Conversational AI Assistant Using Artificial Neural Networks:

Implementation of a contextual chatbot framework in a Point-of-Sale system.

Juan Camilo Díaz Herrera

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

CONVERSATIONAL AI ASSISTANT USING ARTIFICIAL NEURAL NETWORKS: IMPLEMENTATION OF A CONTEXTUAL CHATBOT FRAMEWORK IN A POINT-OF-SALE SYSTEM.

by

Juan Camilo Díaz Herrera

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

Advisor / Co Advisor: Mauro Castelli

September 2021

DEDICATION

Dedicated to my parents, for always supporting me in my academic and professional career.

ACKNOWLEDGEMENTS

Special thanks to the company that gave me the opportunity to work in such a wonderful, open minded, and innovative start-up. It was an invaluable learning process and knowledge that will help me in my future experiences as a Data Scientist.

I would also like to express my special gratitude to my beloved partner and undoubtedly backing in life, Fernanda. Without her unconditional support, this process certainly would not be the same.

Finally, I would also want to extend my appreciation to my supervisor Mauro Castelli who gave me the tools, knowledge, and opportunity to work on this project. I sincerely enjoyed his classes and he motivated me to keep researching the field.

ABSTRACT

Artificial intelligence is changing the way how businesses are affronting their day-to-day difficulties. Chatbots are the perfect demonstration of how simple tasks and queries such as customer support or sales metrics and reporting could be solved without human intervention. This project introduced a task-oriented chatbot framework for Spanish language in a Point-Of-Sale webpage. We applied Natural Language Processing (NLP) techniques such as NER and evaluated two supervised learning methods: (i) an Artificial Neural Network (ANN) and (ii) a Support Vector Machines (SVM) model to create a contextualized chatbot that classifies the user's intention in a text conversation, allowing bidirectional human-to-machine communication. These intents could go from simple chitchatting to detailed reports, always providing a natural flow in conversation. The results using an augmented and balanced corpus suggested that ANN model performed statistically better than SVM. Additionally, a real-word scenario with a small-talk survey made to five users gave positive feedback about the quality of predictions. Finally, a software architecture using a PaaS computing service and an API framework was proposed to implement this dialog system in further works.

KEYWORDS

Natural Language Processing, Name Entity Recognition, Natural Language Understanding, Artificial Neural Network, Support Vector Machine, Word Embeddings, Chatbot, Virtual Assistant.

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
DL	Deep Learning
ML	Machine Learning
NN	Neural Network
ANN	Artificial Neural Network
DNN	Deep Neural Network
NER	Natural Language Recognition
NLU	Natural Language Understanding
NLP	Natural Language Processing
SVM	Support Vector Machines
B2C	Business-to-Consumer
BoW	Bag-of-Words
TF-IDF	Term Frequency Inverse Document Frequency
POS	Point-of-Sale
API	Application Programming Interface

1. INTRODUCTION

Conversational software agents activated by Natural Language Processing (NLP), commonly known as chatbots, have been presented as one of the biggest advances in recent years in the fields of Artificial Intelligence (AI) and Deep Learning (DL). Even though there are still some drawbacks in providing an effective chatbot conversation such as grammatical errors, ambiguity, semantics, or language structure, the advances in the field are very promising and, with the introduction of new methodologies in Deep Neural Networks (DNN), these issues are vanishing gradually.

The presence of chatbots in business has been growing rapidly in the last decade. A survey made by Business Insider showed that 80% of the companies are already using or were planning to use chatbots by 2020[1]. Nowadays, a vast number of economic sectors have implemented this kind of emerging technology in their systems with a positive outcome.

According to Business Insider Intelligence, the prediction on global annual cost savings derived from chatbots across the insurance industry will rise from $0.5 billion in 2020 to $5.8 billion in 2025. And this just in sales representatives' expenditures, with the greatest share in Customer Service representatives[2].

This project has been done in the context of an internship dissertation for the master's degree in Data Science and Advanced Analytics at the faculty of Information Management School (IMS) in the Universidade Nova of Lisbon and has the main goal to develop a task-oriented chatbot framework that serves as a virtual assistant for the start-up company 2Luca.

1.1. COMPANY OVERVIEW

2Luca was created in 2018 in Bogotá, Colombia as a Point-of-Sale (POS) web solution for small businesses that goes beyond simple administration of business operations. 2Luca's goal is to give access to data analytics tools to small or newly created businesses and help them to increase their sales, have control of their day-to-day operations and save them time with a smart sales system. To achieve this, the platform processes the customer's data and transforms it into valuable information so the final business will use it as a key point in the decision-making process.

2Luca presents a *Freemium* business model where their customers only pay for what they use and the resources that adapt to their own necessities. The customer creates a free account that gives access to the regular POS System where they can register products, customers, and transactions for free. At this point, the customer can buy different solutions as subscription packages.

Two examples of these solutions are the analytics solution that grants access to a vast number of analytical tools such as product recommendations, customer segmentation, advanced dashboards, pricing recommendation system, and more. The second solution is the chatbot solution, that is associated with this project. Where the customer has the chance to integrate a private virtual assistant that can handle requests such as create customers, give sales reports, create transactions, or give general insights about the business.

[1] https://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12

[2] https://www.businessinsider.com/business-chatbot-examples

Currently, 2Luca has a local coverage within the Colombian territory with more than 50 active clients. However, the business model could be easily extrapolated in the future to other countries of the region. Furthermore, the company is in the latest phase to get a partnership with one of the principal banks in the country that would help to increase the access to bank services in the country by giving credit facilities to small businesses that are currently using the platform.

The kind of customer that 2Luca has at this moment is wide since the platform is easily adapted to multiple business models. Currently, 40% of customers are restaurants or food-based stores such as healthy/organic food or bakeries, 35% are local food markets with small coverage, 15% are gyms or fit-centers and 10% are other kinds of business such as beauty stores, barbershops, and others.

1.2. The team and activities

My role in the company as Data Scientist Intern was to create the framework for a virtual assistant that understands natural language voice/text commands, completes tasks and generates reports for final users; to achieve that, I was requested to use Named Entity Recognition (NER) and Intent Classification (IC) models which can be integrated into a Natural Language Understanding (NLU) service inside the webpage.

Given the current pandemic situation, all the work was done remotely from Lisbon and with the direct support of my supervisor in Colombia. Some specific tasks were given at the beginning of the project and every Friday in a Work-In-Progress meeting all my work during the week was tracked and changes were introduced as needed.

My responsibilities as Data Scientist in the company where:

- Organize the corpus using baseline samples and create synthetic samples for each intent to have a decent number of records that can be used to test the virtual assistant.
- Train and test an NLP model using ML models that classifies customer’s intents and give a proper response to the user using Python language.
- Create an algorithm that allows extracting relevant entities from the text using NER techniques.

1.3. Internship Goals

The goal of this project is to train and test a contextualized AI chatbot that is intended to be used as a virtual assistant for small businesses in the Point-Of-Sale (POS) webpage solution of 2Luca. This virtual assistant should be able to understand simple queries from customers and answer them in the best way possible. Sentences were categorized into different groups of intents that were previously identified by the company as fundamental for this first phase of implementation.

This project is organized as follows. In Section 2 we discuss the literature review that provides a theoretical framework for the work. In Section 3 we explain the methodological workflow. Section 4 presents the results and finally, Section 5 and Section 6 summarize the conclusions and limitations found during this work.

2. LITERATURE REVIEW

2.1. CHATBOTS

A chatbot can be defined as a conversational software system created to emulate human-like conversations that interacts automatically with a user. Chatbots are mostly based on AI techniques that can understand natural language, identify meaning, emotions, and respond in a meaningful way. (Nuruzzaman & Hussain, 2018)

The former applications of conversational agents began with ELIZA in 1966. This bot was developed by the laboratory of Artificial Intelligence of MIT by Professor Joseph Weizenbaum and it was a benchmark in the creation of human-like conversations (Weizenbaum, 1966). After ELIZA, there were more intentions to create a virtual assistant. In 1972, the psychiatrist Kenneth Colby from Stanford University developed an AI-based bot named Parry that simulated patterns from a schizophrenic person (Colby, Weber, & Hilf, 1971).

During the 80's, 90's and 2000 some other bots were created always with the idea to improve natural language. Some examples are *Jabberwacky* (Carpenter, 1997) in the latest 80's, which was the first to implement human interaction; *Dr. Sbaitso* in 1992, created by Creative Labs in Singapore which simulated the role of a psychiatrist; A.L.I.C.E (Artificial Linguistic Internet Computer Entity), created by Richard Wallace in 1995 that was known for the use of heuristic patterns (Wallace, 2009); and the bot *Watson* in 2006 created by IBM as part of the research project *DeepQA* which used machine learning algorithms to answer questions. It was trained using news, libraries, and other open resources.

In 2010 the company Apple developed one of the most popular virtual assistants used nowadays, SIRI. This chatbot was the first virtual assistant build and distributed massively. SIRI was followed by Google Now in 2012, ALEXA and CORTANA in 2015. All of them are mostly based on task-based ML algorithms to understand intents in conversations.

2.1.1. Types of chatbots

There are two types of dialog systems: open-domain and task-oriented. The former has an open-ended goal without pre-defined tasks or labels and presents more challenges to build, whereas the latter is designed to perform specific tasks and has been broadly implemented in real-world applications (Huang & Zhu, 2020). The conversational agent proposed in this project was built under a task-oriented dialog system, also known as contextualized chatbot, using specific tasks related to customer service, reporting, and POS general actions.

These two systems differ also in their architecture. On one hand, task-oriented bots are created under a specific schema in which a set of pre-labeled intents are defined, and each intent is classified. On the other hand, open-domain dialog systems must learn from a multiple variety of sources and fields to be able to maintain a coherent conversation.

The efforts in open-domain agents have been increased rapidly in the last decade. Recently in 2020, Google introduced Meena, an open-domain chatbot trained with a neural network with 2.6 billion parameters that makes much easier the conversational flow in a more cohesive and comprehensive way. Given that number of parameters, the bot can practically talk about any topic. Furthermore, the authors developed a new metric called Sensibleness and Specificity Average (SSA) that captures key elements in human conversation. This metric goes from 0% to 100%, being 100% the best performance. The full version of Meena scores 79%, whereas the light version 72%, overcome just by a normal conversation between humans that has a SSA of 86%. (Adiwardana, et al., 2020)

Task-oriented dialog agents have played an important role in a vast variety of real-world applications. For instance, in the FinTech sector, the German digital bank N26 has implemented an in-house customer service AI that is available 24/7 and can answer 30% of basic customer inquiries in five different languages.[3] This sort of conversational interface has a positive B2C relationship, where final users are improving their experiences in customer support and business are reducing unnecessary costs from both a human and resources perspective.

2.1.2. How they work?

The way how a task-oriented chatbot understands what the user says could be done in many ways. One possible option is by matching the pattern of user input with some pre-defined commands and try to figure out how to answer each sentence. However, this option is very hard to extrapolate to real-world cases, and its lack of flexibility makes it inappropriate on a big scale.

The second possible option is by using a Machine Learning model that classifies the intents of any user's input with NLP. This technique requires a set of inputs that are already labeled by intents and the idea is to classify new inputs into these categories. The key components of a task-oriented chatbot are (Bocklisch, Faulkner, Pawlowski, & Nichol, 2017):

- The **Entities** are relevant variables or features that are extracting from a given sentence and that represent a person, object or characteristic that could help the bot to interact naturally with the human. For instance, in the sentence: "*I need a sales report of the last year*", the term "*last year*" must be interpreted as a date entity. Other types of entities could be names, locations, zip-codes, phone numbers, time, or currency.
- The **Intents** represent the tasks that the bot needs to perform given the user's message. These are the labels to be predicted by the model, and they need to be as general as possible. In a corpus, each intent comes with many utterances that are related to each action. For instance, the sentence "*give me the number of sales today*" represents an intent that queries the total number of sales in a particular timeframe.
- The **Actions** are the responses to be committed by the bot after an intent classification has been done. There are usually functions that take optional parameters with detailed information also called context.

2.2. NATURAL LANGUAGE PROCESSING

The purpose of Natural Language Processing (NLP) is to develop computational mechanisms that allows bidirectional human-to-machine communication in a "natural" way. NLP represents a subfield of Artificial Intelligence where linguistics interacts with computer science using statistical and computational resources. This kind of mechanism helps computers to understand, respond and interact with any source of communication (text or voice) in much the same way humans do. (IBM Cloud Education, 2020)

[3] https://n26.com/en-eu/blog/building-the-most-reliable-customer-service-in-the-world

NLP itself has many components such as text classification, language translation, summarization, question and answering, sentiment analysis, name entity recognition, part-of-speech tagging, and more. All these features allow machines to interact with humans and have numerous practical applications in multiple fields of science in real-world scenarios: Linguistics, biomedicine, conversational chatbots, contextual chatbots, news topic classification, spam detection, customer service, and much more.

2.3. Natural Language Understanding

The idea behind Natural Language Understanding (NLU) is to extract relevant information from text that can help to understand the main idea of the message. The goal of NLU is machine reading comprehension using the context and grammar structures of the text. This structured information about user messages includes user's intents and entities. (Bocklisch, Faulkner, Pawlowski, & Nichol, 2017)

In a chatbot system, NLU is used to understand what the user is trying to convey. For instance, greetings, book a flight, know the balance in an account, etc. This ensures a natural flow in the conversation and improves the user's experience.

2.4. Name Entity Recognition

Name Entity Recognition (NER) is a subset of NLP that has the goal to identify boundaries and types of entities from a given text. Entities are pieces of relevant information that are inside the text and can be used to understand the context. There are many types of entities, the more general ones are names of persons, names of organizations, and locations. However, they will entirely depend on the task or field of study. For instance, in medicine, the identification of gene and protein entities could be relevant.

There are two types of mechanisms that can be used to extract entities in text: The rule-based approach and the statistical approach. The former has the benefit to be simpler to implement but presents several difficulties identifying specific entities, especially proper names. In Spanish, as in many other languages, proper names are particularly difficult to identify since they could have a vast number of structures and can overlap with other words. The goal of NER using statistical models is to recognize this type of entities and resolve the structural and semantic ambiguity in names. (Wacholder, Ravin, & Choi, 1997).

2.4.1. Rule-based approach

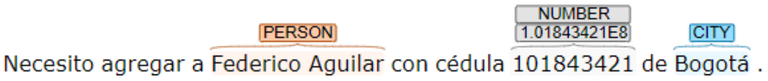
This approach allows extracting entities in text using specific patterns that are previously defined. In many specific cases in which there is a finite number of possibilities and where the pattern is fully and easily recognizable, the rule-based approach is an optimal choice. These kinds of instances can be extracted using specific tokens or regular expressions. Some examples could be IP addresses, URLs, country names, chemical elements, e-mails, or dates (Spacy, 2020).

In practice, this kind of approach is combined with statistical models to boost the results and extract more complex types of entities. However, since statistical models require training data, this approach could be more practical when there is not enough data and could be used as initial method for data collection process.

2.4.2. Statistical approach

There are multiple statistical methods that could be used to extract entities. *Nymble* (Bikel, Miller, Schwartz, & Weischedel, 1998) is a statistical approach used to detect names and other non-recursive entities in text using a slightly modified version of the standard Hidden Markov Model (HMM). Other well-known approach is the Condition Random Fields (CRF) which is essentially a way to combine the conditional probabilities classification with graphical modeling (framework for representation and inference in multivariate probability distributions) and could be used for segmenting and labeling sequence data (Sutton & McCallum, 2012).

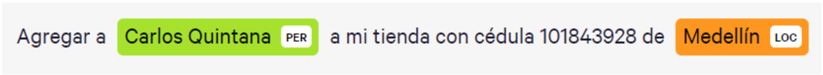
Many open-source entity recognition solutions in the market allow detecting multiple entities from pre-trained models. One example is the *Stanford Named Entity Recognizer* that is a JAVA tool developed by Stanford University and provides a general implementation of linear-chain Conditional Random Fields (CRF) model. Allowing to label sequences of words in a text such as persons, companies, locations, and others. (Finkel, Grenager, & Manning, 2005). CRF models were pioneered by (Lafferty, McCallum, & Pereira, 2001). The package library NLTK offers a module called *StanfordNERTagger* for interacting with the Stanford taggers. This tool also offers a Spanish model available for NER using the CoreNLP toolkit (Christopher, et al., 2014), one example of its usage is illustrated in Figure 1.



Necesito agregar a Federico Aguilar con cédula 101843421 de Bogotá .

Figure 1. Example of NER using Stanford CoreNLP 4.2.0

Another popular tool used for NER is *Spacy Entity Recognizer* that is a transition-based named entity recognition parser. This component recognizes non-overlapping labelled tokens in text using a pre-trained model in which the loss function was optimized for whole entity accuracy. For Spanish language, Spacy uses a pipeline that was trained using the AnCora-ES corpus[4] with more than 500,000 unique vectors in 300 dimensions. The original annotation of this corpus was done as part of the AnCora project directed by the University of Barcelona. This pipeline comes in three different sizes: small (13MB), medium (41 MB) and large (543 MB). Figure 2 illustrates one example using the small model.



Agregar a Carlos Quintana a mi tienda con cédula 101843928 de Medellín

Figure 2. Example of NER using Spacy v2.3 small model.

2.5. TOKENIZATION

Tokenization in NLP refers to the process of chopping a document into pieces called tokens, these tokens correspond to instances of a sequence of characters that are grouped together and provide important semantic information that could be used in posterior models (D. Manning, Raghavan, & Schütze, 2008).

[4] http://clic.ub.edu/corpus/en/ancora

Tokens are crucial factors during the creation of the vocabulary of terms that is used during the vectorization process, and they are relevant to accomplish good results in subsequent NLP analysis such as NER or text classification.

There are many ways to split a text, the most used are by words, characters, sentences, or lines. The simplest type of tokenization method is to divide a string into words using white spaces as separators. However, this method has some drawbacks since punctuation or special cases need to be considered and not all languages have the same grammatical structure which makes tokenization a language-specific problem.

There are many python libraries available for tokenization that can deal with most of these issues. To list just a few of them: Spacy tokenizer, Keras tokenizer, NLTK tokenizer, or Gensim tokenizer. The selection of one of them will depend on the language and the task.

Fortunately, Spanish has not many special cases to consider like hyphenation in English, compound nouns in German, or apostrophes in French. However, there are general cases that are present in multiple languages that we would like to avoid creating a single token, for instance, internal spaces in phone numbers or dates.

The following table shows examples of different methods of tokenization and their common issues:

Original text	“Hola! Necesito ver los reportes del año pasado.”
Word tokens using white spaces	“Hola!”, “Necesito”, “ver”, “los”, “reportes”, “del”, “año”, “pasado.”
Word tokens using white spaces and considering punctuation	“Hola”, “!”, “Necesito”, “ver”, “los”, “reportes”, “del”, “año”, “pasado”, “.”
Char tokens (unigrams)	“H”, “o”, “l”, “a”, “!”, “ “, “N”, “e”, . . . , “p”, “a”, “s”, “a”, “d”, “o”, ”.”,
Bigrams tokens	“Ho”, “ol”, “la”, “a!”, . . . , “pa”, “as”, “sa”, “ad”, “do”, “o.”
Sentence tokenizer	“Hola!”, “Necesito ver los reportes del año pasado.”
Line tokenizer	“Hola! Necesito ver los reportes del año pasado.”

Table 1. Examples of tokenization methods

2.6. Word Representation

Considering that learning algorithms need a numerical representation as input to process the data, the unstructured form in texts needs to be vectorized first in word embeddings. There are several mechanisms that can be used to accomplish that requirement. Namely, Bag-of-words, TF-IDF or World2Vec are the most popular ones. This step is essential to achieve good performance in subsequent natural language processing tasks since it represents the way how the system will interpret the semantics, grammar, and other structures in the text (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

2.6.1. Bag-of-words

Bag-of-words (BoW) is the simplest way to represent text in numerical vectors. It uses a vocabulary of unique words to create a sparse vector that gives a simple and flexible representation of the text, where the final dimension of the embedding will be equivalent to the number of words in the vocabulary (D. Manning, Raghavan, & Schütze, 2008) For this reason, previous steps such as pre-processing or tokenization are highly important to reduce dimensionality since a bad preprocessing means more tokens, and more tokens mean more dimensions.

Consider as example the following three sentences (English translations are inside the parenthesis):

Document 1: *"Me gustaría agregar un cliente a la cuenta"*
(*I would like to add a customer to the bill)*
Document 2: *"Me gustaría agregar un producto a la cuenta actual"*
(*I would like to add a product to the current bill)*
Document 3: *"Me encantaría agregar la cliente a la cuenta"*
(*I would love to add the customer to the bill)*

The vocabulary for these three documents could be represented with the following 11 unique words: "Me", "gustaría", "agregar", "un", "cliente", "a", "la", "cuenta", "producto", "actual", "encantaría". With the following BoW representation:

	Doc. 1	**Doc. 2**	**Doc. 3**
Me	1	1	1
gustaría	1	1	0
agregar	1	1	1
un	1	1	0
cliente	1	0	0
a	1	1	1
la	1	1	2
cuenta	1	1	1
producto	0	1	0
actual	0	1	0
encantaría	0	0	1

Table 2. Example of BoW.

However, there are two drawbacks associated with this approach. The first comes with the high dimensionality that is represented in the sparse vector, which could be mitigated using pre-processing techniques such as lowercase, stop-words, lemmatization, and the use of a good tokenizer. And the second downside, and perhaps the biggest one, is that BoW comes with the lack of retaining important grammatical information in the sentence since the order of the tokens is not taken into consideration. Thus, the document "*Juan está jugando con María*" means exactly the same as the document "*María está jugando con Juan*", in which the subject and object in the predicate are not being part of the vectorization process. (D. Manning, Raghavan, & Schütze, 2008)

2.6.2. Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a statistical method for feature selection that considers the frequency of each term not only inside each document but across the complete corpus. In this method, higher values are given to terms that are rare compared to other terms. This property is particularly important to identify relevant terms that are giving additional information to the document and that are not frequent terms found in several documents. The more times the term appears across all documents (such as stop-words) the less value it has (D. Manning, Raghavan, & Schütze, 2008).

As the name suggests, there are two parts that constitute the TF-IDF statistic, the Term Frequency (TF) and the Inverse Document Frequency (IDF).

Term Frequency (TF)

Term frequency gives the proportion of the number of times a specific term is present inside the document. This could be calculated as follows:

$$tf_{t,d} = \frac{n_{t,d}}{M}$$

Where $n_{t,d}$ represents the number of times the term t appears inside the document d, and M represents the total number of terms in the document d. If the term is highly frequent inside the document, this value will be close to one. If the term is not present in the document, then TF takes the value of zero.

Inverse Document Frequency (IDF)

On the other hand, IDF gives a proportion of how frequent the term is throughout the documents in the corpus. It can be computed as follows:

$$idf_t = \log \frac{N}{df_t}$$

Where N represents the total number of documents and df_t represents the total number of documents with the term t on it. This metric is high when the document is rare to find, and it is close to zero when the term is common. If the term is present in all the document this value should be zero.

TF-IDF Weighting

Once both pieces of the TF-IDF are calculated, it is possible to build the weighted metric as follows:

$$tfidf_{t,d} = tf_{t,d} * idf_t$$

Considering the same three sentences used in BoW for demonstration purpose. We can calculate the terms TF-IDF terms as follows:

Term	$n_{t,d}$			$tf_{t,d}$			idf_t	$tfidf_{t,d}$		
	Doc. 1	Doc. 2	Doc. 3	Doc. 1	Doc. 2	Doc. 3		Doc. 1	Doc. 2	Doc. 3
Me	1	1	1	1/8	1/9	1/8	*log*(3/3) = 0.00	0	0	0
gustaría	1	1	0	1/8	1/9	0/8	*log*(3/2) = 0.18	0.023	0.02	0
agregar	1	1	1	1/8	1/9	1/8	*log*(3/3) = 0.00	0	0	0
un	1	1	0	1/8	1/9	0/8	*log*(3/2) = 0.18	0.023	0.02	0
cliente	1	0	0	1/8	0/9	0/8	*log*(3/1) = 0.48	0.06	0	0
a	1	1	1	1/8	1/9	1/8	*log*(3/3) = 0.00	0	0	0
la	1	1	2	1/8	1/9	2/8	*log*(3/3) = 0.00	0	0	0
cuenta	1	1	1	1/8	1/9	1/8	*log*(3/3) = 0.00	0	0	0
producto	0	1	0	0/8	1/9	0/8	*log*(3/1) = 0.48	0	0.05	0
actual	0	1	0	0/8	1/9	0/8	*log*(3/1) = 0.48	0	0.05	0
encantaría	0	0	1	0/8	0/9	1/8	*log*(3/1) = 0.48	0	0	0.06

Table 3. Example of TF-IDF.

In this toy example, it is possible to notice that stop-words like "*Me*", "*a*", "*la*" have a value of zero in all the documents, which make them less relevant. In contrast, words like "*cliente*" and "*producto*" have higher values and are more relevant. Although this approach gives more information compared to BoW, the issue of not considering the order of tokens is still present. This problem is solved using the embedding representation of the Word2Vec model.

2.6.3. Word2Vec

Word2Vec is a powerful tool to represent words as vectors since it uses embedding-based models to learn the vector representation from the context of the document. In this model, a vocabulary of words is first created from the corpus and then the model learns the vector representation of words. The model was proposed by (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) and has two variants: Continuous Bag-of-Words (CBOW) and Skip-gram model.

In CBOW, the idea is to predict the current word using the embeddings of its context words. In the Skip-gram model, the idea is that given a corpus, the model analyzes each word inside a sentence and tries to predict the surrounding words using the embedding of the current word. (Deng & Liu, 2018)

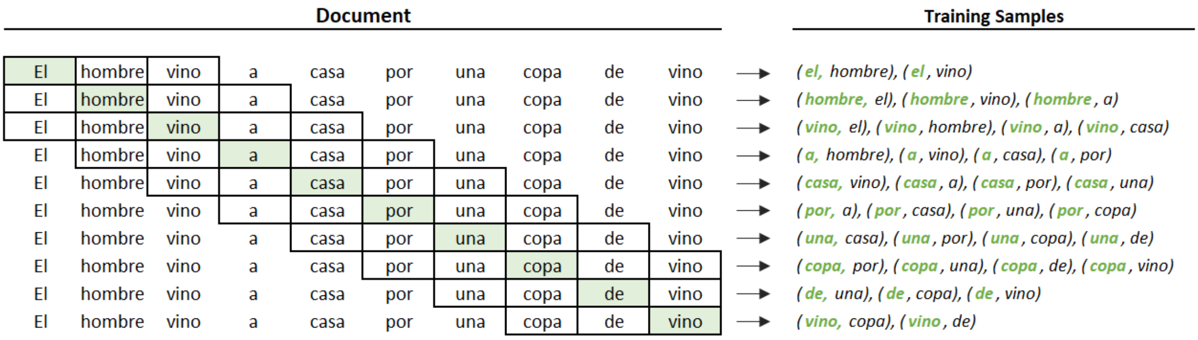
There are some interesting properties of a word vector since it captures many linguistic regularities. For instance, given the transitivity property present in some languages, it is possible to perform vector operations within the word's representation. Furthermore, word embeddings can be also used in unsupervised learning algorithms such as K-means to derive word classes and word similarities. (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013)

Skip-gram model

As mentioned before, the skip-gram model uses the embedding of the current word to predict the surrounding words. The main idea of this model is to obtain the probability of each word in our pre-defined vocabulary set that says how close is each word relative to the current word.

The basic form to represent this model is by using a Feed Forward Neural Network with a single hidden layer, in which the input vector is the embedding of the current word, and the output vectors represent the "proximity" probability of each word in the vocabulary given a window size. Once the model learns, the embedding matrix that carries the weights represents the vectors of each word that captures relevant semantic structures. (Mikolov, Chen, Corrado, & Dean, 2013)

Given the Corpus, the training samples are build using pairs of words in the form *(center, context)* using a specific window size. For instance, if the document is: "*El hombre vino a casa por una copa de vino*" (*"The man came home for a glass of wine"*), then it is possible to create the following 32 training samples using a window size of 2 words:



(el, hombre), (el, vino)
(hombre, el), (hombre, vino), (hombre, a)
(vino, el), (vino, hombre), (vino, a), (vino, casa)
(a, hombre), (a, vino), (a, casa), (a, por)
(casa, vino), (casa, a), (casa, por), (casa, una)
(por, a), (por, casa), (por, una), (por, copa)
(una, casa), (una, por), (una, copa), (una, de)
(copa, por), (copa, una), (copa, de), (copa, vino)
(de, una), (de, copa), (de, vino)
(vino, copa), (vino, de)

Figure 3. Example of skip-gram input.

Notice that the word *"vino"* has two meanings in this sentence, in the first occurence *"vino"* represents the third person of the singular of the verb "come", and in the second occurrence it represents the noun *"wine"*. In simple BoW representation, this word would not be differentiated; however, we expect that the skip-gram model is able to capture this difference since it relies on the context of the surrounding words.

The input vector of the NN system would be a one-hot representation for each center word (the green words in Figure 3). The dimension is given by the vocabulary set. The input is then processed by the NN and the output layer returns the proximity probability for each word in the vocabulary to the input word. The number of units in the hidden layer represents the dimensionality of the final embedding matrix. Figure 4 illustrates this process for the first center word: *"el".*

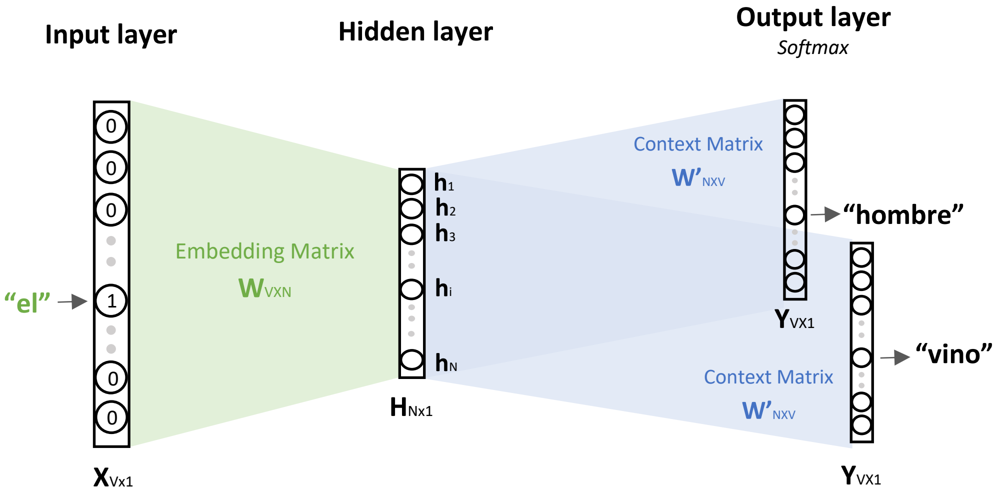


Figure 4. Skip-gram model.

Where V represents the vocabulary size and N the number of units in the hidden layer. The word *"el"* is first represented as one-hot encoding, passing through the hidden layer, and finally producing the output vector for each context word associated with it. Subsequently, the model updates the embedding matrix by back propagating the error that is given by comparing the real labels *"hombre"* and *"vino"* with the predicted vectors, one at a time. The same logic applies for all other center words in the training samples.

At the end of this process, the embedding matrix in the hidden layer (W) contains the vector representation for each of the words. Each word embedding will be N-dimension since it is the number of units in the hidden layer. The number N represents a hyperparameter that needs to be selected. For instance, in (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) a 300-dimensional vector with 3 million English words and a window size of 5 was trained using the Google News dataset.

In practice, there are some open-source python libraries built with pre-trained vector embeddings such as *Gensim* or *Spacy* that can be used for general purposes. The only disadvantage is that the training corpus could not capture specific needs in some close-domain tasks. There are few pre-trained models for the Spanish language, the most reliable is presented by (Cardellino, 2016) which consists of an unannotated corpus of nearly 1.5 billion words with a 300-dimensional vector that was built from multiple web resources.

2.7. SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) is an algorithm used for nonlinear classification and regression that was developed in the early 90's by (Boser, Guyon, & Vapnik, 1992). This approach has shown good performance with small datasets, and it is considered one of the best "out of the box" classifiers. Therefore, there is no need to dedicate extra effort to tune hyper-parameters. SVM are intended for binary classification. However, there are extensions of the model that can be used for multi-class classification problems (James, Witten, Hastie, & Tibshirani, 2013).

SVM are a clear example of learning patterns from examples, and they can achieve high performance in real-world applications. Generally speaking, SVM is a linear model in a high-dimensional feature space where kernel functions are used to perform computations directly in the input space (James, Witten, Hastie, & Tibshirani, 2013).

Furthermore, SVM can be contrasted with a feed-forward neural networks since their structure is similar because both induce an output function which is expressed as a linear combination of simple functions. For instance, the number of support vectors can correspond to the number of hidden layers, the kernel function to the activation function, the support vector to the hidden layer weights, and the coefficients found by the convex optimization problem in SVM correspond to the output layer weights from an ANN point of view (Moraes, Valiati, & Neto, 2013) (Romero & Toppo, 2007) (Joachims, 1998).

Support vectors are coordinates of individual observations that support the decision boundary that is given by the distance between a threshold represented by a hyperplane and the support observations. If the threshold allows for misclassified observations to avoid outliers, this distance is called the soft margin. The Support Vector Classifier (SVC) is the hyperplane that gives the widest separation possible between the classes. This SVC could be a point in a 1-dimensional space, a line in a 2-dimensional space (illustrated in Figure 5), a plane in a 3-dimensional space, or any hyperplane in higher dimensional spaces.

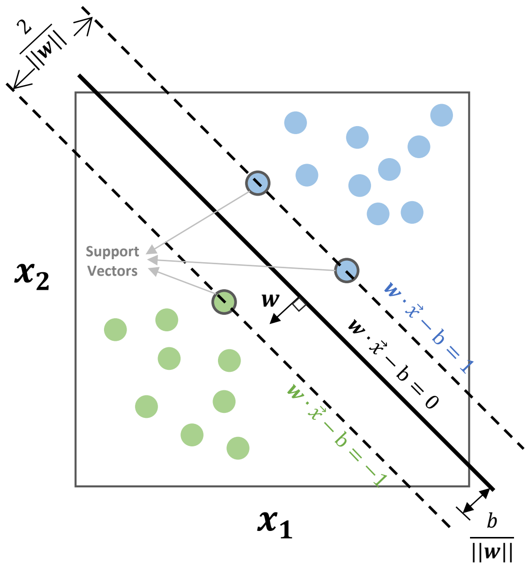

$\frac{2}{||\boldsymbol{w}||}$
$\boldsymbol{x_2}$
Support Vectors
$\boldsymbol{w}$
$\boldsymbol{w}\cdot\vec{x}-b=1$
$\boldsymbol{w}\cdot\vec{x}-b=0$
$\boldsymbol{w}\cdot\vec{x}-b=-1$
$\boldsymbol{x_1}$
$\frac{b}{||\boldsymbol{w}||}$

Figure 5. SVM representation in 2D space.

Given the training samples matrix $\boldsymbol{X}_{N\times M}$ with dimensions equal to N number of observations and M number of features. And the binary label vector $\boldsymbol{y}$ with $N \times 1$ dimensions. Then, the linear model for each observation i could be express as follows:

$$y_i \times (\boldsymbol{w}\cdot\vec{x_i} - b) \geq 1$$

$$\boldsymbol{w}\cdot\vec{x_i} - b \geq 1 \qquad \rightarrow \quad if\ y_i = 1$$

$$\boldsymbol{w}\cdot\vec{x_i} - b \geq -1 \qquad \rightarrow \quad if\ y_i = -1$$

Where $\boldsymbol{w}$ and b correspond to the vector of weights with dimensions $M \times 1$ and the scalar bias, respectively. In order to find these two variables in the equation system, it is necessary to use a cost function and apply gradient descent. One example of cost function is the Hinge Loss function:

$$H = \max\,(0, 1 - y_i \times (\boldsymbol{w}\cdot\vec{x_i} - b))$$

The SVM is created by solving the constrained minimization linear system. To do so, it is necessary to minimize the Lagrange function L as follows:

$$L = \lambda||\boldsymbol{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \max\,(0{,}1 - y_i \times (\boldsymbol{w}\cdot\vec{x_i} - b))$$

This function is composed by two parts, the first element is intended to maximize the margin ($2/||\boldsymbol{w}||$) and the second part is the cost function. L gets the following form depending on the value of the linear model:

$$L = \begin{cases} \lambda||\boldsymbol{w}||^2, & y_i \times (\boldsymbol{w}\cdot\vec{x_i} - b) \geq 1 \\ \lambda||\boldsymbol{w}||^2 + 1 - y_i \times (\boldsymbol{w}\cdot\vec{x_i} - b), & otherwise \end{cases}$$

Therefore, it is possible to calculate the gradients that will be used to update the weights and biases. These gradients correspond to the derivative of the Lagrange function with respect to w and b, and they are calculated for each of the two conditions as follows:

$$\left(\frac{dL}{dw_k}, \frac{dL}{db}\right) = \begin{cases} (2\lambda w_k \ , 0) \ , & y_i \times (\boldsymbol{w}\cdot\vec{x_i} - \mathrm{b}) \geq 1 \\ (2\lambda w_k - y_i\cdot\vec{x_i} \ , y_i) \ , & otherwise \end{cases}$$

Thus, the update rules for each training sample x_i using the learning rate α is expressed as:

$$\boldsymbol{w} = \boldsymbol{w} - \alpha\cdot\frac{dL}{dw_k} \qquad\qquad b = b - \alpha\cdot\frac{dL}{db}$$

The SVC is used for binary classification and if the boundary between the two classes is linear like the one shown in the Figure 5. However, for non-linear boundaries like Figure 6, the Support Vector Machines (SVM) is used. SVM is an extension of the SVC that is the result of expanding the feature space to a higher dimension using kernels in which the idea is to accommodate a non-linear boundary between the classes. (James, Witten, Hastie, & Tibshirani, 2013). There are different types of kernel functions to use. Namely linear, polynomial, Radial Basis Function (RBF) or sigmoid are the most common ones.

Polynomial Kernel

The polynomial kernel function for two observations x_1 and x_2 with a polynomial degree d and polynomial coefficient r is:

$$K(x_1, x_2) = (x_1 x_2 + r)^d$$

In the polynomial kernel, the parameters r and d are considered hyperparameters and are obtain by cross-validation.

Radial Basis Function kernel (RBF)

The Radial Basis Function kernel for two observations x_1 and x_2 could be expressed as follows:

$$K(x_1, x_2) = e^{-\gamma(x_1 - x_2)^2}$$

Where the parameter γ scales the influence between the two observations. A good initial measure to use for this parameter could be $(M \times var(X))^{-1}$. The RBF kernel represents the data in infinite dimensions using Taylor series, contrary to the polynomial kernel, in which the dimensionality is given by the polynomial degree.

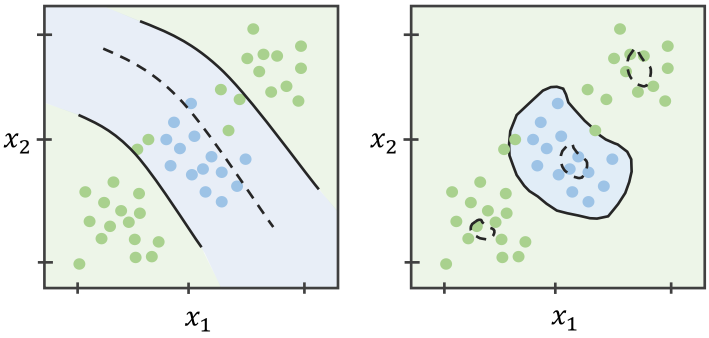


Figure 6. Representation of polynomial and RBF kernel.

2.8. Artificial Neural Networks (ANN)

The main idea of a Neural Network is to recognize underlying patterns and relationships in data. This goal is achieved by using algorithms that mimic the way our brain works. Biologically speaking, the brain operates using neurons which send impulses to communicate with other neurons in a process called synapsis. This biological process could be broadly replicate with algorithms where information is passed through the network in a way the system learns relationships via error correction mechanisms (IBM Cloud Education, 2020).

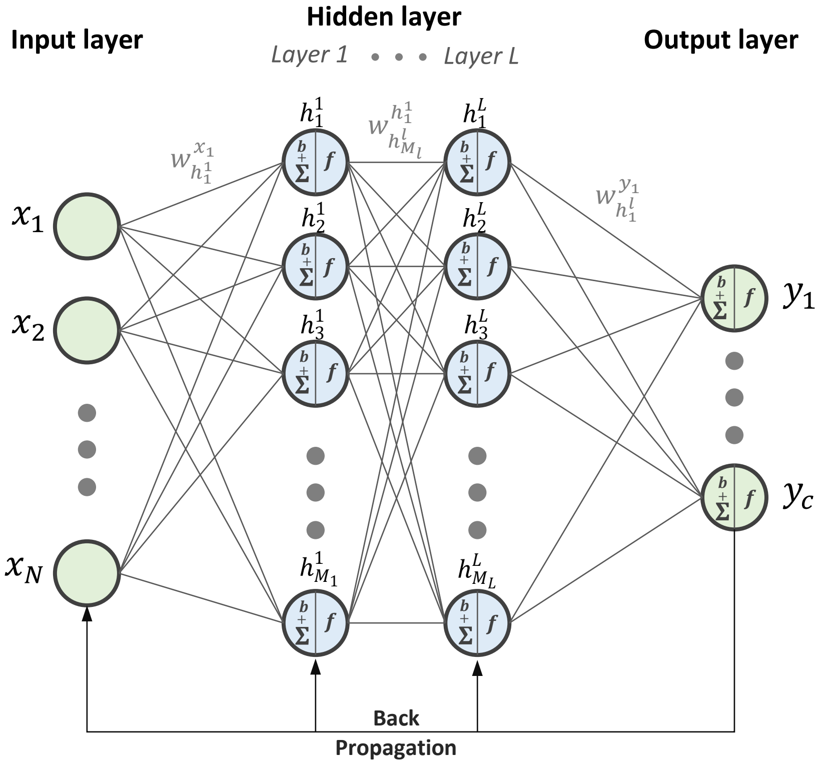


Figure 7. Neural Network scheme.

A network with two or more hidden layers is called a Multi-layer Perceptron Network (MLP) with a feed forward mechanism like the one in Figure 7. In this network, the neurons (or nodes) are interconnected and communicate with each other by passing pieces of information the same way the synapsis does. Each neuron is also called a perceptron, and each perceptron has the job of aggregating and processing the signal produced by prior layers and send new information to posterior layers. This information could be processed in several ways. The most common one is by using an activation function that introduces non-linearity to the system (James, Witten, Hastie, & Tibshirani, 2013).

In an ANN like the one in Figure 7, the aggregation $\boldsymbol{\Sigma}$ and activation $\boldsymbol{f}$ can be represented as follows:

$$\boldsymbol{f}(\boldsymbol{\Sigma}) = \boldsymbol{f}\left(\sum_{i=1}^{N} x_i w_{ij} + b\right) \qquad \forall j = 1, \dots, L$$

The activation function $\boldsymbol{f}(\boldsymbol{\Sigma})$ represents a differentiable and non-linear function that is used to learn complex patterns or relationships in the hidden layers. This activation function makes neural networks robust, since in most cases data comes from a non-linear generation process. There are several types of activation functions that could be used in an ANN, namely sigmoid, ReLU (Rectified Linear Unit), ELU (Exponential Linear Unit), SELU (Scaled Exponential Linear Unit), tanh, SoftPlus, Exponential, SoftMax and more. Figure 8 illustrates some of them. However, in practice the most frequently used are: ReLU, Sigmoid and Tanh.

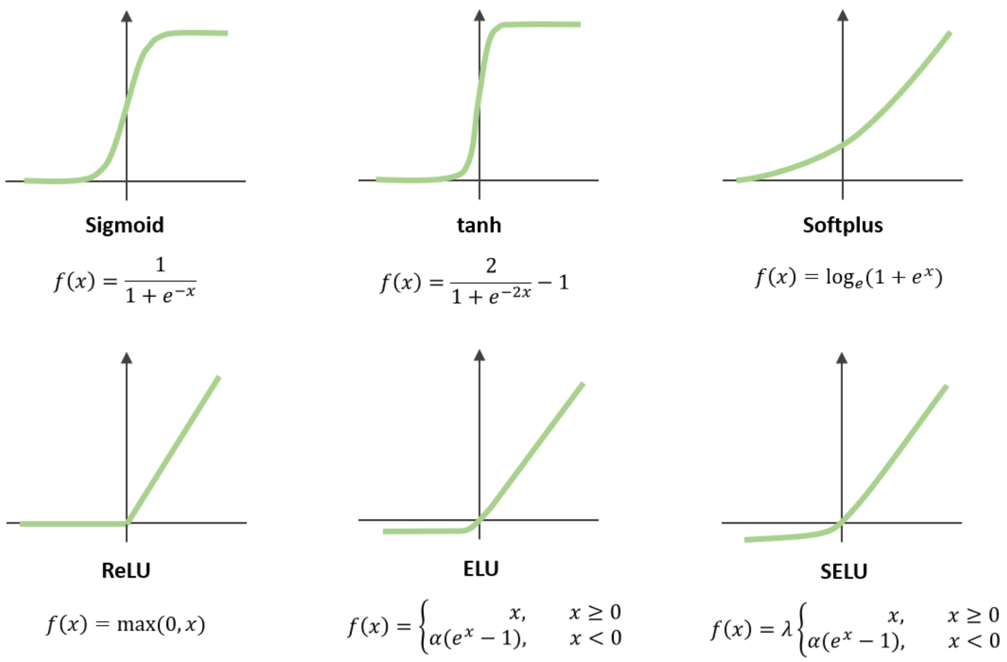


Figure 8. Representations of activation functions.

The simplicity and efficiency of ReLU function makes it perhaps the most popular one nowadays. Its functional form is basic but powerful. It returns zero when the input is negative and returns the same input value otherwise. This function overcomes the problem of vanishing gradient that is present in multi-layer feed-forward neural networks, in which the network is not able to propagate information using the gradient during back propagation process. This issue limits the learning process and makes the model prematurely converge to non-optimal solutions (Goodfellow, Bengio, & Courville, 2016).

The type of activation function used in the hidden layers could be selected via hyperparameter optimization. However, the activation function used in the output layer should be selected depending on the number of classes in the predicted variable. For instance, a binary classification problem should use a *sigmoid* activation function since the outcome has two mutually exclusive classes. Here the output is rounded to get the probability that belongs to one class. On the other hand, for a multiclass classification problem, like the one presented in this project, the *softmax* function is commonly used since it returns a normalized vector of probabilities, and the final class is chosen by taking the maximum argument of that vector.

3. METHODOLOGY

3.1. METHODOLOGICAL WORKFLOW

The implementation of the task-oriented chatbot in this project involves the traditional methodology followed during any NLP task. Figure 9 presents the schematic pipeline with the detailed workflow. This pipeline is composed of three main building blocks, namely data preprocessing, intent classification, and entity extraction.

During the first stage of the pipeline, the corpus is first augmented and then preprocessed to clean the sentences before using them as inputs. These first two steps are crucial to obtain quality results since they help reducing dimensionality and extracting relevant features in the text. After this phase, the data is tokenized and then vectorized to transform the unstructured data to a numerical representation that can be read by the algorithms.

Once the data goes through the first stage, machine learning models are trained to classify the intent attached to each of the utterances. In this stage, two models were evaluated: the first one, a Feed Forward Neural Network, and the second one a Support Vector Machine model.

Once the intent is classified, during the final stage, the entities in the text are detected in order to identify the key pieces of information that will allow the bot to provide a proper response. In the following subsections, these three stages with their components will be fully detailed.

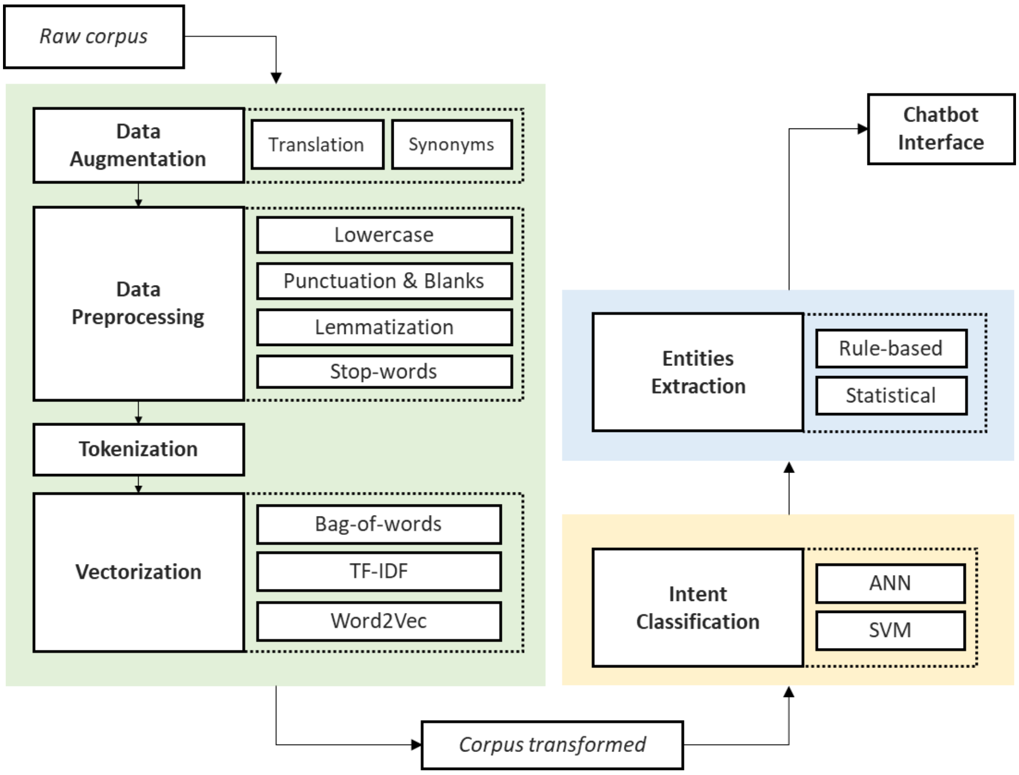


Figure 9. Methodological workflow

3.2. DATA

The dataset used for this project was gathered from a collection of documents provided by the company that was built from private conversations with their final customers. This dataset was hand labeled to match the option that best represents the intention of the utterance. The total number of sentences in this dataset was 7122 with an imbalanced class distribution as shown in Table 4; nevertheless, the next section shows how the data was augmented to balance the distribution.

A total of 18 intentions divided into 4 domains were used. Three of these four domains correspond to the main agents that make up the platform, namely customers, items, sales, and bills. While the fourth is dedicated to the bot's natural talk that allows maintaining a basic conversation with the final user, for instance, greetings, thanks, or support. This last domain was created by hand and contains the main intents used by a simple chatbot system. Table 4 presents a summary of these intents and their description.

Domain	Intent	Description	Sentences
Customer	*ItemAdd*	Creates a new item (product) in the system	1000 (14.1%)
Item	*CustomerAdd*	Creates a new customer in the system	1000 (14.1%)
Sales	*SalesStrategy*	Gives a sales strategy that helps the client	1000 (14.1%)
	SalesReport	General report of sales	664 (9.3%)
Bill	*CurrentBillClearAll*	Clears all information in the current bill	670 (9.4%)
	CurrentBillProcess	Starts process payment in the current bill	390 (5.5%)
Bot Natural Talk (BNT)	*GreetingsHello*	BNT - Hello from user	198 (2.8%)
	GreetingsBye	BNT - Bye from user	183 (2.6%)
	AnswerThanks	BNT - Thanks from user	119 (1.7%)
	AnswerPositive	BNT - User's positive response	432 (6.1%)
	AnswerNegative	BNT - User's negative response	72 (1.0%)
	BotReal	BNT - User asks bot if it is real	531 (7.5%)
	BotName	BNT - User asks for bot's name	181 (2.5%)
	BotCompliment	BNT - User compliments the bot	79 (1.1%)
	BotAge	BNT - User asks bot's age	92 (1.3%)
	BotFunctionality	BNT - User asks bot's functionalities	81 (1.1%)
	AnswerYourWelcome	BNT - Your welcome from user	74 (1.0%)
	GreetingsHowRU	BNT - User asks about the bot status	356 (5.0%)
		Total	**7122**

Table 4. Description of Intents

3.2.1. Data Augmentation

Some studies have found that appropriate data augmentation methods are useful for controlling generalization error for deep learning models and can boost short text classification (Marivate & Sefara, 2020). Similar to computer vision, these techniques could be applied to text-based models and may help the ML algorithms to deal with imbalanced data. However, data augmentation in text should be completed carefully since the exact order of words and characters may contain syntactic and semantic meaning. (Zhang, Zhao, & LeCun, 2016)

Even though, this data augmentation process should be done by rephrasing the sentences one by one, by hand, and maintaining the grammatical structure of the sentences, in practice this approximation is unrealistic and highly expensive. However, there are other methods available for this purpose. For instance, using synonyms (Zhang, Zhao, & LeCun, 2016), using word embeddings (Yang Wang & Yang, 2015), using back-translation (Sennrich, Birch, & Haddow, 2016), or using Natural Language Generation methods (Kafle, Yousefhussien, & Kanan, 2017).

As mentioned before, the data augmentation process in this study was performed since the initial corpus contains an imbalanced number of classes, as shown in the left-hand chart in Figure 10. These issues were solved using three methods. The first one creating synthetic documents using synonyms, the second one using back-translation, and the last method introducing misspellings in the text.

In the first approach, a list of 483 Spanish synonyms was created using the *Real Academia Española* (RAE) dictionary as a reference. To maintain the syntax of the sentence, all synonyms were carefully selected always respecting the semantic of the text. For instance, the grammatical structure of the word was maintained, so verbs were transformed to verbs, nouns to nouns, and so on. Likewise, since the Spanish nouns have lexical gender, their transformations were also considered.

The second approach was performed using the back-translation method. In this method, a percentage of the training samples were translated from the target language, in this case Spanish, to other languages, and then the translated sentence was translated back to the original language. Two python libraries were used for this step: The library *googletrans*[5] that uses the Google Translate API, and *TextBlob*[6] that is a library for processing textual data with common NLP tasks.

Lastly, augmenting data using possible Spanish spelling errors like “s” instead of “c” or skipping an “h” in a world was also considered. A list of all Spanish misspellings can be found in Annex 8.1. Thus, at the end of this process, the dataset was augmented by 153% passing from 7122 to 18022 documents with a balanced class distribution. An overview of the final number of sentences per intent is shown on the right-hand side of Figure 10.

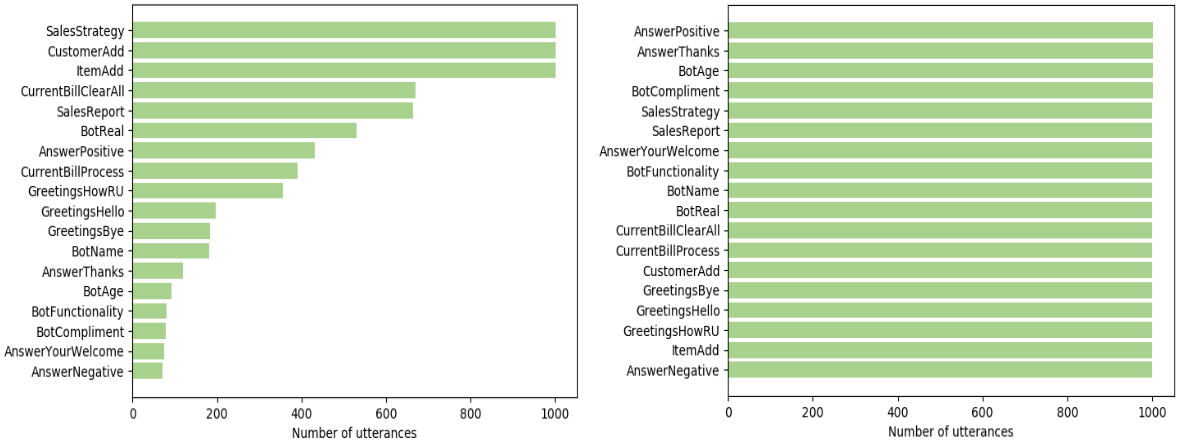


Figure 10. Number of sentences per intent before (left) and after (right) augmentation.

[5] https://py-googletrans.readthedocs.io/

[6] https://textblob.readthedocs.io/

3.2.2. Data Preprocessing

During this stage, the idea is to process the raw corpus that is composed of pairs of sentences and labels and create the dataset that will be used to train the Machine Learning models. Thus, the models will learn to identify only the relevant patterns in data that are needed to classify intents and not any other unnecessary aspect of the text that would not be relevant to include.

As mentioned previously in the workflow schematic in Figure 9, the preprocessing procedure consists of the following ordered steps:

1. **Lowercase the sentences**: This step allows the tokenizer to not duplicate tokens whose meaning is the same, but the only difference is the capitalization.
2. **Delete punctuation marks**: In this step any special character that could unnecessarily enlarge the dimensionality and complexity of the model was removed. Namely, the following characters were removed: !¡?¿|=:()«»;,.*´'_{}#^~<>\
3. **Lemmatization:** The normalization technique of Lemmatization was applied using SpaCy's python library (since it is the only ready-to-use library that allows for Lemmatization in Spanish). This step is especially important to reduce dimensionality since the way a user gives instructions to the bot could be expressed using several forms of the verbs, or in singular/plural form, but the meaning remains the same. Thus, lemmatization takes the root form of the inflected word and considers the morphology of the term. For instance, the sentences "*agregar el cliente*", "*agrega los clientes*" and "*agregando los clientes*" have the same root "*agrega el cliente*" or "*add customer*" and they represent the same idea that is aligned with the intention.
4. **Replace any vocal accent mark from Spanish language for the root character, *i.e. áéíóúü***: This step was done for the same reason as the lowercase step, and also to prevent possible orthographic errors from the final user and bearing in mind that in most of the cases the accent marks in Spanish are used only to emphasis the stressed syllables, so the pronunciation gets clearer but the meaning of the sentence remains the same[7]. It is important to mention that it needs to be done after lemmatization, so it does not interfere with the process.
5. **Stop words:** The following customized stop words were removed since their contribution to the intent classification is assumed to be insignificant: *'de', 'del', 'la', 'con', 'el', 'los', 'las', 'por', 'mi', 'mis', 'tu', 'te', 'su', 'me', 'que', 'pero', 'en', 'es', 'a'*.
6. **Remove multiple white spaces between words**: In this step, any blank space with two or more consequently blank spaces were replaced by a single space using the following regular expression: *\s{2,}*
7. **Strip**: In this step white spaces at the beginning and end of each sentence were removed.

[7] There are circumstances where the meaning of a word could change entirely if the accent marks were not used correctly. Some accent marks in the last vocal of the word could modify both the tense and subject of the sentence. For instance, *camino* (I walk) and *caminó* (You/he/she walked). However, since these special scenarios of confusion are rare to find in the context of this chatbot and especially in short-length sentences, the cost of not including them is presumed to be small.

3.2.3. Data Exploration

To better understand and analyze the text contained in the document, three data exploration analyses were performed. The first corresponds to a simple word frequency, the second a sample length distribution and the third a graphical representation of the word embeddings by unigrams, bigrams, and trigrams. All three methods were done after preprocessing the sentences.

Table 5 presents the top five terms that appear most frequently in all documents by category of intent and overall. Here, it is possible to observe how each category has terms that are more frequently used within the group. For instance, the category *Customer* contains unigrams like "*usuario*" and "*consumidor*" that are synonyms of the way a customer is called; furthermore, it contains bigrams that are related with customer's personal information, such as *"correo electronico"* or *"numero cedula".* The same logic occurs in the rest of the categories. These word frequencies help to understand how future models would classify the intention that is attached to the documents.

	Unigrams	Bigrams
All	gracias (*thanks*)	correo electronico (*e-mail*)
	como (*how*)	nuevo código (*new code*)
	eres (*you are*)	cuanto han (*how much have*)
	nuevo (*new*)	e mail (*e-mail*)
	hola (*hello*)	factura actual (*current invoice*)
Customer	usuario (*user*)	correo electronico (*e-mail*)
	consumidor (*consumer*)	e mail (*e-mail*)
	correo (*mail*)	numero cedula (*national ID number*)
	cliente (*client*)	direccion correo (e-mail *address*)
	nuevo (*new*)	mail (*mail*)
Items	nuevo (*how*)	nuevo codigo (*new code*)
	codigo (*code*)	servicio nuevo (*new service*)
	producto (*product*)	item nuevo (*new item*)
	nombre (*name*)	producto nuevo (*new product*)
	item (*item*)	articulo nuevo (*new article*)
Sales	ventas (*sales*)	cuanto han (*how much have*)
	como (*how*)	han aumentado (*have increased*)
	compras (*purchases*)	han crecido (*have increased*)
	ganancias (*earnings*)	han incrementado (*have increased*)
	han (*have*)	como puedo (*how can I*)
Bill	actual (*actual*)	factura actual (*current invoice*)
	factura (*invoice*)	actual gracias (*current thanks*)
	ayuda (*help*)	cuenta actual (*current bill*)
	gracias (*thanks*)	quisiera ayuda (*I would like help*)
	todo (*all*)	necesito ayuda (*I need help*)
Bot Natural Talk	eres (*you are*)	quien eres (*who are you*)
	hola (*hello*)	eres muy (*you are too*)
	esta (*to be*)	cuantos años (*how many years*)
	como (*how*)	como llamas (*how is your name*)
	gracias (*thanks*)	eso esta (*that is*)

Table 5. Word frequency by category

The second exploration uses the sample length distribution to give an idea of how long the sentences are in terms of number of characters. Figure 11 provides two insights; first, the average sentence has around 25 characters, which means that the user inputs are quite concrete and small which is expected in a conversation with a chatbot, and lastly, there are no outlier sentences in Corpus.

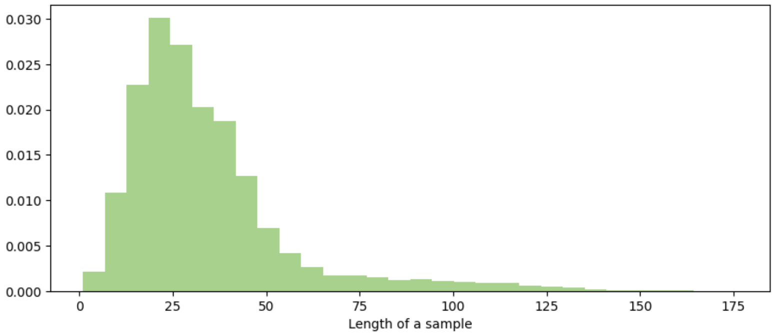


Figure 11. Sample Length Distribution

The last exploration technique used is the graphical representation of word embeddings. In which it is possible to understand how similar terms are grouped together since they presumably represent the same idea. Figure 12 shows on the left side the word embeddings representation using unigrams and in the right side the clusters representation using the unsupervised learning method of K-means with 13 clusters, that were determined using the elbow method.

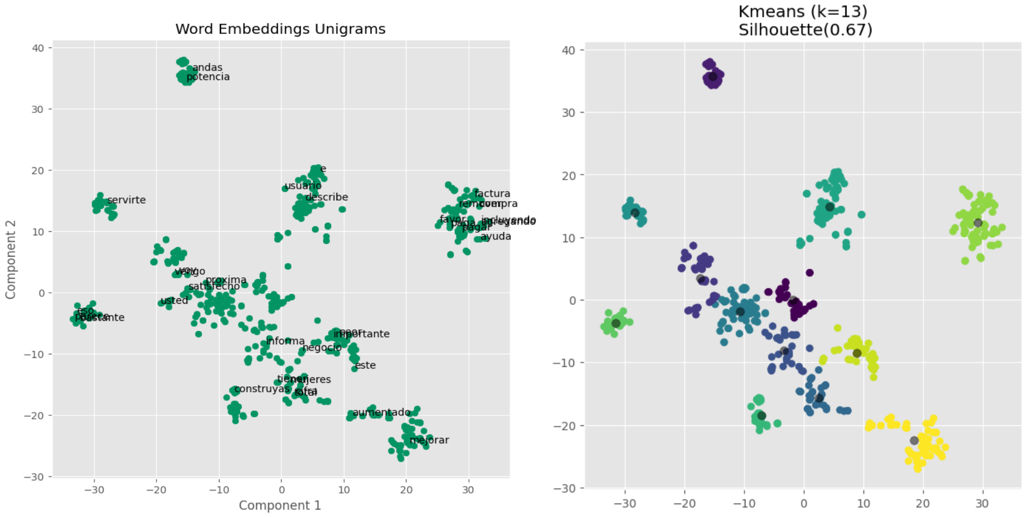


Figure 12. Word Embeddings Unigrams

Among the most representative clusters, it is possible to find the group that helps identifying the purchase process, it includes words like: "*factura*" (*bill*), "*compra*" (*purchase*) or "*pago*" (*payment*). Another relevant cluster detects how the user approaches the bot and asks for support, it includes words like: "*colaborarme*" (*formal help*), "*explícame*" (*explain me*) or "*ayudarme*" (*help me*). The last group to highlight is the cluster that gathers the way the business refers to itself, for instance: "*bar*", "*restaurante*" (*restaurant*), "gimnasio" (*gym*) or "peluquería" (*hair saloon*). All these clusters give an idea of how the algorithm will extract relevant features using word embeddings, grouping them by similarity and use them to identify the intention. Further analysis using bigrams, and trigrams can be found in Annex 8.2.

3.2.4. Vectorization

Once the preprocessing is done, the next step is to transform the unstructured corpora to a numerical representation, so the subsequent ML models will be able to handle the input. To achieve this, all three vectorization methods presented during the literature review were tested independently. Namely BoW, TF-IDF and Word2Vec. For that purpose, the following grid was built:

1. **Three types of analyzers**: Splitting by words or splitting by characters and words.
2. **Multiple types of n-grams**: Unigrams, bigrams, trigrams, and their combinations.
3. **The minimum document frequency**: Remove terms that only appear a specific number of times, varying from 1 time to 3 times.
4. **The maximum document frequency**: Remove terms that appear more than a specific percentage throughout all documents, varying from 85% to 100%.

The Word2Vec methodology was applied using the skip-gram model provided by *genism*[8] and using the pre-trained Spanish embeddings presented in the literature review (Cardellino, 2016), which consists of a corpus of nearly 1.5 billion words with a 300-dimensional vector. This model uses as input the sequence of integers for each document, where each integer represents the index of a token in a vocabulary dictionary, for this purpose, the *Keras* tokenizer was used.

Since the embedding matrix was created using the available corpus, not all the vocabulary was covered by the pre-trained model. In fact, the percentage of vocabulary covered using the augmented dataset was only of 43.41%. This issue is happening since most of the unknown terms come from the misspelling data augmentation. Thus, the dataset used to train the models with Word2Vec vectorization did not considered this type of augmentation. This correction allowed cover up to 84.28% of the vocabulary.

Finally, it is important to note that this optimization was built for each intent classification model, so each of the three vectorization methods was tested independently for every type of run. The results of these three techniques are presented in the following section.

3.3. INTENT CLASSIFICATION

Once the sentences were preprocessed and transformed to a numerical representation, the next step is to train a supervised algorithm that learns how to classify the intention that is attached with the sentence. For this purpose, two approaches were considered. The first is an Artificial Neural Network (ANN) that has been used widely in this subject (Moraes, Valiati, & Neto, 2013) (Chena, Liub, & Chiua, 2011) (Yang, 1999). The second approach is a Support Vector Machine (SVM) model that is commonly compared with a feed-forward neural networks since their structure is similar because both of them induce an output function which is expressed as a linear combination of simple functions (Romero & Toppo, 2007) (Joachims, 1998).

For both algorithms, a K-Fold cross validation technique was applied. Therefore, the database was divided into three blocks or subsamples where the data was shuffled before splitting into batches. Thus, one of the blocks (with 30% of the observation) is used to validate the model while the remaining k-1 blocks (with 70% of observation) are used for training purpose. This technique helps ensuring

[8] https://radimrehurek.com/gensim/index.html

consistency in the results and improves generalization. In the end of this process, the average of the subsamples is measured to determine the performance of the model being tested.

Furthermore, a heuristic hyperparameter optimization was performed using *Optuna*[9] which allows to automate the search for hyperparameters and can be used in any machine learning or deep learning framework. The metric of performance that was selected for both models is the accuracy. It was chosen since there are not special motivations to use other metric that considers for possible penalties for misclassify neither true positives nor false negatives.

Finally, one type of dataset variant was considered in addition to the augmented dataset (18022 sentences) presented in Section 3.2.1 and the original imbalanced dataset (7122 sentences). This variant corresponds to a small dataset with 120 sentences per intent that were randomly chosen, which produced a training set of 2160 utterances. This was done with the intention to assess the performance of the best models under three different data scenarios. (Larson, et al., 2019).

3.3.1. ANN Approach

The ANN model was implemented using *Keras*[10], that is a high-level Deep Learning API that runs on top of *TensorFlow 2*. The final ANN was selected considering the following remarks:

1. The entire hyperparameter optimization process for feature selection, architecture and optimizer was conducted for each of the three vectorizers (*i.e.* BoW, TF-IDF and Word2Vec) presented in Section 3.2.4. The number of iterations in the heuristic optimization was set to 50 iterations, each of them with a 3 K-Fold cross-validator as mention previously.
2. Just in the case of Word2Vec, an embedding layer was included in the first layer of the model. This layer has an input dimension equals to the vocabulary size that resulted from the tokenization process, and it has an output dimension equals to the embedding dimension of the pre-trained model (Cardellino, 2016), which is 300. The maximum length of the input sequences was set to 100, considering that the maximum number of words per sentence has an average of 25 words as shown in Section 3.2.3. Finally, this embedding layer was followed by a global max pooling operation that allowed to downsample (reduce size) the input representation by taking the maximum value over the second dimension.
3. The architecture was selected by optimizing several parameters in a 2 layers neural network. This includes the number of units on each hidden layer in decreasing order and by factor of 32 units (between 32 and 512 in the first hidden layer and between 32 and 224 in the second hidden layer), the type of activation function (*relu, sigmoid, softplus, tanh, selu* and *elu*) and finally the probability of dropout between layers that goes from 0 (no dropout) to 0.5 (dropout half of the time). The batch-size was fixed at 32 and the number of epochs to 20.
4. Two types of optimizers were tested (*Adam* and SGD), varying the learning rate for each of them in the range $\alpha \in \{0.0001, 0.001, 0.01, 0.1, 1, 0.003, 0.03, 0.3\}$. For the specific case of Adam optimizer, the exponential decay rate for the 1st moment $\beta_1 \in \{0.85, 0.9, 0.95\}$ and 2nd moment $\beta_2 \in \{0.99, 0.999, 0.9999\}$ were tested using the general recommendations by the authors (Kingma & Ba, 2015). In the case of SGD optimizer, only the momentum that

[9] https://optuna.org/

[10] https://keras.io/

accelerates gradient descent $\gamma \in \{0, 0.9\}$ was tested. Where 0 represents the vanilla gradient descent and 0.9 is the convention (Ruder, 2016).

5. A learning rate schedule was implemented where the initial learning rate was used for the first 10 epochs and an exponential decay of $\alpha = \alpha e^{-0.1}$ was introduced gradually after that. This adaptive learning rate help reducing the training time and could increase the performance. (Goodfellow, Bengio, & Courville, 2016).

By the end of this process, a total of 450 runs (50 iterations × 3 folds × 3 vectorizer methods) with 20 epochs each were performed using different combinations of parameters, and hyperparameters. The results are presented in the following section.

Lastly, to avoid overfitting, the following techniques were applied:

- An early-stopping mechanism was used to stop training when the accuracy in validation set has stopped improving after three epochs.
- A dropout layer with a certain optimizable probability was introduced between each hidden layer for regularization.
- The training set was shuffled before each epoch, so the ANN does not learn from the order of the training samples and instead only learns relevant patterns that help identifying the intention.

3.3.2. SVM Approach

The methodology followed to select the best SVM model was done similarly to ANN. Thus, the hyperparameter optimization was done using *Optuna*. However, as discussed during the literature review and since SVM are considered "out of the box" classifiers. Only two of the most essential hyperparameters were tuned, namely the kernel function and the regularization parameter.

The final model was selected considering the following:

1. The hyperparameter optimization process was performed for each of the three vectorizers (*i.e.,* BoW, TF-IDF and Word2Vec) presented in Section 3.2.4. The number of iterations in the heuristic optimization was set to 50 iterations, each of them with a 3 K-Fold cross-validator as mentioned previously.
2. The kernel function was optimized considering the following options:
    a. Polynomial kernel with degrees $d \in [2, 6]$.
    b. RBF with scale parameter of $\gamma \in \{1, 0.1, 0.01, 0.001, 0.0001\}$.
3. Lastly, the regularization parameter was optimized with values of $C \in \{0.5, 1, 10 \text{ or } 100\}$.

At the end of this process, a total of 450 runs (50 iterations × 3 folds × 3 vectorizer methods) were performed with different combinations of hyperparameters.

3.4. Entity Recognition

Once the intent classification model was selected, then it is necessary to extract the entities from the text. These entities represent important pieces of information that will help the chatbot to understand the context and give a more natural response based on the information that the user has provided.

Therefore, the rule-based approach and the statistical approach presented in the literature review were applied simultaneously to extract their best advantages.

The rule-based approach was used to extract specific patterns that have a standard form or a finite number of possibilities, such as emails, dates, barcodes of products or phone numbers in Colombian pattern. Depending on the task, the use of regular expressions or a list of specific tokens were used.

On the other hand, the statistical approach implemented to extract more complex entities such as proper names was the spaCy's v.2.0 pre-trained NER module in Spanish (*es_core_news_md*). It was trained using the *AnCora* corpus mentioned in the literature review and contains 20 thousand unique embeddings with 300 dimensions[11]. This model has a F-Score in test of 0.9 for the name entities component and allows to extract locations (LOC), organizations (ORG), and persons (PER). However, it will only be used to extract locations and persons.

The following table summarizes the 16 entities extracted along with their description and approximation used. Further details can be found in Annex 8.3:

Entity Code	Description	NER Approximation
sys-num-any	Extract any number in any form	Rule-based (from regular expression)
sys-units-name	Extract names of units	Rule-based (from tokens)
sys-units-symbol	Extract abbreviations of units	Rule-based (from tokens and regular expression)
sys-date	Extract any date in any form	Rule-based (from regular expression)
sys-person	Extract any person's full name	Spacy NER Model
sys-gender	Extract a gender	Rule-based (from tokens)
sys-location	Extract any location	Spacy NER Model
sys-address	Extract any physical address	Rule-based (from regular expression)
sys-email	Extract any email	Rule-based (from regular expression)
sys-barcode	Extract EAN-13 and EAN-8 barcodes	Rule-based (from regular expression)
sys-phone	Extract Colombian phone numbers	Rule-based (from regular expression)
sys-gel	Extract greater, equal, or less	Rule-based (from tokens)
sys-typeid	Extract any identification type	Rule-based (from tokens)
userItemName	Extract any user's item by name	Rule-based (from tokens)
userItemId	Extract any user's item by id	Rule-based (from tokens)

Table 6. Summary of entities extracted.

[11] *https://spacy.io/models/es*

4. RESULTS AND DISCUSSION

Once the heuristic hyperparameter optimization has been performed, the best model for each vectorizer was selected (further details in Annex 8.4). These models were run one last time using a 5 K-fold cross-validation in order to have a more reliable significance of the results. Additionally, as discussed during the methodology section, these best-performed models were tested using three dataset variations, Augmented, Small and Imbalanced datasets. Figure 13 shows a summarized representation in terms of average accuracy score and standard errors for each combination of model and vectorizer. A detailed table with these results is reported in Table 7.

Furthermore, in order to test whether the two classifiers have the same performance rate or not. A K-fold paired t-test was done with a 5% significance, following (Alpaydin, 2004, pp. 501-502). It was created by taking the difference between accuracy rates for both models during cross-validation and testing the null hypothesis of no difference between classifiers. The results for each method are presented in Table 8.

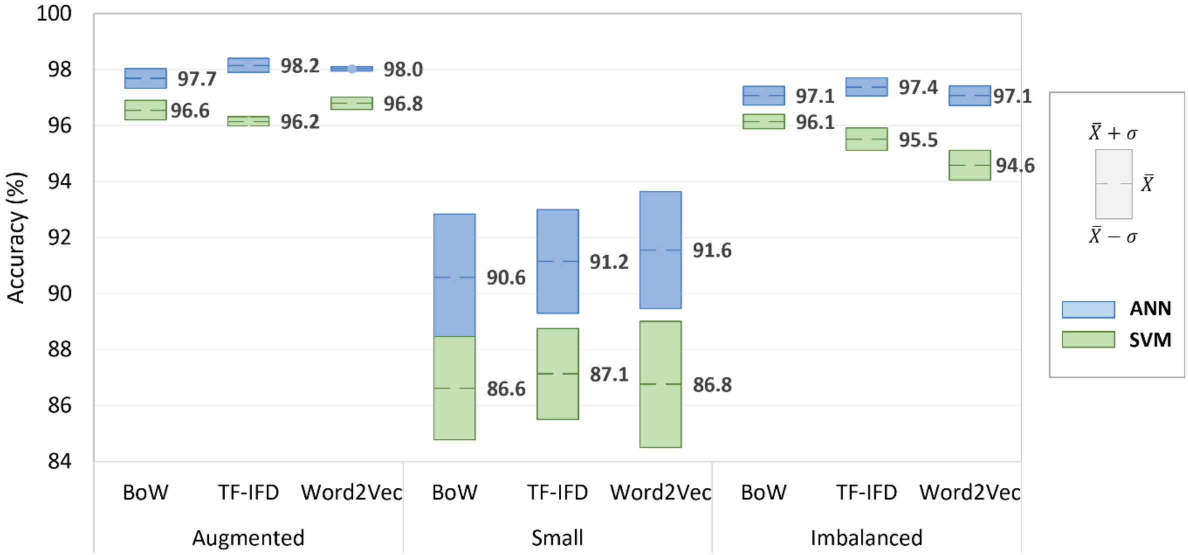


Figure 13. ANN and SVM test accuracy in a 5-folds CV

	ANN			SVM		
	Augmented	**Small**	**Imbalanced**	**Augmented**	**Small**	**Imbalanced**
BoW	97.70% ± 0.35	90.60% ± 2.25	97.08% ± 0.33	96.55% ± 0.35	86.62% ± 1.84	96.14% ± 0.26
TF-IFD	98.16% ± 0.25	91.16% ± 1.85	97.39% ± 0.32	96.15% ± 0.17	87.13% ± 1.62	95.51% ± 0.40
Word2Vec	98.04% ± 0.08	91.57% ± 2.09	97.08% ± 0.35	96.80% ± 0.22	86.76% ± 2.25	94.59% ± 0.53

Table 7. Average and standard deviations of test accuracy in a 5-fold cross validation

	Augmented t-Test	Small t-Test	Imbalanced t-Test
BoW	4.306	7.89	6.796
TF-IFD	16.974	8.036	11.81
Word2Vec	13.384	11.429	14.066

Significance at 5% ($t_{0.05,4} = 2.78$). If $ttest > t_{0.05,4}$ reject Ho: both models have the same performance.

Table 8. K-Fold paired t-Tests at 5% significance level for different dataset variants

Figures 14 and 15 show the multi-class Receiver Operating Characteristic (ROC) curves for two types of comparisons. On one hand, Figure 14, compares different dataset variants between classifiers and vectorizers. On the other hand, Figure 15 compares vectorizers and dataset variants within classifiers. The ROC curve plots two parameters, on the x-axis the False Positive Rate (FPR) and on the y-axis the True Positive Rate (TPR), showing the performance of a specific classification at different threshold levels. Moreover, these ROC curves were zoom-in to have a better picture of the differences between methods.

The multi-class ROC curve was done using the One-vs-rest strategy, where one classifier is fitted per class and tested against all other classes, transforming a multi-class classification problem into a binary classification, and gaining more interpretability and knowledge about the class. Finally, the macro-average Area Under the ROC Curve (AUC) was used to compute the metric independently for each class and taking the average, the higher the number the better the classifier to predict between classes.

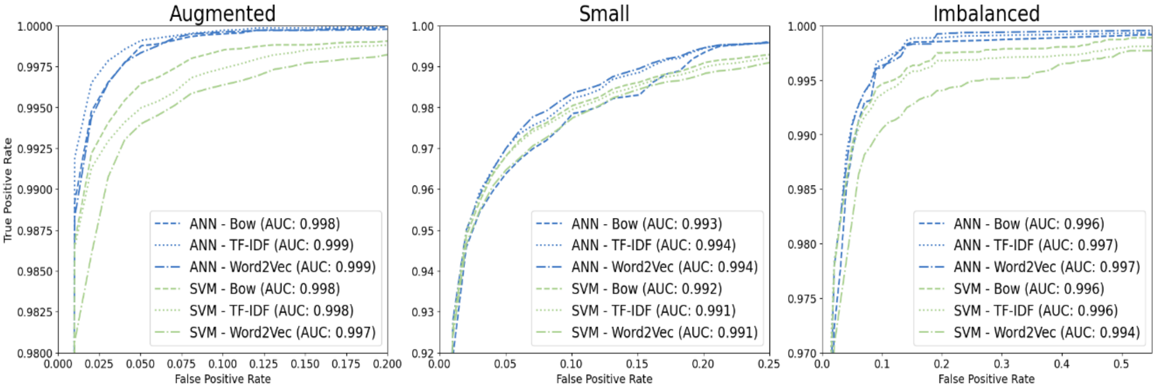


Figure 14. Comparison of One-vs-Rest ROC between-models by dataset variant.

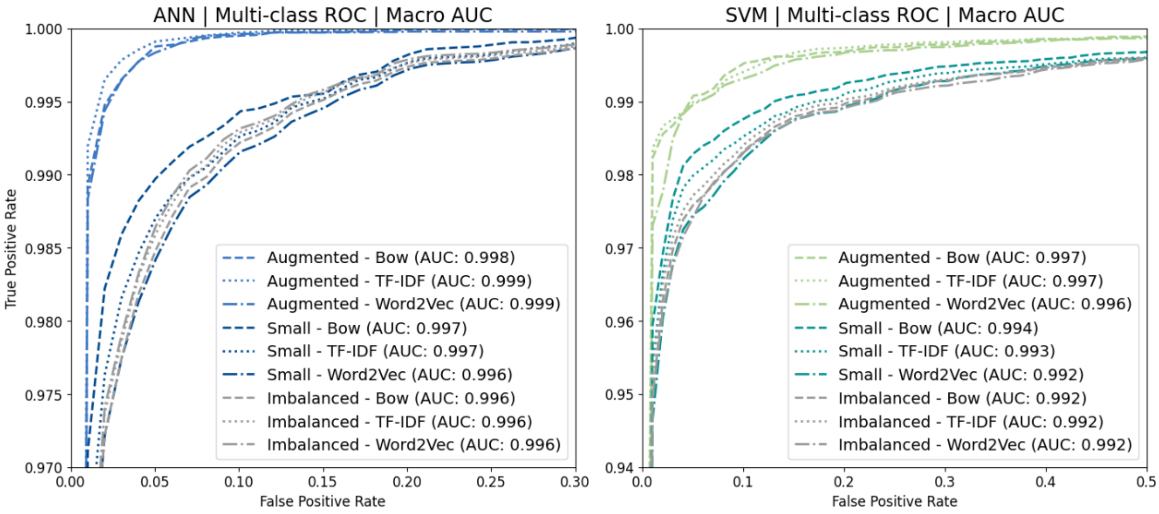


Figure 15. Comparison of One-vs-Rest ROC within-model using macro average AUC.

Considering the results obtained, it is possible to observe the following:

- Although both models achieved good levels of accuracy using the imbalanced dataset. Data augmentation increased performance for both classifiers with a higher effect on SVM using TF-IDF and word embeddings as vectorizers.
- As expected, the small dataset had the worst performance of all three, with a notorious drop of accuracy for both models. Given the volatility in the k-fold estimations, the performance is statistically equal between vectorizers. Furthermore, it was an unexpected outcome the fact that SVM underperformed ANN, given that it is a classifier that tends to respond better in this small-data scenario.
- Even though Word2Vec methodology was not the best vectorizer in all cases (just 2 out of 6 in Table 7), its performance is remarkable considering that it uses a pre-trained model that can be retrained in further studies and could incorporate the vocabulary and sentences used for training the intention in the embedding's representation.
- In terms of accuracy, tables 7 and 8 showed that ANN significantly (with a t-test at 5% of significance) achieved the best accuracy in all datasets and vectorizers with the higher performance using the augmented dataset and TF-IDF as a feature selection method.
- The results obtained by the ROC curves in Figure 14 showed that ANN presented the best results of both classifiers using the augmented and imbalanced datasets; however, in the small dataset, this advantage is less clear.
- Observing the within-model comparison in Figure 15, and aligned with the results using the accuracy metric, the augmented dataset had the best performance for both models. Small and imbalanced datasets had a less clear differentiation.

To sum up, using the accuracy metric and examining the multi-class ROC curves, the best model was reached using the augmented and balanced dataset with ANN as classifier and TF-IDF as vectorizer method. Compared to SVM, this model achieved significantly better performance (at 5% of significance) with an accuracy value of 98.2%. This result is aligned with previous works like (Moraes, Valiati, & Neto, 2013) and (Romero & Toppo, 2007) where ANN produced superior performance compared to SVM models, even using imbalanced or small datasets.

In order to evaluate the convergence to a satisfactory solution of the model that was selected, the neural network was trained one last time using the same configuration of TF-IDF vectorizer in the augmented dataset but without the cross-validation process. Figure 16 shows the loss and accuracy in training and validation that was run for 30 epochs using an early-stopping mechanism that ended the training process after 8 epochs since the model was not achieving a better solution in terms of validation loss. This technique combined with the usage of a dropout layer in the network, helped the model to converge and to avoid overfitting, as shown in Figure 16.

Thus, the model selected reached an accuracy in the test set of 98.28%, which is consistent with the results obtained during cross-validation. This model was chosen for production purposes in the implementation of the dialog system and was used as an intent classifier combined with the NER approach discussed in Section 3.4.

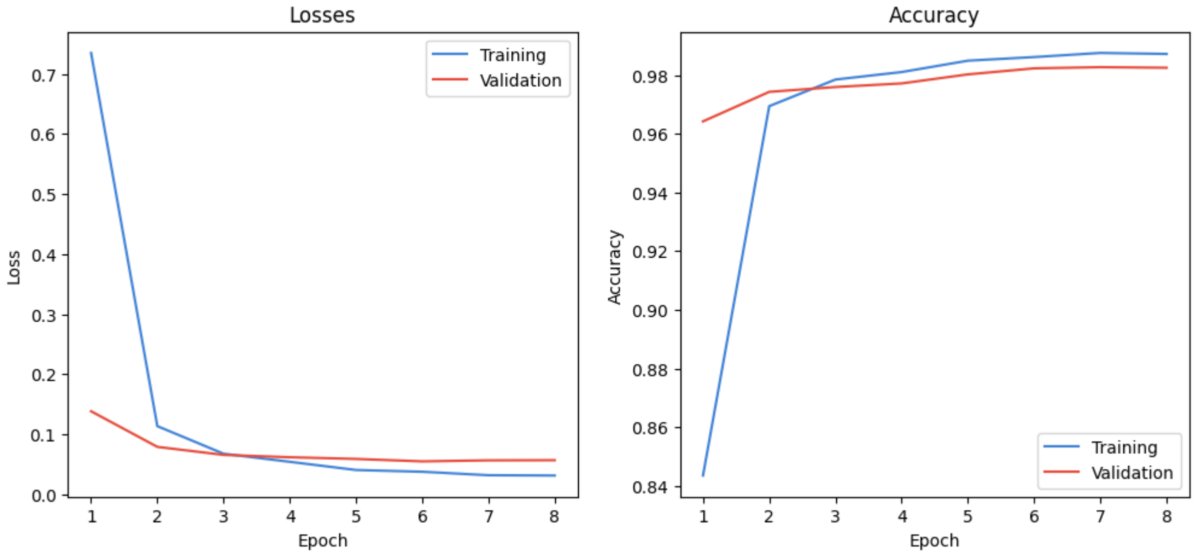


Figure 16. Losses and accuracy by epochs of best model.

Finally, in order to have a deeper knowledge of how the model performs for each class, Figure 17 shows the multi-class ROC separated by intents. Its construction is similar to the one done previously for Figures 14 and 15 but with the difference that each class is shown independently. Thus, it is possible to observe that even though the model achieved good outcomes across all classes, the worst performance (those farthest below the Macro-averaged ROC curve) was achieved in the classes associated with greetings, sales strategy, thanks, and bot compliment. In contrast, the majority of the classes that are related to the specific task for this chatbot performed well. This represents an important point since although the proper identification of task-related queries by the system is more relevant than queries that fall outside of the scope, those out-of-scope intents should not be discarded and could potentially improve the user experience with the dialog system (Larson, et al., 2019).

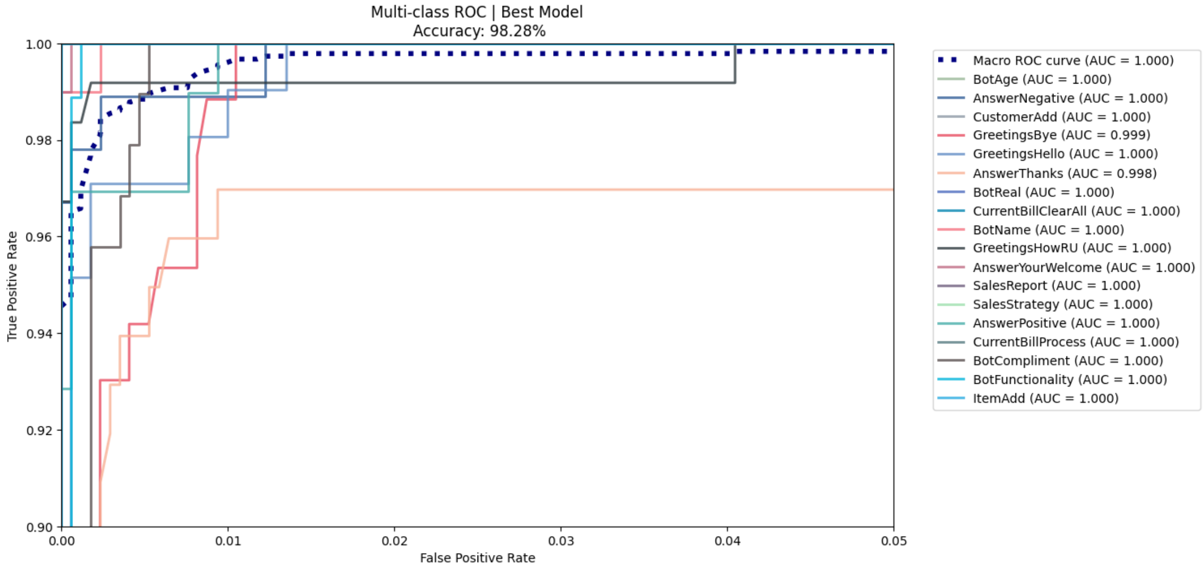


Figure 17. ROC curves of best model selected by classes.

Chatbot in real-word scenario

Once the best model was selected, the entire chatbot system was put to the test in a real scenario. For this purpose, an experimental chatbot was built using Command Line Interface (CLI) where the user can start a conversation and provide a feedback to each answer. Thus, a short survey with 5 active customers was performed just to qualify intent classification and NER. In this case, each user had the task to provide a score from 1 to 3 points for each answer in both intention and entities, where 1 corresponds to a good-quality answer, 2 corresponds to a regular-quality answer, and 3 to a poor-quality answer. It is important to mention that this chatbot was created just for testing purposes, as a prototype in a controlled environment, and the responses are not considered as final.

At the end of this experiment, the quality of each survey was measured by taking the average of the scores provided by the customers. Table 9 summarizes these results in terms of intents, further details were reported in Annex 8.6.

Group	Intent Expected	Average	Average of group
Specific Task	CurrentBillClearAll	1.0	1.3
	CurrentBillProcess	1.0	
	CustomerAdd	1.0	
	ItemAdd	1.0	
	SalesReport	1.8	
	SalesStrategy	2.0	
Bot Natural Talk (BNT)	AnswerThanks	1.0	1.3
	BotCompliment	3.0	
	BotFunctionality	1.0	
	BotName	1.0	
	BotReal	1.0	
	GreetingsBye	1.0	
	GreetingsHello	1.0	
	GreetingsHowRU	1.0	
Default	Default Answer	2.5	2.5
OVERALL		**1.4**	

Table 9. Results of user's feedback

Considering the results obtained by the survey, the following was observed:

1. The chatbot produced favorable results in terms of intent classification with an overall score of 1.4. The main drawback in terms of specific-task intents was detected in *SalesStrategy*, whereas in the BNT group the intent *BotCompliment* produced the poorest performance.
2. In terms of NER classification, the chatbot achieved an overall score of 1.3, which indicates good results. However, we noticed that further tuning is required for entities predicted by the SpaCy's pre-trained model used for detecting persons and locations, since in some cases the model predicted false positives, especially in cases where capitalize letters were used.
3. Lastly, the fact that the chatbot had poor results in terms of the default answer means that it needs to be considered in further detail. In this case, the addition of new observations in an out-of-scope class could be implemented (Larson, et al., 2019); additionally, a reduction in the threshold value considered as the minimum level of uncertainty in the resulted probability could also be contemplated for the final system.

5. CONCLUSIONS

In this project, the framework for a specific-domain dialog system was built as part of an internship program with the start-up 2Luca. This conversational agent is based on Artificial Neural Networks and NER models, which can be tuned and enhanced using an improved corpus based on future real conversations with final users.

The results obtained in this project showed that MLP neural networks are one of the best options for text classification problems. These results are consistent with other works like (Moraes, Valiati, & Neto, 2013) where ANN showed better performance compared to SVM in a binary classification problem or (Larson, et al., 2019) in which MLP outperformed several models, including platforms such as Rasa NLU[12] or Dialogflow[13] that are built for the development of task-oriented agents.

Furthermore, three dataset variants were tested including an augmented dataset that was built using different techniques found in literature, the results showed that this augmentation process produced an improvement in terms of accuracy level and ROC curves.

Additionally, multiple vectorizer methods were tested including a Bag-of-Words, a TF-IDF and a word embeddings vector representation. The best results were obtained using the TF-IDF vectorization method. However, the role of Word2Vec should not be discarded in further analysis, as it would potentially improve the results since word embeddings would capture particular needs if the model was trained using a specific-task corpus.

Finally, the implementation was tested using a small-talk survey made by 5 customers. The results showed positive feedback from the final users in terms of both intent classification and NER. Moreover, good feedback was extracted from these conversations, since it highlighted the importance to incorporate more out-of-scope training data in order to improve the user's experience.

[12] https://rasa.com/

[13] https://dialogflow.cloud.google.com/

6. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

Nowadays, the main challenge for any intelligent chatbot is to reach the same conversational capability as human conversation. To achieve this goal, the system must understand and convey complex concepts, sentences, and semantics using NLP techniques. The key aspect of such systems is the communication of information in an efficient manner, which allows the conversation to be very rich both, in content and context.

To improve the performance in the intent's classification made by the bot, the proposed model could be modified using a Bi-directional Transformer Model (BERT), where the use of encoders and decoders could dramatically outperform the current results as shown in (Larson, et al., 2019). However, to use this model it is recommended first to increase naturally the number of observations in the dataset, potentially by storing future conversations of the current model and then tagging the intents of the final users.

Another recommendation to be considered in future implementations is the Name Entity Recognition model. Here, the use of online text annotation resources such as *doccano*[14] or *Prodigy*[15] could be useful to create a customized dataset that helps to detect more precisely the entities that are included in the text.

One of the main limitations encountered during the elaboration of this project was the lack of NLP resources available for the Spanish language. There is still a long way to go in this area. For instance:

1. The development of new Word2Vec pre-trained models with further Corpus sources in addition to the one presented by (Cardellino, 2016).
2. Additionally, the need for more NLP tools in addition to *SpaCy* package that include more rigorous work on Spanish lemmatization.

Software infrastructure

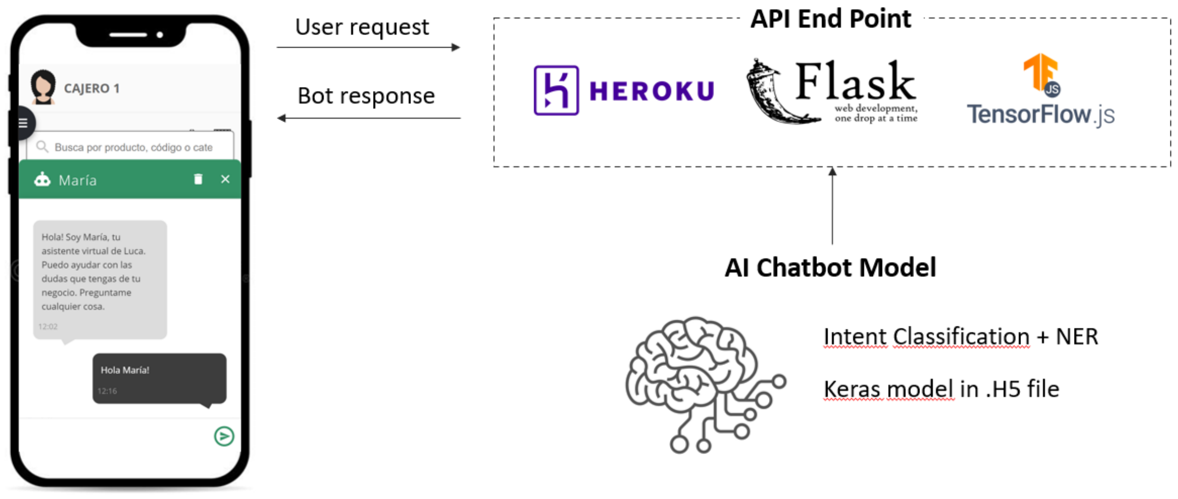
Although the final implementation of the dialog system software is beyond the scope of this project, a software ecosystem was proposed to implement the framework developed here. For this purpose, an Application Programming Interface (API) could interact with the backend of 2Luca and serve as endpoint for all user inquiries send to the chatbot.

Figure 18 illustrates this process, in which each user's request (each sentence in the conversation) is sent to the API endpoint, where the model processes the input and passes it through the neural network to detect intentions; lastly, the system detects entities using the proposed model, which combines statistical and rule-based approaches. The ANN model can be exported as an .H5 file which can be easily used within the API for prediction. Once the best response from the AI model is obtained, it is sent back to the user.

[14] *https://doccano.herokuapp.com/*

[15] *https://prodi.gy/*

Finally, the API can be easily mounted in the cloud on a Platform as a Service (PaaS) solution such as Heroku using Flask as a web development framework, which allows to easily create a secure API solution and will help to define the endpoints.



Hola! Soy María, tu asistente virtual de Luca. Puedo ayudar con las dudas que tengas de tu negocio. Preguntame cualquier cosa.

12:02

Hola María!

12:16

Figure 18. Software implementation proposed.

7. BIBLIOGRAPHY

Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., . . . Le, Q. V. (2020). Towards a Human-like Open-Domain Chatbot. *arXiv*, 38.

Alpaydin, E. (2004). *Introduction to machine learning.* Cambridge: The MIT Press.

Bikel, D., Miller, S., Schwartz, R., & Weischedel, R. (1998). Nymble: a High-Performance Learning Name-finder.

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa: Open Source Language Understanding and Dialogue Managemen. *CoRR*, 1-9.

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *COLT '92*.

Cardellino, C. (2016, March). Retrieved from Spanish Billion Words Corpus and Embeddings: https://crscardellino.ar/SBWCE/

Carpenter, R. (1997). *Jabberwacky*. Retrieved from http://www.jabberwacky.com/

Chena, L.-S., Liub, C.-H., & Chiua, H.-J. (2011). A neural network based approach for sentiment classification in the blogosphere. *Journal of Informetrics*, 313–322.

Christopher, M., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55-60.

Colby, K. M., Weber, S., & Hilf, F. D. (1971). Artificial paranoia. *Artificial Intelligence 2*, 1–25.

D. Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

Deng, L., & Liu, Y. (2018). *Deep Learning in Natural Language Processing.* Springer.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating Non-local Information into Information Extraction Systems by Gibbs. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, 363-370.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning (Adaptive Computation and Machine Learning series).* MIT Press.

Huang, M., & Zhu, X. (2020). Challenges in Building Intelligent Open-domain Dialog Systems. *arXiv*, 1-33.

IBM Cloud Education. (2020, 07 02). *IBM Education*. Retrieved from IBM: https://www.ibm.com/cloud/learn/natural-language-processing

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R.* New York: Springer Texts in Statistics.

Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. *Proceedings of the 10th European conference on machine learning*, 137–142.

Kafle, K., Yousefhussien, M., & Kanan, C. (2017). Data Augmentation for Visual Question Answering. *Proceedings of The 10th International Natural Language Generation conference*, 198-202.

Kingma, D., & Ba, J. (2015). Adam: A Method For Stochastic Optimization. *ICLR*.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the 18th International Conference on Machine Learning*, 282-289.

Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., . . . Mars, J. (2019). An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. *CoRR*. Retrieved from http://arxiv.org/abs/1909.02027

Marivate, V., & Sefara, T. (2020). Improving Short Text Classification Through Global Augmentation Methods. *Machine Learning and Knowledge Extraction*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations. *In Proceedings of Workshop at ICLR*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *arXiv*, 1-9.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, 1-9.

Moraes, R., Valiati, J. F., & Neto, W. P. (2013). Document-level sentiment classification: An empirical comparison between SVM and ANN. *Expert Systems with Applications*, 621-633.

Nuruzzaman, M., & Hussain, O. K. (2018). A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks. *International Conference on e-Business Engineering*, 9.

Romero, E., & Toppo, D. (2007). Comparing support vector machines and feedforward neural networks with similar hidden-layer weights. *IEEE Transactions on Neural Networks*, 959–963.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*.

Sennrich, R., Birch, A., & Haddow, B. (2016). Improving Neural Machine Translation Models with Monolingual Data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Spacy. (2020). *Spacy*. Retrieved from https://spacy.io/usage/rule-based-matching

Sutton, C., & McCallum, A. (2012). An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*, 267-373.

Wacholder, N., Ravin, Y., & Choi, M. (1997). Disambiguation of Proper Names in Text. *Proceedings of 5th Conference on Applied Natural Language Processing*, 1-7.

Wallace, R. S. (2009). The anatomy of ALICE. In Parsing the Turing Test. *Springer*, 181–210.

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

Yang Wang, W., & Yang, D. (2015). That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets. *Association for Computational Linguistics*, 2557–2563.

Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 69–90.

Zhang, X., Zhao, J., & LeCun, Y. (2016). Character-level Convolutional Networks for Text Classification. 1-9.

8. ANNEXES

8.1. Spanish Spelling Errors

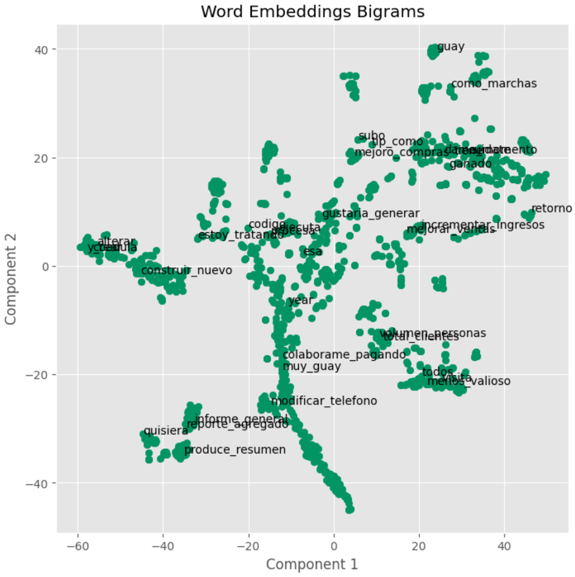
sa	ca
se	ce
si	ci
so	co
su	cu
cion	sion
ción	sion
ha	a
he	e
hi	i
ho	o
hu	u

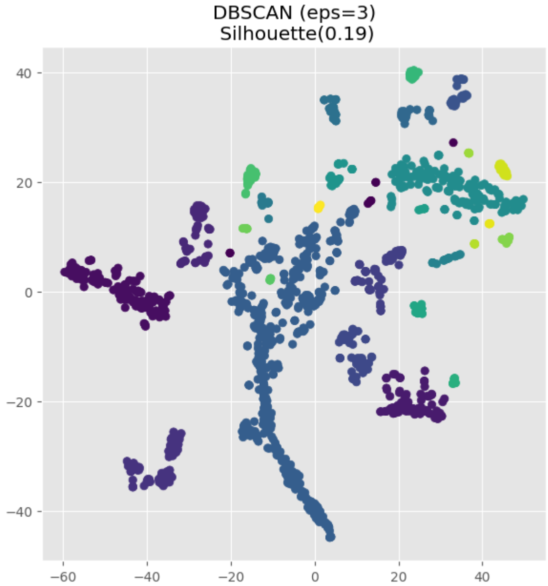
ch	sh
rr	rrr
rr	r
va	ba
ve	be
vi	bi
vo	bo
vu	bu
amp	anp
emp	enp
imp	inp
omp	onp

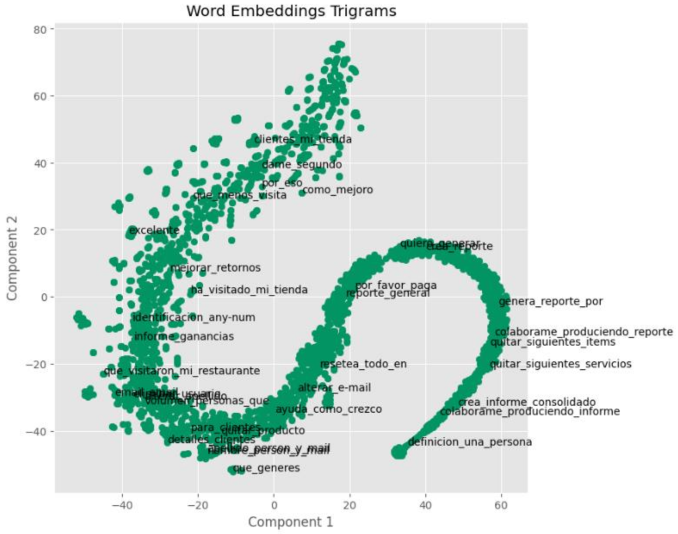
za	sa
ze	se
zi	si
zo	so
zu	su
yi	lli
yo	llo
yu	llu
lla	ya
lle	ye
ga	ja
ge	je

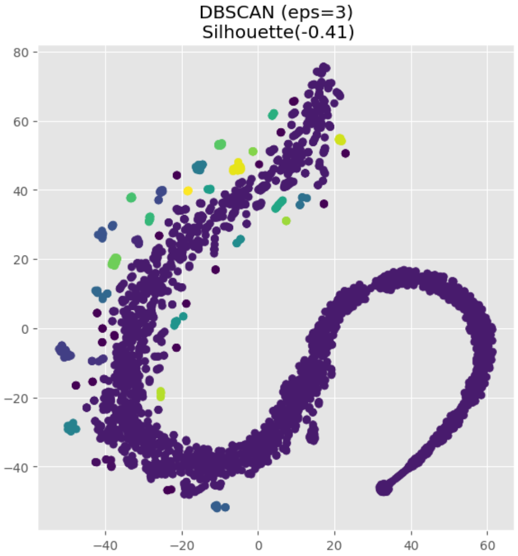
gi	ji
go	jo
gu	ju
ay	ai
oy	oi
uy	ui
que	q
wa	gua
gui	gi
gue	ge
sc	s
por	x

8.2. WORD2VEC REPRESENTATION









8.3. NER, FURTHER DETAILS

<table>
<tr><th>Entity Code</th><th>NER Approximation</th><th>Details</th></tr>
<tr><td>sys-num-any</td><td>Rule-based (from regexp)</td><td>"[-]?[0-9]*\\.?[0-9]+",
"[-]?\\d{1,3}(\\,\\d{3})+"</td></tr>
<tr><td>sys-units-name</td><td>Rule-based (from tokens)</td><td>"unidades", "unidad", "botellas", "botella", "vasos", "vaso", "barriles", "barril", "cajas", "caja", "sobres", "sobre", "piezas", "pieza", "retazos", "retazo", "equipos", "equipo", "cubos", "cubo", "micra", "micras", "milimetro", "milimetros", "centimetro", "centimetros", "metro", "metros", "kilometro", "kilometros","pulgada", "pulgadas","pie", "pies", "yardas","yarda","milla","millas","pie cuadrado", "pies cuadrados", "yarda cuadrada", "yardas cuadradas", "millas cuadradas", "milla cuadrada", "metro cuadrado", "metros cuadrados", "hectarea", "hectareas", "kilometro cuadrado", "kilometros cuadrados", "centimetro cúbico", "centimetros cúbicos","metro cúbico", "metros cúbicos", "pie cúbico", "pies cubicos","yarda cúbica", "yardas cúbica", "microlitro", "microlitros", "mililitro", "mililitros", "mili litros", "litro", "litros", "microgramo", "mirco gramos", "microgramos","miligramo", "miligramos", "mili gramos", "mili gramo", "gramo", "gramos", "kilo", "kilos", "kilogramo", "kilo gramo", "kilogramos", "kilo gramos","tonelada metrica", "toneladas metricas","onza", "onzas","libra", "libras","libras por pulgada cuadrada", "libra por pulgada cuadrada","partes por mil","partes por millon","segundo", "segundos", "minuto", "minutos", "horas", "hora", "kilovatio-hora", "kilovatio", "kilovatio hora"</td></tr>
<tr><td>sys-units-symbol</td><td>Rule-based (from tokens)</td><td>"k", "kg", "gr", "grm", "g", "km", "mtr", "m", "mm", "dm", "hec", "m2", "m3", "ft"</td></tr>
<tr><td>sys-date</td><td>Rule-based (from regexp)</td><td>"((\\d{1,2})(\\\\|\\/|-|;|\\.|\\s)(\\d{1,2})(\\\\|\\/|-|;|\\.|\\s)([12]\\d{3}))",
"([12]\\d{3})(\\\\|\\/|-|;|\\.|\\s)(\\d{1,2})(\\\\|\\/|-|;|\\.|\\s)(\\d{1,2})",
"(\\d?\\d)\\s*(de)?\\s*(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre|ene|feb|mar|abr|may|jun|jul|ago|sep|oct|nov|dic)\\s*(del|de)?\\s*(año)?\\s*(\\d{2,4})?",
"\\b(lunes|martes|miercoles|jueves|viernes|sabado|domingo|lun|mar|mie|jue|vie|sab|dom)\\,?\\s?(\\d?\\d)\\s*(de)?\\s*(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre|ene|feb|mar|abr|may|jun|jul|ago|sep|oct|nov|dic)\\s*(del|de)?\\s*(año)?\\s*(\\d{2,4})?\\b",
"\\b(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre|ene|feb|mar|abr|may|jun|jul|ago|sep|oct|nov|dic)\\s*(del|de)?\\s*(año)?\\s*([12]\\d{3}|\\d{1,2})+\\b"</td></tr>
<tr><td>sys-person</td><td>Spacy NER Model</td><td>-</td></tr>
<tr><td>sys-gender</td><td>Rule-based (from tokens)</td><td>"masculino", "femenino", "M", "F", "hombres", "mujeres", "hombre", "mujer", "trans", "transexual", "transgenero", "FTM", "MTF", "androgino", "androgynous", "neutrois", "genderqueer", "agenero", "bigenero", "pangenero"</td></tr>
<tr><td>sys-country</td><td>Spacy NER Model</td><td>-</td></tr>
<tr><td>sys-zipcode</td><td>Rule-based (from regexp)</td><td>"\\b\\d{6}\\b"</td></tr>
<tr><td>sys-address</td><td>Rule-based (from regexp)</td><td>-</td></tr>
<tr><td>sys-email</td><td>Rule-based (from regexp)</td><td>“([a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+\\.[a-zA-Z0-9_-]+)”</td></tr>
</table>

sys-barcode	Rule-based (from regexp)	"\\b\\d{1}\\s*\\d{6}\\s*\\d{6}\\b", "\\b\\d{1}\\-*\\d{6}\\-*\\d{6}\\b", "\\b\\d{4}\\s*\\d{4}\\b", "\\b\\d{4}\\-*\\d{4}\\b"
sys-phone	Rule-based (from regexp)	"\\b\\d{3}\\s*\\d{4}\\b", "\\+\\d{2}\\s*\\d{3}\\s*\\d{4}\\b", "\\(\\d{3}\\)\\s*\\d{3}\\s*\\d{2}\\s*\\d{2}\\b", "\\+\\d{2}\\s*\\(\\d{3}\\)\\s*\\d{3}\\s*\\d{2}\\s*\\d{2}\\b", "\\b\\d{3}\\s*\\d{3}\\s*\\d{2}\\s*\\d{2}\\b", "\\+\\d{2}\\s*\\d{3}\\s*\\d{3}\\s*\\d{2}\\s*\\d{2}\\b"
sys-gel	Rule-based (from tokens)	“mayor que”, “mayor a”, “mayor de”, “menor a”, “menor que”, “menor de”, “mayor o igual”, “menor o igual”, ”igual a” , ”igual que”, ”igual” , “mayores a”, “mayores que”, “mayores de”, “menores a”, “menores que”, “menores de”
idtype	Rule-based (from tokens)	"cedula de ciudadania", "cedula", "cc", "c.c", "tarjeta de identidad", "cedula de extranjeria", "pasaporte", "numero de identidad", "identidad", "NIT", "numero de identificacion tributaria", "numero de identificacion", "número de identificacion fiscal"
userItemName	Rule-based (from tokens)	Tokens are extracted from the database
userItemId	Rule-based (from tokens)	Tokens are extracted from the database

8.4. RESULTS OF HYPERPARAMETERS OPTIMIZATION

The following table summarizes the specification of the best models that were selected after 50 trails of hyperparameter optimization using Optuna. These models were obtained independently for each classifier and each vectorizer method using the augmented dataset and the preprocessing techniques mentioned in Section 3.2.2.

Classifier	Vectorizer	Specification	
ANN	***BoW***	*Analyzers:*	Characters and words.
		N-gram:	Trigrams
		Min and Max DF:	1 and 0.9
		Optimizer:	Adam
		Learning Rate:	0.01
		Hidden Layer 1:	352 units with ReLU activation function
		Dropout 1:	0.4 probability
		Hidden Layer 2:	160 units with *Softplus* activation function
		Dropout 2:	0.1 probability
	TF-IFD	*Analyzers:*	Characters and words.
		N-gram:	Bigrams and Trigrams
		Min and Max DF:	3 and 0.9
		Optimizer:	Adam
		Learning Rate:	0.01
		Hidden Layer 1:	480 units with ReLU activation function
		Dropout 1:	0.5 probability
		Hidden Layer 2:	224 units with tanh activation function
		Dropout 2:	0.5 probability
	Word2Vec	*Analyzers:*	Words
		N-gram:	Unigrams
		Optimizer:	Adam
		Learning Rate:	0.001
		Hidden Layer 1:	320 units with ReLU activation function
		Dropout 1:	0.3 probability
		Hidden Layer 2:	96 units with ReLU activation function
		Dropout 2:	0.1 probability
SVM	***BoW***	*Analyzers:*	*Characters and words*
		N-gram:	*Bigrams and Trigrams*
		Min and Max DF:	*3 and 0.85*
		C:	*100*
		Kernel:	*RBF*
		Gamma:	*Scale*
	TF-IFD	*Analyzers:*	*Characters and words*
		N-gram:	*Bigrams and Trigrams*
		Min and Max DF:	*3 and 0.85*
		C:	*100*
		Kernel:	*RBF*
		Gamma:	*Scale*
	Word2Vec	*Analyzers:*	*Words*
		N-gram:	*Unigrams*
		C:	*1*
		Kernel:	*RBF*
		Gamma:	*Scale*

8.5. K-Folds Results

			Test Accuracy K-Fold						
			Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Std**
ANN	**Augmented**	***BoW***	97.78	98.03	97.42	98.09	97.20	97.70	0.35
		TF-IFD	98.36	98.25	97.97	98.45	97.78	98.16	0.25
		Word2Vec	98.05	98.02	97.99	98.20	97.96	98.04	0.08
	Small	***BoW***	87.96	87.73	92.59	92.36	92.36	90.60	2.25
		TF-IFD	88.66	89.35	92.82	91.67	93.29	91.16	1.85
		Word2Vec	89.35	88.89	92.36	93.06	94.21	91.57	2.09
	Imbalanced	***BoW***	96.91	96.56	97.12	97.26	97.54	97.08	0.33
		TF-IFD	97.05	97.12	97.47	97.96	97.33	97.39	0.32
		Word2Vec	97.27	96.56	97.27	96.79	97.51	97.08	0.35
SVM	**Augmented**	***BoW***	96.51	95.89	96.84	96.78	96.73	96.55	0.35
		TF-IFD	96.28	96.00	95.89	96.28	96.28	96.15	0.17
		Word2Vec	96.76	96.94	97.03	96.88	96.40	96.80	0.22
	Small	***BoW***	85.65	83.80	89.35	87.04	87.27	86.62	1.84
		TF-IFD	85.19	85.65	89.58	88.19	87.04	87.13	1.62
		Word2Vec	84.95	83.33	88.19	89.35	87.96	86.76	2.25
	Imbalanced	***BoW***	95.65	96.14	96.35	96.21	96.35	96.14	0.26
		TF-IFD	94.74	95.58	95.86	95.65	95.72	95.51	0.40
		Word2Vec	94.19	94.18	94.54	94.42	95.61	94.59	0.53

8.6. RESULTS OF CHATBOT SURVEY FROM USERS

User	Intent Score		NER Score	
	Count	Mean	Count	Mean
1	7	1.0	7	1.3
2	9	1.4	9	1.3
3	13	1.6	13	1.2
4	7	1.3	7	1.7
5	10	1.4	10	1.0
OVERALL	**1.4**		**1.3**	