-learning processes. Computers can ‘understand’ by means of the right software/programming and are able to optimize their own behavior on the basis of their former

behavior and their experience. This includes automatic networking with other machines, which leads to a dramatic scaling effect.

2 Deep learning

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech [2, 3] .

3 Chatbots

Speech or text interaction between a human and a computer is gaining more and more popularity nowadays. People want to communicate with computers in the same manner they communicate with other human beings. One of the main tools used for analyzing speech and providing human-like answers is Natural Language Processing (NLP). In order to provide suitable responses based on phrases or keywords taken from questions as well as to keep the communication continuous, like any other language processing program, chatbot architectures fall into two classes: rule-based systems and corpus-based systems. Rule-based systems include the early influential chatbots. Corpus-based systems mine large datasets of human-human conversations, which can be done by using information retrieval (IR-based systems simply copy a human's response from a previous conversation) or by using a machine translation paradigm such as neural network sequence-to-sequence systems, to learn to map from a user utterance to a system response.[4, 5]

3.1 Customer service chatbot

The communication between brands and their clients has never been so intense as it is nowadays. With the rapid development of technology, the customer experience is changing dramatically. Customers want more autonomy and self-service options, preferring to make a purchase or get information without interacting with the human representative of the brand. In order to fit the expectations of their customers, companies are reshaping the experience from human-to-human interactions into the advanced self-service experience. Therefore, the use of chatbots in customer service can be a solution to the crucial issue of improving customer-brand communication. Companies are using this technology to create a better engagement with their clients with the help of messaging platforms to offer a regular chat function, in-message purchases, and many other advanced functions.

4 Natural Language Processing

Natural Language Processing (NLP) is an active research area that explores how computers could understand and manipulate natural language text or speech to do useful things. NLP researchers aim to collects knowledges on how we could make machine understand and use human language, so that appropriate tools and techniques can be developed to make computer systems manipulate natural languages to perform the desired tasks. The foundations of NLP lie in a number of disciplines, via. computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, etc. [6] Applications of NLP include a number of fields of studies, such as natural language text processing, machine translation, user interfaces, multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems, and so on [6].

4.1 Tokenizing

When training a neural network on a word to word basis, it is not very practical to use the actual words as input. The interest does not lie in how the word is constructed, only what word it is. Therefore, the text is tokenized before it is passed on to the network, which simply means that every unique word is replaced by an identifying number.

A vocabulary is created by finding each unique word that occurs in the chosen data set (corpus) and adding them to an indexed list. Having a vocabulary that covers all possible worlds is usually both unnecessary and unfeasible, since a larger vocabulary will increase

computation of the losses/gradients. Instead, it is common to use a subset of the full vocabulary, containing only some of the most commonly occurring words.

Because some of the more unusual words in the corpus will appear in the training data, but not in the vocabulary, there must be some way of representing them. The simplest way to solve this is by replacing all occurrences of an out-of-vocabulary word with a special token that is treated by the network as a single word. Every instance of out-of-vocabulary words would then be treated as though they were the same words.

It is common to represent the input to the network as a one-hot vector. A permutation of this vector represents a word in the input vocabulary. This vector is the same length as the size of the vocabulary used, and all elements in the vector except for one is set to 0. A one-hot vector having its element at index i set to 1 will thus represent the word at index i in the input vocabulary

4.2 Natural Language Understanding

The natural language understanding (NLU) component, takes the user input as a text and extract from it the semantic representation. A travel system which has the goal of helping a user find an appropriate train, would have a frame with slots for information about the train ticket [7]. User input such as the sentence Show me available trains from Algiers to Constantine on Tuesday might correspond with the following filled-out [7].

```
SHOW:
  FLIGHTS:
    ORIGIN:
      CITY: Algiers
      DATE:
        DAY-OF-WEEK: Tuesday
      TIME:
        PART-OF-DAY: morning
    DEST:
      CITY: Constantine
```

LISTING 1.1: Natural Language Understanding filled-out

4.3 Natural language generation

Natural Language Generation (NLG) aims at producing understandable text from non-linguistic representations of information (concepts). NLG applies in too many areas, from

summarization of emails, information about weather [8], and spoken dialogue systems [9, 10, 11].

The objective of NLG component is to take a meaning representation of what to say from the dialogue manager and transform this into a natural language string. This can be done in two ways [12]. The easiest and most common technique is template-based in which you have a template of the response with slots so you can fill it with words, for example:

```
Train from <src> to <dest> on <date>
```

The advantages of this method are that it's conceptually simple and tailored to the domain so often of good quality, and also to be able to generate a grammatical mistake free sentence. The disadvantages are that it lacks generality and variation in style, so you will still fell that you are chatting with a bot. An alternative method is to make use of a rule-based generator [13]. Such a generator generally consists of three components, a sentence planner, a surface realizer, and a prosody assigner. The sentence planner determines what words and syntactic structures will be used for expressing the content. It also determines what elements can be grouped together for more natural-sounding succinct output. The surface realizer combines everything into a syntactically and morphologically correct sentence. Finally, the prosody assigner annotates the text with rhythm, stress and intonation markers for use by the speech synthesizer.

5 Artificial Neural networks

Artificial neural networks (ANNs) were originally developed as mathematical models that was able to detect patterns in a dataset. It is inspired by how the human brain is believed to function.

In the brain, neurons connect to each other via their synapses, which is a one-way connection. These connections can be of varying strength. A neuron can fire an electrical signal and does this when it has gotten enough stimulus from other neurons via their synapses. This signal is then transmitted to other neurons which this neuron's synapses connects to [14]. The basic ANN structure are some small processing units, or we can call it nodes or a perceptron, which are connected to each other by weighted connections. And if we compare it to the original biological model, the nodes represent neurons, and the connection weights represent the strength of the synapses between the neurons.

The output of a perceptron x_i is calculated by the values of all other perceptron x_k that connects to it. The output of these perceptron x_k is strength by a factor $w_(i,k)$. Additionally, each perceptron has an associated bias value $b_(i)$ to allow to work event when we don't have an input or a zero as a result of an input. A single connection to a perceptron can be calculated with this equation $x_i = w_{i,k}.x_k + b_i$ and the value of a perceptron that have n Connection is calculated in equation 1.1.

$$x_i = g(b_i + \sum_{k \neq i}^{n} w_{i,k}.x_k) \tag{1.1}$$

We also can see in equation 1.1 that there is a function g and we call it activation function, and it should be a non-linear function, and the most popular ones are Sigmoid Function or Hyperbolic Tangent. The role of this function is to allow the result to stays inside some interval. The Sigmoid function, is defined in equation 1.2.

$$\sigma(a) = \frac{1}{1+e} \tag{1.2}$$

If we start connecting those perceptron's to each other, we start forming a ANN, and it will be divided into 3 parts, the first part we call it input layout which is a group perceptron with a starting state, the second part is a group of ANN layout, we call it hidden layout, and the last one is the output layout.

5.1 Feed forward neural networks

One of the most basic and common type of ANN is the feed-forward type neural networks. Feedforward neural networks are called like that because the signal flows in forward direction without feedback. In other words, they are directed acyclic graphs (DAGs) [15]. where perceptron's from one layer are partly / fully connected to the one in the next layer. The simplest form of feed forward network is a fully connected one, where each perceptron in one layer is connected to all perceptron in the next one [15]. See figure 1.1 for an example.

The input layer has no connection to it, so its values are set manually. In the other side, the output layer has no connections from it, because its values are taken as the result. As we know the network is arranged as a DAG. To make the network output something meaningful, we need some means of adjusting the weights to improve the output. For this a method called back propagation is used and is described in section 5.2 [15].

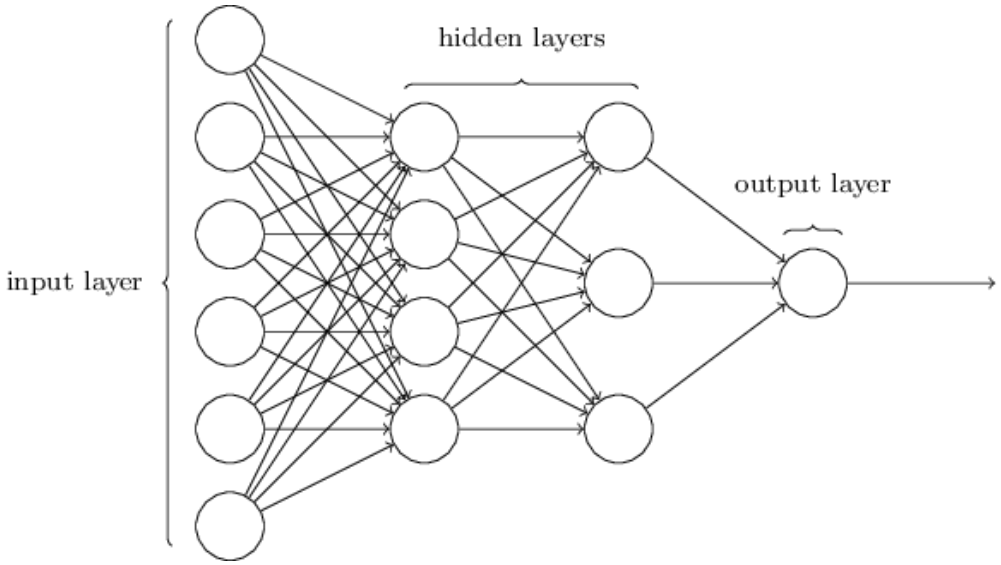


FIGURE 1.1: Feed forward neural network [16]

5.2 Weight Optimization

As we have seen before that each perceptron has a weight connected to it, and with the right weight it will give the right output, but to be able to find the right weight is another problem, let's have an example to make things clear, for example we have a network that can predict if you are going to pass your class or no, and the perceptron have 3 different weight connected to it, and as an output it gives q(x) = (0.1, 0.75, 0.) and by giving the network the right data that will compare to be able to use the right weight next time, for example (0, 1, 0) it will know the most probable weight that will give us the right output, and we can tell if the network made a correct prediction (in this case it did).We define H (p, q) as the entropy, where $p(x) \in \{0, 1\}^{|x|}$ is the true probability (the correct label) and q(x) is the predicted probability of data point x .. For this we must define a loss function. One of the most common loss functions is called cross entropy and is defined in equation 1.3 . A subset of all data points N, called mini-batch, is usually used when calculating the loss to make the training more tractable .

$$H(p, q) = -\frac{1}{N} \sum_{N}^{x} p(x). \log q(x) \tag{1.3}$$

We also need an algorithm to minimize this loss function with respect to the weights of the network. This algorithm can vary greatly depending on the task, but usually involves calculating gradients of the weights in the network. This is called backpropagation

and is done by applying the chain rule on the derivative of the loss function back through the layers in the network. The most common optimization method for neural networks is Stochastic Gradient Descent (SGD). The SGD-algorithm uses backpropagation to calculate the gradients for each weight in the network and minimizes the loss function by iteratively taking small steps in the direction of the negative gradient. An update of SGD is defined in equation 1.4, where w_t is the weights at time t of the network, $\eta > 0$ is the step size or learning rate and $\bigtriangledown H(p, q)$ are the gradients.

$$W_{t+1} = W_t - \eta \bigtriangledown H(p, q) \tag{1.4}$$

The learning rate η needs to be set manually, but there are more advanced optimization methods which can lessen the importance of this parameter slightly. One of these is the Adaptive gradient algorithm (AdaGrad) which was introduced by Duchi et al [17]. It is an optimizer that assigns individual learning rates for each weight in the network dynamically during training. AdaGrad is especially useful for optimizing on sparse features, such as when the output classes are a bag of words. The algorithm will then give more importance to less frequently occurring features, enforcing a form of equity.

5.3 Recurrent Neural Networks

The normal FNN won't be able to have an input with a varied length. And this was the need of many applications such as text processing so the input of the network will vary from each sentence. So, if we want to keep track of each word placement in the sentence to detect dependencies between cohesive words being input to the network at different times. Recurrent neural networks solve this problem by implementing a recurrent connection from a neuron to itself, so the neuron will be able to have its previous state as an input [18]. A visualization of a single recurrent cell can be seen in figure1.2.

This transformation is equivalent and a shorthand to the equation 1.4 for a feed forward network in section 5.1, but without the activation function applied [19]. For a standard recurrent neural network,h_t^l is updated according to equation 1.5, where layer $l-1$ and l contains m and n neurons respectively.

$$h_t^l = \sigma(T_{m,n}(h_t^{l-1}) + T_{n,n}(h_{t-1}^l)) \tag{1.5}$$

The first part inside the function in equation 1.5 contains the input from the previous layer $l-1$ at time step t, while the second part contains the input from the current layer l at the

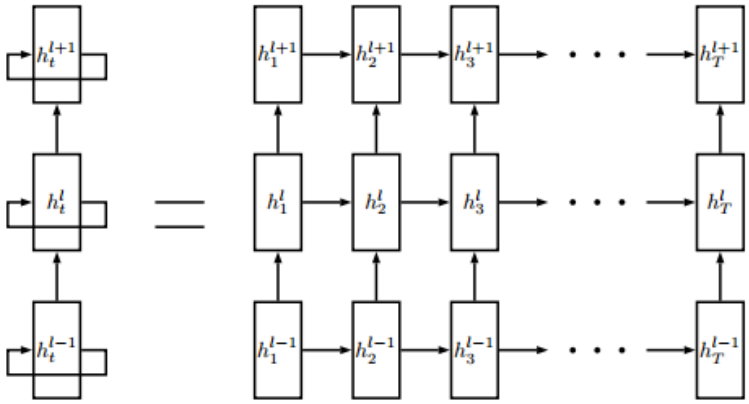


FIGURE 1.2: A visualization of a single recurrent cell.

previous time step t–1 [20].

5.4 Long short-term memory

When we found that RNN can keep a memory up forward in time step, but it will also forget after a couple of steps. because it is very difficult to train a standard RNN to keep its memory up over large data. To solve this problem, an extension to the standard RNN called Long Short-Term Memory (LSTM) is often used instead [21].

An LSTM network is built up by LSTM cells. Every cell has the same architecture like RNN (input and output and is connected to itself to get the temporal aspect). An LSTM network is formed exactly like a simple RNN, except that the nonlinear units in the hidden layer are replaced by memory blocks. It usually has multiple layers with many cells in each layer. Figure 1.3

The basic concept behind an LSTM cell is to use something called gates, the multiplicative gates allow LSTM memory cells to store and access information over long periods of time, and also to control what information the cell will remember, forget and output [22].

5.5 Sequence-to-sequence

The concept of a sequence to sequence (seq2seq) model was originally proposed by Cho et. al [23] and Sutskever et. al [24].

An RNN or a DNN are very powerful machine learning models that have achieved a very good result with problems like speech recognition or visual object recognition. But

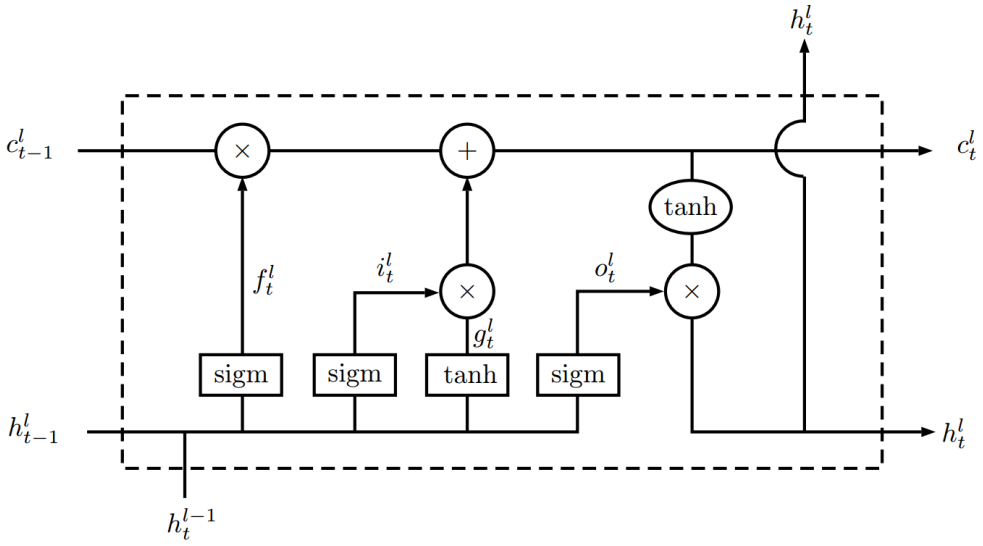


FIGURE 1.3: Long Short-Term Memory Cell.

there was another problem after the length one which is the sequence problem, let's have an example: imagine we have those two sentences as an input, "The food was good, not bad at all." Vs "The food was bad, not good at all." We have the same word and length but the sequence matter because it could change the whole meaning of the sentence.

And for that we have a new powerful technique that uses RNNs, and it's called $sequence-to-sequence$ (seq2seq).

To create a sequence-to-sequence model we first have to create our recurrent neural network with L layers and then divided into an encoder and a decoder. the encoder's job is to encapsulate the information of the input text into a fixed representation. The decoder's is to take that representation and generate a variable length text that best responds to it. [25] The basic model was extended to use multi-layer cells (specifically LSTMs) [26].

The input is a tokenized sentence with length T, which is converted into a string of one-hot vectors $(x_1, x_2, ..., x_t)$. At each time step, a word x_t is embedded into a vector x'_t. The embedded word x'_t is then fed as the input to the encoder, which consists of a multi-layered recurrent neural network with LSTM-cells.

When all T inputs has been fed to the encoder, the network will be used for decoding. At each time step $t > T$, the network will output a word w_t. The network will be run until a special end-of-sentence symbol is produced. The input to the decoder w_t at time $t > T$ consists of the last word w_{t-1} that the network generated, where the first decoder input is a special GO symbol. The predicted word w_t at decoding time t is calculated in equation

1.6, where $V(i_w)$ is the i'th word in the vocabulary and O_t is the output of probability. An example run of a sequence-to-sequence network can be seen in figure 1.4. The string ”How are you?” are fed one word at a time to the network, and it will generate words until it generates a special EOS token [27].

$$\begin{aligned} i_w &= argmax(O_t) \\ w_t &= V(i_w) \end{aligned} \tag{1.6}$$

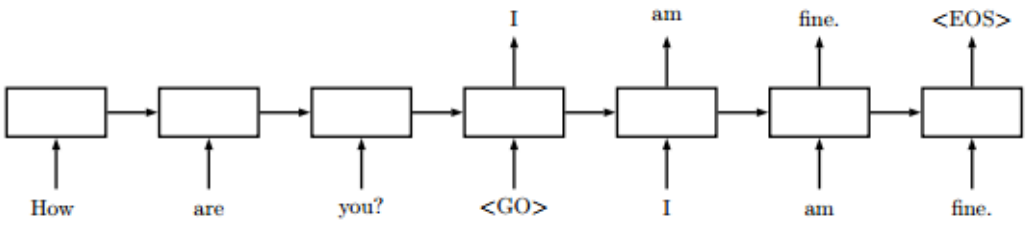


FIGURE 1.4: A sequence-to-sequence model which encodes the sentence ”How are you?” and produces during decoding the sentence ”I am fine. <EOS>”. When decoding, the previously generated output is used as input for the next time step, except for the first word, where <GO> is used as input. The decoder stops when an <EOS> is generated.

6 Conclusion

From the information presented in this chapter we understand the model used for this thesis, some background knowledge in the area of neural networks and deep learning, also we have described the basics of neural networks, how they are trained and the architecture in which they are used for dialog modeling.

Chapter 2

Introduction to Chatbots

The chatbot industry has grown a lot in the past few years, the methods used to make chatbots are numerous and advancing quickly, now there are several mature and field-tested techniques for building chatbots.

In this chapter we give an overview of the designing processes of a chatbot and what we call the chatbot design pipeline, then we describe the different models used to create a chatbot. After that, we describe the conversation types, then we delve into the challenges faced during the development.

1 Designing a chatbot

Designing a chatbot consist on many steps or like we call workflow, in figure 2.1, we can see that the first step is to identify business use cases, then defining tasks that the bot would perform, next you have to use the power of NLP to identify the need of Customers, and create a deep learning model that could understand and generate sentences, after you are done with the model you have to train it and it's going to take a while, so you have to test it during training, when could find a satisfy result you could launch the bot.

To start the designing process, several questions related to the use of the bot and how it will interact with users need to be answered, some of these questions are:

- What does the chatbot need to do?
- What business goals will the chatbot be made to achieve?
- How satisfying does the chatbot perform in front of customers?

All these questions and more need to be answered before starting to build a chatbot, but the main questions must be:

- Retrieval-Based or Generative Models?
- Does the chatbot have to be able to handle long conversations or short conversations?
- Will the chatbot interact in an open domain or a closed domain setting?

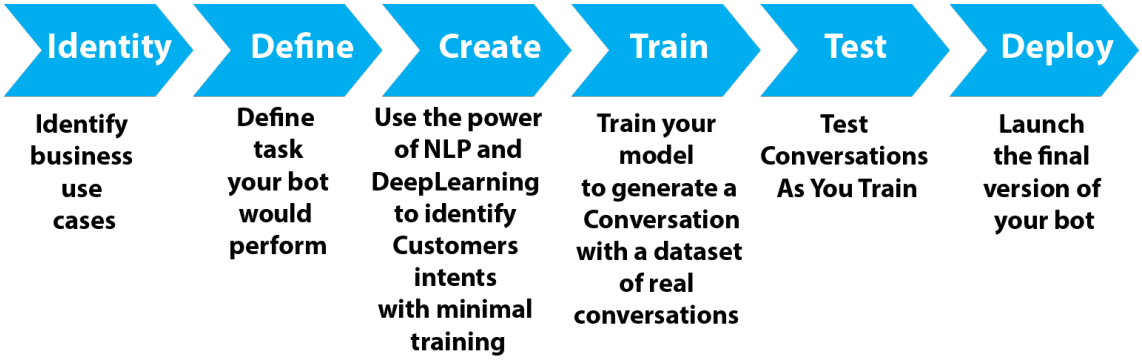


FIGURE 2.1: Chatbot design

2 A Taxonomy of models

2.1 Retrieval-Based vs. Generative Models

Retrieval-Based Models

These types of models use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context. The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of machine learning classifiers. These systems don't generate any new text, they just pick a response from a fixed set [28, 29, 30].

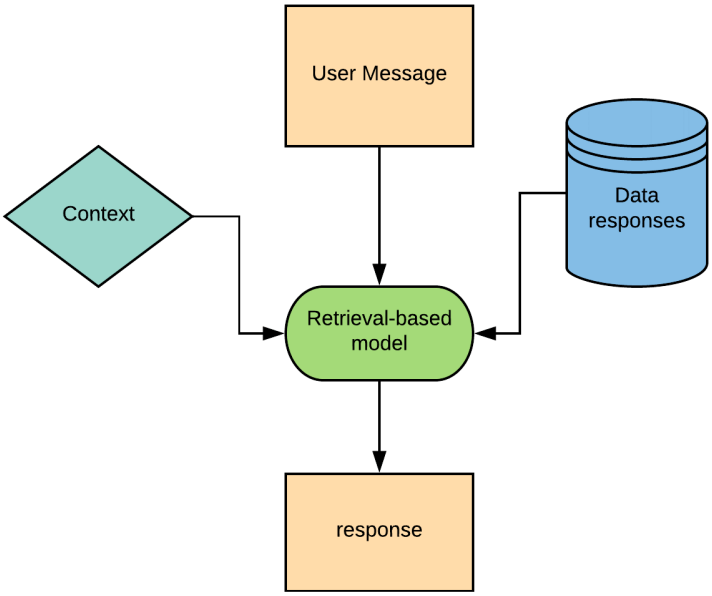


FIGURE 2.2: Flow of Retrieval-based model [31]

Generative Models

Generative models don't rely on pre-defined responses, they generate new ones from scratch. Generative models are typically based on machine translation techniques, but instead of translating from one language to another, they "translate" from an input to an output (response) [29, 1].

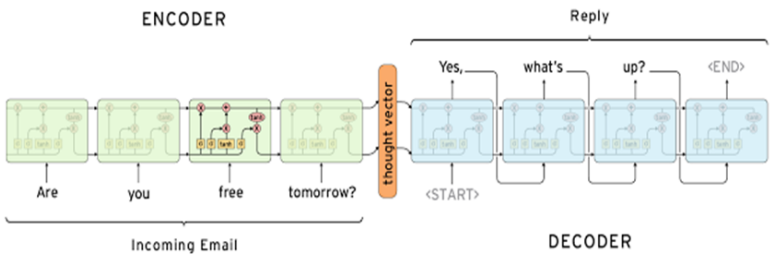


FIGURE 2.3: Sequence to Sequence Learning with Neural Network [30]

Both approaches have some obvious pros and cons. Due to the repository of hand-crafted responses, retrieval-based methods don't make grammatical mistakes. However,

they may be unable to handle unforeseen cases for which no appropriate predefined response exists. For the same reasons, these models can't refer back to contextual entity information like names mentioned earlier in the conversation. Generative models are "smarter", they can refer back to entities in the input and give the impression of talking to a human. However, these models are hard to train, are quite likely to make grammatical mistakes (especially on longer sentences), and typically require huge amounts of training data [1].

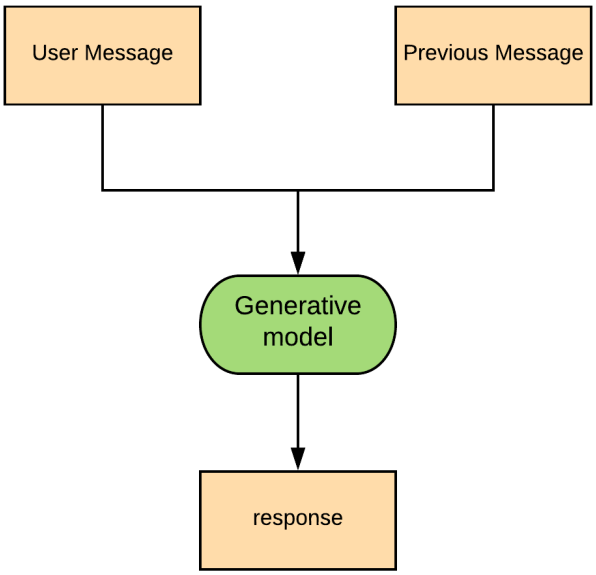


FIGURE 2.4: Sequence to Sequence Learning with Neural Network[31]

Deep Learning techniques can be used for both retrieval-based or generative models, but research seems to be moving into the generative direction. Deep Learning architectures like Sequence-to-Sequence are uniquely suited for generating text and researchers are hoping to make rapid progress in this area. However, we're still at the early stages of building generative models that work reasonably well. Production systems are more likely to be retrieval-based for now.

2.2 Long vs. Short Conversations

Short Conversations tend to be some answers to a simple question and most commonly to FAQ (Frequently Asked Questions), the bot only has to generate a single response to a single input.

Then there are long conversations, the goal of it is to be able to answer questions that are not common and questions that might be buried deep, so the bot must go through multiple turns and need to keep track of what has been said and identify the intent of the user and be able to ask for the entities it needs to know, then generate a final answer, so sometimes the bot could generate an answer or even another question. Customer support conversations are typically long conversational threads with multiple questions.

Short Conversation agents are created to make the user experience simple and the process more effective and streamlined. In the other side long conversation agents are more general and can adapt to a larger set of questions and can act more as a human in that they have memory and a little more intelligence [30].

Long conversations take much longer to implement, because it takes a long time to train it on the domain that it is going to work in. Often a new Deep learning model is needed for the bot to be able to fully control the domain and be able to understand, learn and reason. not only learning model is needed, a large conversations dataset on the specific domain to be able to understand the more detailed and niche questions that might be asked [30].

3 Open Domain vs. Closed Domain

3.1 Open Domain

Open domain bots tend to take the conversation anywhere, they can be used as a personal assistant or as a friend, because there is not a predefined goal or intention for the conversation.

Open domain bots are not that hard to implement comparing to close domain ones, because training data sets are available, bots can be trained with any dataset, whether it is conversations on social media sites like Twitter and Reddit, or movie scripts because they are typically open domain conversations, also because there is no predefined knowledge or problem solving situation like in customer service bots (closed domain), but also we can say that the infinite number of topics and the fact that a certain amount of world knowledge is required to create reasonable responses makes this a hard problem[1, 32].

3.2 Closed Domain

When we speak about a closed domain, that means that the bot has less to do compared to our tasks, because close domain bots tend to solve specific problems and help users achieve very specific goals. Technical Customer Support or Shopping Assistants are examples of closed domain problems.

But most of closed domain bots must have human decisions at the end because the conversation flow must be predefined, and they can't answer or do tasks that are not pre-programmed, they just need to fulfill their specific tasks as efficiently as possible. Sure, users can still take the conversation anywhere they want, but the system is not required to handle all these cases and the users don't expect it to [30].

4 Common Challenges

There are some undeniable and not really evident difficulties when building conversational operators, the majority of which are dynamic research regions.

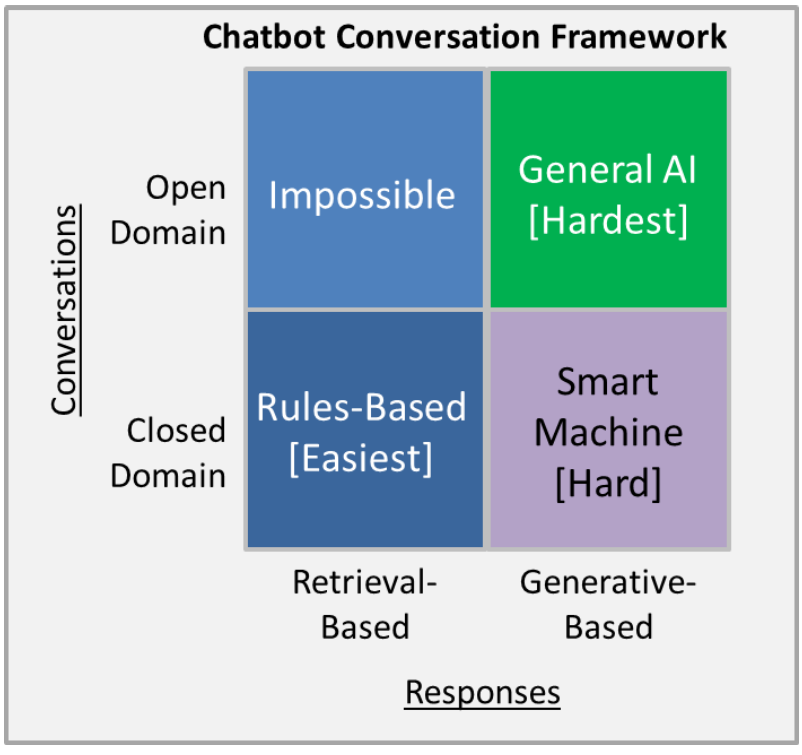


FIGURE 2.5: Domain complexity[30]

4.1 Incorporating Context

To produce sensible responses, systems may need to incorporate both linguistic context and physical context. In long dialogs people keep track of what has been said and what information has been exchanged, that's an example of linguistic context. The most common approach is to embed the conversation into a vector but doing that with long conversations is challenging. Experiments in Building End-To-End Dialog Systems using Generative Hierarchical Neural Network Models and Attention with Intention for a Neural Network Conversation Model both go in that direction. One may also need to incorporate other kinds of contextual data such as date/time, location, and information about the user [30].

4.2 Coherent Personality

When generating responses, the agent should ideally produce consistent answers to semantically identical inputs. For example, you want to get the same reply to "How old are you?" and "What is your age?". This may sound simple but incorporating such fixed knowledge or "personality" into models is very much a research problem. Many systems learn to generate linguistic plausible responses, but they are not trained to generate semantically consistent ones. Usually that's because they are trained on a lot of data from multiple different users. Models like that in a Persona-Based Neural Conversation Model are making first steps into the direction of explicitly modeling a personality [30].

FIGURE 2.6: Coherent personality

4.3 Evaluation of Models

The ideal way to evaluate a conversational agent is to measure whether or not it is fulfilling its task, e.g. solve a customer support problem, in a given conversation. But such labels are expensive to obtain because they require human judgment and evaluation. Sometimes there is no well-defined goal, as is the case with open-domain models. Common metrics such as BLEU that are used for Machine Translation and are based on text matching are not well suited because sensible responses can contain completely different words or phrases. In fact, in "How NOT to Evaluate Your Dialog System: An Empirical Study of Unsupervised Evaluation Metrics for Dialog Response Generation" researchers found that none of the commonly used metrics really correlate with human judgment [30].

4.4 Intention and Diversity

A common problem with generative systems is that they tend to produce generic responses like "That's great!" or "I don't know" that work for a lot of input cases. Early versions of Google's Smart Reply tended to respond with "I love you" to almost anything. That's partly a result of how these systems are trained, both in terms of data and in terms of actual training objective/algorithm. Some researchers have tried to artificially promote diversity through various objective functions. However, humans typically produce responses that are specific to the input and carry an intention. Because generative systems (and particularly open-domain systems) are not trained to have specific intentions they lack this kind of diversity [30].

5 Conclusion

This chapter contained some required information that we needed to understand before we start modeling a chatbot, starting with some fundamental information about the chatbot design process, then we explored some models we used in our work. Then we gave the chatbot types to clarify what we are going to deal with, lastly, we had an introduction to the most common challenges. In the next chapter we will share our system configuration and the development tools we used throughout our work.

Chapter 3

Design and implementation

In this chapter we give an overview of the basic customer service problem that lead businesses to using chatbots, then we present the way each part of our chatbot work. Finally, we examine the implementation of our bots and how both of them work and share the results we obtained.

1 Customer support services

Customer support services have changed drastically over the past few years. Before, a team of people in an office with headsets and computers, answer call after call, and email after email personally responding to customer requests. While that is still the case at many organizations, the way businesses communicate with their customers have changed driven by the advent in communication channels and automated information processing.

Today, people use messaging applications as their preferred method of personal and professional communication. They interact with brands the same way they would with their family and friends. With that in mind, companies are following their audiences where they're most likely to be.

Allocating the resources necessary to interact with millions of potential consumers on a daily basis is near impossible, so businesses started using bots. These bots are powered by artificial intelligence, allowing them to interact with their consumers in a way similar to a human representative.

With the fast-paced development of technology, the customer service experience is changing dramatically. Customers want more autonomy and self-service options, preferring to make a purchase or get information without interacting with a human representative of the service. In order to fulfill the expectations of their customers, companies are reshaping the experience from human-to-human interactions into the advanced self-service

experience. Therefore, the use of chatbots in businesses can help increase customer satisfaction by improving accessibility through enhanced communication.

There are several problems that have driven the customer care into automation, the first common problem customers encounter is long resolution time of their problems, and there are two reasons why it happens:

- Difficult troubleshooting is needed (e.g. IT team needs to fix a bug or you deal with a third party to have the problem solved).
- The lack of Communication in the team (lots of stuff happening and nothing really makes sense).

Also, one of the reasons why customers lose their temper is because they are being switched between departments. It can happen in a call center, it also may happen on a chat.

In most cases, the reason behind it is that a customer service agent doesn't know what to do and they hope someone else will know. Suddenly, a customer turns into a ball and no one wants to be left holding it.

What are the benefits that chatbots bring to businesses and their customer communication? Three main positive aspects that chatterbots have are: reducing stress, allowing instant response and extending the service. Previously, it took quite a lot of effort to contact the company. One way was to call the company via their hotline, which could not be free and might work on specific hours only, which is not suitable for emergency situations at all. Another way was to write an email to the company and wait for their response, which sometimes could happen in days at best or even weeks. Chatbots can solve this problem by being easily accessible, and most people always have their smartphones at hand with messaging applications installed and internet access. Chatbots are available 24/7 and reply immediately, so customers do not get frustrated waiting and get the response they want as soon as possible. Moreover, there is a possibility to integrate the chatbot with other services to offer a set of products or services at one place. As can be seen, chatbots can be a solution for improving communication between businesses and customers. Although the development of customer service chatbots is still in its early stages, businesses are becoming more and more interested in this technology.

2 Project description

In our work, we have explored two deferent chatbot systems, because we faced several problems during research and design, the first bot is an open domain deep learning chatbot that has been trained on my personal computer, and the second one is a customer service chatbot that we designed and set its training in Google's cloud platform.

In the beginning we were looking to create a full deep learning customer service chatbot and train it on my personal computer, but while researching for the required data sets, I couldn't find the right data set to use in our solution. Because the training sets used in deep learning differ widely from one business to another because of the chatbot goal or mission. So, to create a full deep learning customer service chatbot it is required to get the data set from the company's own databases, and because it's deep learning a large amount of conversational data is needed, and acquiring that much data is, in our case, highly improbable if not altogether impossible due to being outside of an enterprise environment and not finding a company that is willing to share its databases for our work. because most of the customer service done by the them is in person, other ones run Facebook pages to communicate with their customers and offer their different services, but the conversation with the client is made in English, French, Arabic, Arabic written in Latin letters, so even if we could get the data it will be impossible to process it and extract the useful information from the conversations. The second problem is hardware, in order to train a deep learning neural network model, a really powerful computer with expansive GPUs is needed, to at least be able to have some efficient results in a relatively short period of training (a week), a lot of related works benefit from access to Supercomputers which accelerate their development cycles considerably.

Because I had already created a model, I trained it with an open domain dataset, and the training took about 3 weeks of my personal computer crunching numbers non-stop, and the results weren't satisfying.

During that time, I was looking to find a similar customer service chatbot that can be optimized for our work, or another way to train the model on the cloud, the only way to get the dataset was to choose a foreign company that has a customer service system on Twitter, and try to dump their tweets into a document database, but the process is not that efficient for our purposes, because it will need a lot of cloud storage and will take a long time to pre-process and filter to get only the right entries we need for our work.

And that's way we decided to start working on a second chatbot. Google offers this system where models can be implemented and trained on their powerful cloud , it helps

by getting around most of the common challenges like (example: datasets, preprocessing, training time, expenses… etc), as well as provides the option of adding a personality to the bot by giving it the ability to respond to personal questions such as the bot's name and other predefined information, also with the power of google machine server the bot will be trained perfectly to be able to generate a sensible responses, it also allows the bot to connect to other services via webhooks which are cloud functions that can take a series of parameters and do some work, like for example connect to a database to save or query some data, then return a response, the service also allows for easy integration with the most used social media and chat platforms, and even smart home devices like Amazon's Alexa or Google's own Google Home..

The bot that we implemented is an SNTF (Société Nationale des Transports Ferroviaires) chatbot, that helps SNTF clients figure out train times, book tickets, report problems, and receive answers to some of the most common questions.

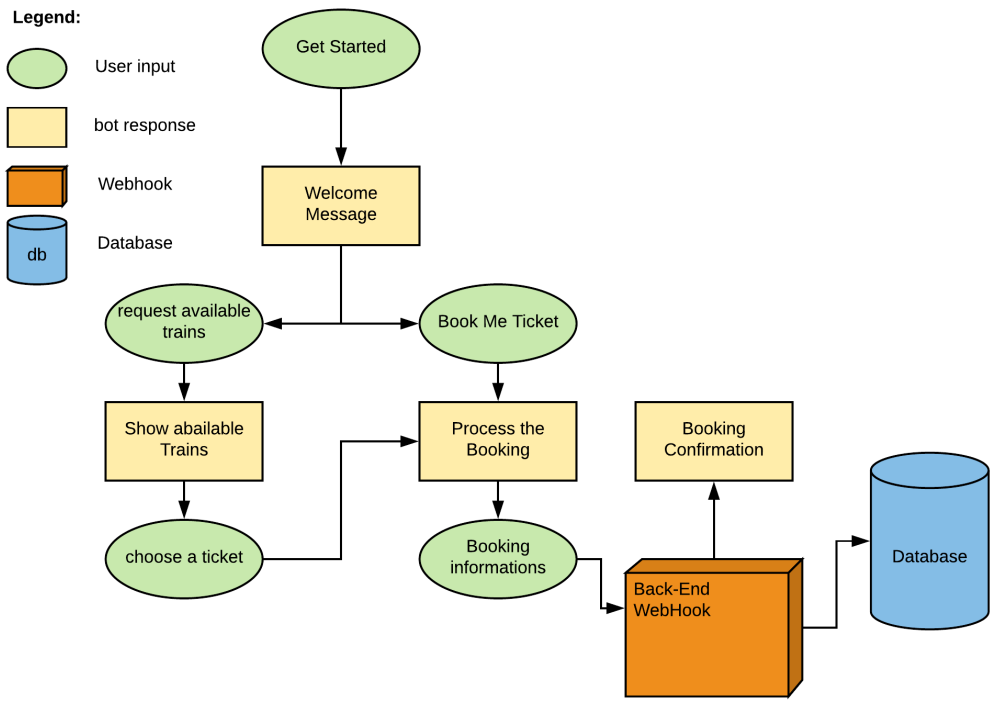


FIGURE 3.1: CONVERSATIONAL FLOW DIAGRAM

In Figure 3.1 a representation of the conversation between a client and our bot. It illustrates the user's inputs, the bot's responses and the different calls to external sources,

allowing us to have an overview over the whole conversation and the parts of our system

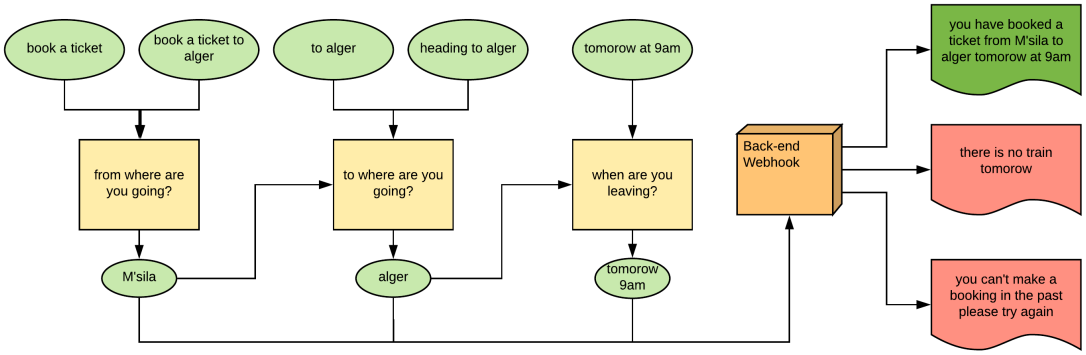


FIGURE 3.2: diagram represents only the Booking flow

In Figure 3.2 we can see an example of a booking conversation flow and how it is processed, if a required piece of information is missing the bot will prompt the user to enter it.

To break things down a little bit, we can see in Figure 3.3 that the user interface can be any conversation channel, a string is passed through to the bot, when the user input arrives to the bot we need to use NLP and a neural network to break the user request into intent and entities, the intent is the task that the bot needs to preform, the entities are the parameters that the bot needs to perform the intent. To be able to extract those information, we need to pass it through the neural network that have been trained with a large amount of data, because that information can be in a multitude of forms, after that the bot should process his task successfully with the help of our backend system. when the task is completed the bot uses a neural network and NLG to generate a correct response sentence that is then sent back to user.

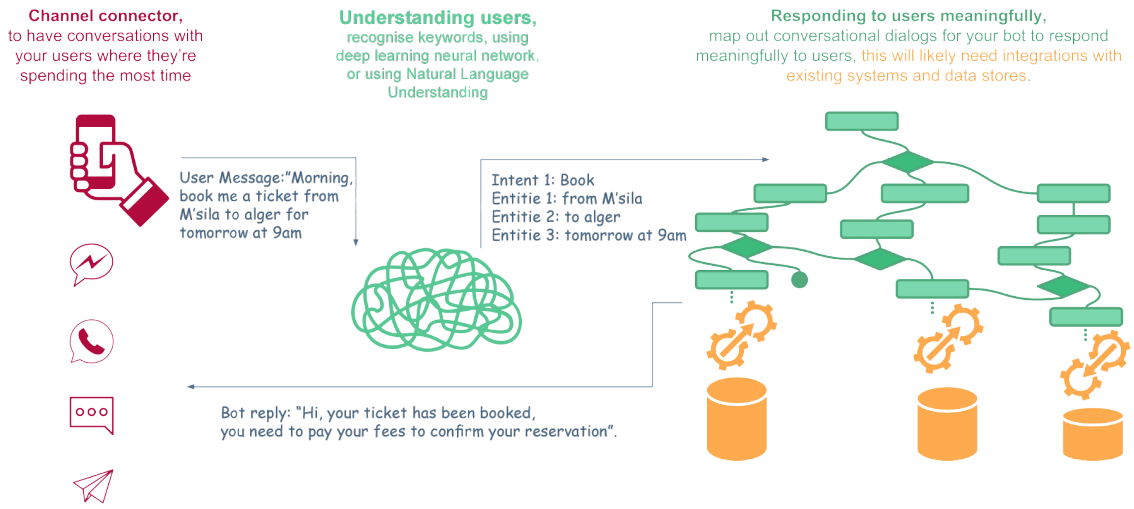


FIGURE 3.3: Anatomy of a Chatbot

Also, we can see in Figure 3.4 that the bot has a Decision engine which will decide if it should use the neural network, or if the request is just a simple question, like a greeting to begin the conversation, or it could be a question about the bot's personality, and like we mentioned before, the personality is one of the common challenges, so the common question about the bot personality must not be generated from the neural network because it will have a deferent answer every single time.

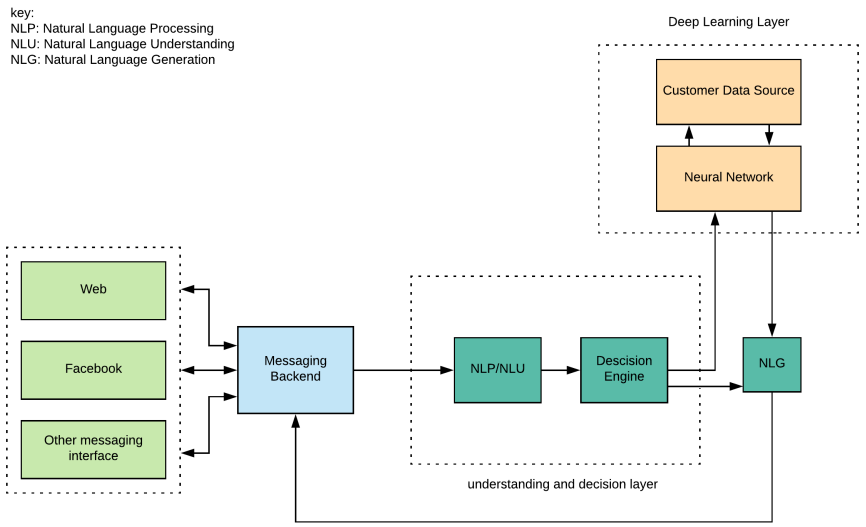


FIGURE 3.4: Architecture Diagram for Chatbots

2.1 Datasets

Cornell Movie–Dialogs Corpus

The Cornell Corpus is a large set of imagined conversations, starting from movie scripts crawled from various sites. Metadata for conversation analysis and duplicate-script detection involved mostly automatic matching of movie scripts with the IMDB movie database; clean-up resulted in 617 unique titles tagged with genre, release year, cast lists, and IMDB information.

This corpus contains a metadata-rich collection of fictional conversations extracted from raw movie scripts:

- 220,579 conversational exchanges between 10,292 pairs of movie characters.
- involves 9,035 characters from 617 movies.
- in total 304,713 utterances.

Reddit conversation dataset

That's 1.7 billion comments total, with data about the author, subreddit, position in the comment tree, and comment score for each post, this dataset is over 1 terabyte uncompressed, so this would be best for larger research projects, you can download each month's comment separately.

3 Tools

3.1 Python

FIGURE 3.5: Python Logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed [33].

3.2 TensorFlow

FIGURE 3.6: TensorFlow Logo

Released on the 15th of November 2015 by Google [34], TensorFlow [35] is the newest open source library written in Python for numerical computation. It has immediately a great success in the Machine Learning community and in less than one year it also had

a lot of support and development by Google [34] itself, more over by many community projects, developed in any area of Deep Learning. The peculiarity of TensorFlow is its work flux, made by data flow graphs. Where Nodes represent mathematical operations, edges represent the multidimensional data arrays communicated between them; the latter can be considered, as in electronics, a Tensor, from here its name. In May 2016, Google [36] has revealed that it has used TensorFlow in AlphaGo project, with a special hardware dedicated to boost library's performances.

To reach rapidly this goal, Google dedicated a special attention to the user experience of TensorFlow [35], which arrives with a great basic support and a well grown GitHub community [37], the key of this quick improvement. All these are the peculiarities that pushed us to choose TensorFlow [35], over more developed and bigger communities. The irruption made in the Deep Learning environment, shows up its great future potentialities, that are rolling out day by day.

3.3 Dialogflow

FIGURE 3.7: Dialogflow Logo

Dialogflow is a platform owned by Google that allows you to create a natural language interface by providing actionable data based on the input given. The platform includes speech recognition, deep learning, natural language understanding, machine learning as well as text to-speech capabilities. The platform works on the basis of intents and entities recognized from the user's utterances rather than on a predefined flow pattern branching only based on the response of the user.

Dialogflow includes machine learning capabilities to further improve the detection of the intentions from the user utterances. Intents include the following sections: user says, action, response and contexts. Contexts can be used to pass information from previous conversations or external sources. For the intent to be triggered, all the contexts defined for the intent must be active. It is possible to prioritize the intents in case several intents are

identified, define fallback and follow-up intents, and define text responses. Rich responses can be used in case of using one of the following one-click integrations that supports rich responses: Facebook Messenger, Slack, or Telegram.

The entities can be copied or moved to another agent easily through the UI. Exporting the entities in JSON or CSV format and uploading them back is also supported.

3.4 Node.JS

FIGURE 3.8: NodeJS Logo

Node.JS is an open source framework built on top of Google's v8 JavaScript engine for Google Chrome [6, 7]. JavaScript is a light-weight programming language often used in Web browsers [38]. The v8 engine uses an event driven and non-blocking IO model to handle concurrency. Its server applications can be created and executed from command line or Unix Shell [6, 7]. The package manager NPM allows for easy installation and publishing of third-party modules from a large and growing repository [7, 6]. Each module has a manifest file that describes its name, version details and dependencies [39]. In addition to the third-party modules, Node.JS has built-in core modules with support for file system operations, HTTP services, and assertion testing among other things [39]. In node applications, the environment variables in shell are available through the object process. env [40]. It is also possible to listen to signals shutting down the application, which can be used to close down the application gracefully [40].

3.5 Web Hooks

FIGURE 3.9: WebHooks Logo

Web Hook is a design pattern for creating a simple subscription feature enabling users to define callbacks on the Web [41]. By using Web Hook, servers can push event notifications to other servers with HTTP POST requests [42]. The incoming request can be connected to a callback in order to trigger remote events.

3.6 Natural Language Toolkit

Natural Language Toolkit or, more commonly, NLTK is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language. NLTK includes graphical demonstrations and sample data. It is accompanied by extensive documentation, including a book that explains the underlying concepts behind the language processing tasks supported by the toolkit.

NLTK is ideally suited to students who are learning NLP or conducting research in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. The NLTK Project is led by Steven Bird [43].

Using NLPTK [44]. The NLTK website contains excellent documentation and tutorials for learning to use the toolkit. It would be unfair to the authors, as well as to this publication, to just reproduce their words for the sake of this arteicle. Instead, I will introduce NLTK by showing how to perform four NLP tasks, in increasing order of difficulty. Each task is either an unsolved exercise from the NLTK tutorial or a variant thereof. Therefore, the solution and analysis of each task represents original content written solely for this article.

Some of the task which is done through NLPTK is:

- Predicting Words.
- Discovering Part-Of-Speech Tags.
- Word Association.

Python and the Natural Language Toolkit (NLTK) allow any programmer to get acquainted with NLP tasks easily without having to spend too much time on gathering resources. This article is intended to make this task even easier by providing working examples and references for anyone interested in learning about NLP.

The Natural Language Toolkit is a suite of program modules, data sets, tutorials and exercises, covering symbolic and statistical natural language processing.

NLTK is written in Python and distributed under the GPL open source license. Over the past three years, NLTK has become popular in teaching and research.

3.7 Firebase

FIGURE 3.10: Firebase Logo

Basically, Firebase is a platform (Google's product) which let you build the web and mobile application without server-side programming. Firebase provides a real-time database which allows you to store data as well as sync data among users in real-time. It also provides multiple authentication system, various APIs, hosting system, cloud messaging etc [45].

3.8 Firestore

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. Like Firebase Realtime Database, it keeps

the data in sync across client apps through Realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud Platform products, including Cloud Functions [46].

4 Our work

4.1 Reddit dataset preparation

To start we need to know the format of the data, bellow is an example of single entry:


```
{"author": "Arve",
 "score": 0,
 "body": "I just got the new reddit app. So actually, no. :/",
 "score_hidden": false,
 "subreddit": "reddit.com",
 "edited": false,
 "retrieved_on": 1427426409,
 "name": "t1_c0299ap",
 "parent_id": "t1_c02999p",
 "controversiality": 0,
 "id": "c0299ap",
 "subreddit_id": "t5_6",
 "archived": true }
```


First, we need to extract only the useful information from the data, we definitely need the body, comment_id and parent_id, and also the score to be able to filter the comments a little bit more.

The data is divided into months, and because a single month's dataset is more than 50GB, we can't work on it like that because of RAM limitation, so our solution for that is to buffer through the dataset file and save the useful data into an SQLite database. The first comments that we read are going to be parents and will not have a "parent_id", then we get to the replies of those comments where we store their respective "parent_id", so that we have in our database: the parent comment and the reply. And with that we won't be forced to process the whole dataset at once, because we have our pairs (comment, reply), then we divide our SQLite database into samples to load into RAM.

First, we define the function that will create our database.

```
def create_table():
c.execute("CREATE TABLE IF NOT EXISTS parent_reply(
          parent_id TEXT PRIMARY KEY, comment_id TEXT UNIQUE,
          parent TEXT, comment TEXT, subreddit TEXT, unix INT,
          score INT)")
```

Then we start reading from the dataset and writing to our database, and during this process we are going to filter our data a little bit to get rid of useless entries. We created this "acceptable" function, which filters the comment if it is too long or too short, or if it is tagged as "deleted" or "removed" so we don't even need it.

```
def acceptable(data):
        if len(data.split(' ')) > 1000 or Len(data) < 1:
                return False
        elif len(data) > 32000:
                return False
        elif data == '[deleted]':
                return False
        elif data == '[removed]':
                return False
        else:
                return True
```

We also filter the data using the comment score, so that we have some certainty that the comment is reliable, the comment score here is considered a trust score, in our function we go through the replay of each comment, to pair each comment with the replay that have the highest score .

```
if score >= 2:
        existing_comment_score = find_existing_score(parent_id)
        if existing_comment_score:
                if score > existing_comment_score:
                        if acceptable(body):
                                sql_insert_replace_comment
                                (comment_id,parent_id,parent_data
                                ,body,subreddit,created_utc,score)
```

After the preprocessing of the data finished, we ended up with a 4GB database that contains 78 million rows.

The seq2seq model requires 4 files: "train.from" and "train.to", "test.from" and "test.to". The structure of those files is as follows: each line in the ".from" file contains the question, the answer is in the ".to" file in the same line, and the two "train" files are then fed into the neural network to be trained, while the other two test files are used to test the

model at specific checkpoints during the training, where displays the real question with a corresponding answer and the chatbot's answer.

4.2 The model

We used a sequence-to-sequence model. The encoder is a single utterance, and the decoder is the response to that utterance. An utterance could be a single sentence, more than one sentence, or even less than a sentence, or in short anything people say in a conversation. The chatbot is built using a wrapper function for the sequence-to-sequence model.

As mentioned in Chapter 1, the sequence-2-sequence model is combined with an encoder and a decoder which use LSTM:

```
single_cell = tf.contrib.rnn.BasicLSTMCell(
        num_units, forget_bias=forget_bias)
```

Our encoder is setup as follows:

```
encoder_outputs, encoder_state = tf.nn.dynamic_rnn(
    ncoder_cell, encoder_emb_inp,time_major=True,
    sequence_length=source_sequence_length)
```

Note that sentences have different lengths, to avoid wasting computer resources we tell dynamic_rnn the exact source sentence lengths through source_sequence_length.

Our decoder is setup as follows:

```
decoder = tf.contrib.seq2seq.BasicDecoder(
    decoder_cell, helper, encoder_state,
    output_layer=projection_layer)
```

The sequence-to-sequence function is defined as follows:

```
def _seq2seq_f(encoder_inputs,  decoder_inputs,  do_decode):
    return  tf. nn . seq2seq . embedding_attention_seq2seq(
        encoder_inputs,  decoder_inputs,  self. cell,
        num_encoder_symbols=config . ENC_VOCAB,
        num_decoder_symbols=config . DEC_VOCAB,
        embedding_size =config . HIDDEN_SIZE,
        output_projection =self. output_projection,
        feed_previous=do_decode)
```

By default, do_decode is set to be True, which means that during training we will feed in the previously predicted token to help with predicting the next token in the decoder even

if the token was the wrong prediction. This helps approximate the training to be closer to the real environment when the chatbot has to make the prediction for the entire decoder from solely the encoder inputs.

In chapter 2 we talked about evaluating models, we used the BLEU (bilingual evaluation understudy) implementation in the Tensorflow library. Using Tensorboard which is a TensorFlow tool, we see the following graph, Figure 3.11. We see in Figure 3.11 that the BLEU score starts rising after about 10 thousand iterations until it reaches a little bit over 1.4 after 15 thousand iterations, then we notice some fluctuations, but after that we can see that after 40 thousand iterations it starts steadily rising again, but this time in a much slower rate, and we get an increase of about 0.4 over the next 50 thousand iterations, that is why we decided to stop running the training. State of the art machine translation scores are around 27 [26]. To reach a significantly better result will require drastically more computational power for the increase to be worth for us, as we didn't have dedicated resources to run the training on.

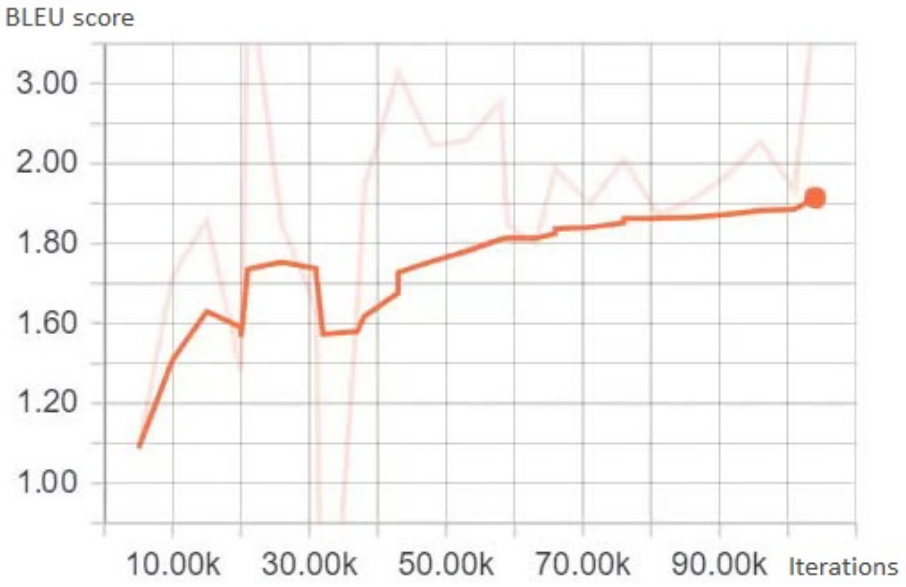


FIGURE 3.11: Progression of BLEU Score during the training of our model

Now let's talk about the Dialogflow chatbot. To create a Dialogflow chatbot, first we create our intents and entities, the entities are the variables that are needed to process the user's request, and the intents are what is expected from the user to say.

Our chatbot can perform many user requests, but one of its most important tasks is to process a ticket booking. First, we created our entities, we have 3 types: train station,

ticket class, and ticket type. The train station entity contains all the available train stations, below we show an example of a couple of values of the train station entity.

```
{
  "value": "Médéa",
  "synonyms": [
    "Médéa",
    "medea"
  ]
},
{
  "value": "Boumerdès",
  "synonyms": [
    "Boumerdès"
  ]
}
```

The ticket class entity contains the two types of tickets: economic and first class. The ticket type lets us know if the user is booking a roundtrip or a one-way ticket.

The intents capture the intention of a user who is interacting with our Agent. Whatever is expected from the user's request from the Agent gets mapped to an intent. When setting up our intents, we fill in some "User Expressions", these are the example phrases/sentences we expect the user to say to our agent, only some examples of user expressions need to be declared because the user's inputs are unpredictable, but with the power of deep learning, the input and our predefined intents can be matches, so that the conversation context is extracted and the required parameters for the intent are validated.

Consider our SNTF Chatbot, the user says: "Book me a ticket from M'Sila to Algiers for tomorrow". The Agent should recognize that the user needs some info on the train leaving M'Sila tomorrow and heading to Algiers. An intent can have "parameters", these are extracted from the user's input. There are required parameters and non-required ones, required parameters must be given by the user for the agent to be able to perform the task, and the agent can ask for those parameters if they are missing by setting up some "prompts". Our SNTF bot requires three parameters to book a ticket: the departure station, the destination station and the time of departure. Bellow, is the code we used for setting up the "from" parameter.

```
{"parameters": [ {
      "id": "65d6d9a0-13a7-4b0e-9864-bff440cb7db1",
      "required": true,
      "dataType": "@Train_Station",
      "name": "from",
      "value": "$from",
      "prompts": [
        {
          "lang": "en",
          "value": "from which station"
        },
        {
          "lang": "en",
          "value": "from which city"
        }],
      "isList": false}]}
```

We can see in the code above that the parameter's required filed is set to true, the "from" parameter takes a data type train_station, and the variable name $from.

Next up we will talk about our webhook. We actually work with real data stored in a Firestore database which is a NoSQL document database, where we store all the train schedules and available trips, as well as the user tickets after booking.

To connect the bot to the database we created a webhook, that we implemented in Firebase. Our webhook function has an intent map which is a dictionary of intent-function, it corresponds the intent send by Dialogflow with a function.

```
let intentMap = new Map();
intentMap.set('Default Welcome Intent', welcome);
intentMap.set('Default Fallback Intent', fallback);
intentMap.set('new intent test', book);
```

When the webhook is called, it automatically maps the intent by parsing the POST request, so the function corresponding with the requested intent will be executed.

```
agent.handleRequest(intentMap);
```

Because we require access to our database, we create a variable and initialize it with a connection to our Firestore database, since we are using Firebase for both the webhook and the database, it provides a configuration field that we can use directly, if we had other

databases in other clouds, we can connect to them as well by providing the connection string and the required authentication. The code bellow shows connecting to our database.

```
const admin = require('firebase-admin');
admin.initializeApp(functions.config().firebase);
var db = admin.firestore();
```

Bellow, in Figure 3.12 we show the structure of our document database.

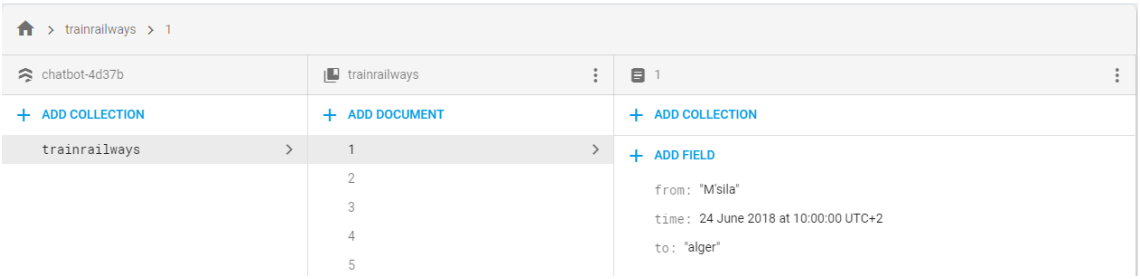


FIGURE 3.12: database infrastructure

In order to get the available trips, we need to get the data from the collection 'trainrailways', then access the document 'railways'.

```
var cityRef = db.collection('trainrailways').doc('railways');
```

Then we run a callback function that is shown below.

```
var getDoc = cityRef.get()
    .then(doc => {
        if (!doc.exists) {
        // handle missing document
        ...
        } else {
        // read the data from the doc variable and filter it
        // use agent.add() to add responses which are then sent back
        ...
        }
    })
    .catch(err => {
        // handle database access errors or other problems
        ...
    });
```

The call-back function will access the database and get the desired document, and then we do a search with the requested parameters from the user, we do any data manipulation required, then the response is sent back to Dialogflow.

4.3 Results and discussion

Full deep learning chatbot

After 3 weeks of non-stop training using this huge dataset and my personal computer, the resulting chatbot was not able to produce good responses, it did not have a personality, it made some grammatical mistakes, all of that was expected due to the dataset we used and the amount of time we were able to run the training. The chatbot did however show some results. We managed to train a model that always generates complete and correct sentences, though they do not make sense in the context of the conversation.

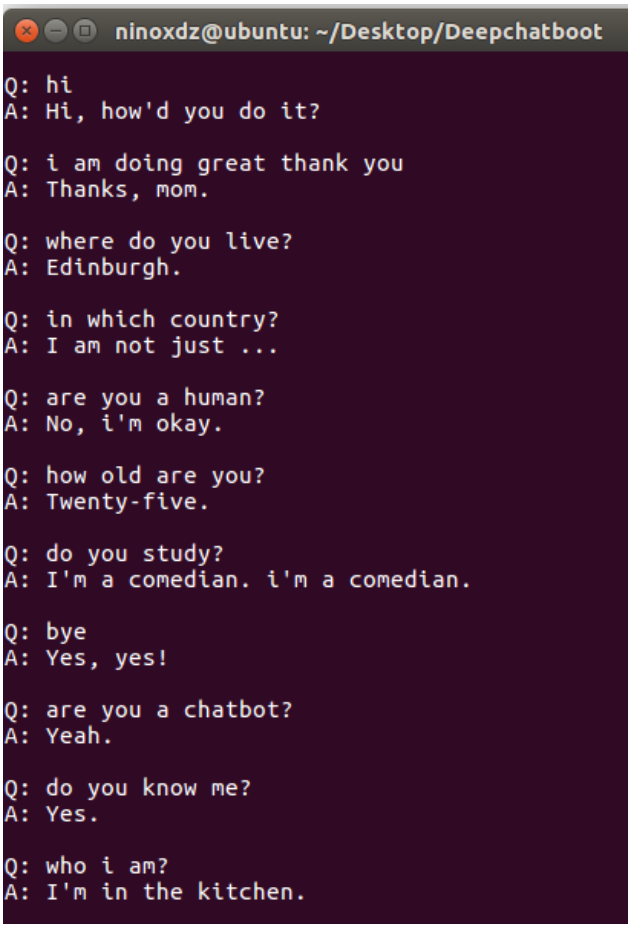


FIGURE 3.13: Deep Learning Chatbot Conversation

The results we achieved with this approach were not completely satisfactory, but still a valuable experience, as we learned first-hand about the difficulties in this field, and we now have several ideas on how to tackle the problems in future works. One of the things that we can use to directly improve our bot, could be using a more specialized dataset, acquiring such a dataset will be a challenge in itself, one way would be to start a community project to build a test dataset for a specific customer service scenario, so that researches can use it to test new methods. Another problem we faced is processing power, that problem is solved by acquiring better hardware or gaining access to a super computer or a computer powerful enough to cut down on training times.

Dialogflow chatbot

With Dialogflow we had a head start with the platform's already trained neural network that can handle conversations with users. It provides automatic extraction of the essential information from the user's input and allows a good level of customizability which is important from both technical and business standpoints. It also allows giving the bot a personality that corresponds with the company's vision.

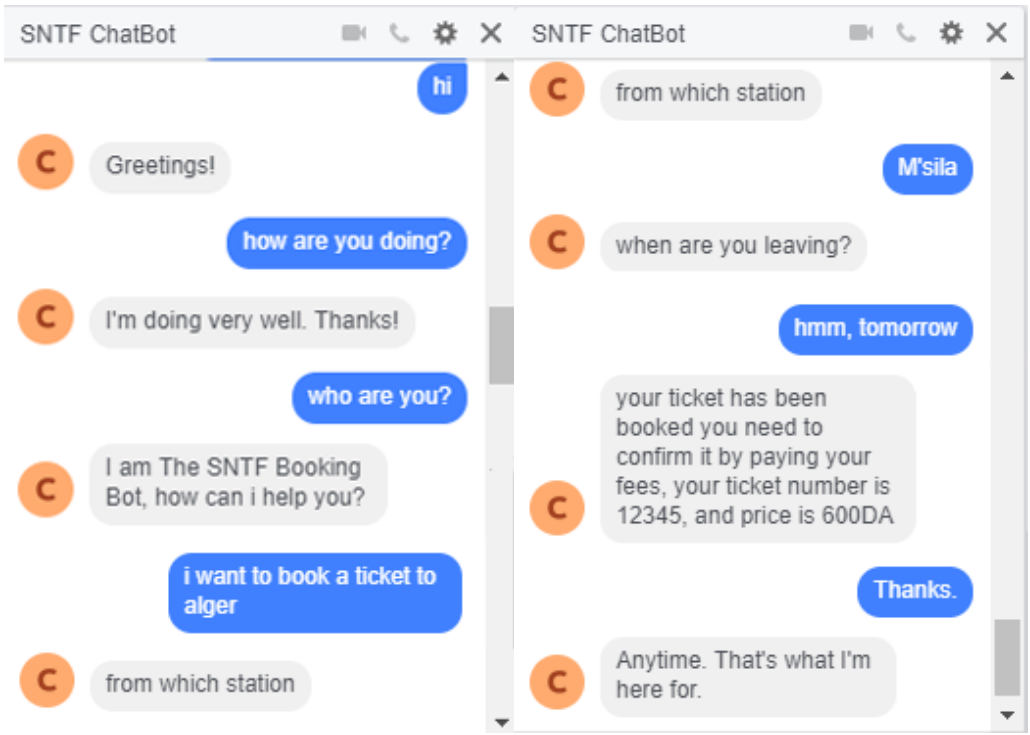


FIGURE 3.14: SNTF bot Conversation on facebook messenger

As can be seen in figure 3.14, the chatbot has the ability to handle normal questions that has nothing to do with the task, from greeting messages to simple questions about its personality. It also detects the user's intent, detect the parameters and can even prompt the user to provide missing ones. Sometimes an external call to webservice or a database is needed, and the bots sends a POST request to a backend webhook that does any work outside the Dialogflow platform.

Next up we present a general conclusion, and future works.

Conclusion and Future works

In this thesis, we set out to make a chatbot geared toward customer support, to tackle the problem companies face in that aspect of dealing with their customers, to offer a more flexible and efficient experience. Our work had taken an unexpected turn in the middle of implementing our model. After weeks of researching, developing, implementing, and finally training the model with the available hardware resources and datasets, as well as exploring some cloud services, we found that the results were unsatisfactory. The resulting chatbot was able to reply to greetings but was not capable of answering domain-specific questions or show a kind of personality. Due to our inability to get better hardware or better datasets, as well as the time restriction, we decided to switch our approach and use a chatbot framework to build our solution, since the requirements of our solution could be met by using such frameworks.

But even though were unable to achieve the desired results from our trained chatbot, we were able to gain valuable information about chatbots and deep learning, mainly about the techniques used in deep learning, such as how to design a conversational agent model, and using the Sequence-to-sequence model, which in turn uses Recurrent Neural Networks and long-short term memory cells, we also learned about the TensorFlow library that we used to implement our model. We also discovered the challenges that can be faced when tackling this issue, as well as what could be the possible solutions to some of the problems we encountered.

For our second chatbot, we used the Dialogflow cloud platform, to implement our SNTF bot that is able to interact with clients, book tickets, answer FAQs, and provide schedule information. The bot is available on Facebook's Messenger, Telegram, and many more potential online services. Learning about Dialogflow and similar platforms, shows that the interest chatbots have and are still generating in all industries and across disciplines, and that the level of reliability that those platforms offer, is significant enough for some startups and even some giant tech companies to invest in such platforms.

We believe that the deep learning approach is still far from being able to result in a customer service conversational agent, due to the inability to control the bot response, making it really hard to produce relevant and correct responses to the customers' inquiries. This problem is being actively researched, and some approaches are being developed to solve it. We also believe that the use of platforms like Dialogflow is going to increase, due to the relatively short time and small resources it takes to develop a reliable chatbot compared to an in-house solution using deep learning or other approaches.

As mentioned before, the building of a customer service chatbot using only deep learning is a new and active area of research. The main aim of future works will be on acquiring a proper customer service dataset, try to explore new deep learning approaches, and maybe use open-source libraries for all parts of our work. One recent and promising approach consists of using two deep neural networks, the first one is a conversation response generator like the one that we worked with, while the second one is a decision making neural network that takes the output of the first one, and decides whether to accept the response, or send it back to generate another one.

As with all research, more work is needed to achieve the desired outcome.

Bibliography

[1] Denny Britz. *Deep Learning for Chatbots*. Apr. 7, 2018. URL: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. 2012.

[3] Lee L. J.s Frey B. J Leung M. K. Xiong H. Y. "Deep learning of the tissue regulated splicing code". In: *Bioinformatics* (2014).

[4] Adit Deshpande. *Chatbot Architecture*. Apr. 2, 2018. URL: https://medium.com/@surmenok/chatbot-architecture-496f5bf820ed.

[5] Matt Schlicht. *The Complete Beginner's Guide To Chatbots*. May 27, 2018. URL: https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca.

[6] G. Chowdhury. "Natural language processing". In: *Annual Review of Information Science and Technology* (2003), pp. 51–89.

[7] R. Ingria S. Miller R. Bobrow and R. Schwartz. "Hidden understanding models of natural language". In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. 1994, p. 25.

[8] R. I. Kittredge E. Goldberg N. Driedger. "Using natural-language processing to produce weather forecasts". In: *IEEE Expert* 9.2 (1994), pp. 45–53. ISSN: 0885-9000. DOI: 10.1109/64.294135.

[9] S. Bangalore O. Rambow and M. Walker. "Natural language generation in dialog systems". In: *Proceedings of the 1st International Conference on Human Language Technology Research*. 2001.

[10] Prasad Rashmi Higashinaka Ryuichiro and Walker Marilyn A. “Learning to Generate Naturalistic Utterances Using Reviews in Spoken Dialogue Systems”. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. ACL-44. 2006.

[11] Lemon Oliver Janarthanam Srinivasan. “Learning Lexical Alignment Policies for Generating Referring Expressions in Spoken Dialogue Systems”. In: *Proceedings of the 12th European Workshop on Natural Language Generation*. ENLG ’09. 2009.

[12] Songsak Channarukul. *YAG: A Template-Based Natural Language Generator For Real-Time Systems*. Tech. rep. 1999.

[13] M. Walker and O. Rambow. “Spoken Language Generation”. In: *ScienceDirect* (2002).

[14] Robert Stufflebeam. *Neurons, synapses, action potentials, and neurotransmission*. Mar. 5, 2018. URL: http://www.mind.ilstu.edu/curriculum/neurons_intro/neurons_intro.php.

[15] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain”. In: *Psychological Review* (1958), pp. 65–386.

[16] Mandar Chandorkar. *DynaML User Guide*. Apr. 10, 2018. URL: https://transcendent-ai-labs.github.io/DynaML/core/core_ffn_new/.

[17] Singer Yoram Duchi John Hazan Elad. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *J. Mach. Learn. Res.* ().

[18] Alex Graves, Santiago Fernández, and Schmidhuber Jürgen Gomez Faustino. “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. 2006.

[19] Schmidhuber Jürgen Graves Alex. “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks”. In: *Advances in Neural Information Processing Systems 21*. 2009, pp. 545–552.

[20] Alex Graves et al. “Unconstrained On-line Handwriting Recognition with Recurrent Neural Networks”. In: *Advances in Neural Information Processing Systems 20*. 2008.

[21] P. Frasconi Y. Bengio P. Simard. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* (1994).

[22] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* ().

[23] Çaglar Gülçehre Fethi Bougares Holger Schwenk Yoshua Bengio Kyunghyun Cho Bart van Merrienboer. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* (2014).

[24] Quoc V. Le Ilya Sutskever Oriol Vinyals. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014).

[25] Adit Deshpande. *How I Used Deep Learning To Train A Chatbot To Talk Like Me*. Apr. 24, 2018. URL: https://adeshpande3.github.io/How-I-Used-Deep-Learning-to-Train-a-Chatbot-to-Talk-Like-Me.

[26] Yoshua Bengio Dzmitry Bahdanau Kyunghyun Cho. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014).

[27] Terry Koo Slav Petrov Ilya Sutskever Geoffrey E. Hinton Oriol Vinyals Lukasz Kaiser. "Grammar as a Foreign Language". In: *CoRR* abs/1412.7449 (2014).

[28] Brendan T Johns and Michael N Jones. "Generating Structure from Experience: A Retrieval-Based Model of Language Processing". In: 69 (May 2015).

[29] Cheng-Te Li Jian-Yun Nie Ming Zhang Dongyan Zhao Yiping Song Rui Yan. *An Ensemble of Retrieval-Based and Generation-Based Human-Computer Conversation Systems*. 2018. URL: https://openreview.net/forum?id=Sk03Yi10Z.

[30] Krupal Modi. *Taking Neural Conversation Model to Production*. Apr. 7, 2018. URL: https://haptik.ai/tech/neural-conversation-model-production/.

[31] Kumar Shridhar. *Generative Model Chatbots*. Apr. 8, 2018. URL: https://medium.com/botsupply/generative-model-chatbots-e422ab08461e.

[32] Sigmoidal LLC. *Deep Learning chatbot – analysis and implementation*. Apr. 9, 2018. URL: https://sigmoidal.io/chatbots-for-b2c-and-deep-learning/.

[33] Python. *What is Python?* May 27, 2018. URL: https://www.python.org/doc/essays/blurb/.

[34] Google. *Alphabet*. Feb. 27, 2018. URL: https://abc.xyz/.

[35] Google. *TensorFlow*. Mar. 20, 2018. URL: https://www.tensorflow.org/.

[36] The Verge. *Google revelead AlphaGo secret Hardware with ASIC chip-tensor Processor Unit for Machine-Learning*. May 27, 2018. URL: http://www.theverge.com/circuitbreaker/2016/5/19/11716818/google-alphago-hardwareasic-chip-tensorprocessor-unit-machine-learning.

[37] Google. *Tensorflow GitHub repository*. Apr. 17, 2018. URL: https://github.com/tensorflow.

[38] David Flanagan. *JavaScript: the definitive guide*. O'Reilly Media, Inc, 2002.

[39] Pedro Teixeira. *Instant Node. js Starter*. Packt Publishing Ltd, 2013.

[40] Guillermo Rauch. *Smashing Node. js: JavaScript Everywhere*. Jhon wiley and sons, 2012.

[41] Liviu-Gabriel Cretu. "Smart cities design using event-driven paradigm and semantic web". In: *Informatica Economica,* (2012).

[42] Trifa Vlad Kamilaris Andreas Pitsillides Andreas. "The Smart Home Meets the Web of Things". In: *Int. J. Ad Hoc Ubiquitous Comput.* ().

[43] WIKIPEDIA. *Natural language toolkit*. May 27, 2018. URL: http://en.wikipedia.org/wiki/Natural_Language_Toolkit.

[44] N. MADNANI. *Getting started on natural language processing with Python. University of Maryland*. May 27, 2018. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.2178\&rep=rep1\&type=pdf.

[45] Vibhu Kumar. *What is Firebase and Its Features*. May 27, 2018. URL: http://www.webtoblog.com/what-is-firebase-and-its-features/.

[46] Google. *Cloud Firestore Docs*. May 27, 2018. URL: https://firebase.google.com/docs/firestore/.

ملخص

يرغب الناس في التواصل مع أجهزة الكمبيوتر بالطريقة نفسها التي يتواصلون بها مع البشر الآخرين، ولم يكن التواصل بين العلامات التجارية وعملائها بهذا القدر من الشدة كما هو في أيامنا هذه. مع التطور السريع للتكنولوجيا، تتغير تجربة العملاء بشكل كبير. يريد العملاء المزيد من خيارات الاستقلالية والخدمة الذاتية، مفضلين إجراء عملية شراء أو الحصول على معلومات دون التفاعل مع الممثل البشري للعلامة التجارية. لذلك، يمكن أن يكون استخدام روبوتات الدردشة في خدمة العملاء حلاً للمسألة الحاسمة المتعلقة بتحسين التواصل مع العملاء. تستخدم الشركات هذه التقنية لخلق تفاعل أفضل مع عملائها بمساعدة منصات الرسائل لتقدم وظيفة دردشة منتظمة، عمليات الشراء عبر الانترنات، والعديد من الوظائف المتقدمة الأخرى. في عملنا، قمنا باستكشاف نظامي روبوتات دردشة مختلفين، أول روبوت هو برنامج روبوت دردشة باستعمال تقنية التعلم العميق للمجال المفتوح، والثاني هو روبوت دردشة لخدمة العملاء قمنا بتصميمه و تدريبه في سحابة جوجل.

كلمات مفتاحية: التعلم العميق ، روبوت دردشة ، خدمة العملاء ،نموذج تسلسل لتسلسل ,TensorFlow

Abstract

People want to communicate with technology in the same manner they communicate with other human beings, and the communication between brands and their clients has never been so intense as it is nowadays. With the rapid development of technology, the customer experience is changing dramatically. Customers want more autonomy and self-service options, preferring to make a purchase or get information without interacting with the human representative of the brand. Therefore, the use of chatbots in customer service can be a solution to the crucial issue of improving customer-brand communication. Companies are using this technology to create better engagement with their clients with the help of messaging platforms, to offer a regular chat function, in-message purchasing, and many other advanced functions. In our work, we have explored two deferent chatbot systems, the first bot is an open domain deep learning chatbot that has been trained on our personal computer, and the second one is a customer service chatbot that we designed and set its training in Google’s cloud platform.

Keywords: Deep learning, chatbot, customer service, Dialogflow, sequence-to-sequence, TensorFlow

Résumé

Les personnes veulent communiquer avec les technologies de la même manière qu'ils communiquent avec d'autres êtres humains, et la communication entre les entreprises et leurs clients n'a jamais été aussi intense qu'aujourd'hui. Avec le développement rapide de la technologie, l'expérience client change radicalement. Les clients veulent plus d'autonomie et de libre-service, préférant faire un achat ou obtenir des informations sans interagir avec le représentant humain. Par conséquent, l'utilisation de chatbots dans le service client peut être une solution à la question cruciale de l'amélioration de la communication avec les clients. Les entreprises utilisent cette technologie pour créer un meilleur engagement avec leurs clients à l'aide de plates-formes de messagerie, pour offrir une fonction de chat régulière, l'achat en ligne et de nombreuses autres fonctions avancées. Dans notre travail, nous avons exploré deux systèmes de chatbot déférents, le premier bot est un chatbot d'apprentissage en profondeur ouvert qui a été formé sur notre ordinateur personnel, et le second est un chatbot de service client que nous avons conçu et mis en place dans le cloud de Google.

Mot clé: Apprentissage en profondeur, chatbot, service client, Dialogflow, séquence-à-séquence, TensorFlow