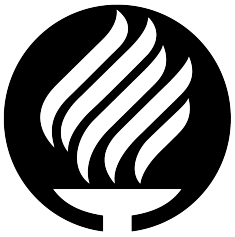
Monterrey Institute of Technology and Higher Education
Campus Queretaro
Computing and mechatronics department



Tecnológico de Monterrey

Implementation of NLP model for concept detection in omnichannel communication platform messages

by

Roberto Carlos Guzmán Cortés

Integrative Project for the Development of Business Solutions

Computer Systems Engineering

Adviser: Dr. Pedro Leopoldo Martínez Quintal

Santiago de Querétaro, Querétaro, Mexico
11/22/2022

" *Essentially, all models are wrong, but some are useful.* "

George Box

" *AI is likely to be either the best or worst thing to happen to humanity.* "

Stephen Hawking

Acknowledgements

First of all, I would like to thank Dr. Pedro Leopoldo Martínez Quintal for all the support and shared knowledge during the development of this project. This couldn't have been possible without him.

Secondly, I'd also like to thank all the professors that taught me a wide variety of Computer Science and math topics throughout different courses, you're amazing at what you do!

Finally, I would like to thank my parents for all the support, everything I have achieved so far as a student has been possible because of you. You've been always with me and I deeply appreciate that. I'll be eternally grateful.

Abstract

Social media has led to an increment of unstructured data which currently represents 80% of the digital universe, reaching this last a size of 163 ZB by 2025. Experts have estimated that organizations loss billions of dollars every year because of the difficulty to analyze unstructured data. This paper selects the Natural Language Processing (NLP) model that according to the state-of-the-art suits the best to the analysis of customer messages in an omnichannel communication platform: The Transformer architecture. Since their introduction in 2017 transformers outperformed the previous state-of-the-art Deep Learning (DL) architectures for NLP tasks, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). The implemented pre-trained model presents two completely different scopes: a) A local solution coded in a Jupyter notebook, and b) A cloud function deployed in Google Cloud Platform (GCP) designed to be consumed by an external tool. Both prototypes reached an accuracy of 43%, proving that despite Transformers overtake the classical NLP techniques in omnichannel platforms we need more investigation to tackle problems that Spanish analysis involve.

Contents

Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
DL	**D**eep **L**earning
DT	**D**ecision **T**ree
GCP	**G**oogle **C**loud **P**latform
GPU	**G**raphic **P**rocessor **U**nit
GRU	**G**ated **R**ecurrent **U**nit
IDC	**I**nternational **D**ata **C**orporation
LDA	**L**atent **D**irichlet **A**llocation
LR	**L**inear **R**egression
LSTM	**L**ong **S**hort-**T**erm **M**emory
ML	**M**achine **L**earning
NB	**N**aive **B**ayes
NER	**N**amed **E**ntity **R**ecognition
NLG	**N**atural **L**anguage **G**eneration
NLP	**N**atural **L**anguage **P**rocessing
NLU	**N**atural **L**anguage **U**nderstanding
NOSQL	**N**ot **O**nly **S**QL
POS	**P**arts **O**f **S**peech
RF	**R**andom **F**orests
SQL	**S**tructured **Q**uery **L**anguage
SVM	**S**upport **V**ector **M**achine
TPU	**T**ensor **P**rocessor **U**nit

Chapter 1

Introduction

"Data is the new oil." Clive Humby

Unstructured data is the most common type of data present in the digital universe reaching a proportion of 80% and growing 3 times faster than its counterpart structured data. Only 10% of the unstructured data has being analyzed by organizations, but at least 95% of businesses need solutions to manage this type of data, and a few of them intend to do it frequently. In addition, International Data Corporation (IDC) reported that 37% of the unstructured data can benefit businesses if adequately treated and analyzed, leading to $430 billion dollars in productivity gains [1][2].

On the other hand, e-commerce activity increased by 55% during the COVID-19 pandemic reaching $1.7 trillion dollars in USA, where some phenomena such as grocery shopping with minimal discounts highlighted the necessity of delivery speed and convenience [3].

Even though analyzing unstructured data represents an increase in incomes not only for e-commerce companies but all the companies in general, an initial investment is required for the research and development of specialized tools, specially those powered by AI.

This paper will be about concept detection in Spanish text messages using a pre-trained NLP model combined with another data cleaning techniques. The structure will be as follows:

Chapter 2 will touch unstructured data and NLP models and how they work together to perform concept detection tasks. Token classification techniques will be discussed as well as advances

in concept detection using ML techniques. Similar solutions for concept detection will be mentioned, specially one that performs intent identification using classical ML techniques.

Chapter 3 will deep dive on what unstructured data is, the challenges involved on analyzing this type of data, why an initial investment is required to analyze it and why it is worth. In addition, it will explain why this research is focused on a specific kind of unstructured data.

Chapter 4 is about a detailed description of the implemented NLP model, the tasks it performed, data cleaning techniques, etc. Relevant code on the implementation of the pre-trained NLP model and other functions mentioned in this chapter can be found in Appendix A.

Chapter 5 will be about the evaluation method to be used to determine the success of the NLP implementation. This includes the description of the different experiments to be done.

Chapter 6 will include the tests described in the previous chapter. The experiments were done firsts in a local environment to test if all the technologies involved in the solution matched according to the needs of this project and later on a Cloud Computing environment to get the final results. This will include all the obtained graphs and tables gotten from all the experiments. The interpretation of the results gotten is also included here.

Finally, chapter 7 contains the conclusions gotten after finishing the investigation and the future work to improve this project and probably get it ready to be integrated into a commercial software solution focused on Chat Applications for messaging.

1.1 Problem statement

Having compared the most advanced work done in the state of the art (Chapter 2), it is clear that there is a problem that has not been tackled yet. Here I explain why said problem has not been solved at all and why nowadays it is relevant to be solved.

As it was mentioned previously, unstructured data is hard to be analyzed, specially for those organizations that cannot afford an initial investment in ML technologies. Once they have dispensed with getting profit from unstructured data analysis, they have started to lose money and will be in a serious disadvantage against those competitors that have been investing in this type of AI projects [1].

1.2 Objectives

The main objective of this project is to implement a pre-trained NLP model for concept detection in omnichannel communication platform messages. This objective can be divided into the following secondary objectives:

1. Identify top 3 AI algorithms for concept detection in message apps.
2. Implementation of pre-trained NLP model for local environment.
3. Integration and testing of implemented pre-trained NLP model to be consumed by external tool.
4. Final thesis delivery which will include all relevant metrics, graphs, conclusions and future work.

1.3 Hypothesis

An NLP solution that involves a DL architecture for concept detection in omnichannel communication messages will get an accuracy greater or equal than those focused on classical NLP techniques.

Chapter 2

State of the Art

This chapter describes unstructured data and Natural Language Processing (NLP) models. These concepts are necessary to understand how Deep Learning (DL) models can be used for concept detection and how Artificial Intelligence (AI) can help to organizations to get profit from unstructured data. Then, an analysis of some of the most important work on the area will be made. With this analysis, we will be able to see the areas in which there is still work to be done to detect concepts in omnichannel platform messages and what contributions can be made on the subject.

2.1 Prior knowledge

An omnichannel communication platform is a especial kind of application that allows customers to contact retailers through multiple communication channels such as email, social media, instant messages in social networks, short messages and phone calls even. This concept was first introduced to the marketing world in 2010 and in September, according to a report from IDC Retail Insights, retailers utilizing multichannel strategies in 2010 saw a 15-35% increase in average transaction size, along with a 5-10% increase in loyalty customers' profitability. IDC cited the growing ecommerce market as the key reason retailers needed to implement omnichannel strategies [4].

Omnichannel communication platforms contributes to increase the generation of unstructured data, which increases 3 times faster than its counterpart structured data. Also, unstructured data is not only hard to analyze but also only 10% of it has been analyzed. Both represent a

loss of 430 billion dollars annually for most of the organizations [1].

2.2 Relevant work

A considerable amount of investigation work has been done making use of classical NLP techniques for concept detection reaching high accuracy percentages in some cases. However, not much work has been done using DL techniques for Spanish chat messages.

In the area of concept detection using classical NLP processing techniques the following was found:

- **Support Vector Machine (SVM):** Studies show that SVM is one of the most famous and common used algorithms used for text classification due to its good performance. SVM is a non-probabilistic binary linear classification algorithm which performs by plotting the training data in multi-dimensional space. Then SVM categories the classes with a hyperplane. The algorithm will add a new dimension if the classes can not be separated linearly in multi-dimensional space to separate the classes. This process will continue until a training data can be categorized into two different classes [5].
- **Naive Bayes (NB):** It's been proved that NB are a group of different classification algorithms based on the Bayes Theorem. All NB algorithms have the same assumption, i.e., each pair of features being classified is independent of others. The NB classification algorithms are widely used for information retrieval and many text classification tasks [5].
- **Logistic Regression (LR):** It's referred that this is a statistical model and is one of the earliest techniques used for classification. LR predicts probabilities rather than classes. It works well in the case of categorical results but fails in the case of continuous results [5].
- **Decision Tree (DT):** It's referred to be one of the earliest classification models for text and data mining and is employed successfully in different areas for classification task. The main intuition behind this idea was to create tree-based attributes for data points, but the major question is which feature could be a parent and which will be a child's level. DT classifier design contains a root, decision and leaf nodes which denote dataset, carry out the computation and performs classification respectively. The advantages of DT classifier

are; the amount of hyper-parameters which require tuning is nearly zero, easy to describe, can be understood easily by its visualizations [5].

- **Random Forests (RF):** Results have demonstrated concentrates on methods to compare the results of several trained models in line to give better classifier and performance than a single model. This classifier is quick to train for textual data but slow in giving predictions when trained. Performs good with both categorical and continuous variables, can automatically handle missing values, robust to outliers and less affected by noise whereas training a vast number of trees can be computationally expensive, require more training time and utilize much memory [5].

It is obvious that classical techniques reached good results before the DL era, and were used in production environments before modern neural networks.

When it comes to the use of DL techniques for text classification, it's proved that convolutional neural networks (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU) and Transformers show higher accuracy than classical NLP techniques. Furthermore, tasks that use deep learning for NLP have been classified into 2 general groups: Natural Language Understanding (NLU) and Natural Language Generation (NLG). This indicates that the use of CNN, LSTM, GRU and Transformers architectures could produce models with higher accuracy because text analysis is required and token classification comprehend tasks that fall into Named Entity Recognition (NER) and Parts of speech (POS) [5].

Hugo D. Rebelo developed a model using classical NLP techniques combined with probabilistic models such as Latent Dirichlet Allocation (LDA) and clustering algorithms like as K-Means to differentiate messages according to specific topics in a Customer Support system from a Brazilian electric energy company. The model was trained using chatbot messages written in Portuguese corresponding to the category of "wrong classified messages". This system was able to allocate 62% of the failed queries in the previous mentioned category. While there's still the need to train to test with more text messages, results suggests that deep neural networks are a useful tool for analyzing text messages [6].

The main advantage of this model is that it can classify text messages according to a reduced number of tags. The problem is that in the real world we've got several tags rather than only 7. For this reason, there's no certainty that this model could achieve a high accuracy but with Spanish messages datasets coming from omnichannel applications. Furthermore, the model was

trained using a small portuguese dataset with no more than 800 instances. When it comes to the choice of using Transformers, while this model outperforms GRU and LSTM architectures in small datasets, the biggest problem could be the grammar rules violated by some users while writing messages and the idioms or regionalisms that could confuse the neural network. A pre-trained model could require less time to be implemented compared to one that requires training from scratch. So, a pre-trained model could achieve similar accuracy [6].

To perform concept detection, a Transformer model developed Google in 2018 known as BERT Multilingual, obtains an accuracy of 97.10% for token classification tasks like Parts of speech in multiple languages. This models has been trained with more than 100 languages including English, Portuguese, Russian, Spanish, German, French and Chinese even, but training this Transformer will require hours in a computer using a cheap Graphic Processor Unit (GPU) or 30 minutes running on a Cloud Tensor Processor Unit (TPU). Said model promises a good performance when implementing a software solution reaching higher accuracy levels for Spanish messages [7].

Spanish focused models have been developed as well, for example Cañete, J and Chaperon, Gabriel and Fuentes, Rodrigo and Ho, Jou-Hui and Kang, Hojin and Pérez, Jorge from Universidad de Chile created in 2020 an implementation based on BERT Multilingual called BETO which reached an accuracy of 98.97% for Parts of speech (POS) tasks, outperforming the accuracy reached by BERT Multilingual which stands in 97.10% [8].

2.3 Advances in Omnichannel platforms using DL

The most interesting work in this case has been made by Cañete, J and Chaperon, Gabriel and Fuentes, Rodrigo and Ho, Jou-Hui and Kang, Hojin and Pérez, Jorge from Universidad de Chile, the model they have developed has an outstanding accuracy for Spanish datasets.

Universidad de Chile BETO model is great for Parts of speech (POS) tasks [8]. On the other hand, the Multilingual BERT devloped by Google supports more than 100 languages including Spanish where it showed an excellent performance as well [7]. However, none of them have been used to predict providers in Spanish messages from omnichannel communication platforms.

2.4 Conclusion

What's most important about this chapter is that among all the analyzed projects and models, none of them fully satisfy the need of predicting providers by analyzing omnichannel communication platform messages. In order to do that, a model that suggests providers by analyzing text messages needs to be developed. This model have to achieve an accuracy level similar to those found on the studied models.

Chapter 3

Why concept detection in unstructured data is so hard

Unstructured data is everywhere, Facebook, Messenger, WhatsApp, Twitter, Telegram, Instagram, TikTok, YouTube, Amazon Shopping, Uber, Rappi, LinkedIn, Github, Zoom, Dropbox, Snapchat, Discord, Microsoft Teams, Tinder, Leadsales, Google Drive, Reddit, etc. In addition, as mentioned in Chapter 1, unstructured data represents 80% of the digital universe and grows 3 times faster than its counterpart structured data, reaching proportions of 212 zettabytes (ZB) vs 53 ZB by 2025 [1].

Meanwhile structured data — typically categorized as quantitative data — is highly organized, easily decipherable and can be queried by using a structured query language (SQL), unstructured data — typically categorized as qualitative data — does not have a predefined data model, it is best managed in non-relational (NoSQL) databases. Some examples of unstructured data are: chat messages, emojis, stickers, images, audio notes, videos, URLs, songs, text files, HTML documents, etc [9].

Nowadays, organizations could gain $430 billion dollars annually by analyzing at least 37% of its own unstructured data adequately. However, most of them do not have specialized tools and could not afford an initial investment in AI powered tools or data science systems [2], and that's the problem we will tackle in the next section and incoming chapters.

3.1 Concept detection in omnichannel platform messages

Unstructured data can be found in several file formats: .mp3, .jpg, .txt, .png, .gif, etc. All of them are present in omnichannel communication platform messages, so using classical NLP techniques to get profit from this kind of data by performing sentiment analysis, concept detection, named entity recognition (NER) or parts of speech (POS) classification turns into a high complexity problem, needing to transform audio into text, images into emotion classes, text files into word frequency reports, videos into tagged data, etc.

Thus, the solution will be focused on analyzing text messages only, i.e., the implementation will perform concept detection in Spanish text messages to suggest fruit and vegetable providers from a dataset, expecting to reach an accuracy between 60-70% like the intent recognition model mentioned in Chapter 2. The following chapters will show a detailed description of the implemented solution, the results gotten from every experiment, the reasons of such results and the conclusions.

Chapter 4

Solution

To solve the problem of concept detection in omnichannel communication platforms in Spanish messages, a pre-trained NLP model was implemented combined with another data cleaning tools. The pre-trained NLP model follows a Transformer architecture, it was chosen because it requires no previous training compared to one training from scratch, like so, we can start doing predictions as soon as possible. Furthermore, this type of neural network since its release in 2017 has outperformed the previous state-of-the-art neural neural network architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) [10]. The pre-trained model was developed by José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang and Jorge Pérez from Universidad de Chile and trained on a dataset of 32,001 Spanish words, implemented in python using the PyTorch framework and hosted in the Hugging Face site as a pipeline [11]. The pre-trained model receives the name of "BETO", it's BERT-based, has 12 self-attention layers with 16 attention-heads each and uses 1024 as hidden size. In total the BETO has 110M parameters [8]. The code that calls the BETO pipeline can be found in Appendix A under the title "Spanish BERT (BETO) + POS".

The most important task to be solved by BETO consist of token classification: given an input string, the words (tokens) will be classified according to a specific task such as Named Entity Recognition (NER) or Parts of Speech (POS). Since NER requires a very specialized dataset where concepts receive specific tags like John: Name, Doe: Last Name, etc. POS requires an input text only to classify words according to the grammar sentence structure, for example, "I need an apple" will be classified as I: Pronoun, need: Verb, an: Preposition and apple: Noun Countable. Thus, that's why we opted for BETO to perform POS tasks in order to detect fruits

and vegetables. The following diagram describes what has been mentioned in this paragraph:

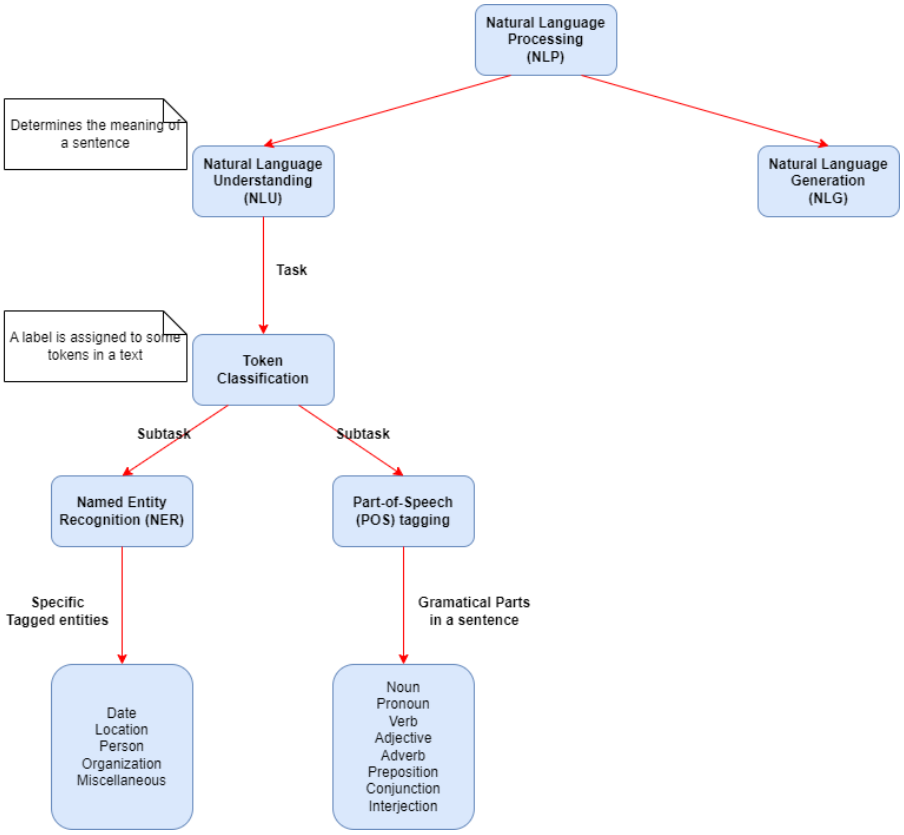


FIGURE 4.1: Token classification tasks

The code to install all the required dependencies can also be found in Appendix A under the title "Dependencies". Actually all of the code found in Appendix A - except Lemmatization (Cloud Function) and Cloud Function Requirements - just need to be copied into a jupyter notebook code snippet and be executed to verify the results.

The dataset used to measure the solution accuracy was built from scratch, it consisted of the results gotten from a survey in a Google Form that contained 2 questions only:

1. *Selecciona una fruta o verdura que más compres en el supermercado.* (Select the fruit or vegetable that you have bought most of the times when you went to the supermarket)
2. *¿De qué forma le pedirías a alguien que te lo consiga en el supermercado? (Ejemplo: ¿Podrías comprarme un kilo de plátano por favor? o ¡Tráeme unas manzanas del super!).* (Write how would you ask to someone to get it for you (Example: Could you buy a kilogram of bananas for me please? or Bring me some apples from the supermarket!)).

A total of 100 people older than 18 years old answered the survey, they chose between 20 different fruits/vegetables and answered the corresponding open-ended question mentioned above. The results for the first question behave in the following way:

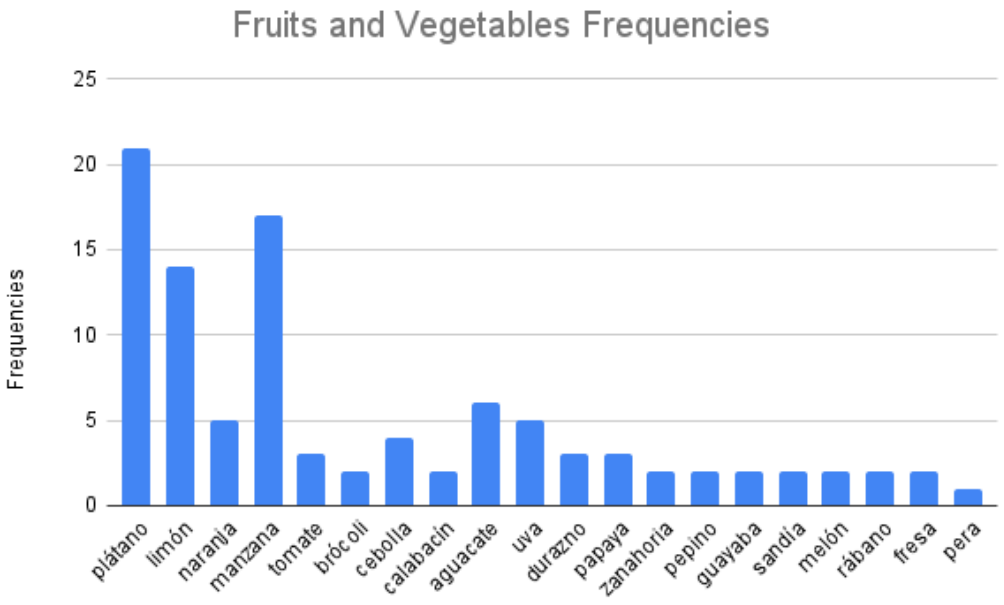


FIGURE 4.2: Fruits and vegetables frequencies

Where *plátano* (banana) was the most common fruit bought in the supermarket with a frequency of 21 and *pera* (pear) was the least common with a frequency of 1. On the other hand, the results for the second question are a mixed of words, some of them are well-written and another ones contained grammatical errors, emojis, and so on. Reflecting that in fact, intent has a higher priority than grammar at the time of writing messages. Some examples are:

- *Trae unas 5 manzanas.* (Bring 5 apples)
- *Me traes una cebolla?.* (Could you bring me an onion?)
- *Compra 1kilo de limón* . (Buy 1 kilogram of lemons)

The first sentence has been written correct according to Spanish grammar rules, but the second one has an starting question mark missing and the third sentence has a blank space missing between number 1 and the spanish word *kilo*. The code to load this dataset is can be found in Appendix A under the title "Messages dataset".

Another important part of this project system the "Providers dataset" which can also be found in Appendix A under the title "Providers Knowledge base". This dataset originally has been

saved as a local python dictionary because we didn't want to share original data from real providers in order to avoid some kind of legal issues and manipulate data that follows a clean format, i.e., standardized product names (no accents, no plurals), standardized email addresses, normalized telephone numbers, etc. The providers dataset contains the following attributes:

- **Provider_name:** provider's name
- **Keyword:** product name such as *manzana* (apple), *limon* (lemon), *naranja* (orange), etc.
- **Category:** product category. For academic purposes we have included only 2 different categories: fruit or vegetable.
- **Email:** provider's email.
- **Telephone:** provider's telephone number.
- **Score:** punctuation assigned in a random way, it'll be used to prioritize the providers to be shown when a prediction is done. Providers will be displayed in a descending way.

Another problem to be tackled was one related to the words that have been written in plural and transform them into their root form (singular) in order to perform a successful search into our providers dataset. So, we had to execute a linguistic process known as lemmatization. To achieve such goal we have used a popular library for NLP tasks known as Spacy. This library contains several NLP models for different languages such as English, German, French, Chinese, and Spanish of course! So in order to progress with such process we downloaded the model "es_core_news_sm" from the official Spacy website: https://spacy.io/models/es. Thus, the implemented system consists mainly of 5 parts:

1. **User Input:** text message in Spanish language. It can be provided in 3 different ways:
   (a) **Default text:** will use a default string provided by us.
      - *Necesito comprar un kilogramo de naranjas, manzanas, tomate, limón, plátanos, cebollas y uvas para mañana.*
   (b) **Random text:** will choose a string from the Spanish dataset.
   (c) **Write your own input:** users will be free to write a sentence which must consists of the way how they would ask to someone to get them some fruit o vegetables from the supermarket.

2. **BETO (POS):** Transformer pipeline that will output the Parts of speech (POS) gotten from the user input.

3. **Lemmatization:** Spacy model that will convert every Noun Countable gotten from BETO to its root form.

4. **Providers Query:** will search every lemmatized noun countable from Spacy into the providers dataframe.

5. **Suggested Providers:** dataframe with all the suggested providers, displayed in descending order according to their score attribute.

The following diagram shows what has been explained lines above:

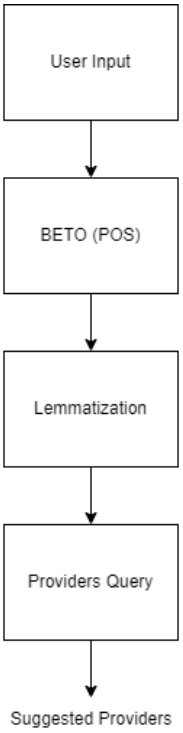


FIGURE 4.3: System workflow overview

Chapter 5

Evaluation Method

To evaluate if the solution works, different tests will be made. These tests will be measured in a similar way the related model [6] mentioned on Chapter 2 were evaluated: calculating the model's successful rate. In order to say the model is able to make predictions correctly, it will need to achieve greater or equal accuracy than that of the related model. That accuracy is 62%. So, an accuracy greater than 62% is expected.

First, the solution will be implemented in a local environment using a Jupyter Notebook in a Google Colab, coded in python using pandas, spacy, unidecode, hugging face transformers, and the dependencies previously mentioned in Chapter 4 which can be installed using the code found in Appendix A under the title "Dependencies".

If the previous implementation works as expected (regarding of the accuracy gotten), then a solution that involves a Cloud Function from Google Cloud Function (GCP) will be implemented in order to simulate how this solution can be consumed as an external service.

5.1 Local environment implementation

This solution is focused on testing if the following software components work together to fulfil the creation of our first prototype according to the investigation scope:

1. **Google Colab:** will be our Jupyter Notebook editor and execution environment. Worked with the released version corresponding to 2022/9/16.

2. **Python:** programming language that will be used to code the solution. The python version used is 3.7.15.
3. **Pandas:** dataset manipulation tool included in Python as a library. The pandas version used is 1.3.5.
4. **Hugging Face Transformers:** external library provided by Hugging Face company used for implementation that require transformers (neural network architecture). The version used for this project is 4.24.0.
5. **Spacy:** popular library for NLP tasks. The Spacy version used for this project is 3.4.3.

The key aspects to be measured are: **average successful rate** (how many times the solution has predicted something vs how many has returned empty) and **average message accuracy** (how many products have been detected). In addition, another relevant metrics will be measured such as model size and number of words used for training. Here is how these metrics will be gotten:

- **Average successful rate:** proportion of how many times the solution has predicted providers divided by how many times the solution has been executed.
- **Average message accuracy:** proportion of the number of Lemmatized nouns from BETO divided by the number of Lemmatized Words from the original sentence. Just in case this last parameter were equal to zero, this metric will result in 0 (to prevent NaN results).
- **Model size:** this property will be taken from the BETO model called from the hugging face transformers module.
- **Words for training:** this property will be taken from the BETO model called from the hugging face transformers module.

5.2 Cloud environment implementation

This solution is focused on demonstrate how some parts of this project can be consumed as an external service:

1. **Google Colab:** will be our Jupyter Notebook editor and execution environment. Worked with the released version corresponding to 2022/9/16.

2. **Python:** programming language that will be used to code the solution. The python version used is 3.7.15.

3. **Pandas:** dataset manipulation tool included in Python as a library. The pandas version used is 1.3.5.

4. **Hugging Face Transformers:** external library provided by Hugging Face company used for implementation that require transformers (neural network architecture). The version used for this project is 4.24.0.

5. **Spacy:** popular library for NLP tasks. The Spacy version used for this project is 3.4.3.

6. **Cloud Function:** event-driven serverless compute platform, i.e., it'll consume GCP resources only when called. Uses Python 3.7 under the 1st generation standard.

All the steps described can be summarized in the following pipeline diagram that represents the final solution workflow:

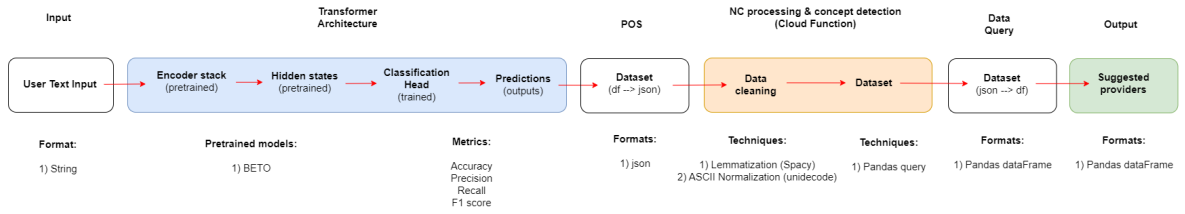


FIGURE 5.1: System final flow

All metrics mentioned in the previous section will be evaluated and one extra will be included: **latency**, this is because the response time plays an important factor when using an external service. Here is an explanation of how this metric will be gotten:

- **Latency:** a total of 10 runs will be done (because it costs GCP credits) and then the average and standard deviation will be calculated.

Finally, the successful rate gotten from both prototypes will be compared against the expected accuracy mentioned in Chapter 2.

Chapter 6

Results and Interpretation

Both prototypes gave unexpected results. However, there are a couple of thing to note: the dataset size could affect due to we had 100 instances only and all of the simulated text messages contained 1 fruit/vegetable only. Having a larger dataset with more fruit/vegetables written into every message could have been really beneficial for both prototypes.

The first prototype implemented in a local environment got an **Average Successful Rate** of 43.0% and a **Average Message Accuracy** of 43.0% as well, this is because concept detection worked in 43/100 of the messages in the dataset. The reason behind such results is because some messages contain grammatical errors such as missing accents, misspelling words, and so on. On the other hand, some fruits or vegetables were not detected by BETO because this transformers separated some words into 2 parts. The following table will illustrate some examples:

Concept	Classification
cebolla	*cebol la*
platano	*plata no*
limon	*limo n*
papaya	*papa ya*
plátano	*pláta no*
plátanos	*plátanos*
pera	*per a*

Table 6.1: Countable Nouns found by BETO POS task.

According to Table 6.1 we noticed that using the Spanish grammar rules correctly is not enough to perform concept detection, because some well-written Spanish words such as *papaya* are not

detected by BETO POS algorithm. In addition, some words written correctly in a singular form weren't detected as well, for example *plátano* wasn't detected, but *plátanos* was. So, we can conclude that not only following the Spanish grammar rules play an important factor but also the texts used for the training process when designing a Transformer. This last is so relevant because BETO was trained using a big Spanish Corpus which consisted of wikis mainly [8]. The following bar graph illustrates the behaviour previously explained:

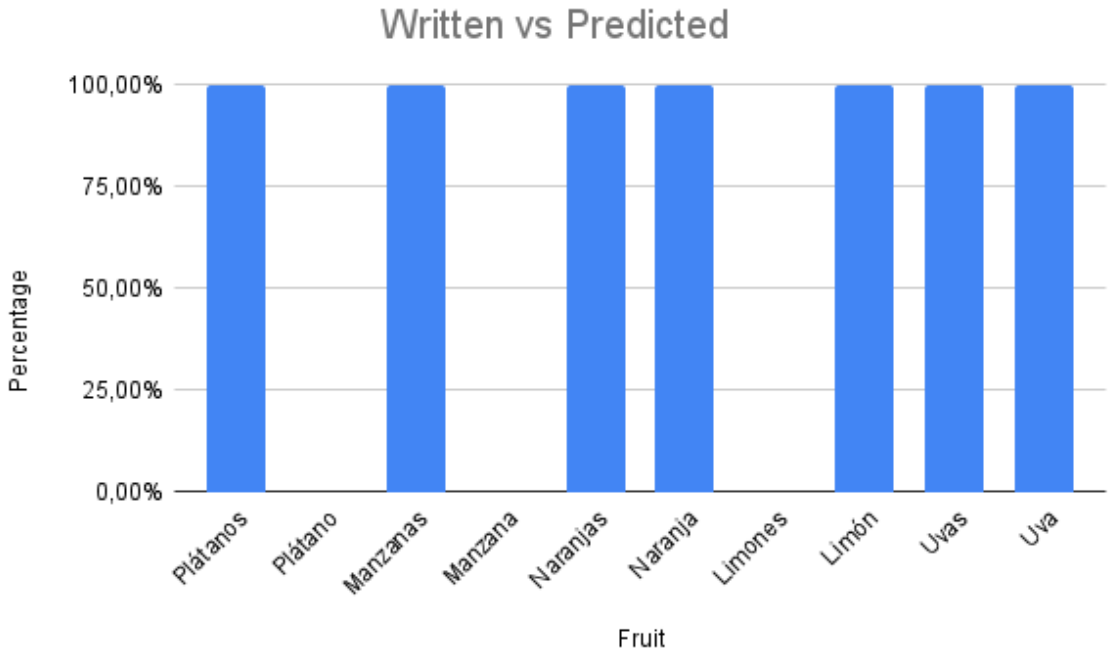


FIGURE 6.1: Detection of concepts written

Another relevant metrics for this first prototype were those aimed at BETO:

- **Model size:** 391 MB
- **Vocabulary training size:** 31002 tokens

The second prototype which used a Cloud Function from GCP had the same results in all the previously mentioned metrics. Only one parameter regarding to the response time was measured, we got the following results:

- **Average latency:** 1.89 s
- **latency standard deviation:** 0.2799 s

This means that everytime we call the Cloud Function we will get an average response time of

$$1.89 \pm 0.2799s$$

where the best case would be 1.6101 s and the worst case would be 2.1699 s.

The results of both prototypes demonstrate how complex the Spanish language is when trying to solve tasks that involve token classification. We couldn't improve the percentage mentioned in Chapter 2, but we trust that a third prototype could reach such percentage if some improvements are taken into account, also this research could contribute to future works in order to avoid some mistakes and get higher accuracy values.

Chapter 7

Conclusions and Future Work

The most important takeaway from this investigation is that concept detection in omnichannel Spanish messages cannot be tackled by using parts of speech (POS) only, there are several factors that have a big influence in such task, like grammar and the training dataset used for pre-trained models. In addition, in case of improve such implementation we trust that it can be turned into a profitable software solution for companies that have omnichannel communication platforms or chatbots even.

It would be also interesting to verify how **Average Successful Accuracy** and **Average Message Accuracy** could improve with the help of some external tools like the Fasttext library from Meta (previously known as Facebook). This amazing library provides some wonderful algorithms such as semantic similarity and cosine of similarity [5], both could help significantly our solution in those scenarios where nothing has been predicted. Finally, the creation of a dataset using Mexican Spanish messages could be helpful for BETO during the training process in order to recognize common words used by us and avoiding unknown behaviour when processing simple words as saw in Table 6.1 in Chapter 6.

Appendix A

Relevant Code

A.1 Dependencies

```
# G O O G L E   C O L A B
from google.colab import drive
drive.mount('/content/drive')

# P A N D A S
import pandas as pd
import random

# T R A N S F O R M E R S
# https://huggingface.co/docs/transformers/installation
%pip install transformers
%pip install unidecode

# B E T O
# https://huggingface.co/mrm8488/bert-spanish-cased-finetuned-pos
from transformers import pipeline

# S P A C Y
%pip install -U pip setuptools wheel
%pip install -U spacy
!python -m spacy download es_core_news_sm

# https://pharos.sh/python-para-pnl-tokenizacion-derivacion-y-lematizacion-con-la-biblioteca-spacy/
import spacy
```

A.2 Providers Knowledge base

```
d = {
    "Provider_name" : ["citrex", "alameda", "dorada", "citrex", "alameda",
                       "citrex", "alameda", "citrex", "alameda", "citrex",
                       "citrex", "citrex", "citrex", "chiapanex", "real",
                       "real", "real", "real", "real", "real",
                       "citrex", "dorada", "dorada", "dorada", "dorada"],

    "Keyword"       : ["manzana", "manzana", "manzana", "aguacate", "aguacate",
                      "durazno", "durazno", "tomate", "tomate", "calabacin",
                      "pera", "naranja", "limon", "platano", "uva",
                       "melon", "sandia", "pepino", "fresa", "papaya",
                       "guayaba", "zanahoria", "cebolla", "brocoli", "rabano"],

    "Category"      : ["fruta", "fruta", "fruta", "fruta", "fruta",
                      "fruta","fruta", "verdura", "verdura", "verdura",
                      "fruta", "fruta", "fruta", "fruta", "fruta",
                       "fruta", "fruta", "fruta", "fruta", "fruta",
                       "fruta", "verdura", "verdura", "verdura", "verdura"],

    "Email"         : ["manzana@gmail.com", "manzana@hotmail.com", "manzana@yahoo.com",
                       "aguacate@gmail.com", "aguacate@hotmail.com", "durazno@gmail.com",
                       "durazno@hotmail.com", "tomate@gmail.com", "tomate@hotmail.com",
                       "calabacin@gmail.com", "pera@gmail.com", "naranja@gmail.com",
                       "limon@gmail.com", "platano@gmail.com", "uva@gmail.com",
                       "melon@gmail.com", "sandia@gmail.com", "pepino@gmail.com",
                       "fresa@gmail.com", "papaya@gmail.com", "guayaba@gmail.com",
                       "zanahoria@yahoo.com", "cebolla@yahoo.com", "brocoli@yahoo.com",
                       "rabano@yahoo.com"],

    "Telephone"     : ["555-555-55-50", "555-555-55-51", "555-555-55-52",
                       "555-555-55-50", "555-555-55-51", "555-555-55-50",
                       "555-555-55-51", "555-555-55-50", "555-555-55-51",
                       "555-555-55-50", "555-555-55-50", "555-555-55-50",
                       "555-555-55-50", "555-555-55-53", "555-555-55-54",
                       "555-555-55-54", "555-555-55-54", "555-555-55-54",
                       "555-555-55-54", "555-555-55-54", "555-555-55-52",
                       "555-555-55-52", "555-555-55-52", "555-555-55-52", "55-555-55-52"],

    "Score"         : [1, 3, 7, 1, 3, 7, 6, 8, 11, 0,
                      2, 24, 36, 7, 93, 9, 6, 33, 4, 25,
                       0, 19, 12, 15, 18]
}

df_prov = pd.DataFrame(data=d)
```

A.3 Messages dataset

```
# M E S S A G E S   D A T A S E T
# https://colab.research.google.com/notebooks/snippets/sheets.ipynb
from google.colab import auth
auth.authenticate_user()

import gspread
from google.auth import default
creds, _ = default()

gc = gspread.authorize(creds)

worksheet = gc.open('FruitMessage_POS').sheet1

# get_all_values gives a list of rows.
rows = worksheet.get_all_values()

# Convert to a DataFrame and render.
df_messages = pd.DataFrame.from_records(rows, columns=["Fruit_Vegetable", "Message"])
```

A.4 Spanish BERT (BETO) + POS

```
# B E T O
def getNounCountablesFromBeto(user_input):
  # https://iq.opengenus.org/bert-cased-vs-bert-uncased/
  # cased models hava been trained using capital letters and accents
  nlp_pos = pipeline(
      "ner",
      model="mrm8488/bert-spanish-cased-finetuned-pos",
      tokenizer=(
          'mrm8488/bert-spanish-cased-finetuned-pos',
          {"use_fast": False}
  ))

  text = user_input

  pos_list = nlp_pos(text)
  df_pos = pd.DataFrame(pos_list)
  query = df_pos.query("entity=='NC' and word != '[UNK]'")
  keywords = query["word"]

  return keywords
```

A.5 Cloud Function Requirements

```
# Requirements to be installed before cloud function execution
unidecode
pandas
spacy
es_core_news_sm @ https://github.com/explosion/spacy-models/releases/download/es_core_news_sm-3.4.0/es_
```

A.6 Lemmatization (Cloud Function)

```
import pandas as pd
import spacy
import unidecode
import json

# P R O V I D E R S   K N O W L E D G E   B A S E
def getProvidersDictionary():
    d = {
        "Provider_name" : ["citrex", "alameda", "dorada", "citrex", "alameda",
                           "citrex", "alameda", "citrex", "alameda", "citrex",
                           "citrex", "citrex", "citrex", "chiapanex", "real",
                           "real", "real", "real", "real", "real",
                           "citrex", "dorada", "dorada", "dorada", "dorada"],

        "Keyword"       : ["manzana", "manzana", "manzana", "aguacate", "aguacate",
                           "durazno", "durazno", "tomate", "tomate", "calabacin",
                           "pera", "naranja", "limon", "platano", "uva",
                           "melon", "sandia", "pepino", "fresa", "papaya",
                           "guayaba", "zanahoria", "cebolla", "brocoli", "rabano"],

        "Category"      : ["fruta", "fruta", "fruta", "fruta", "fruta",
                           "fruta","fruta", "verdura", "verdura", "verdura",
                           "fruta", "fruta", "fruta", "fruta", "fruta",
                           "fruta", "fruta", "fruta", "fruta", "fruta",
                           "fruta", "verdura", "verdura", "verdura", "verdura"],

        "Email"         : ["manzana@gmail.com", "manzana@hotmail.com", "manzana@yahoo.com",
                               "aguacate@gmail.com", "aguacate@hotmail.com",
                           "durazno@gmail.com",
                           "durazno@hotmail.com", "tomate@gmail.com",
                           "tomate@hotmail.com", "calabacin@gmail.com",
                           "pera@gmail.com",
                           "naranja@gmail.com", "limon@gmail.com",
                           "platano@gmail.com", "uva@gmail.com",
                           "melon@gmail.com",
```

```
                        "sandia@gmail.com", "pepino@gmail.com", "fresa@gmail.com",
                        "papaya@gmail.com", "guayaba@gmail.com", "zanahoria@yahoo.com",
                        "cebolla@yahoo.com", "brocoli@yahoo.com", "rabano@yahoo.com"],


        "Telephone"     : ["555-555-55-50", "555-555-55-51", "555-555-55-52",
                           "555-555-55-50", "555-555-55-51",
                           "555-555-55-50", "555-555-55-51", "555-555-55-50",
                           "555-555-55-51", "555-555-55-50",
                           "555-555-55-50", "555-555-55-50", "555-555-55-50",
                           "555-555-55-53", "555-555-55-54",
                           "555-555-55-54", "555-555-55-54", "555-555-55-54",
                           "555-555-55-54", "555-555-55-54",
                           "555-555-55-52", "555-555-55-52", "555-555-55-52",
                           "555-555-55-52", "555-555-55-52"],

        "Score"         : [1, 3, 7, 1, 3,
                        7, 6, 8, 11, 0,
                        2, 24, 36, 7, 93,
                        9, 6, 33, 4, 25,
                        0, 19, 12, 15, 18]
    }

    return d


def suggest_providers(request):
    """Responds to any HTTP request.
    Args:
        request (flask.Request): HTTP request object.
    Returns:
        The suggested providers a response text can be turned into a
        Response object using
        `make_response <http://flask.pocoo.org/docs/1.0/api/#flask.Flask.make_response>`.
    """
    request_json = request.get_json()

    # keywords_json must be converted from json to -> dictionary to -> dataframe
    keywords_json = request.args.get('keywords_json')
    keywords_dict = json.loads(keywords_json)
    keywords = pd.DataFrame(data=keywords_dict["data"])
    # print("keywords_json: ", request.args.get('keywords_json'))

    # df_prov_json must be converted from dictionary to -> dataframe
    df_prov_dict = getProvidersDictionary()
    df_prov = pd.DataFrame(data=df_prov_dict)

    import es_core_news_sm
```

```
    sp = es_core_news_sm.load()

    keyword_list = []

    for k in keywords["word"]:
        sentence = sp(k)
        for word in sentence:
            keyword_list.append(word.lemma_)

    df_final = pd.DataFrame()
    LNB = 0

    for l in keyword_list:
        normalizedKeyword = unidecode.unidecode(l)
        value = normalizedKeyword
        df_query = df_prov.query(f"Keyword==@value")

        if df_query.empty == False:
            df_final = df_final.append(df_prov.query(f"Keyword==@value"))
            LNB = LNB + 1

    metric = {"LNB":LNB}

    if df_final.empty == True:
        df_final = df_final
    else:
        df_final = df_final.sort_values(by="Score", ascending=False)

    df_final_result = df_final.to_json(orient="table")
    df_final_parsed = json.loads(df_final_result)
    # https://www.geeksforgeeks.org/append-to-json-file-using-python/
    df_final_parsed.update(metric)

    # print(df_final_parsed)

    return df_final_parsed
```

A.7 User Input

```
from google.colab import data_table
import json
import requests
import sys
import time
```

```
print("Selecciona una opcion a utilizar: ")
print("1) Texto Default")
print("2) Texto aleatorio de un dataset")
print("3) Ingresa tu propio mensaje")
print("4) Salir")

user_input = ""
option = input("Selecciona una opcion: ")
option = int(option)

if option == 1:
  user_input = 'Necesito comprar un kilogramo de naranjas y uvas'
elif option == 2:
  r = random.randint(0, 99)
  user_input = df_messages.iloc[r]["Message"]
elif option == 3:
  user_input = input("Ingresa como le pedirias a alguien una fruta o verdura del supermercado: ")
elif option == 4:
  print("Se eligio no hacer recomendaciones")
  sys.exit()
else:
  print("Opcion no encontrada en el menu")
  sys.exit()

LNB = 0

# keywords must be converted from dataframe to -> json
keywords = getNounCountablesFromBeto(user_input)
keywords_result = keywords.to_json(orient="table")
keywords_parsed = json.loads(keywords_result)
keywords_formatted = json.dumps(keywords_parsed, indent=2)

# df_prov must be converted from dataframe to -> json
df_prov_result = df_prov.to_json(orient="table")
df_prov_parsed = json.loads(df_prov_result)
df_prov_formatted = json.dumps(df_prov_parsed, indent=2)

url = 'https://us-central1-tesina-368609.cloudfunctions.net/suggestProviders'
params = { "keywords_json": keywords_formatted }

print("\n\nMessage: ", user_input)
print("Best providers:")

start = time.time()
response = requests.get(url, params)
end = time.time()

# https://bobbyhadz.com/blog/python-response-object-is-not-subscriptable
```

```
response_dictionary = response.json()

suggestedProviders_df = pd.DataFrame(response_dictionary["data"])
LNB = int(response_dictionary["LNB"])

print("Query time: ", end - start)

suggestedProviders_df
```

A.8 Metrics

```
def calculateLWT(user_input):
  import unidecode
  sp = spacy.load('es_core_news_sm')

  original_tokenized_text = list(user_input.split(" "))

  keyword_list_from_tokenized_text = []

  for k in original_tokenized_text:
    sentence = sp(k)
    for word in sentence:
      keyword_list_from_tokenized_text.append(word.lemma_)

  #print(keyword_list_from_tokenized_text)

  LWT = 0

  for s in keyword_list_from_tokenized_text:
    normalizedWord = unidecode.unidecode(s)
    value = normalizedWord
    df_tokenized_text = df_prov.query(f"Keyword==@value")

    if df_tokenized_text.empty == False:
      LWT = LWT + 1

  return LWT

LWT = calculateLWT(user_input)

# M E T R I C S
print("Model size: 391 MB")
print("Vocabulary training size: 31002 tokens")
print("Average query latency: 1.89 s +- 0.2799 s" )
print("LNB: ", LNB)
```

```
print("LNT: ", LWT)

percentage = 0

if LWT != 0:
  percentage = LNB/LWT

print("Message accuracy = LNB / LWT = ", percentage)
```

Bibliography

[1] How to tap the power of unstructured data in 2022 and beyond?, Saxon, Apr 04, 2022. Available at https://saxon.ai/blogs/how-to-tap-the-power-of-unstructured-data-in-2022-and-beyond/

[2] Data - structured and unstructured – is the key to meaningful insights, Straive, 2021. Available at https://www.straive.com/e-books/straive-transforming-unstructured-data-into-actionable-insights

[3] E-Commerce Jumped 55% During Covid To Hit $1.7 Trillion, Koetsier, 2022. Available at https://www.forbes.com/sites/johnkoetsier/2022/03/15/pandemic-digital-spend-17-trillion/?sh=4b7d81ea5035

[4] S. Louie. A brief history of omnichannel marketing, 2015. Available at https://nectarom.com/2015/01/05/brief-history-omnichannel-marketing/

[5] U. Naseem, "A Comprehensive Survey on Word Representation Models: From Classical to State-Of-The-ArtWord Representation Language Models" arXiv:2010.15036v1 [cs.CL], Oct. 2020.

[6] H. Revelo, "Intent Identification in Unattended Customer Queries Using an Unsupervised Approach". Journal of Information & Knowledge Management. Vol. 20, No. 03, 2150037 (2021). Accessed: Nov. 18, 2022. doi:10.1142/S0219649221500374. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219649221500374

[7] J. Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" arXiv:1810.04805 [cs.CL], May. 2019.

[8] Cañete, J. (2020). Spanish pre-trained model BERT model and evaluation data. Presented as a workshop paper at PML4DC, ICLR 2020. [Online]. Available at: https://users.dcc.uchile.cl/~jperez/papers/pml4dc2020.pdf

[9] IBM Cloud Education. A look into structured and unstructured data, their key differences and which form best meets your business needs, 2021. Available at https://www.ibm.com/cloud/blog/structured-vs-unstructured-data

[10] F. Chollet. Deep Learning with Python, 2nd ed. Shelter Island, United States of America. Manning, 2021, pp. 336-348.

[11] Hugging Face. Spanish BERT (BETO) + POS, 2021. Available at https://huggingface.co/mrm8488/bert-spanish-cased-finetuned-pos?text=Mis+amigos+y+yo+estamos+pensando+en+viajar+a+Londres+este+verano.