André Francisco Gonçalves Gouveia

Template Generation for Automatic Summarization

1 2 9 0 UNIVERSIDADE D COIMBRA

UNIVERSIDADE D
COIMBRA

André Francisco Gonçalves Gouveia

# Template Generation for Automatic Summarization

**Internship Report in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Hugo Gonçalo Oliveira (DEI), Pedro Verruma (Talkdesk), and Bruno Antunes (Talkdesk) and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.**

June 2020

Faculty of Sciences and Technology

Department of Informatics Engineering

# Template Generation for Automatic Summarization

André Francisco Gonçalves Gouveia

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Professor Hugo Gonçalo Oliveira (DEI), Pedro Verruma (Talkdesk), and Bruno Antunes (Talkdesk) and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering.

June 2020


UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

## Abstract

The plethora of unstructured textual data continually growing at an immense rate via call center logs where interactions, using natural language, between agents and customers, happen at a scale of millions per day elevate the urgency of solutions capable of identifying and summarizing relevant content in dialogues.

This internship took place at Talkdesk's Innovative Lab located in Coimbra. Regarding the framework developed, it will integrate the Agent Assist Product. With the clear target of revolutionizing the Contact Center space, Talkdesk Agent Assist is an intelligent conversational guide that supplies real-time contextualized suggested actions to aid and support agents in delivering quality customer interactions.

Agent Assist team already implemented a solution for summarizing calls. However, in this solution, templates are created manually. So, the purpose of the internship is to automate this process by developing a framework, receiving a set of already identified and classified moments, that automatically generates template summaries.

Towards this goal, methods proposed by different researchers for the task of extracting information from dialogues, or even, that summarize dialogues, were first investigated. Later, some datasets were examined based on the linguistic richness, variation, and complexity, as well as the dimension, and therefore was chosen the one most capable of representing the unstructured information present in dialogues.

We opted for an approach relying on triple extraction and clustering to extract the facts that will enter the final templates. Results were obtained that reached 78% F1 for some domain types, which leads us to conclude that our framework is capable of automating the process of generating templates/structural summaries.



## Keywords

Natural Language Processing, Information Extraction, Template Generation, Summarization, Triple Representation

This page is intentionally left blank.

## Resumo

O excesso de dados textuais não estruturados que cresce consistemente a um elevado ritmo via "call centers", onde as interações, usando linguagem natural, entre agentes e clientes ocorrem numa escala de milhões por dia, elevam a urgência de soluções capazes de identificar e resumir o conteúdo relevante nos diálogos.

Este estágio ocorreu no Laboratório Inovador da Talkdesk, localizado em Coimbra. Em relação ao produto desenvolvido, este integrará o produto Agent Assist. Com o objetivo claro de revolucionar a forma como os Contact Center funcionam, o Talkdesk Agent Assist é um guia inteligente de conversação que fornece ações sugeridas contextualizadas em tempo real para ajudar e guiar os agentes na entrega de interações de qualidade com os clientes.

A equipa do Agent Assist já implementou uma solução para resumir as chamadas de "call centers". No entanto, nesta solução, os templates são criados manualmente. Portanto, o objetivo do estágio é automatizar esse processo, desenvolvendo um produto, recebendo um conjunto de momentos já identificados e classificados, que gere automaticamente resumos de templates.

Para alcançar esse objetivo, foram estudados os métodos seguidos por diferentes autores na tarefa de extrair informações de diálogos e de sumarização dos mesmos. Posteriormente, alguns conjuntos de dados foram examinados com base na variação e complexidade da riqueza linguística, bem como na dimensão, sendo, depois, escolhido aquele que tinha mais capacidades de representar as informações não estruturadas existentes nos diálogos.

Optamos por uma abordagem baseada na extração de triplos e no clustering para extrair fatos que irão entrar nos templates finais. Foram obtidos resultados que atingiram 78% F1 para alguns domínios, o que nos leva a concluir que o nosso produto é capaz de automatizar o processo de geração de templates.



## Palavras-Chave

Processamento de Linguagem Natural, Extração de Informação, Geração de Templates, Sumarização, Representação de Triplos

This page is intentionally left blank.

## Acknowledgements

In carrying out this thesis, I have relied on the direct or indirect support of multiple people and institutions to which I am deeply grateful. Thank you:

To the Talkdesk advisors Pedro Verruma, Bruno Antunes, and Liliana Medina for the excellent support throughout this journey, sharing their valuable knowledge and experience. They are truly a source of inspiration for me, I could not have gotten better advisors;

To Talkdesk for the unique opportunity for professional and personal growth, striving for quality and opportunity to learn;

To my DEI advisor, Professor Hugo Oliveira, for his unconditional availability and dedication, teachings, encouragement, support, and sharing of his knowledge and valuable contributions to my work, correcting me when I made mistakes without ever being discouraged. Thank you for accompanying me on this journey and for stimulating my interest in natural language processing;

To my friends, whom I dare not even name at the risk of forgetting some, but whom they know who they are, by their unconditional friendship, support and encouraging words throughout this journey;

Finally, to my family for always believe in my value and praise my potential, helping me to fight for my dreams and goals, supporting me economically throughout my academic life.

This page is intentionally left blank.

# Contents

This page is intentionally left blank.

# Acronyms

**AI** Artificial Intelligence. 1, 11, 44, 45

**ANN** Artificial Neural Networks. 12, 32

**BLEU** Bilingual Evaluation Understudy Score. 34

**CNN** Convolutional Neural Networks. xvi, 13, 14, 33, 38

**CRM** Customer Relationship Management. 2, 3, 45

**DEI** Department of Informatics Engineering. 1

**DL** Deep Learning. xvi, 11–13, 32

**HMM** Hidden Markov Model. 9, 31

**IE** Information Extraction. 93

**LSA** Latent Semantic Analysis. 32

**LSTM** Long Short-Term Memory. 15, 37, 38

**ML** Machine Learning. xvi, 5, 6, 8, 10–12, 30, 39, 40

**MT** Machine Translation. 25

**NER** Named Entity Recognition. 22, 58, 77

**NLP** Natural Language Processing. x, 15–17, 19, 21, 24, 25, 39–41, 44, 65, 67, 72, 77, 93, 94

**QA** Question Answering. 26

**RDF** Resource Description Framework. 23

**RE** Regular Expression. 18

**RNN** Recurrent Neural Networks. 13–15, 33, 37, 38

**ROUGE** Recall-Oriented Understudy for Gisting Evaluation. 34

**RST** Rhetorical Structure Theory. 32

**TDX** Talkdesk's Innovation Lab. 1, 2

**TF-IDF** Term Frequency - Inverse Document Frequency. 30

# List of Figures

This page is intentionally left blank.

# List of Tables

This page is intentionally left blank.

# Chapter 1

# Introduction

The present document reflects the work implemented by the author. It describes the framework aimed to be integrated as a microservice in the Talkdesk product. This project was developed in the scope of the Master's degree in Informatics Engineering, Specialization in Intelligent Systems, at the University of Coimbra, during the academic year 2019/2020. The internship took place at Talkdesk's Innovation Lab (TDX), and it was supervised by both Pedro Verruma and Bruno Antunes, the Talkdesk advisors, and Professor Hugo Gonçalo Oliveira, the Department of Informatics Engineering (DEI) advisor.

This introductory chapter intends to explain the fundamental guidelines of this Master Thesis, clarifying its motivation and established objectives. Furthermore, it provides a contextualization of this proposal and describes the structure of the document with a brief review of each chapter's content.

## 1.1 Motivation

We are witnessing a relentless technological evolution where the field of Artificial Intelligence (AI) has assumed a leading role. There are more and more intelligent systems developed to perform the most diverse tasks, which imply solutions that will mimic the human capabilities. One of these will be the ability to understand language leveraging the power of Natural Language Processing [38].

With the burgeoning amount of unstructured information that is available all over the web, which is trending to increase and not the opposite [18], knowledge workers are forced to manually read and analyze texts, in order to collect valuable knowledge for their assignments. However, for many companies, performing such task and simultaneously keep up with the fast-growing amount of information every day, from emails to documents, reveals to be extremely time-consuming, and consequently financially unviable.

In a contact center, for example, the interactions are typically managed using natural language and are processed at a scale of millions per day. In order to obtain worthy incomes and take actionable insights from

these interactions, when a call ends agents need to be capable of distinguishing the relevant highlights from all redundant information that conveys in between, and write down these highlights for later review. This process leads to significant disadvantages. First, manually capturing and populating the call information into a Customer Relationship Management (CRM) system is an exhausting and time-consuming task conducted by agents. Secondly, difficulty in stick towards a streamlined process of capturing information leads to inconsistent reporting. Last, but not least, if the customer contacts a new agent, and there is no trail of the previous conversation, the latest agent is not prepared to provide the proper support that ultimately impacts customer satisfaction. All these problems motivate and accelerate the need for an automated technological process that helps optimize and reduce After Call Work.

## 1.2 Context

As previously said, the internship took place at TDX, an Innovation Lab of Talkdesk located in Coimbra, whose purpose is to fast-track key technology initiatives. Talkdesk is a could-based communication software that provides users with a call center hub that empowers companies to continuously improve their customer experience. With the clear target of revolutionizing the Contact Center space, the company eliminates the need to set up an on-premises call center, thus saving companies from the added related costs, and has the capability of being integrated with business apps and systems, such as CRM, sales software, and customer databases.

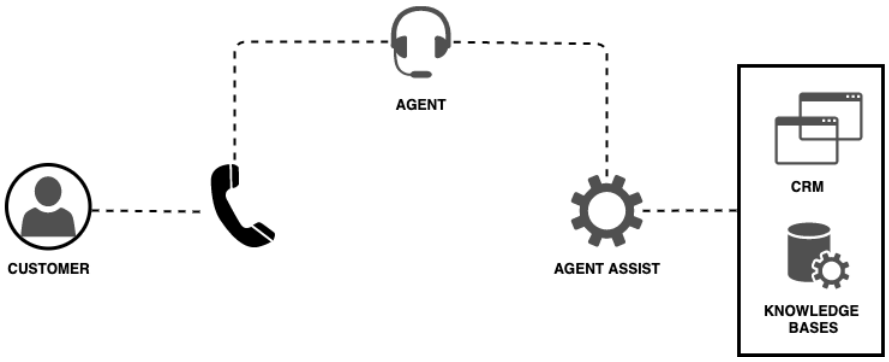


Figure 1.1: Agent Assist Process.

The framework developed during this internship will integrate the Agent Assist product. Talkdesk Agent Assist is an intelligent conversational guide that supplies real-time contextualized suggested actions to aid and guide agents in delivering quality customer interactions and outcome excellence.

The main objective of Agent Assist is to support agents guide customers towards a solution with reduced effort boosting agent's productivity and proficiency. The core components of Agent Assist, as shown in figure 1.1, are:

- Proactively deliver recommended responses in real-time, from Knowledge Bases, to any request form the customer.
- Integrated with CRM systems, it also automates actions such as follow up tasks.
- Automate data entry tasks.

With the framework implemented in this internship, the Agent Assist product is a step closer to facilitate interactions between customers and agents and make the process more efficient since the beginning of the call.

## 1.3 Goals

At Talkdesk there is already a solution for summarizing call center calls, however, this solution requires that templates have to be created manually, which is impracticable since this task is high consuming and ineffective. Hence my internship, whose objective is to automate the process mentioned above, that is, to automatically generate summary templates. This solution poses great challenges because it is necessary to process multiple and varied examples of moments and find patterns on them that can be used in the summary. This solution allows reducing the agent's work and his After Call Work. By moments, we mean a selected segment of dialogue, between a customer and an agent, with an intent. The system's components should be simple to change and upgrade, and, as such, its architecture should be as modular as possible. The project should support the English language and be deployed in different domains.

From an academic point of view, the objectives of this project were to consolidate and acquire knowledge in several areas of Artificial Intelligence and gain experience in the development of solutions at an enterprise level, including processes and expected deliverables.

## 1.4 Structure

As for the structure of this thesis, it will be divided into the following chapters (this introductory chapter is not included):

**Chapter 2 - Background Knowledge**

This chapter presents the most relevant concepts for the implementation of this project. It starts with a concise description of the three main fields on Artificial Intelligence, Machine Learning (section 2.1), Deep Learning (section 2.2), and Natural Language Processing (section 2.3), and an explanation on Summarization (section 2.4).

**Chapter 3 - State of the Art**

This chapter describes the existing methods related to the purpose of this internship, as well as the approaches that the different authors adopted (section 3.1), an analysis of resources and tools that can help in the intern implementation (section 3.2), and an outline of the competitors to this project alongside a comparative study between them (section 3.3).

**Chapter 4 - Approach**

This chapter describes the entire process that led to the specification of the framework (section 4.2), and then, the system's use cases, requirements, and constraints are presented (section 4.3). Following that, a risk analysis is displayed (section 4.4), and an overall perspective on the architectural design of the major constituents is given (section 4.5). The last section 4.6 provides a summary of the development methodologies used and work planning.

**Chapter 5 - Implementation**

This chapter outlines the work that was done throughout the internship, the final result, as well as some implementation details.

**Chapter 6 - Validation and Results**

This chapter presents all the validation tests realized. For each variant, a description is given, and the results outlined.

**Chapter 7 - Conclusion**

Lastly, this chapter addresses the overall conclusions and retrospectives about this internship. It serves as a recapitulation of the proposed work done through both semesters, as well as analyzing some challenges that we had to surpass. We also reflected on possible improvements to incorporate in the future work.

# Chapter 2

# Background Knowledge

This chapter sets the scene for this thesis and is divided into four sections. It starts by introducing, in section 2.1, the concept and the purpose of Machine Learning, as well as the different types of learning tasks in this field, and a brief description of the stages that compose a Machine Learning workflow. Section 2.2 explains the need for Deep Learning and how it makes use of Artificial Neural Networks. Section 2.3 presents Natural Language Processing and the various levels of knowledge concerning each type of analysis. Ultimately, in order to understand the intent of this project and achieve an initial perception of the main topic, section 2.4 specify the many variances of Summarization and the fundamental strategies that set a tone for future researchers.

## 2.1 Machine Learning

Machine Learning (ML) is a branch of Computer Science and Artificial Intelligence that has become, over the last two decades, a powerful and global tool in a wide variety of applications, such as medical services, pattern recognition, or in language processing fields. The primary aim of ML is to create intelligent machines that can learn, modify, or adapt their actions and therefore improve their performances by gathering more data and experience without the intervention of humans. So, let us see the following example: you are playing Chess, a famous board game, against a computer. In the beginning, it seems that you are dominating, and the victories are succeeding. However, with an increase of games played, the paradigm shifts, and triumphs start tending towards the computer side. Why does this happen? The machine starts to learn and adjust actions accordingly, converting experience into expertise.

In today's world, we are surrounded by machine learning-based technology. Anti-spam software, accident prevention systems, and intelligent personal assistants are just a few examples we can list. Nevertheless, how did they come about? Who introduced the concept of ML? Although this field coalesced in the 80s, its roots lie way back. Alan Turing [79], in 1950, besides having published a paper in which he asked "Can

machines think?" created the so-called "Turing Test" to determine if a computer could be considered intelligent. Two years later, Arthur Samuel [71] created the first learning-program that helped an IBM computer get better in the game of checkers. Another critical achievement occurred in 1967 when the first pattern recognition program was able to detect a pattern in data [49].

Since then, with the exponential growth of data available online, coupled with the fact that there are more and more subfields of ML dealing with different types of learning tasks, scaling up ML is an emerging obligation. Reasons such as:

- Algorithm's complexity and the need for adaptivity.
- Increase in the amount of observation that could be used as training data and data points with several features.
- High dimensionality input.
- Dealing with inference time constraints.

substantiate the improvement observed in the last years, as well in the future ahead.

The next sections describe in more detail some of the current algorithms used and how they are classified, as well as a further explanation of the steps that form a project in this field.

### 2.1.1 Machine Learning Approaches

Because ML is a very wide domain, ML algorithms are classified into these four main categories:

**Supervised Learning**: "a training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalizes to respond correctly to all possible inputs" [50]. This type of learning is the most used among ML algorithms and stands out in the applications indicated in figure 2.1. The main difference between the two applications presented is that classification is the task of predicting a categorical, or discrete, class label while regression stands for predicting a continuous quantity. Despite the various advantages that supervised learning leverages, problems when dealing with non-linear relationships/decision boundaries, or intricate patterns can be time-consuming.

**Unsupervised Learning**: the system is not provided with information classified or labeled to train, but instead the algorithm tries to identify similarities between pairs of objects and thus model the underlying structure of data. The algorithms in this category can be further classified into clustering (grouping data points according to their similarities), association (find associations/relations amongst data objects), and dimensionality reduction tasks [50]. The figure 2.2 shows the benefits of solving them. However, since the data handled is labeled and not known, getting precise information regarding data sorting and high accuracy on results is not feasible.

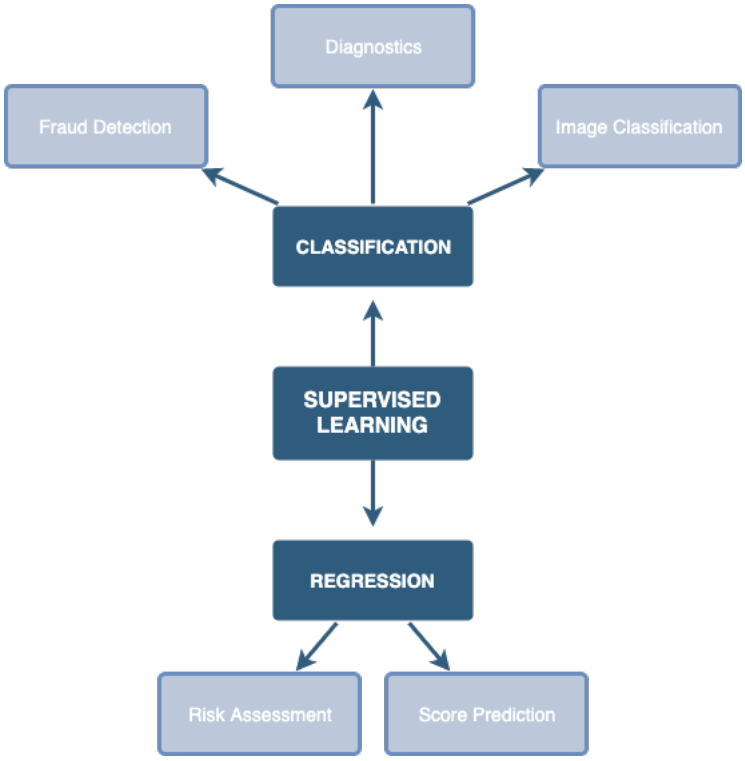


Figure 2.1: Applications of Supervised Learning.

**Semi-Supervised Learning**: is midway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is supplied with some supervision information — not required for all examples [77]. It was introduced to counter the disadvantages of the two above mentioned types of learning. That is, for supervised learning the dataset has to be hand-labeled, while for unsupervised the application spectrum is limited.

**Reinforcement Learning**: agent, environment, and actions are the three components of this type of learning. The algorithm is told when the answer is wrong, with a reward feedback, but is not told how to correct it. The agent has to explore the environment and try out different possibilities (actions) until it works out how to get the right answer [50].

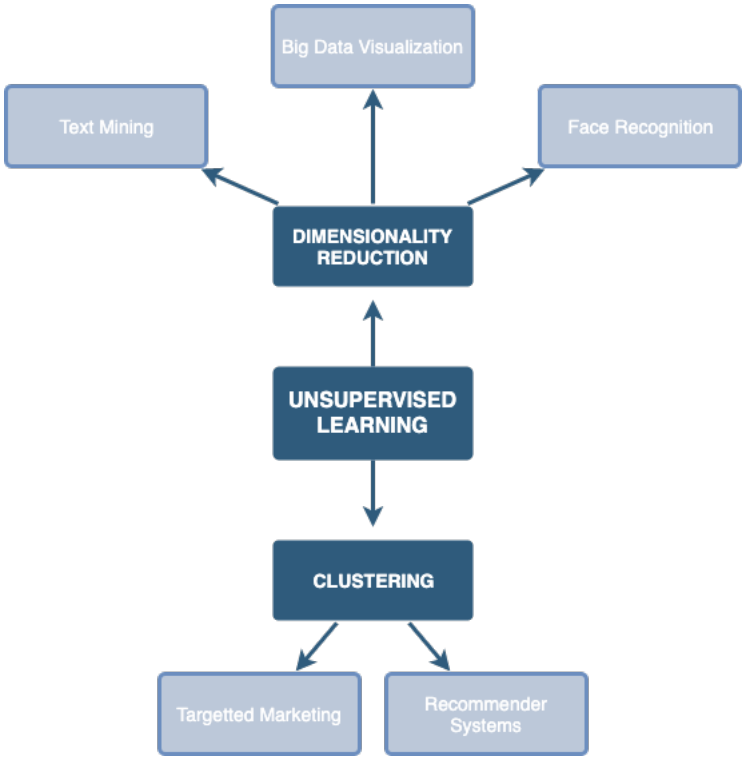


Figure 2.2: Applications of Unsupervised Learning.

### 2.1.2 Machine Learning Algorithms

ML has an extensive range of algorithms, including:

**Logistic Regression**: a statistical algorithm that builds a linear model based on a transformed target value [10]. It is generally used to predict a binary outcome but can also be used to predict multi-class problems if required.

**K-Nearest Neighbors**: learning algorithm that spends no time during the learning process, but only for pre-processing and testing. The straightforward algorithm, memorizing the training set directly, makes predictions based on a similarity measure (e.g., Euclidean, Manhattan) and by checking the k-nearest neighbors of the sample to classify. Then, according to the majority vote, the class label is assigned to the sample [32].

**Decision Trees**: classifiers in the form of a tree-like structure built for regression and classification applications based on a sequential decision process. That is, they evaluate instances/features by starting from the root of the tree and moving along the branches until reaching a final leaf node,

which generally represents the classification target. In figure 2.3, the classification target can be either: Stay in, Go to beach, Go running, or Go to movies.

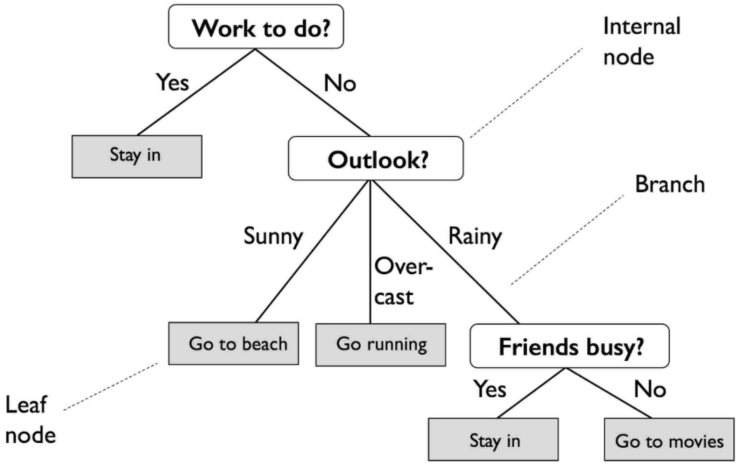


Figure 2.3: A Decision Tree example [67].

**Support Vector Machines**: capable of operating with both linear and non-linear scenarios is a binary classifier that defines a separating hyperplane differentiating data points of distinct categories. With this optimized hyperplane, the margin of training data is maximized, and new entry points can be classified [78]. Figure 2.4 displays an example illustrating the separating hyperplane between class -1 and 1.

The following equation represents the best hyperplane found, with $x$ being the dataset of feature vectors to classify:

$$\bar{w}^T \bar{x} + b = 0, \tag{2.1}$$

**Naive Bayes**: a non-parametric probabilistic model that is used for classification tasks. Contingent on Bayes' theorem assumes that the presence of a variable value on a certain class is unrelated to other variables [53]. In other words, if $X$ and $Y$ are conditionally independent given the event of $Z$, knowledge of whether the variable $X$ occurs presents no information on the likelihood of variable $Y$ occurring, and vice versa.

**Hidden Markov Model (HMM)**: used for modeling sequences, that is, by observing a sequence of states, it is possible to calculate the probability of a certain sequence of states. It is also often used for classification applications [74]. For instance, we use an HMM to model a male voice and a female voice. Then we can predict if a new voice sample whether or not belongs to a male or a female.

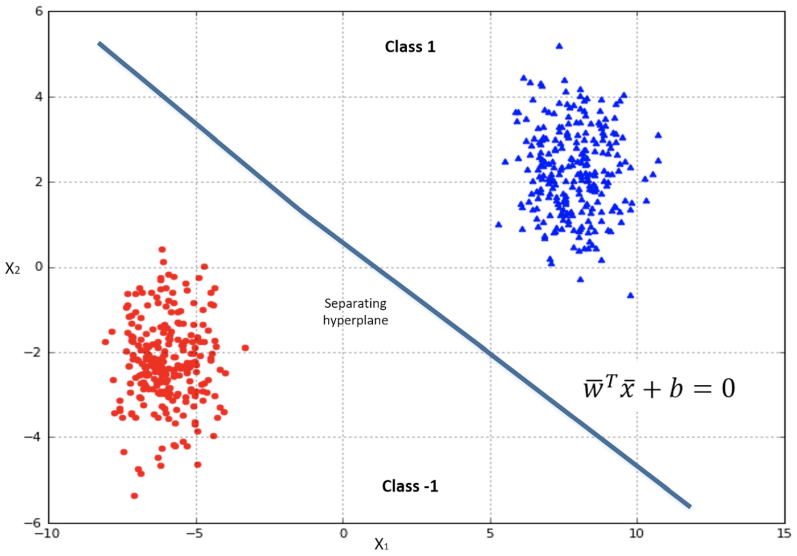


Figure 2.4: Bidimensional representation of a hyperplane [15].

### 2.1.3 Workflow of a Machine Learning Project

To build and guarantee a proper ML project is a highly iterative and experimental process. Although these projects vary in scale and complexity, they follow a common step-by-step workflow:

**Pre-processing**: To achieve great results, before performing the following steps, raw data (collected from various sources) must be clean and wrangle into an acceptable format. Missing data, inconsistent text data, or the existence of values that deviate strongly from other observations in data (known as outliers) are clear examples of the need for this step. Also, feature scaling is another main process to accomplish. It is applied to independent variables and normalizes the data within a specific range.

**Learning**: The first step is researching and choosing the model (between supervised and unsupervised) that will be best and more suitable for the pre-processed data is the first step. Automatically optimizing the hyperparameters of a model is an optional step. Hyperparameters are referred to as the parameters which are not updated during the learning phase and applied in the model configuration. Then the training process starts, and the data is divided into three groups:

1. Training set: build up the model from training data. At this stage, feature importance may be employed. This process aims to assign a score to features based on how valuable they are at predicting a target variable
2. Validation set: validate the model with k-fold cross-validation. This technique splits the input data into k subsets of data. The model is trained on all but one of the subsets and then

evaluated on the subset that was not used for training

3. Testing set: implement an unbiased evaluation of a model fit on the training dataset

**Evaluation** and **Prediction**: Model evaluation and prediction are integral parts that helps in understanding how well the chosen model will work and assess his performance in previously unseen test data.

## 2.2 Deep Learning

A dominant origin of difficulty in numerous real-world Artificial Intelligence (AI) applications is that many of the factors of variation influence all pieces of data we are able to see. "Most applications require us to disentangle the factors of variation and discard the ones that we do not care about" [30]. For instance, a speaker's attributes, such as gender or accent, are undoubtedly abstract features that require powerful and almost human-level implements for being extracted from raw data. Deep Learning (DL) is considered the answer to overcome these problems.

### 2.2.1 What is Deep Learning?

DL, an approach to ML, is about learning multiple levels of representation and abstraction, enabling the computer to create complex concepts out of simpler ones like a nested hierarchy [30]. Unlike shallow ML algorithms that fail to handle high-dimensional data or that are not very successful at tasks such as recognizing patterns, this approach, due to its needlessness of domain expertise and hardcore feature extraction, has seen tremendous growth lately. The ability to learn an implicit representation of the raw data without manual effort was one of the main reasons why Deep Learning models increased accuracy, complexity and prompted real-world impact solving complex problems, see figure 2.5.

Although DL seems like a recent field, it dates back to the 1940s. To be more precise, in 1943, when Warren McCulloch and Walter Pitts "introduced models of neurological networks, they recreated threshold switches based on neurons and showed that even simple networks of this kind are able to calculate nearly any logic or arithmetic function" [41]. Since then, there were three waves of development, reflecting different philosophical viewpoints: Deep Learning known as cybernetics (in the 1940s-1960s), connectionism (in the 1980s-1990s) and, since 2006 the current uprising under the name of Deep Learning.

But the following question that arises is why now? Why only recently this field became so popular, catching the attention of the science community? Two main reasons justify the enormous impact in areas like automated driving, speech and audio processing, Natural Language Processing or medical research:

1. DL has become more useful as the amount of available training data has increased [30]

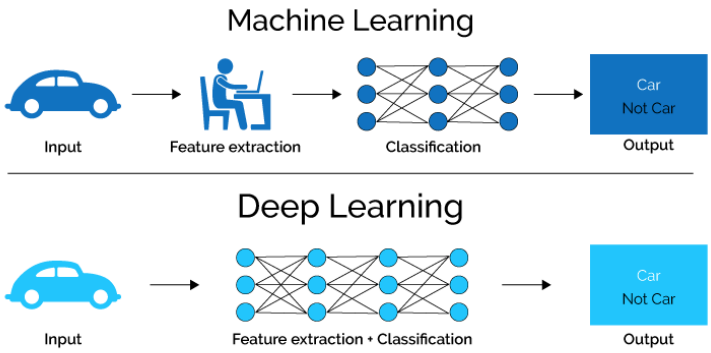


Figure 2.5: Differences between ML and DL [1].

2. DL models have grown due to the advance of computing power and infrastructures (high-performance GPUs/CPUs and faster network connectivity)

### 2.2.2 How does Deep Learning work?

Work on artificial neural networks, commonly referred to as "neural networks", has been triggered undeviatingly from its inception by the recognition that the human brain operates in an entirely distinct way from the conventional digital computer. The brain is profoundly complex, nonlinear, and act as a parallel computer. It can organize its structural constituents, recognized as neurons, to perform some computations much quicker than the fastest digital computer in existence now [33]. DL algorithms are roughly inspired by these abovemencioned networks and the structure of the brain. For example, in image processing, the input of these algorithms is presented at the visible layer, containing the variables that we can observe. Then a group of hidden layers extracts increasingly abstract features from the image [30]. In these layers, each node performs an Activation Function to convert the input into output which can then be used as input for other nodes. Finally, the output is obtained by combining all layers and their information to identify objects present in the image in question. Figure 2.6 illustrates the previous explanation.

When we approach the different types of Artificial Neural Networks (ANN), it is imperative to talk about **FeedForward Neural Networks**. These networks, extensively used in pattern recognition, were among the first and quintessential learning algorithms and have just one condition: the information must proceed in only one direction with no back-loops, from the function being assessed from $x$, through the intermediate computations applied to determine $f$, and subsequently to the output $y$ [30]. They compose together several distinct functions, and there are no limitations on the number of layers or connections between neurons.

Feedforward networks, and especially their specializations, are a key example of the successful use of ma-

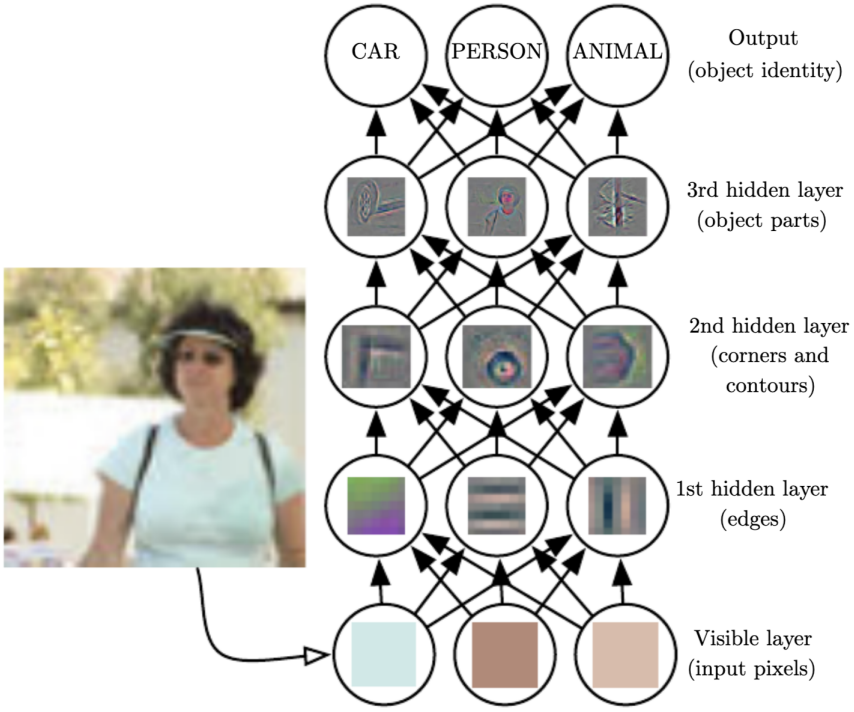


Figure 2.6: Illustration of a Deep Learning model [30].

chine learning applications. For instance, Convolutional Neural Networks (CNN) are particularly helpful at problems like text classification, while Recurrent Neural Networks (RNN) play an important role in finding patterns to predict what will happen next, given certain circumstances based in the past. These networks are described next regarding their purpose and advantages.

#### Convolutional Neural Networks

CNN are a specialized type of neural network that has the particularity of eliminating the need for manual feature extraction. They consist of an input and output layer, and $n$ hidden processing layers between them. Due to these hidden layers and their powerful aptitude to learn and extract hierarchical representations automatically from data, these networks are currently one of the most prominent algorithms for DL in various competitive benchmarks.

The topology of CNN is split into multiple learning stages constituted by a combination of three types of hidden layers, as displayed in figure 2.7 with a concise explanation of each of them below.

1. **Convolution layers**: The layer's parameters consist of learnable kernels that spread through the entire depth of the input. From this, the network will learn which kernels 'fire' when they detect a specific feature at a certain spatial position of the input. These are generally identified

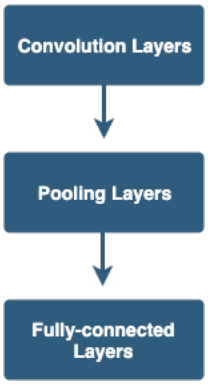


Figure 2.7: Examples of hidden layers in CNN.

as activations. "Every kernel will have a corresponding activation map, of which will be stacked along with the depth, dimension to form the full output volume from the convolutional layer" [60].

2. **Pooling layers**: These layers will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation [60].
3. **Fully-connected layers**: They perform classification based on the features extracted by the previous layers and then output the final result.

### Recurrent Neural Networks

This type of neural network distinguishes itself from FeedForward or CNN in being capable of processing and model sequential data, instead of fixed-data and assuming that all inputs are independent of each other.

To better demonstrate the advantages of RNN lets study the following sentence:

*After my allergy **attack**, the doctor resolved to **attack** the problem with a new remedy*

When analyzing and classifying the previous example, understanding the surrounding context is an obligation to do it successfully. In this sentence, the first use of the word “attack” is a noun, while the second is a verb. So, how would a FeedForward Network, that models only a word at a time, will be aware of the difference? We can infer now that features, such as internal memory, which captures information about what has been "seen" so far, and a looping mechanism, that allows data flow between steps, have a profound influence on the learning ability of the RNN and its performance.

However, when dealing with long sequences, RNN suffer from short-term memory, which means they can only look a few steps backward and leave out valuable information. This problem is caused by the vanishing gradient problem - small gradients mean small updates in the weights of a neural

network, which implies that the early layers will not learn. To avoid this outcome, a main specialized RNN named **Long Short-Term Memory (LSTM)** makes use of a memory block composed by gating units that helps the network distinguishing which data can be thrown away and which needs to be kept. This way, the network is more than capable of processing large sequences.

## 2.3 Natural Language Processing

Natural Language Processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and human (natural) languages [42]. The ultimate purpose of NLP is to examine, understand and achieve human-like comprehension of languages. It is a discipline that has become embedded in our daily lives and is scaling to a myriad of industries related to finance or healthcare, among others.The term *natural language* refers to any language (such as Portuguese or Russian) used in the communication between humans, as distinct from formal languages (such as C+ or Python) which are specifically developed to interact with a computer and understood by a machine [42].

NLP has come a long way, and despite the high growth we have witnessed in recent years, this field faces yet many challenges. Dealing and extracting valuable insights from unstructured data hard to manipulate, generated from social media posts like Twitter, text messages, reviews or other roots, and present in the majority of enterprise's data, is a clear example. Also, a very critical problem is the ambiguity, that arises whenever an expression can be interpreted in different ways, which occurs at each level of analysis/knowledge, and it has the potential to multiply across levels [27]. Just as an illustration, here are presented two cases of the ambiguity problem [42]:

1. When the word "bank" can have two different meanings:
   *Withdraw some money from the* ***bank***
   *Do not go near the* ***bank*** *of the river*

2. When the same sentence can be interpreted in more than one way:
   *I saw the boy with the telescope*
   The same sentence can be referred as "you saw the boy using a telescope", and at the same time as "you saw the boy having a telescope with him". Who had the telescope?

Language processing essentially relies on the representation of knowledge related to the language accordingly with specific levels that tend to focus on meaning and structure. The more capable an NLP system is, the more levels it will utilize. Table 2.1 describes the different levels of knowledge in natural language:

| | **Description** |
|---|---|
| Phonology | Describes the systematic arrangement of sound. Deals with speech recognition and generation. |
| Morphology | "Study of the way words are built up from smaller meaning-bearing units, morphemes" [37]. There are two main types: **derivational** morphology (use of affixes to turn a word from one grammatical level to another or to switch the meaning of the word [25]) and **inflectional** morphology (addition of aspects such as gender, number, person, and tense [25]) |
| Lexical | Responsible for identifying and analyzing all the relevant linguistic information about each word, such as their type (noun, verb, etc). |
| Syntax | Refers to the study of formal relations between words from sentences. In this phase, the legitimacy of a sentence according to the grammar rules is checked. To perform syntactic analysis, both the knowledge of grammar and parsing technique is required [42]. |
| Semantics | Knowledge about converting natural language sentences into a meaning representation. |
| Pragmatics | Focuses on the relation between language and context of use, that is, trying to perceive how speakers express their intended meaning and what deductions the listeners draw in a given context. |
| Discourse | Knowledge about linguistic units larger than a single utterance [37]. Set of well-formed and individually interpretable sentences regularly produces coherent discourse [42]. |

Table 2.1: Summary of the distinct levels of knowledge

Notwithstanding the varied existence of levels, their usage depends on the purpose of the project to be applied. Since the nature of this internship is to develop a framework mainly converged on text analysis and template generation based on relevant content identified, the next sections will focus on the following levels: *Lexical*, *Syntax*, *Semantics*, *Pragmatics*, and *Discourse*. It is worth saying that the *Phonology* level will not be analyzed since Talkdesk have its speech-to-text service and in this work, we will be working directly with transcriptions.

### 2.3.1 Lexical Analysis

In NLP, the first stage of processing input text is to examine each word in the sentence and is followed by the lexical analysis that verifies the validity of words according to the lexicon. Lexicon stands for a dictionary, and it is a "collection of words and/or phrases along with associated information, such as

part-of-speech and sense definitions" [42][13]. To perform this kind of analysis some operations need to be executed:

- **Sentence Segmentation**

  One of the primary steps performed on texts is the process of breaking a running text into separate sentences.

  ***input:*** *Hi there! Anything I can help with?*

  ***output:*** *[Hi there!] [Anything I can help with?]*

  However, this is not a trivial task or straightforward since punctuation signs that mark the end of a sentence are ambiguous. More explicitly, the period sign can denote an abbreviation ("Mr." or "P.S."), a decimal point or the end of a sentence.

- **Tokenization**

  "A word is the minimal unit that can convey meaning to a human. So any text string cannot be further processed without going through tokenization". Tokenization is the process of dividing the raw string into tokens. The complexity of tokenization diversifies according to the demand of the NLP application, and the complexity of the language itself [31]. For instance, there are some cases such as hyphens and apostrophes that need particular attention.

  ***input:*** *TDX is located in Coimbra.*

  ***output:*** *[TDX] [is] [located] [in] [Coimbra] [.]*

- **Stop Words**

  Stop Words are high-frequency words that we often want to filter out of a document before further processing. Stop words regularly have limited lexical content, and their appearance in a text folds to discriminate it from others [13].

  ***input:*** *At Talkdesk we are reimagining how people experience contact centers*

  ***stop words identified:*** *"we", "are", "how"*

- **Lemmatization**

  Lemmatization is a more systematic way of converting all the inflected forms to the root of the word, named lemma, as it occurs in language dictionary. It uses context, to accurately lemmatize ambiguous forms, and part of speech to determine the inflected form of the word and involves different normalization rules for each part of the speech to get the root word [31].

  ***E.g:*** *The word "**run**" is the lemma of the words "runs", "running", and "ran"*

- **Stemming**

  Stemming consists of collapsing the morphological variants of a word together. This process when applied, utilizes simple pattern matching to strip suffixes of tokens (e.g., remove “ing”, remove “s”).

The primary advantage of utilizing stemming is that it enables a particular query term to match documents comprising any of the morphological variants of the term [37].

***E.g:*** *In the words* ***"booking"*** *and* ***"booked"****, when executed the stemming process, the output will be the common form "book".*

- **N-Grams**

  An n-gram is a contiguous chain of n items from a provided sample of text or speech. N-gram models formalize the idea of word prediction with probabilistic models foretelling the next word from the prior n - 1 words [37]. Figure 2.8 exemplifies this operation with sequences of one, two, and three items.

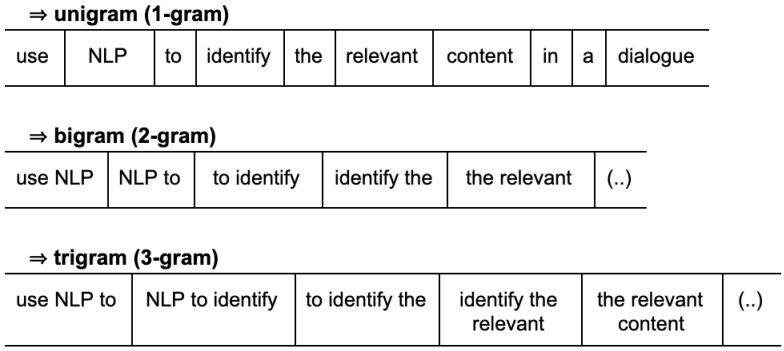


Figure 2.8: N-Grams example.

There are some challenges when dealing with n-grams. A short n-gram size advances high bias, and an extensive n-gram size presents high variance. "Language is full of long-range dependencies that we cannot capture because $n$ is too small; at the same time, language datasets are full of rare phenomena, whose probabilities we fail to estimate accurately because $n$ is too large" [25].

- **Spell Checking**

  Given an input file of text, the goal is to identify the words that are not properly written. A spelling corrector both exposes misspelled words and attempts to determine the most likely correct word. Correcting a misspelling requires a search of the word space to pick the nearest neighbor(s) of the wrongly spelled word [63].

- **Regular Expressions**

  A Regular Expression (RE), regex, is a sequence of characters that define a search pattern. They are especially valuable for searching in texts when we have a pattern to seek for and a corpus of texts to search through. A RE search function will explore through the corpus, replying all texts that match the pattern[38]. In table 2.2 we present some of the variants of the regular expressions:

| | **Match** |
|---|---|
| /[abc]/ | 'a','b', or 'c' |
| /[1234567890]/ | any digit |
| /[a-z]/ | a lower case letter |
| /[0-9]/ | a single digit |
| /[^A-Z]/ | not an upper case letter |
| /[a^ ]/ | either a 'a' or a '^' |

Table 2.2: Different types of regular expressions.

### 2.3.2 Syntactic Analysis

Syntactic Analysis is the second phase of the NLP pipeline and has the purpose of validating a sentence according to certain grammar rules. To perform the syntactic analysis, the knowledge of grammar and parsing techniques is required. Grammar is a specification of rules admissible in the language and parsing is a technique for analyzing a sentence to discover its structure according to a grammar [42]. For instance, the sentence *"Boy the go the to store"* [20] would be rejected after performed a syntactic analysis as a result of the structural relationships between words (the knowledge required to order and pair words together). However, this type of analysis is susceptible to problems that can arise due to reasons such as a word that may function as distinct parts of speech in different contexts, or a structure of a sentence that has several possible interpretations. Certain processes that involve this analysis include:

- **Part-of-speech Tagging**

  This process consists of grammatical classes to which words are assigned, based on its context, that is, its relationship with adjacent and associated words in a sentence, and role. Parts of speech are considered a useful resource because they can unveil a lot about a word and its neighbors. Discerning whether a word is a noun or a verb indicates us the likelihood of certain neighboring words (nouns are usually preceded by determiners and adjectives, and verbs by nouns) and syntactic structure as well, making this process a key aspect [38]. There are four main parts-of-speech tags, among others - Noun (**NOUN**), Verb (**VERB**), Adjective (**ADJ**), and Adverb (**ADV**). Let's consider the following sentence with the result of this operation on the table 2.3.

  ***input:*** *I am going to Amsterdam next Friday*

- **Chunking**

  The following technique parts and labels multi-token sequences as illustrated in 2.9. The larger boxes display higher-level chunking, and each of them is designated a chunk. Similar to tokenization, which omits whitespace, chunking usually selects a subset of the tokens. Also like tokenization, the items produced by a chunker do not overlap in the source text. This collection typically includes phrases that match to the content-bearing parts-of-speech [13] [37]. There are generally 5 five major

| | **Tag** |
|---|---|
| I | PRP (pronoun) |
| am | VBP (verb) |
| going | VBG (verb) |
| to | ADP (adposition) |
| Amsterdam | NNP (noun) |
| next | JJ (adjective) |
| Friday | NNP (noun) |

Table 2.3: Output after performing PoS Tagging.

categories of phrases, which are Noun phrase (**NP**), Adjective phrase (**ADJP**), Verb phrase (**VP**), Adverb phrase (**ADVP**), and Prepositional phrase (**PP**).

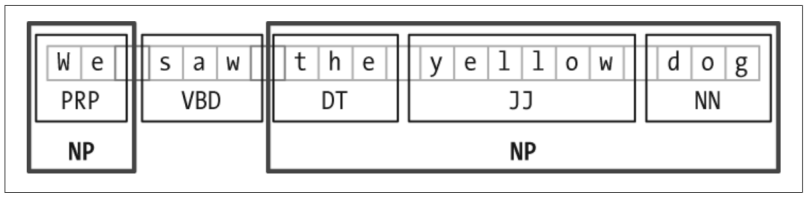


Figure 2.9: Chunking example [13].

After performing the chunking process, you can have chunks with their chunk tags, along with individuals words with their POS tags. These individual words are not relevant to the final result and can be withdrawn. So, the concept of chinking is introduced. **Chinking** allow us to remove from the chunk what we want and deal with the issue presented previously.

- **Parsing**

  Parsing is the automatic analysis of a sentence concerning its syntactic structure based on an underlying grammar (of the language) and plays an important role in applications such as Question Answering or Information Extraction. There are two main types of parsing, which will be described briefly:

  – Constituency Parsing

    This method strives to extract a constituency-based parse tree from a sentence representing its syntactic structure according to a phrase structure grammar and in terms of a hierarchically ordered structure of their constituents.

    However, ambiguity can be a real problem, especially structural ambiguity that occurs when the grammar can assign more than one parse to a sentence. Considering the following figure 2.10 of this type of ambiguity, here the "parse on the left corresponds to the humorous reading

in which the elephant is in the pajamas, while the parse on the right corresponds to the reading in which the photographer did the shooting in his pajamas" [38].

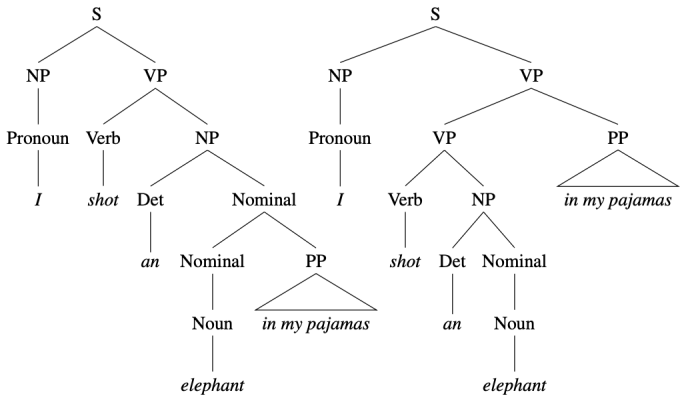


Figure 2.10: Two parse trees for an ambiguous sentence [38].

– Dependency Parsing

The associations between words in a sentence can be represented in a directed graph, based on the lexicalized phrase-structure parse: form an edge (i, j) if word $i$ is the peak of a phrase whose child is a phrase headed by word $j$. Thus, in figure 2.11, we obtain "scratch" -> "cats" and "cats" -> "the". These edges specify syntactic dependencies, a bi-lexical relationship among a head and a dependent [25].

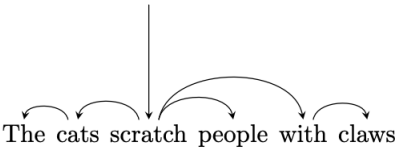


Figure 2.11: Unlabeled dependency tree [25].

### 2.3.3 Semantic Analysis

As the following phase of the NLP pipeline, semantics aims to study the meaning in language by pivoting on the interactions among word-level meanings in the sentence. To be able to understand what a given sentence means, it is important to understand some of the lexical-semantic relations, as shown in table 2.4.

| | Description | Example |
|---|---|---|
| Hyponymy | Word or phrase whose semantic field is included within that of another word, its hypernym | The word **color** is hypernym and the color **green**, **red** are hyponyms |
| Homonymy | Two different words with the same spelling and with the same sound | The words **peer** (person belonging to the same group in age and status) and **peer** (look searchingly) |
| Synonymy | Word with the same or nearly the same meaning as another word | The words **destiny** and **fate** |
| Antonymy | Words which have the opposite (or nearly opposite) meaning | The words **hot** and **cold** |

Table 2.4: Lexical-semantic Relations.

Performing this analysis involves two steps. The first one is called lexical semantics which studies the meaning of individual words, and the last one has the purpose of studying the combination of the individual words. Some of the processes that encompass the semantic analysis will be presented:

- **Named Entities**

  Aside from PoS Tagging, a common labeling problem is locating meaningful entities in the text. Examine the next example:

  *The United States Army captured a hill overlooking Atlanta on May 14, 1864* [25]

  For the presented sentence, the first intention is to extract the called entity mentions that are composed by the spans: "United States Army", "Atlanta", and "May 14, 1864", and then associate with each a particular entity type. This two-step process is designated as Named Entity Recognition (NER) and it is a critical subtask of information extraction which focuses on recognize and classify information units according to a category group already defined, e.g, person names, organizations, locations, time and dates, etc. However, there are complications with current named entity recognition systems. The necessity to fine-tune a system for every new domain constitutes a significant drop in the performance and the quality of the output. Another quandary is the lack of semantic/context and external knowledge particularly in informal texts where the public needs to recognize names [31]. The other intent is to perform the entity linking task, which links the spans identified in the previous step to entities in a knowledge base.

  Additionally, techniques such as feature-based (e.g., embeddings, part-of-speech, and chunk label of neighboring words) or neural sequence labeling (based on a bi-LSTM) achieve accurate results and are set as viable solutions [38].

- **Relations**

  After obtaining the entities detected, the next task consists of identifying relations among them. Relations, considering the order of the inputs, take precisely two arguments. Table 2.5 shows some relation samples.

| **Relations** | **Types** | **Examples** |
|---|---|---|
| Physical-Located | PER-GPE | **He** was in **Tennessee** |
| Part-Whole-Subsidiary | ORG-ORG | **XYZ**, the parent company of **ABC** |
| Person-Social-Family | PER-PER | **Yoko**'s husband **John** |
| ORG-AFF-Founder | PER-ORG | **Steve Jobs**, co-founder of **Apple**... |

Table 2.5: Semantic relations with the involved named entity types [38].

From the previous table, factoids such as state = "Tennessee" and founder = "Steve Jobs" can be extracted and turned into associations - *located-in* and *founder-of*, in this order. Another alternative is to follow a metalanguage named Resource Description Framework (RDF) that represents the relations through a **triple**, a tuple composed of a subject, predicate, and object. Triples represent information in the form of relations among data elements and can interpret, without losing context, a sentence in fewer words.

Three main approaches are capable of performing relation extraction, and each is dependent on a specific circumstance:

  - Relation Extraction via **Supervised Learning**: when is available a hand-annotated training corpus with the entities and relations, and a trained classifier for searching pairs of related entities.
  - **Semisupervised** Relation Extraction via Bootstrapping: when seed tuples or patterns are available. The Bootstrapping process is defined by utilizing entities into the seed pair and finding sentences containing those entities.
  - **Unsupervised** Relation Extraction: when no labeled data is provided, Open IE, for instance, is responsible for the extraction of relation tuples from segments of text [8].

- **Word Sense Disambiguation**

  In the majority of the cases, words share more than one meaning or sense, and those subtle distinctions indicate the paramountcy on choosing the correct interpretation of each word in order to perceive the semantics of an entire sentence. The designated formal process of correctly interpreting the contextual meaning of words is understood as word sense disambiguation. Word sense disambiguation is the problem of identifying the intended sense of each word token in a document [25]. Consider the following examples:

*The little girl saw a **bear***

*Your efforts will **bear** fruit*

These sentences are ambiguous because they contain the word "bear" that has multiple meanings, or senses.

- **Sentence Embeddings**

  Sentence Embeddings are representations of sentences through vectors of real numbers. Such representation becomes very handy especially for tasks in the language understanding domain like similarity and sentiment analysis. In order to obtain these representations a straightforward and clear option is to use word embeddings as a proxy for a sentence embedding approach. To be more precise, calculate the embeddings for a specific sentence based on the embedding of each word. For instance, the approach known as Smooth Inverse Frequency (SIF) applies a weighting function to word embeddings to obtain the final representation [9]. However, these types of approaches are associated with some problems such as the incapacity to capture the semantic meaning of a sentence or ignoring word order.

  In chapter 5, this topic will be analyzed more deeply.

### 2.3.4 Pragmatic Analysis

After completing the lexical analysis, checking the grammatical rules through the syntactic analysis, and extracting the true meaning of the sentence, it is time to study the relationship between language and context of use. This analysis consists of deriving and reinterpreting several aspects of language in order to understand the real intentions, plans, or goals present in a text. For example, for the following sentence:

*Andre saw Pedro in a garden with a dog.*

In this case, we have multiple interpretations, and we cannot say whether Andre is with the dog or, instead, Pedro is with the dog. To infer who is with the animal, we require real-world knowledge and appropriate context. Another example but this time associated with ambiguous references in noun phrases.

*The X is writing a novel.*

The above sentence requires resolution of the encountered reference "The X" with pragmatics. As sustained, the majority of text's meaning does have to do with the context in which a term appeared. Ambiguity, and limiting ambiguity, are at the core of natural language processing, and that is the reason why pragmatic analysis is critical for extracting meaning. Without this level, an NLP system may correlate improbable or even unwittingly incompatible attributes/relations to such an object or term [19].

### 2.3.5 Discourse Analysis

Natural language does not consist of unrelated sentences which can be interpreted isolated. Rather, it consists of structured and coherent groups of sentences. The discourse level of NLP considers these groups as a whole that conveys meaningful connections between. Different types of discourse processing can occur at this level, being **anaphora** resolution and **coherent** discourse two of the most common. Consider this example from [34].

*John took a train from Paris to Istanbul.* ***He*** *likes spinach.*

*Jane took a train from Paris to Istanbul.* ***She*** *had to attend a conference.*

We can infer from this sequence of sentences that they require resolution of the anaphoric terms "she" and "he" in order to associate them with an object and a relation. The appearance of anaphoras indicates how discourse is constructed and maintained.

Furthermore, inherited the perspective of coherent discourse, for the first example, it is unclear what is the connection between a train trip and the appreciation for spinach. In the opposite, for the latter case, there is a coherence relation between the sentences that connects them. More explicitly, a reason that justifies the necessity of Jane taking a trip.

### 2.3.6 NLP Applications

As stated earlier, although the many challenges and complex tasks ahead that NLP will face, it is a field developing at a remarkably quick pace and reaching high levels of advancement. Large companies are exploiting this technique, taking profits of the numerous applications that exist, such as:

**Machine Translation**: focused on translating text from one language to another, this field has the potential to transform society by facilitating communication between people anywhere in the world. However we are still far from translation systems that match the nuance and depth of human translations [25]. The major challenge of Machine Translation (MT) is the language barrier, that is, the multitude of languages with totally different grammar and dialects while maintaining intact the meaning of sentences after being translated.

**Speech Recognition**: ability to convert sound waves to individual letters and ultimately sentences. With successfully obtaining a text dictation of a person's speech, these systems become a real vantage for home automation, virtual assistants, hands-free computing, and so on. The current use of speech recognition is still limited due to noisy acoustic environments and systems for natural, free-style speech, preventing achieving a recognition accuracy close to the human-like performance [82].

**Sentiment Analysis**: this automatic process allows businesses to determine how their customers are reacting (positively, negatively, neutral) and what they are saying about a product or service launched and, simultaneously, extract valuable insights that lead to profitability [16]. The growth of sentiment analysis coincide with those of the social media on the web, for example, forum reviews, blogs, and social networks like Instagram and Twitter.

**Question Answering**: these systems automatically answer different templates of questions posed in a natural language. Question Answering (QA) can be split into two paradigms. The first, **information-retrieval** question answering, has the goal of find text segments or passages obtained in documents or web and draw a response immediately. While the second, **knowledge-based** question answering, map the natural language into a query over a structured database [38].

**Information Extraction**: with the current information overload in the form of news, social media, and corporate files, for example, the automatic extraction of information embedded from unstructured sources opened a window for searching, organizing, and analyzing data by "drawing upon the clean semantics of structured databases and the abundance of unstructured data" [73].

The **Summarization** application will be more in-depth and detailed in the succeeding sections.

## 2.4 Automatic Summarization

Recently, we are witnessing an exponential increase in the availability of online information spread from various sources. This mass of text is a valuable origin of data and knowledge which needs to be accessed and digested adequately to be useful. These concerns have provided influential impetus to the development of automatic summarization systems accountable for producing a coherent, fluent and condensed summary, conserving key content and meaning.

Text summarization can be used for multiple domains, for instance, a cluster of news and journals, scientific articles, meeting recordings, contact centers dealing with millions of calls per day, legal contract analysis, and so on. Despite these advantages that enhance the use of automatic systems, there are yet some hurdles to take into account and give particular attention. The computer's lack of language capability, cohesion in text, information significance, detecting redundancy in information, sentence ordering, and the dependency in use cases make very challenging and non-trivial the task of generating summaries.

### 2.4.1 Summarization Factors

There are many types of summarization according to certain criteria, which are grouped and classified by the following factors: input, purpose, and output [36].

- **Input Factors**: these factors can be categorized according to table 2.6.

| | **Type** | **Description** |
|---|---|---|
| **Number** of input documents | Single Document | Can accept only one document per summary, whether a news story, scientific article, etc. |
| | Multi Document | Multi-document summarization was driven by use cases on the web. Given the amount of redundancy online, summarization was often more helpful if it could provide a brief digest of several documents on the same topic or the same event. In the first deployed online systems, multi-document summarization was applied to clusters of news articles on the same event [58]. |
| **Language** | Mono Lingual | Input text only allows documents with a specific language and the generated summary created on that language. |
| | Multi Lingual | Can summarize input documents with distinct languages and produce summary of different languages, but it is difficult to implement because of the various rules of grammar and semantics to deal. |

Table 2.6: Input factors with the corresponding types and descriptions.

- **Purpose Factors**: in contrast with the input factors, the purpose factors are distinguished based on the following three levels displayed 2.7.

| | Type | Description |
|---|---|---|
| **Details** | Informative | With the difference of being shorter and without redundant information, this summaries can be read in place of the original text. |
| | Indicative | It presents only the main idea of the original text and enables the reader to figure out if that text is worth reading. |
| **Content** | Generic | Generic summarization makes few assumptions about the audience or the goal for producing the summary. Typically, it is assumed that the audience is a universal one: anyone may end up studying the summary. Furthermore, no assumptions are made about the genre or domain of the elements that need to be summarized [58]. |
| | Query-based | Summarize solely the information in the input document(s) that is related to a specific user query. For instance, in the context of information retrieval, "given a query issued by the user and a set of relevant documents retrieved by the search engine, a summary of each document could make it easier for the user to determine which document is relevant" [58]. |
| **Limitations** | Domain Dependent | Restraining the input text to certain knowledge domains such as medicine, biology, etc. |
| | Genre Specific | The input text is restricted to accept only a specific structure of text like newspaper articles or instruction manuals. |

Table 2.7: Purpose factors with the corresponding types and descriptions.

- **Output Factors**: regarding this type of factors, there are two approaches for summarization: the summarization by **extraction**, which consists of concatenating and reusing certain snippets of the original input text as illustrated in the figure 2.12; and summarization by **abstraction**, which involves creating a summary with novel and shorter sentences that describes the same content of the original text.

  Despite the emergence of abstractive techniques, extractive techniques are still attractive as they are less complex, less expensive, and generate grammatically and semantically correct summaries most of the time [57].

Figure 2.12: Illustration of Extractive Multi Document Summarization [39]

### 2.4.2 Early Work

Automatic text summarization started gaining recognition at the beginning of the 1950s. Most early works focused on approaches that primarily had the purpose of identifying important content at the sentence level and worked on technical papers. Luhn, considered one of the pioneers of this subject, proposed that the "significance" factor of a sentence derives from an analysis of its words. He introduced a method to extract and score sentences from the text, applying a combination of measurements like frequency of word occurrence in an article and the relative position within a sentence of significant words [48].

Edmundson's [24] work was the foundation of several other trends in summarization research, which eventually led to machine learning approaches in summarization [58]. He expanded on Luhn's approach by adding another combination of features:

- Cue Method: the presence or absence of certain words matching a pre-compiled list of cue words.
- Title Method: the sentence weight is measured as the sum of all the content words that appear in the title or in section headings.
- Location Method: sentences appearing at a certain position or section of the text have a higher probability of being relevant.

Since then, countless approaches have been published to address the problem of automatic text summarization. Next, recognized methods will be presented following different based strategies.

- **Unsupervised Data-driven Methods**

  Unsupervised methods for sentence extraction are the prevailing paradigm in extractive summarization. Labeled data in the form of human annotations for training statistical methods, for instance, is not required. They are "data-driven" because they do not rely on any models or linguistic processing and interpretation [58].

**Term Frequency - Inverse Document Frequency (TF-IDF)**

This weighting technique assesses the relative frequency of words in a particular document, then comparing with the inverse proportion of those words but concerning the collection of every document. That is, it can determine if a word is just a frequent word or a meaningful one. Assuming that *c(w)* will be the number of appearances of a word $w$ in a given document, $D$ the total number of documents and *d(w)* the number of documents containing the word $w$, the TF-IDF value is calculated as follows:

$$TF * IDF = c(w) * log\frac{D}{d(w)} \tag{2.2}$$

**Clustering**

Clustering is an automatic text summarization technique used mostly for documents containing more than one topic. Radev, Jing, Stys, and Tam [65] proposed an approach that produces summaries using cluster centroids and topic detection. This last one is responsible for clustering together news articles that describe the same topic, and by generating TF-IDF vector representations of the documents. Then, an algorithm operates over the TF-IDF vectors, successively adding documents to clusters if the similarity measure between those TF-IDF values and the centroids is within a threshold. Following this step, this method uses the centroids to distinguish sentences in each cluster that are central to the topic of the entire cluster. For this to be accomplished, the authors defined two metrics: cluster-based relative utility (CBRU) and cross-sentence informational subsumption (CSIS). The first metric measures how relevant is a particular sentence to the general topic of the entire cluster, while the second estimate the redundancy amongst sentences.

**Graph-Based**

A graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information [54]. Web Page ranking algorithms such as *PageRank* or *TextRank* can be applied in summarization, where the graph is assembled by adding a vertex for each existent sentence in the text. Using sentence inter-connections, vertices are connected by weighted edges according to a similarity relation and later, sorted by their scores.

- **Machine Learning Methods**

  ML approaches can model the summarization as a classification problem categorizing sentences into two classes — include or exclude from summary — through a set of features, given a training and testing set of documents.

  **Naive Bayes**

  Kupiec describes a method using a Naive Bayes classifier that determines the probability of a sentence to be included in an extract. New extracts can then be generated by ranking

sentences according to this probability and selecting a user-specified number of the top-scoring ones [43]. Considering a particular sentence $s$ and $k$ features assumed independent, the system will estimate if that sentence will be included in a summary $S$ using Bayes' rule in equation 2.3:

$$P(s\epsilon S|F_1, F_2, ..., F_k) = \frac{\prod_{i=1}^{k} P(F_i|s\epsilon S) \cdot P(s\epsilon S)}{\prod_{i=1}^{k} P(F_i)} \tag{2.3}$$

There are a set of features per sentence, which are all discrete:

1. Sentence Length Cut-off Feature: given a settled threshold (e.g., 6 words), the feature has the value true for all sentences longer than the threshold
2. Fixed-Phrase Feature: sentences including key phrases or occurring instantly after a section heading that contains a keyword (e.g., "conclusions", "discussions", etc) tend to be included in summaries
3. Paragraph Feature: sentences are categorized according to their position in a paragraph
4. Thematic Word Feature: the most frequent content words are named thematic words, and few of them are selected. Sentences are classified based on the frequency of those words.
5. Uppercase Word Feature: sentences are ranked according to the frequency of uppercase words present in the document

**Log-Linear Model**

Osborne [59] used maximum entropy modeling for sentence extraction. The parametric form of this model can be declared in equations 2.4 and 2.5:

$$P(c|s) = \frac{1}{Z(s)} exp\left(\sum_i \lambda_i f_i(c, s)\right) \tag{2.4}$$

$$Z(s) = \sum_c exp\left(\sum_i \lambda_i f_i(c, s)\right), \tag{2.5}$$

where $c$ is a label from the set $C$, $s$ the item that will be labeled, $fi$ the features applied, and $\lambda i$ the corresponding feature weight. $C$ consists of two types of labels: "keep" and "reject" meaning if the sentence is to be extracted or it is not. Also, to find the optimal set of weights the conjugate gradient descent was used.

**HMM**

Conroy and O'Leary [22] determined how a HMM can be applied for summarization. Only three features were considered: the position of a sentence in the document, number of terms in a sentence, and likeliness of the sentence terms given the document terms. The authors also deliberately exploit Markov dependencies: they suggest that the probability that the next sentence in a document is included in a summary will depend on whether the current document sentence is already part of the summary [58].

- **Methods using Semantics and Discourse**

  Language-dependent methods either rely on manually composed semantic resources, knowledge concerning lexical items, or coreference tools.

  **Lexical Chains**

  Lexical chains are sets of semantically related words spread over an entire text. Computed by the surface level analysis of a text, they can identify the topic of a document provided or identify the sense of a certain word in a given context. For instance, a set composed of the words: "students", "lecture", "science" can represent a specific topic or a context. These chains, with the advantages mentioned, have the power to generate a cohesive summary for a document. The process of obtaining lexical chains relies heavily on *WordNet* [56], a manually compiled thesaurus which lists the different senses of each word, as well as word relationships such as synonymy, antonymy, part-whole and general-specific [58].
  Barzilay and Elhadad [11] presented a summarizer that contained the following steps: segmentation of the text analyzed, identification of lexical chains, and sentence selection using strong lexical chains. Each strong chain is associated with an aggregate score representing how valuable it is.

  **Latent Semantic Analysis (LSA)**

  LSA is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other [44]. This method is applied to cluster similar documents, and to obtain the most relevant sentences in the document.

  **Rhetorical Structure Theory (RST)**

  RST requires the text to follow a fully connected discourse structure that can be represented by a tree. In RST trees, the tiniest units of text analysis, are designated elementary discourse units (EDU), which are mostly sub-sentential clauses. This approach is based on finding rhetorical relationships between two non-overlapping pieces of text spans: the nucleus and the satellite. The first is considered more essential representing important information in a text, whereas satellites contribute to the nucleus containing additional information about it.

- **Neural-Based Methods**

  Since the bloom of DL and ANN, neural-based summarizers have captivated noteworthy attention for automatic summarization. Unlike previous mentioned traditional models, with plentiful training data, these deliver better performance with less human involvement and can also either be abstractive or extractive.

  Most neural approaches general use the following pipeline:

  1. with *BERT* or *GloVe*, for example, word embeddings, words from the vocabulary that are mapped to vectors of real numbers, are produced;

2. with CNN or RNN as encoders, sentences/documents are represented as continuous vectors based on the word embeddings;
3. sentence/document representations are then fed to a model for selection-based extraction or generation-based abstraction, where neural models act like regressors for ranking or decoders.

Kågebäck proposed a method that leverages the use of continuous vector for sentence representation as a basis for similarity measuring. Through the addition of word embeddings from Collobert and Weston's model, or using an unfolding Recursive Auto-encoder (RAE), taught in an unsupervised way by the backpropagation method, on word embeddings, were obtained the continuous vectors [39].

Meantime, Cao and his associates developed a method, PriorSum, which applies CNN to obtain the prior summary independent features. The three features applied were: sentence position, averaged term frequency of words in a sentence based on the document, and averaged term frequency of words in a sentence based on the cluster. About CNN, the layers were composed of multiple pooling and convolution operations, and the filters had different window sizes [17].

### 2.4.3 Evaluation

Evaluating a summary is complex and challenging task, mainly due to lack of agreement or consensus in the research community on what an ideal summary should resemble. Also, there are other serious hurdles to be knowledgeable of:

- Widespread application of diverse metrics. There is an absence of a standard human or automatic metric.
- Approaches assessing the quality of the summaries by human evaluators are expensive, time-consuming, and the obtained results are subjective and inconsistently.
- Cases where it is hard to arrive at a concept of what the correct output is. For example, it is feasible that a system generates a good summary that is moderately different from a human-generated summary.
- The increase in scale and complexity evaluating summaries at different compression rates.

Text Summarization evaluation methods can be broadly classified into two categories. First, an intrinsic evaluation operates on the characteristics of the summary itself, assessing mainly the coherence and informativeness [29]. The second, an extrinsic evaluation, tests the quality of a summary based on how helpful it can be for a specific task like relevance assessment or reading comprehension.

Next, some standard intrinsic automatic methods are reviewed.

**Recall-Oriented Understudy for Gisting Evaluation (ROUGE)**

Lin [46] proposed a recall-oriented work that measures how much the words in the machine generated summaries appeared in the human reference summaries. It became the most commonly used metric for automatic evaluation between researchers due to its speed plus cheapness. There are several varieties of ROUGE, and here only the most broadly used will be mentioned:

- **ROUGE-N**: measures the overlap of n-grams between the candidate summary and the reference summaries
- **ROUGE-L**: employs the notion of the longest common subsequence (LCS) between two series of text. The longer the LCS between two summary sentences, the more alike they are
- **ROUGE-S**: calculates the overlap of skip-bigrams between two summaries. Skip-bigram means any pair of words in a sentence in the order they appear, allowing for arbitrary gaps

Some variants were later extended to consider both the recall and precision. Also, stemming and stopword removal can be regarded.

**Bilingual Evaluation Understudy Score (BLEU)**

On the other hand, BLEU [61] is a precision-oriented metric measuring how much the words in the human reference summaries appeared in the machine generated summaries. It includes a brevity penalty that penalizes machine-generated results which are shorter than the general length of a reference.

For intrinsically evaluating a summary, other notable metrics are precision, recall, and F-measure [76]:

- **Precision**: the ratio of sentences occurring in both machine-generated and ideal summaries to the number of sentences in the machine-generated summary
- **Recall**: the ratio of sentences occurring in both machine-generated and ideal summaries to the number of sentences in the ideal summary
- **F-Measure**: weighted average of Precision and Recall

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.6}$$

# Chapter 3

# State of the Art

This chapter intends to present, in the first section, closely related work covering our approach domain or subphases included, followed by an analysis of resources and tools that can help in the building of several tasks related to summarization and information extraction and to the implementation of this project. Additionally, the final section performs an outline of the applications/products available on the current market that compete with the purposed approach of this internship. It represents a critical step in a project as this competitor's study may unveil if it is reasonable to proceed with the proposal and consequently any likelihood of success.

## 3.1 Related Work

Several papers about information extraction or summarization in the context of dialogue conversations were analyzed. Since they are related to our approach pipeline sub-steps, it is important to comprehend how they developed these components. Performing this review has the ultimate goal of understanding which approaches have already been evaluated and considered the most relevant. Prior to this step, a strategy constituted of two criteria was followed to select and shorten the list of possible articles or scientific papers most appropriate to an accurate review. Firstly, there was a focus on articles accepted by relevant conferences such as the Association for Computational Linguistics (ACL), a premier conference in the field of computational linguistics. Ultimately the number of citations linked was a key factor in the criteria of acceptance.

In [64], the authors presented the GECOR, an end-to-end generative model responsible for the resolution of ellipsis and co-reference in multi-turn dialogues. Ellipsis and co-reference occur regularly due to rhetorical principles such as saving words and avoiding repetitions but leave utterances pragmatically incomplete if separated from context. While humans understand and recognize utterances with anaphorically referenced or absent information, as displayed in figure 3.1 (e.g., Q2 and Q3), dialogue systems fail to discern such

utterances correctly. According to the dialogue context existent, to achieve the goal of automatically supplement user utterances with information that is left out or substituted by anaphora, the model applies a sequence-to-sequence method with two encoders, for reading the user utterance as well as the dialogue context, and a decoder that generates the complete utterance. During the implementation phase of our work, anaphora situations are also treated to associate entities with attributes or relations correctly. However, this process, integrated with the proposed platform, is only one step to reach our final goal of automating the template generation process.

| **Turn** | **Dialogue** |
|---|---|
| $Q_1$ | I would like an Italian restaurant. |
| $A_1$ | What price range do you have in mind? |
| $Q_2$ | I want cheap ones. |
| $A_2$ | Pizza Hut Cherry Hinton serves Italian food in the south part of town. Would you like their phone number? |
| $Q_3$ | Yes, please. |
| **User utterances after resolution** | |
| $Q_2$ | I want cheap Italian restaurants. |
| $Q_3$ | Yes, I would like the phone number please. |

Figure 3.1: Examples of ellipsis and co-reference resolution [64].

Regarding [81], the author's approach, a novel scheme of spoken dialogue systems that use up-to-date knowledge of the web, is based on information extraction defined by a predicate-argument structure and realized by semantic parsing operation. The process of information extraction starts by converting a natural language web text, and the user's query, into a machine-understandable representation. The system then matches the extracted information against the domain corpus. According to the groundtruth, the system either answers the user's question or returns a proactive presentation that matches partially the most relevant information of the user's request.

| | Conversations | User Attributes |
|---|---|---|
| *Usr* | Hello, how are you doing today? | none |
| *Sys* | I am fine! Where do you live? | |
| *Usr* | I am originally from California but now I live in Florida for long. | (I, live_in, Florida) |
| *Sys* | Florida! You must have a good work-life balance. | |
| *Usr* | Oh, I no longer work at banks but for exercise I walk often. | (I, previous_profession, banker)<br>(I, has_hobby, walking) |
| *Sys* | Good to hear that! Do you live with your family? | |
| *Usr* | My son. I bring him to church every Sunday with my Ford. | (I, has_children, son)<br>(I, like_goto, church)<br>(I, have_vehicle, ford) |
| *Sys* | Wow sounds good! You can meet many people. | |
| *Usr* | Sure, but my son is afraid of talking to others. | (My son, misc_attribute, shy) |

Figure 3.2: Example of dialogue between a user and a system, with the user attributes column the potential extracted information [80].

In this paper [80], the researchers leverage dialogues between conversational agents to automatically extract user attributes. User attributes can be defined as structured representations or descriptions of a person's identity. The project's purpose is to predict user information and represent that as a triple format (subject, predicate, object). For instance, in figure 3.2, the triple (I, live-in, Florida) is extracted from the second user utterance. A pipeline of three steps composes this information extraction task. The first one uses a context encoder to capture utterance semantics, using a sequence of word embeddings as input. Then the predicate prediction task takes actions by determining whether there is a predicate triggered by a user utterance. In case there is a triggered predicate, the last step, the entity generation task, performs and generates the subject and object phrases to complete the entire user attribute. In work developed, triples are also extracted from dialogues and used to represent the utterances and the relevant information contained on them. Again, this process is integrated into the proposed platform, which covers many other steps towards our final goal.

In [51], the authors suggested a novel online call scene segmentation method for dialogues occurring in call centers between an agent and a customer. Advantageous for subsequent tasks such as summarization and information retrieval, they define call scene segmentation as "utterance-level sequential labeling" with the following tag types: opening, requirement confirmation, response, customer confirmation, and closing. Capture interactions among the agent and the customer are the main goal of this proposal. In order to execute this step, a fully neural network-based method processes hierarchical LSTM-RNN that handles both the sentence-sequence and the speaker role sequence with the advantage of capturing long-range interactive information.

Raheja V. and Tetreault J. [66], in turn, developed a context-aware self-attention mechanism united with a hierarchical recurrent neural network, for dialogue act classification across multiple domains. That

is, each utterance is linked with a Dialogue Act label describing a function. The authors affirmed that knowing the Dialogue Act labels of the previous utterance is valuable for help in the prediction of the current Dialogue Act. The model used involves three major components: a RNN which role consists in encoding the information within the utterances at both word and character-level; a context-aware mechanism aggregating word representations into utterance representations; and a conversation-level RNN working on the utterance encoding output of the previous mechanism.

| DA type | speaker | sentence |
|---|---|---|
| statement-opinion | B | I always kind of think it would be neat to be able to watch them and be there for them all the time |
| back-channel | A | Uh-huh |
| question-yes-no | B | Is that what you do? |
| yes-answer | A | Uh yeah |
| statement | A | Actually I teach my kids at home |
| statement | A | so I'm here all the time |
| summarize/reformulate | B: | Oh so they don't go to school |

Figure 3.3: Example of a conversation with the Dialogue Act labels [47].

Finally, [47], has the same goal as the previous one, dialogue act classification with each utterance in a conversation associated with a predefined Dialogue Act tag set. Figure 3.3 exhibits the different types of tags that exist. All the methods proposed by the authors incorporate context information with various baselines, including a hierarchical RNN/LSTM and CNN to model the utterance sequence in the conversation, CNN vectors for input classification, and finally, sequence-level decoding based on the foretold Dialogue Act probabilities.

The last three enounced papers were all related to a sub-task integrated on our pipeline, intent classification. These intents are relevant to our approach because they can distinguish if a dialogue utterance represents an important value or if it can be discarded and minimize the possibility of non-relevant information enter on the final templates.

## 3.2 Libraries and Frameworks as Support Tools

The magnitude of tools accessible today and the increasing demand by the companies and researchers trying to exploit these technologies, has led to a leap forward in the field of artificial intelligence. Nevertheless, it is vital to identify the most useful and serviceable libraries and frameworks for the development of this project.

These tools were carefully chosen following certain restrictions at different levels. At the first level, they were selected based on certain factors defined by the company: must be able to be deployed on-cloud, the need to support the English language, feedback from the community, the last release, as well as

the amount of enriching documentation available on their platforms—more details of these restrictions in section 4.3. Simultaneously, the tools are required to be capable of helping in tasks dealing with computational approaches and components to natural language or machine learning applications like clustering, classification, and regression without jeopardizing the performance or computational resources.

A shortened list of the libraries and their description follows.

- **Apache OpenNLP**[1]: Java open-source ML-based toolkit for the processing of natural language text. The last stable release dates from 2018, and it is available in multiple languages. Provides modules for tasks such as tokenization, named entity recognition, coreference resolution, language detection, and speech recognition.
- **FreeLing**[2]: C++ open-source configurable linguistic analysis tool suite providing functionalities such as morphological analysis, Word Sense Disambiguation, and Semantic Role Labelling, for a variety of languages. FreeLing, stable released in 2016, also renders a command-line front-end that analyzes documents and distinguish exporting the output in a particular wanted format (e.g., JSON, XML, and CoNLL).
- **MITIE**[3]: built on top of a high-performance toolkit containing ML algorithms, dlib [40], this library makes use of state-of-the-art information extraction tools like distributional word embeddings, named entity extraction and Structural Support Vector Machines [35]. MITIE, which uses a variety of linguistic resources such as Wikipedia and Gigaword, supports diverse pre-trained models for particular languages and software languages, including Python, R, Java, C, and MATLAB.
- **NLTK**[13]: a leading open-source platform for Python applications in natural language and computational linguistics. With the latest stable version in 2019, that offers over 50 corpora and lexical resources such as WordNet, as well as several text processing libraries for classification, semantic reasoning, among others. It also intends to support research in the field of NLP with his open-source modules and tutorials.
- **SpaCy**[4]: Python open-source library meant to develop natural language understanding systems and integrate or preprocess text for deep learning by interoperating seamlessly with TensorFlow, Keras, Scikit-learn, or PyTorch. Stable released recently this year includes modules such as named entity recognition, PoS tagging, labeled dependency parsing, and syntax-driven sentence segmentation.
- **TextBlob**[5]: stable released in 2020, it is a Python library for processing textual data that offers a flexible and straightforward API to access its methods and perform standard NLP tasks. Meantime, provides an easy to use interface to the NLTK library. It includes several features such as PoS tagging, tokenization, parsing, n-grams, spelling correction, and word inflection.

[1] http://opennlp.apache.org/
[2] http://nlp.lsi.upc.edu/freeling/
[3] https://github.com/mit-nlp/MITIE
[4] https://spacy.io/api/doc
[5] https://textblob.readthedocs.io/en/dev/

- **SciPy**[6]: Python open-source library used for mathematics, science, and engineering with his last stable release in 2019. Built to work with NumPy arrays, provides routines for clustering, optimization, statistics, and linear algebra.
- **LingPipe**[7]: a paid Java toolkit for processing text and gathering data using computational linguistics. Stable released in 2016, it is responsible for performing tasks such as detecting the names of people, organizations, or locations in the news. It will also classify and categorize automatically search results of the platform Twitter, and suggest spelling corrections.
- **Scikit-learn**[8]: the last stable released occurred in 2020. A complete Python open-source ML library intends to process features such as classification, regression, dimensionality reduction, model selection, preprocessing, and clustering algorithms. Capable of interoperating with the scientific library SciPy.

Despite all of the following frameworks identified having suitable features for this proposal, the majority of them will not be applied in the product developed during this internship due to compliance with certain restrictions imposed. Regarding frameworks, high cost charged, and the fact of these services are external to the company are acknowledged as the main impediments to their usability. However, it is relevant to grasp how they operate. Plus, these tools can become useful through the testing stage by working as comparison metrics when we reach some results.

- **Google Cloud Natural Language API**[9]: provides powerful ML models to reveal the structure and meaning of text. This API allows the following operations:
  - Extraction of multiple classes of entities within a document and label them by types such as a person, location, events, and much more.
  - Understand the overall sentiment or feeling expressed in a segment of text.
  - Perform a syntax analysis for a block of text.

  All the above features are made available via a REST API that supports multiple languages.
- **Microsoft Azure Cognitive Search**[10]: constituted by cloud-based services that contribute to the implementation of some advanced NLP tasks:
  - The Text Analytics API's Sentiment Analysis feature applies a machine-learning classification algorithm to evaluate a document or a sentence and return a sentiment score between the values 0 and 1. Scores proximal to 1 indicate positive sentiment. This feature is useful when detecting sentiment in social media posts, customer reviews, and more.

[6] https://www.scipy.org/
[7] http://alias-i.com/lingpipe/index.html
[8] https://scikit-learn.org/
[9] https://cloud.google.com/natural-language/

  - The Text Analytics API's Language Detection feature evaluates text input and detects a wide variety of languages according to some identifiers. Returns a score that indicates the strength and confidence of the analysis.
  - The Text Analytics API's Named Entity Recognition feature detects different entities in text. When the occurrence of an entity with double meaning (e.g., "Mars") happens, the system disambiguates that entity by associating text to supplementary information on the web.
  - The Bing Search API's Spelling Correction feature is capable of correct word-breaking issues, spot common name errors and fix homonyms in a given context, and detect slang language.

- **Stanford's CoreNLP**[11]: integrated framework for application development and NLP research. It is extensible, multilingual, and open-source, unlike the remaining toolsets. Stanford CoreNLP can run as a simple web service and integrates all Stanford NLP tools, including the POS tagger, the parser, the coreference resolution, sentiment analysis, and information extraction tools.

- **Amazon Comprehend**[12]: uses NLP to uncover insights about the content of documents, as well as finding relationships in the unstructured data. It includes, among others, the main features: keyphrase extraction, sentiment analysis (identify negative reviews or estimate how well are the interactions between customer and service agents), and topic modeling. This framework distinguishes itself from the others due to his continuous and consistent learning from information sources such as Amazon.com product specifications and consumer evaluations.

To complete this section tables 3.1 and 3.2 display the libraries and frameworks mentioned above and their most relevant features, respectively. Those tables include information about their software licenses (paid or free) and the programming languages supported. Concerning this approach's domain, all these services handle features relevant and helpful during the implementation phase. However, our priority is to keep up with the imposed restrictions, as previously mentioned. Therefore the supported tools must possess all or the most part features without applied charges associated, for instance.

Regarding the performance of these tools, despite various have similar modules, it was crucial to analyze and effectuate a comparable study capable of distinguishing which of them are the preferable and go-to tools. The justification for this comparative study focus on our primary goal: to implement an efficient and friendly framework. Accordingly, for the following libraries, it was obtained an aggregate of comparisons from different papers and repositories. Simultaneously, since Talkdesk already deals and processes with these modules, we were capable of having critical feedback of which performs better under certain circumstances.

[10] https://azure.microsoft.com/en-us/services/cognitive-services/
[11] https://stanfordnlp.github.io/CoreNLP/
[12] https://aws.amazon.com/comprehend/

| | Apache OpenNLP | FreeLing | MITIE | NLTK | SpaCy | TextBlob | SciPy | LingPipe | Scikit-learn |
|---|---|---|---|---|---|---|---|---|---|
| **License** | Free | Free | Free | Free | Free | Free | Free | Paid | Free |
| **Programming Language** | Java | C++ | Multi | Python | Python | Python | Java | Java | Java |
| **Lemmatization** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **N-Grams** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Sentence Segmentation** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Stop Words** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Spell Checker** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Stemming** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Tokenization** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Chunking** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Constituency Parsing** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Dependency Parsing** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **NER** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **POS Tagging** | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| **Clustering** | ● | ● | ● | ● | ● | ● | ● | ● | ● |

Table 3.1: Libraries and its features.

| | Google Cloud Natural Language API | Microsoft Azure Cognitive Search | Stanford's CoreNLP | Amazon Comprehend |
|---|---|---|---|---|
| **License** | Paid | Paid | Free | Paid |
| **Programming Language** | REST API | REST API | Multi | REST API |
| **Lemmatization** | 🔴 | 🔴 | 🟢 | 🔴 |
| **N-Grams** | 🟢 | 🔴 | 🔴 | 🔴 |
| **Sentence Segmentation** | 🟢 | 🟢 | 🟢 | 🔴 |
| **Stop Words** | 🔴 | 🔴 | 🔴 | 🔴 |
| **Spell Checker** | 🔴 | 🟢 | 🔴 | 🔴 |
| **Stemming** | 🔴 | 🔴 | 🔴 | 🔴 |
| **Tokenization** | 🟢 | 🔴 | 🟢 | 🟢 |
| **Chunking** | 🔴 | 🔴 | 🔴 | 🔴 |
| **Constituency Parsing** | 🔴 | 🔴 | 🟢 | 🔴 |
| **Dependency Parsing** | 🟢 | 🔴 | 🟢 | 🔴 |
| **NER** | 🟢 | 🟢 | 🟢 | 🟢 |
| **POS Tagging** | 🟢 | 🔴 | 🟢 | 🟢 |

Table 3.2: Frameworks and its features.

## 3.3 Competitors

A competitor analysis constitutes a crucial and initial step during the development of a project. Once it is established the vision of the product, and the problem it is attempting to solve, becomes imperative to understand all the available alternatives - direct and indirect competitors - and gather information about them. Direct competitors offer a similar product regarding the one whom we will implement within the same industry. In contrast, indirect competitors advance a distinct type of product within the identical need. This way, we can anticipate shifts in the market, establish a sustainable competitive advantage, and show the stakeholders the viability and innovation of our proposal.

### 3.3.1 Comparative Analysis

**Dialpad**

The exclusive direct competitor here presented was founded in 2011 and headquartered in San Francisco, California, is a cloud-based platform for enterprise communications that operates across all personal devices. Dialpad offers voice calling, video conferencing, text messaging, and screen/document sharing services and provides a suite of advanced tools designed to help the streamline of the business communication, including a custom Interactive Voice Response (IVR), and Google Apps integration. The company also automatically provides a call summary in the form of actions and snippets, moments, and transcript access employing NLP. They provide two products using these features: one oriented for call summaries and another towards meetings and conferences [3].

**Voicea**

Based in San Jose, California, Voicea was founded in 2016 and then acquired by Cisco Systems. It is a voice collaboration platform powered by an Enterprise Voice Assistant named EVA, which leverages AI to facilitate the productivity of meetings by making these last ones searchable. It captures the highlights and shares them to create actionable recaps. The meetings are automatically summarized within two types, notes or action items (e.g., action schedule a meeting at 3 pm) and can be shared with collaborators via Slack or Salesforce [7].

**RaeNotes**

Recently endowed in the past year in Boston, MA, is an AI browser-based voice processing platform. Converts voice data toward a searchable source of knowledge by providing automated video/audio transcriptions, media content management, high-value search, and collaboration for meetings, interviews, talks. RaeNotes extracts topics like decisions, questions, action items addressed in meetings, as well as provides sturdy structure and mark-up aptitudes to annotate the transcript with own categories linked with the recording [4].

**Avoma**

Founded in late 2017 in Palo Alto, CA, provides an end-to-end AI Meeting Assistance and Conversation Intelligence solution for customer-facing teams. The company analyzes conversation insights and capture relevant customer information in the moment's pre, during, and post meetings. With Avoma, it is possible to automatically record all the meetings, obtain complete transcripts and summarized notes, and actionable insights synced instantly to Customer Relationship Management (CRM) [2].

**Reason8 AI**

Established in 2016 and with its head office situated in Boston, MA, is a cloud-based plus mobile-oriented speech-to-text solution to automate note-taking and summary preparation from in-person business and scrum meetings. The platform supports multi-devices and is available in iOS and Android. Reason8 AI is currently working on collaboration features for teamwork, high-level summarization, and as well as integrations with project management services/communication tools [5].

**Sonia AI**

With his headquarters located in New York and founded in 2017, Sonia AI is a workflow automation platform in the voice AI space geared, currently, towards helping consultants and financial advisors. Transcribes conference calls and extracts client meeting notes, automates CRM entries, and drafts client follow-ups (this integration allows sending/amending composed templates) [6].

The identified competitors were analyzed according to the following set of features, relevant for this work:

- Summarization by **extraction** or **abstraction**
- Summarization of conversations supported in **multi-languages**
- Summarization of key sentences based on **trigger words**
- Break down conversations into **automated detected moments** such as action items, next steps, recommendations, and requests for cancelation. By definition and in this approach's domain, moments are mainly topics that capture the activity performed by a client or a call center agent. For instance, if a client reaches an agent with a request to reserve a night in a specific hotel, the moment associated is designated as hotel booking
- Review of **snippets** considered quintessential as pointers towards transcript history. Snippets are small segments of a conversation or call that help keeping track of past moments

Table 3.3 summarises the analysis of the coverage of the selected features by the competitors.

| | Dialpad | Avoma | Reason8 | Sonia AI | RaeNotes | Voicea |
|---|---|---|---|---|---|---|
| **Abstractive Summarization** | ● | ● | ● | ● | ● | ● |
| **Multi Lingual Summarization** | ● | ● | ● | ● | ● | ● |
| **Trigger Words Summarization** | ● | ● | ● | ● | ● | ● |
| **Automated Detected Moments** | ● | ● | ● | ● | ● | ● |
| **Availability of Snippets** | ● | ● | ● | ● | ● | ● |

Table 3.3: Competitors and their features.

### 3.3.2 Discussion

From this comparative analysis, it becomes evident the lack of solutions opting for generating novel sentences from information obtained from the corpus, instead of the prevailing approach - directly extract segments/blocks of text from the transcripts into the summary. Current offers are more focused on solutions considered more straightforward, faster to run, and especially **cheaper** to implement.

The analysed systems present all a similar structure: the majority of them relies on conversation elements to compose the summary, rather than truly summarizing into segments or a set of paragraphs. That

is, dialogs are broken down into moments/categories that describe their intent. Regarding this feature, **Dialpad** stands out from the rest and presents the widest spectrum of moment types detecting situations such as an agent apologizing to the customer, a customer indicating if their problem has been solved, price inquiries, and requests for cancelation/refund of an order.

On a final balance analyzing this comparison made and the information obtained by these competitors, with a greater emphasis on the direct competitor Dialpad, it was not easy to position the intern approach solution with the competitor's solution. Since this approach intends to automate the process of summarization calls by automatically generating template summaries, and the competitors do not explicitly publicly, as expected, the pipeline beyond their process for summarization, clarifying the technological and product differentiation factors prevails a challenge. Theoretically and by searching for scientific papers, articles, documentation, or announcements, generating automatic templates remains a process until date not explored by related companies, which translates, in reality, a unique differentiator in this domain. If successfuly developed, the agent after-call work will reduce dramatically, and therefore a product with these capabilities will be advertised at the level of marketing and directly influence the call center market and behavior of the correlated companies.

This page is intentionally left blank.

# Chapter 4

# Approach

The choice of the dataset for this project influenced and directly dictated the selection of the adopted approach. At an early stage, we were inclined to follow an approach based on Deep Learning. Nevertheless, despite the several advantages, following an approach in this field entails specific requirements that we are unable to meet at all. It requires a large quantity of data to train effectively and a substantial amount of computing resources — both conditions considered impracticable to accomplish. Another feature that led to a deviation from the initial plan correlates with the final product proposed to develop in this internship. The main objective was the development of an end-to-end framework responsible for automatic summarization a transcription of a dialogue handled by an agent in a contact center. However, since the Agent Assist team was already dealing and attempting a promising solution, it was therefore agreed that the internship role in this project would be to automate one of the components that compose the process implementation mentioned above. The ultimate proposal, illustrated by figure 4.1, consists of automating the template generation process through the representation of triples describing each moment of the call, which is previously identified and classified. Subsequent to this representation, related and similar triples are then grouped through a clustering algorithm. Ultimately, the information that will be provided with the templates are extracted from the clusters generated. The following sections cover and explain each step of the flow to obtain this final result.

## 4.1 Summarization Approach

There is already an approach being addressed by the Talkdesk team to extract call summaries and that is outside the scope of this project. The approach follows three main components - Moment Identification, Moment Classification, and Template Filling - which are explained in the next sections.

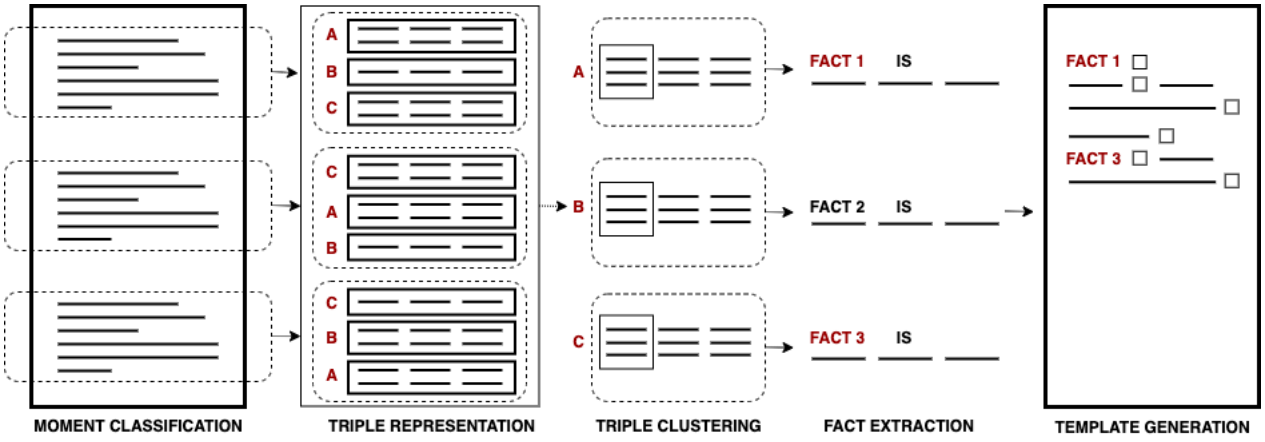


Figure 4.1: Process Flow.

### 4.1.1 Moment Identification and Classification

Moment Identification is the primary phase of the Talkdesk approach pipeline. It is based on the following procedure: existing sentences on the input transcript are split, and each separate sentence is automatically classified with an intent. This sentence classification allows the detection of moment shift passages, and therefore the moment identification. Consequently, the moment classification starts taking action and returning a topic assignment for each moment.

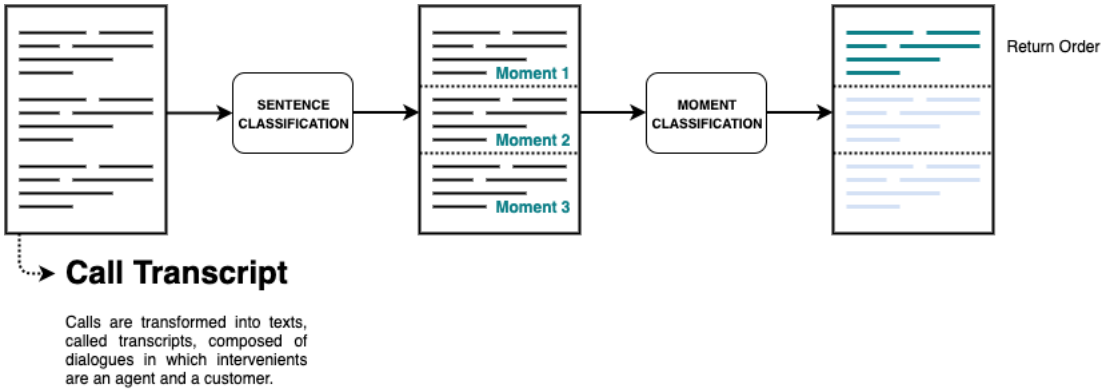


Figure 4.2: Moment classification process.

In figure 4.3 we show with a real scenario, extracted from the MultiWOZ dataset, the process explained previously. Each sentence was labeled with an intent - greeting, information relay, information request, information acknowledgment, farewell, and moment shift. The moment shift intent is used to split the transcript into moments, which are then classified as **train booking** and **hotel booking**.

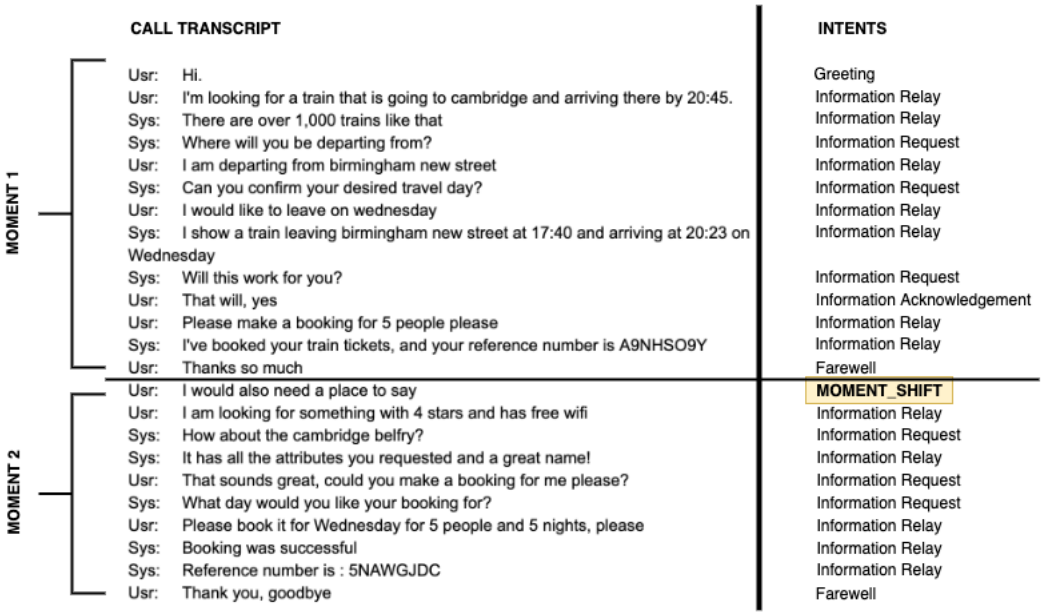


Figure 4.3: Moment classification example.

### 4.1.2 Template Filling

A template can be described as an efficient approach to structure information through a sample document. Fundamentally, it behaves as a structural summary, with the difference of not allowing a call center agent with the freedom to elaborate sentences. This template combines two components. The first one refers to already defined sections of text. Regarding the second, it is a set of fill-in-the-black segments that complement the previously mentioned sections when populated with relevant highlights extracted from dialogue conversations. In the third phase of the approach pipeline, templates created manually are used and filled with extracted content from the moments to generate a summary.

## 4.2 Automatic Template Generation

Since the Talkdesk approach requires that templates have to be created manually, the intern's contribution will focus on automating this time consuming and ineffective process. The solution developed automatically generate summary templates, by processing multiple examples of moments, representing those moments with triples, and finding patterns through clusters and facts which can be used in the summary.

### 4.2.1 Triple Extraction

The internship implementation starts with the triple extraction task. As the human conversation is inherently complex, ambiguous, and sometimes does not follow a sequential order, mastering how to represent and associate properties to an intent can lead to misinterpretations and, consequently, a final result with a performance below average. Another hurdle consists of dealing with phenomena such as interruptions, and the question that arises here is how we perceive relations between terms. Furthermore, when an anaphoric term emerges in a dialogue, associate that term with an entity can reveal to be intricate. These challenges were addressed in the second semester and referred on the next chapter. As of now, let us look at the following example to understand how the process evolves:

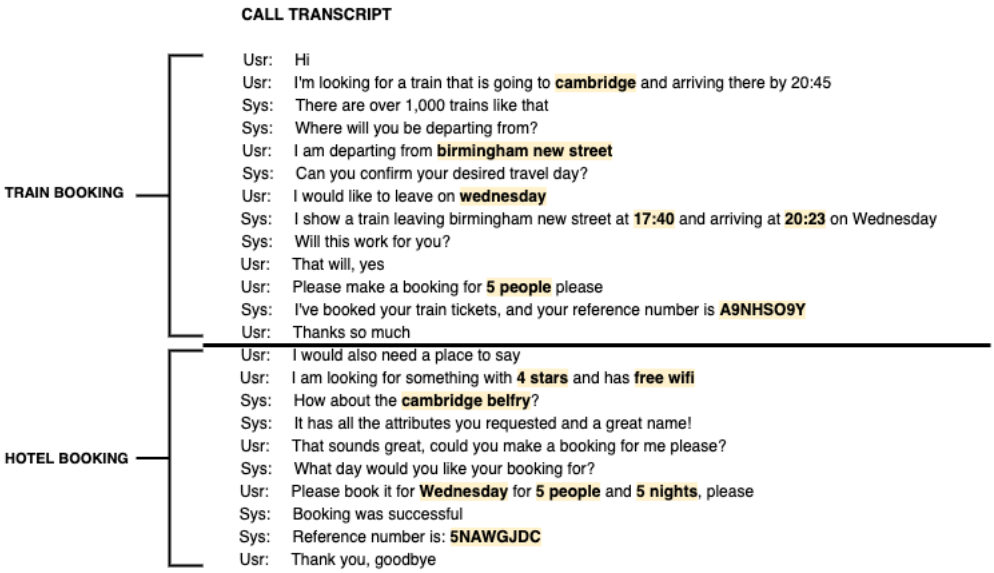


Figure 4.4: Triple Representation process.

For each moment, the dialogue between a customer and an agent will be composed by sets of subject-predicate-object triples trying to identify and represent the intent and the needs of the first-mentioned "actor." In figure 4.4, we have a moment in which a customer desires to book a train but has specific demands such as the departure and arrival station, or the schedule of the voyage. Represented in yellow boxes, are the vital information required to be extracted in the form of triples. The output expected for this process is exhibited in figure 4.5.

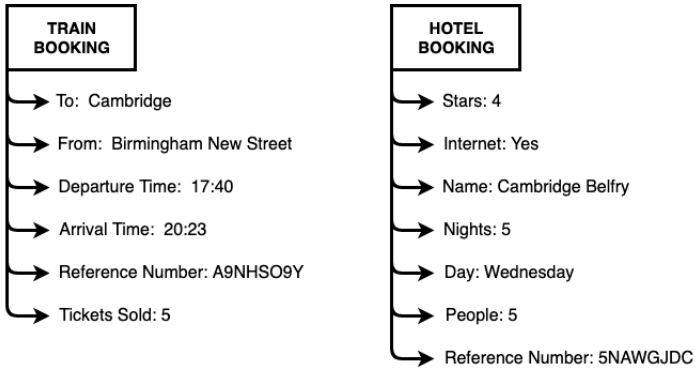


Figure 4.5: Triple Extraction example.

### 4.2.2 Clustering Formation

To fill templates with relevant facts extracted from triples, we start by placing those representative triples into groups, or clusters, such that triples in a given cluster tend to be similar to each other in some grammatically component or being content relatable. In figure 4.6, the system receives as input some Hotel Booking dialogs and generates combined clusters with the triples that aggregate the utterances. In this case, there are five clusters, and it is possible to see the similarity between the elements. For instance, cluster 1 exemplifies the future extraction of the fact "Hotel Name."

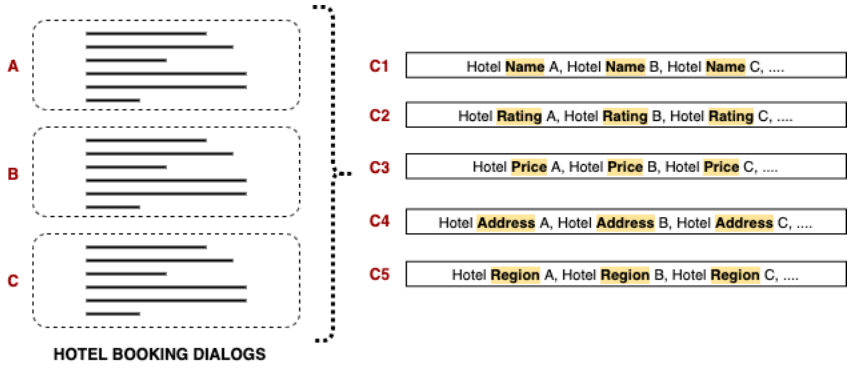


Figure 4.6: Clustering Formation process.

By receiving several dialogs composing the same moment type, these clusters will facilitate the fact extraction task, explained next, and identify useful patterns.

### 4.2.3 Fact Extraction and Template Generation

Towards the purpose of automating the task of generating templates, we require to comprehend first how to distinguish relevant information from factoids, and then find a way of performing this automatically. Figures 4.7 and 4.8 allows a better understanding of what these factoids signify.

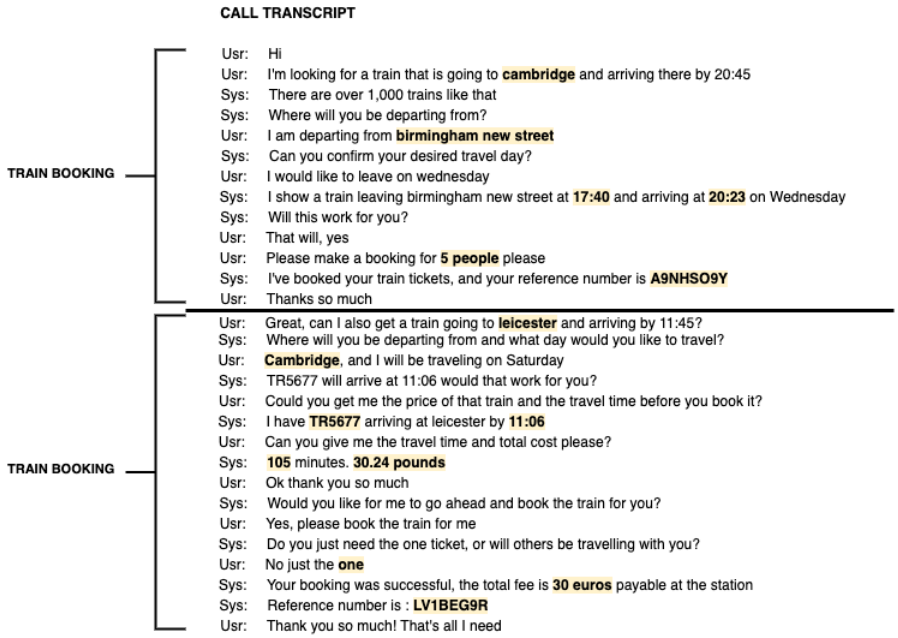


Figure 4.7: Fact Extraction process

In figure 4.7, two moments with the same topic, train booking, are outlined. Although both describe the same action, different characterizing arguments are provided. As observed in figure 4.8, features that are not highlighted, such as cost, train ID, and travel time are included in merely one moment. These features are designated as factoids, the term mentioned earlier. Factoids can be defined as particular characteristics to an individual dialog: a peculiar and non-common piece of information representing some unique attribute. A summary englobing and compiling a sample of dialogues will never present this kind of facts.

So how to generate a template of a particular moment if it can be represented in various styles? The answer lies in **finding patterns** through the triples and the clusters. Patterns in this circumstance are the similarities found between the triple representations of each moment, and those similarities help define what type of information will constitute the template. More precisely, if we extract from a few number of clusters and, consequently, their triples, the same fact, we can conclude that a pattern was encountered. The more times a fact is extracted from a cluster, the more likely it is to be selected to join the template.

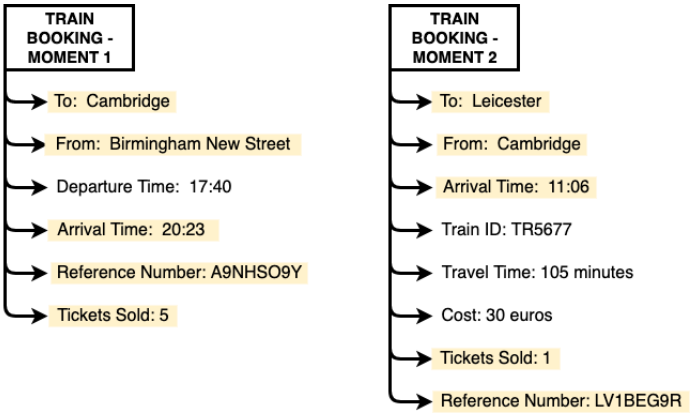


Figure 4.8: Illustration of factoids.

The intern implementation based on these four main steps is not as linear as explained with these examples, but the final product should resemble the demonstrated in figure 4.9. The process of filling the blank boxes is outside the internship domain and therefore, will not be solved.

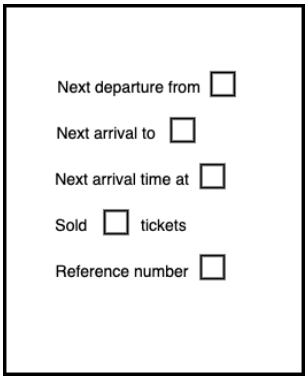


Figure 4.9: Template Generation example.

### 4.2.4 MultiWOZ Dataset

For completing this pipeline, we need a dataset with enough samples to perform these tasks. MultiWOZ is a recently released, fully-labeled collection dataset composed of human-to-human conversations spanning over multiple domains/topics including 3,406 single-domain dialogues, 7,032 multi-domain dialogues, and a train, test, and development set. The linguistic richness variation and the significant size facilitate and make it a benchmark for ongoing research within dialogue tasks [26]. Also, the various and broad possible dialogue scenarios allow us to recreate some of the moments or situations that may occur in the contact

center calls. These details motivated the preference of this dataset, especially during the implementation phase, instead of others accessible online.

## 4.3 Requirements

Comprehending what components, referred beforehand, are obliged and how they all should integrate, translates in a need for the requirement elicitation process. It does represent not only a guideline for the development but serving also as a way to evaluate the project's progress. The subsequent subsections will specify the possible use cases, along with all the requirements, including the functional as well as the non-functional requirements and the constraints that can be triggered. To accomplish these steps, first, we need to define the structure of each requirement and the general labels that compose them:

- **[ID - Name]**: Unique identifier and corresponding designation
- **[Description]**: A succinct summary that explains its function

### 4.3.1 Use Cases

One of the most helpful techniques to organize and analyze the requirements is by designing user stories. User stories are a tool adopted in agile software development that focuses on explaining how the system should behave under various conditions [21]. They are constituted by a set of possible sequences of interactions between systems and actors in a particular environment and related to a particular goal. Table 4.1 displays the only use case defined for this project.

### 4.3.2 Functional Requirements

This type of requirement state the main functionalities the system should do, and since they are based on the use cases, it becomes more straightforward to settle them. To achieve a better understanding of how significant and how much effort each of them requires, they were prioritized following a MoSCoW analysis and according to the following nomenclature:

- **M** (Must Have) - mandatory to be included.
- **S** (Should Have) - are not vital, but add significant value.
- **C** (Could Have) - helpful to have but with a small impact if left out.

| | **UC01 - Template Generation** |
|---|---|
| Description | One external system will send a document to the framework so that it can generate the template. At the end of the process, the template obtained returns to the external system. |
| Scope | System. |
| Actor(s) | External System. |
| Preconditions | The system receives a dialog transcript between an agent and a customer. |
| Basic Flow | The framework will receive the given document and will perform all of the tasks, described in figure 4.11, and then return the result obtained from them. |
| Exception Flow | In case of an impediment when performing the respective feature, the framework will report the error occurred. |
| Postconditions | Template generated from the initial document and the information contained on it. |

Table 4.1: Use Case 1.

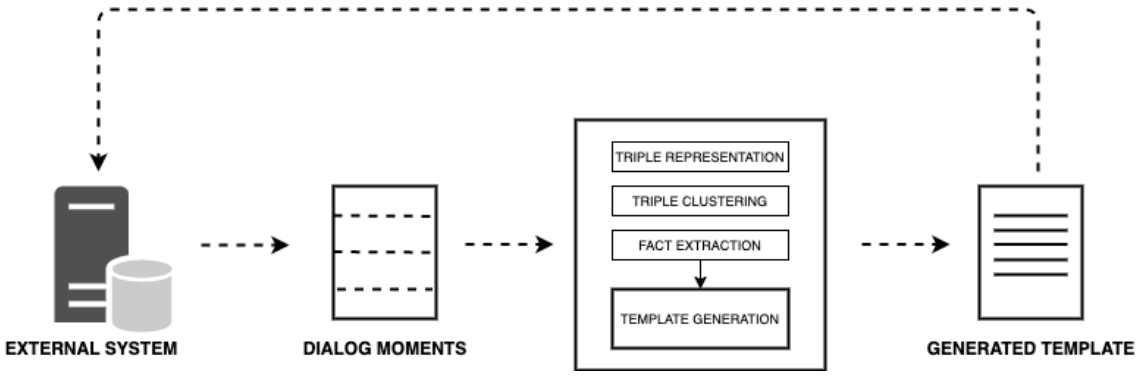


Figure 4.10: Diagram for Use Case 1.

- **W** (Will not have) - for this specific time frame they are not a priority

Knowing the priority and impact, it was also relevant to specify further, which were the dependencies. That is, which requirements are subservient to others. By addressing these metrics, the current analysis has become more detailed and reliable, but, simultaneously, has urged the necessity to combine two more labels to the structure of the functional requirements - **priority**, and **dependency**. Therefore this project shall fulfill the resulting requirements:

**FR1** - Anaphora representation
**Description**: The framework must be able to resolve anaphora situations and associate the correct relations
**Priority**: C
**Dependency**: None

**FR2** - Triple representation
**Description**: The framework must be able to extract triples of each moment in the form of (subject, predicate, object)
**Priority**: M
**Dependency**: None

**FR3** - Entity extraction
**Description**: The framework must be able to perform NER task
**Priority**: M
**Dependency**: FR2

**FR4** - Part-of-speech tagging
**Description**: The framework should be able to generate a part-of-speech tag for each token in order to identify the grammatical class
**Priority**: S
**Dependency**: FR2

**FR5** - Triple relevance
**Description**: The framework should be able to deal with triples that represent no value
**Priority**: M
**Dependency**: FR2, FR3

**FR6** - Utterance relevance
**Description**: The framework should be able to remove redundant and not relevant utterances
**Priority**: M
**Dependency**: FR5

**FR7** - Utterance representation
**Description**: The framework should be able to represent an utterance with sentence embeddings
**Priority**: S
**Dependency**: None

**FR8** - Clustering representation
**Description**: The framework should be able to group into clusters utterances from the same type of dialog
**Priority**: M
**Dependency**: FR7

**FR9** - Clustering performance
**Description**: The framework should be able to find the optimal number of clusters
**Priority**: S
**Dependency**: FR8

**FR10** - Fact extraction
**Description**: The framework should be able to distinguish which facts should enter to the templates
**Priority**: M
**Dependency**: FR3, FR4, FR8, FR9

**FR11** - Generate synonyms
**Description**: The framework should be able to deal with synonyms and disambiguation
**Priority**: C
**Dependency**: FR10

**FR12** - Template generation
**Description**: The framework must be able to generate structural summaries that characterize a type of moment
**Priority**: M
**Dependency**: FR2, FR8, FR10

**FR13** - Grant access to the modules via REST API
**Description**: All the components must be accessible from a REST API for external use.
**Priority**: S
**Dependency**: None

It is noteworthy that, after the development phase starts, before every sprint, these requirements probably will be discussed again with their priorities reviewed.

### 4.3.3 Non-Functional Requirements

Contrasting with functional requirements that define how a system should work, non-functional requirements detail the qualities that a system must-have. Associated with properties such as response time or security, they can impose constraints or restrictions on the design of the system across the different agile backlogs and become liabilities when failed to suffice [75]. The next list displays the existing non-functional requirements:

**NFR1** - Performance of the full pipeline
**Description**: The value for the metric F-Measure for the finished pipeline should be at least 70%.

**NFR2** - Modifiability
**Description**: It must be possible to apply alterations or include further functionalities in the module without affecting the current features.

**NFR3** - Robustness
**Description**: The framework must be fault-tolerant. Events, such as a failed request, an invalid input served, or invalid data, need to be handled appropriately without creating any consequence.

**NFR4** - Reuse of internal components
**Description**: All the components must be built independently of each other to future re-utilization.

### 4.3.4 Business Constraints

Business constraints arise from business decisions and must be satisfied with the architecture of the system. In this case, are:

**BC1** - Licenses (Commercial Level)
**Description**: The licenses of the tools used had to be under a license that allows their utilization for commercial purposes.

**BC2** - Licenses (Free Use)
**Description**: The licenses of the tools and data used should be available without any fee associated.

**BC3** - Schedule
**Description**: Product development must be terminated by June 16, according to the internship legislation, and the thesis delivered on June 29.

**BC4** - Development Time
**Description**: The initial version of the software has to be developed throughout the internship's second semester.

**BC5** - The system relies on external training data
**Description**: All training data used by the system comes from external sources, specifically from the MultiWOZ dataset.

### 4.3.5 Technical Constraints

Similarly to business constraints, technical constraints are unalterable. The system under development must obey the following constraints:

**TC1** - Programming Language
**Description**: All development should be done using either Python or Java.

**TC2** - Docker container
**Description**: The application must run inside a docker container.

**TC3** - Micro-service
**Description**: The product developed must consist of a micro-service.

### 4.3.6 Linguistic Constraints

The linguistics constraints that the development must follow are here stipulated:

**LC1** - Language
**Description**: The framework must be able to deal with the English language.

## 4.4 Risk Management

Performing this analysis focused on identifying all potential risks and their likelihood, as well as inward-looking the possible impact of events that escalate them, is a helpful tool that can avoid the probability of failure when a project is underway. Anticipate these threats with prepared strategies becomes crucial to reduce the possible adverse outcomes resulting in deviations on time estimates and therefore improves the chances of obtaining a successful outcome. This section addresses the task of risk management on this project, concentrating on the risk management procedure and the risk log.

### 4.4.1 Risk Management Procedure

Risk Management procedure sets how the risks associated will be analyzed, and managed, outlining the actions performed throughout the lifecycle of this project.

Before proceeding to risk identification, we must define the risk attributes and their levels. All will be ranked according to their probability (the likelihood of occurring) , impact (representative of the level of threat it presents on the project thresholds of success), and timeframe (time to react and solve). This classification will settle which risks are the most urgent to pursue and respond promptly.

The possibility of the occurrence of a specific risk can be classified as:

- **L** (Low) - the probability of happening is below 25%.
- **M** (Medium) - the probability of happening is between 25% and 75%.
- **H** (High) - the probability of happening is higher than 75%.

As for the impact, it can have the following criteria:

- **L** (Low) - a risk with relatively little impact to affect project schedule or performance.
- **M** (Medium) - a risk that can slightly affect project schedule or performance.
- **H** (High) - a risk that has the potential to affect project schedule or performance significantly.

Regarding timeframe values, they are categorized as:

- **S** (Short) - required to deal with the risk within one week.
- **M** (Medium) - required to deal with the risk within three weeks.
- **L** (Long) - required to deal with the risk within one month.

### 4.4.2 Risk Log

Risks identified throughout the lifecycle of this project are:

**R1** - Limited development time
**Description**: Since there is restricted development time (one semester), the implementation of some functional requirements may not be fulfilled
**Mitigation Plan**: The development will be adjusted and concentrated only on the functional requirements with higher priority, or the deadline can be postponed for September
**Probability**: S
**Impact**: H
**Timeframe**: M

**R2** - Little expertise with summarization tools
**Description**: Not being familiarized with summarization techniques can lead to delays in the development plan, demanding additional effort to accomplish the tasks
**Mitigation Plan**: Study the tools documentation and practice by doing some tutorials and examples online. Also, approach experienced colleagues at Talkdesk
**Probability**: H
**Impact**: M
**Timeframe**: M

**R3** - Quality of the tools and resources
**Description**: Since this project can rely on preexisting tools, the quality of the chosen tools can drastically influence the results
**Mitigation Plan**: Perform extensive tests that will compare all the tools and choose the most competent ones. In the case of unsatisfactory performances, the intern developing its solutions may be an answer but as a last resort
**Probability**: M
**Impact**: H
**Timeframe**: M

**R4** - Quality of the training resources
**Description**: The training resources (e.g., computational resources, validation datasets) available may turn out to be insufficient, which would have a significant influence on the performance of the product
**Mitigation Plan**: One solution could be the identification of new training datasets or the manual creation of the corpus
**Probability**: M
**Impact**: H
**Timeframe**: L

**R5** - Insufficient data
**Description**: Available data may be lacking, dependent on the chosen approach
**Mitigation Plan**: Request for data with the existing clients, or identify new testing datasets
**Probability**: S
**Impact**: M
**Timeframe**: M

## 4.5 Architecture

In this section, taking into account the previous requirement analysis, the platform architecture is exposed through a logical diagram, presented by figure 4.11, containing the core components and modules of the system, clarifying which connections between them exist and how other services will be integrated.

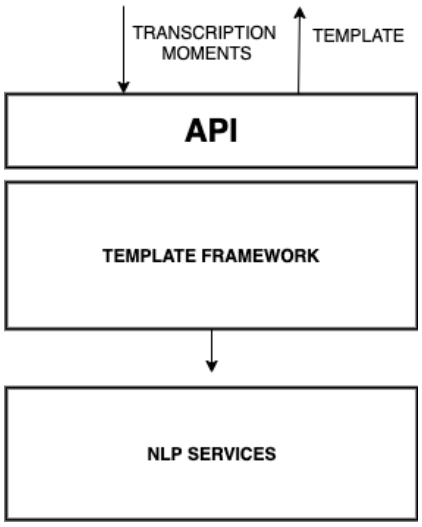


Figure 4.11: Process Flow.

As previously stated, the primary intent of this project is to implement a framework capable of automatically generating templates from dialogue transcripts. It is based on a micro-service architecture and exposes an API if an external system/application wants to make use of this framework. In case of being invoked the method for generating a template, and with the use of modules already implemented and inserted in the NLP services, the framework returns a template to the external system to deploy.

Since this framework invokes implemented modules from the NLP services, it is crucial to reflect on the propagation of errors through the services that precede the intern module. Firstly and because the product will work with dialogue transcriptions, we need to perceive the error rate of those transcriptions. Talkdesk uses the services of Deepgram for speech tasks, and their word error rate varies between 20 to 26%, which presupposes that it is an aspect to upgrade for the future. Also, the sentence and moment classification are modules to be considered. The module's precision is around 75 to 80%, so there is still the capacity to improve. Therefore, the performance of the framework developed during this internship can be jeopardized and influenced.

Additionally, to this error reflection, it is relevant to have an assessment of the system as a whole. The integration of this framework in the global context of the product architecture is still to be defined. As for now, the framework is essentially an isolated project but with the potential to become a reality.

## 4.6 Methodology and Work Plan

### 4.6.1 Work Methodology

This internship follows a development methodology adopted by Talkdesk, and the teams composed in the different clusters, based on *Scrum* [70]. *Scrum* is, besides being iterative and incremental, an "adaptative software development method" that offers the ability to adjust quickly to new features or changes in requirements properties [12].

In the *Scrum* methodology, there are three essential roles, and the project responsibilities are divided among them:

- ***Product Owner***: responsible for maximizing the value of the product and deciding upon the principal features.
- ***Scrum Master***: responsible for the Scrum process, leading meetings and delegating tasks to the development team.
- ***Development Team***: responsible for building the product.

In this project, the role of the *Product Owner* is played by Bruno Antunes and the *Scrum Master* by Pedro Verruma (both Talkdesk advisors). Concerning the *Development Team*, the intern integrates a Data Science team, although it will focus only on the intern framework.

The methodology used produces three major artifacts:

***Product Backlog*** - prioritized features list, including short descriptions explaining their functionality and new changes to implement.

***Sprint Backlog*** - forecast by the Development Team to identify which functionality in the product backlog will be developed in the next Increment.

***Increments*** - integration of all the product backlog items resolved in the previous and current sprints.

The *Sprints* on this project have a fixed duration of two weeks and take place one after the other, without any pause. They usually start with a Sprint planning meeting, and every week a meeting is held to recap the progress made and ponder future steps to take.

In terms of software, *Jira* [1]was initially planned to be used but since the different company teams at the time were not aligned in a SCRUM process, it would not make sense to create a Jira project destined to a single person. Therefore, we opt to use *Trello* [2].

[2]https://trello.com/
[2]https://www.atlassian.com/software/jira

### 4.6.2 Planning

Plans and schedules were established, adjusted, and debated frequently. It was fundamental to know where we were on the timeline, what time was available to do what remained, and revise the plan accordingly with sudden objectives and unforeseen events.

#### 4.6.2.1 First Semester

Looking back at the first semester, the first weeks of the internship carried out a robust research component and acquisition of knowledge in the field of Artificial Intelligence with great emphasis on the fields of NLP, and Summarization which is the main focus on this proposal. During this period, the time was also spent gathering the background knowledge, exploring the resources and tools that could become useful, and studying the market and the competitors we are facing.

Posterior to this phase, specifying the use cases plus requirements/constraints determined for this project and mapping these into architecture design were the tasks performed. After having the architecture defined, the possible risks that could occur, and the respective contingency and mitigation plans, were identified and described.

The last phase focused on developing the intermediate report for this internship, as well as preparing for the intermediate presentation. With each week corresponding to 16 hours of work, the internship workload initially established, the Gantt diagram of figure 4.12 shows the realistic temporal plan completed during the first semester.

The Gantt chart does not represent the first estimates since the scope of the intern project changed during the semester and consequently forced to redefine our related work, use case, and requirements.

#### 4.6.2.2 Second Semester

Regarding this last semester, with each week corresponding to 40 hours of work, it was split into three phases — implementation, validation, and thesis writing. As mentioned in section 4.6, every two weeks, a sprint starts, and in total, eight sprints occur. Each sprint is now described:

1. **Sprint #1**: after receiving the dialogues constituted by different types of moments, a dataset was generated that groups part of these dialogues in moments such as hotel booking, restaurant booking, or train booking. Also existing triple extraction approaches were studied and scrutinized.

2. **Sprint #2**: implementation of an approach to extract triples from a dataset with ClausIE. Additionally we examined the many variances that triples with the same feature identified (e.g., price range or departure time) can have in terms of subject, predicate or object.

3. **Sprint #3**: implementation of an approach to measure the similarity between triples. This task deals with metrics of similarity. Initially, we started to compare according to the subject, predicate, or object. The other approach was to join these three components and compare the triples as if they were sentences.

4. **Sprint #4**: study of different approaches for triple clustering, such as KMeans and DBSCAN, and comparison of the outputs obtained.

5. **Sprint #5**: implementation of an approach for executing sentence embeddings. Several types of sentence embeddings were tested. Also, we started to perform hierarchical clustering and find the optimal number of clusters.

6. **Sprint #6**: investigation of approaches for feature extraction and feature relevance among triples and utterances. Entity Extraction with SpaCy, Duckling and SpaCy Matcher was experimented.

7. **Sprint #7**: implementation of an approach to remove/clean utterances and triples not relevant and based on the Entities, tested in the previous sprint, and part-of-speech tags. Duplicate triples were also discarded.

8. **Sprint #8**: implementation of an approach for fact extraction after obtained the clusters. With the extracted facts, we also developed a method to distinguish which facts will enter the final template.

Completed the implementation task, and respective sprints, to ensure that all the functionalities work and meet all the defined requirements, the validation phase started. This phase consisted of requesting participants to manually annotate templates from a dialogue. The final report started being written after May.

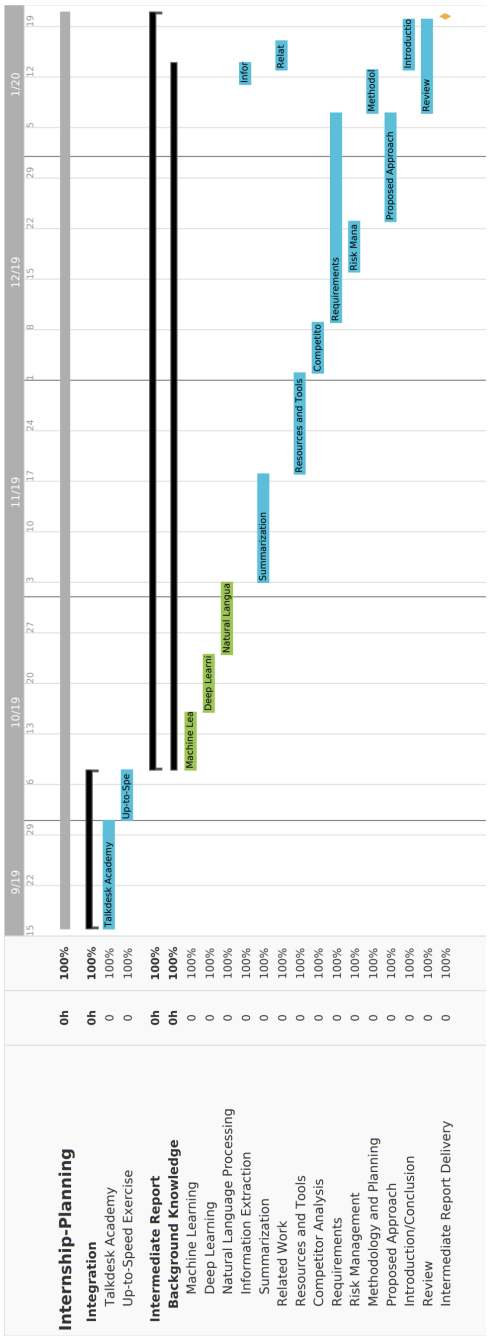


Figure 4.12: Internship plan for the 1st semester.

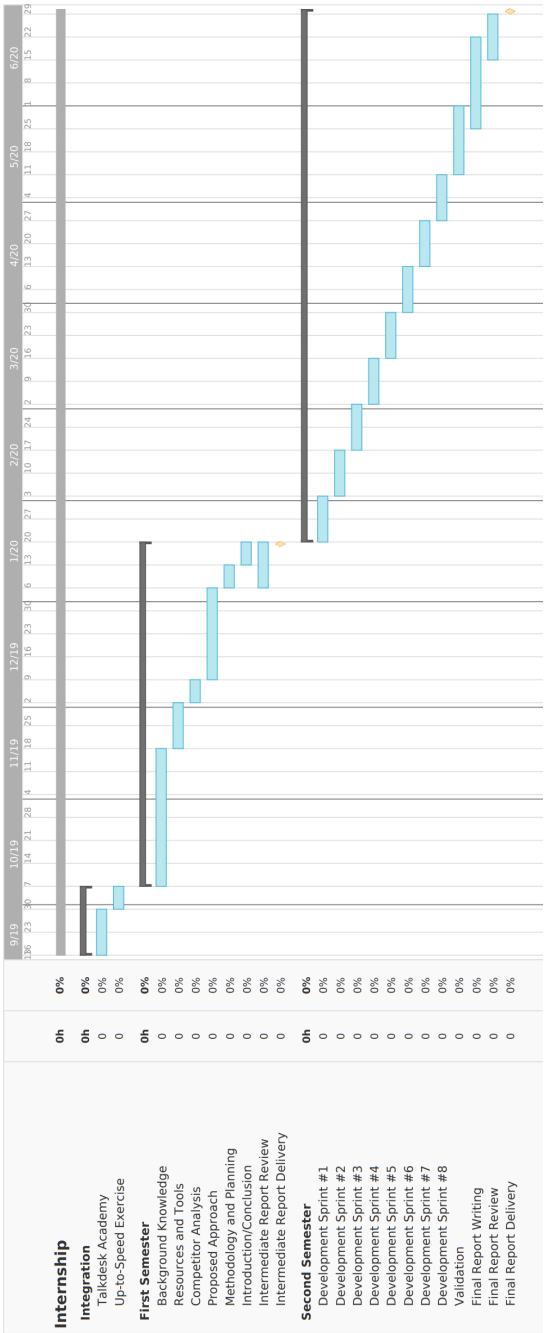


Figure 4.13: Internship plan for the 2nd semester.

# Chapter 5

# Implementation

The chapter introduced here aims to present the system developed in more depth in terms of the more technical aspects of implementing the functional requirements settled earlier. Further, this section states the additional components in the planned architecture and the obstacles encountered during the implementation phase, as well as the decisions that mitigate them. One aspect to take into account was the intention to develop a platform that uses minimal supervision or the need for specific configurations, and therefore directly influenced the pipeline of the approach. Also, due to the high degree of novelty of this pipeline and the lack of ideal results, the implementation is coupled with a strong research component.

As previously outlined, this internship consists of developing a framework proficient of automatically generating templates, or structural summaries, from dialogue transcripts. To attain this target, and respective functionalities, the tasks required were either fully implemented without the recourse to external libraries, or built-on dependent on their use. The upcoming sections describe the full pipeline of this project and the libraries and resources of support indispensable.

## 5.1 Dataset Construction

The experiments conducted used a dataset compiled by Cambridge Dialogue Systems [26] and previously described in section 4.2.4. This dataset was composed of various dialogue scenarios ranging from demanding necessary information about attractions through scheduling a train. However, due to the small samples of dialogues from each scenario, only the data from the Train Booking, Hotel Booking, and Restaurant Booking intents was handled. Table 5.1 presents a summary of the scenarios mentioned before:

| | Hotel Booking | Train Booking | Restaurant Booking |
|---|---|---|---|
| Number of utterances | 814 | 788 | 796 |
| Number of dialogs | 59 | 60 | 63 |

Table 5.1: Summary of the scenario dialogue used

After receiving and grouping the dialogues constituted by these different types of moments, we generated an exploratory dataset that, for each utterance, four other fields were associated:

- "intent"
- "triples"
- "triple_relevance"
- "utterance_relevance"

The field "intent" corresponds to the utterance intent obtained from the already implemented NLP module Sentence Classification. The following table 5.2 displays the current possible intents and demonstrates an example of each of these intents.

| **Intent Type** | **Utterance Example** |
|---|---|
| GREETING | *Good morning.*<br>*Hi.* |
| FAREWELL | *Thanks for using our service.*<br>*Have a great day.* |
| MOMENT_SHIFT | *Great! Can I assist you with anything else today?*<br>*Can I help you with anything else?* |
| INFORMATION_ACKNOWLEDGEMENT | *Of course.*<br>*Ok great.* |
| INFORMATION_REQUEST | *How many nights will you be staying?*<br>*Where will you be departing from?* |
| INFORMATION_RELAY | *I found 1 cheap hotel for you that includes parking.*<br>*Reference number is : 7GAWK763.* |

Table 5.2: Intent examples.

Regarding the remaining fields, their concepts and relations among them are going to be addressed more explicitly in the following chapter sections.

With the structure of the dataset defined, initial pre-processing was necessary to convert the data to the desired format—notably, punctuation removal and contraction expansion using a pre-defined mapping. For example, when performing contraction expansion in the sentence:

*The hotel didn't have a garage to park.*

Would result in the normalization of the expression "didn't" to "did not".

Additionally, we addressed the linguistic phenomenon described in section 2.3.5 - anaphora resolution. This functionality was implemented using NeuralCoref [1], a pipeline extension for SpaCy, which annotates and determines coreference clusters using a neural network. To be capable of using NeuralCoref, we needed to have an English model for SpaCy. When applying this method to the following text sample:

*How about the University Arms Hotel? It has free parking and wifi, and it is a 4 star rated hotel.*

The output will be: [University Arms Hotel: [University Arms Hotel, It, It]]. Thus, it should detect and associate the references to the entity mentioned.

To recap and infer the dataset structure, figure 5.1 exemplifies a simple dataset entry with the already associated fields.


```
"utterance_text": "I need three tickets to Portland for the March 9th.",
"intent": "INFORMATION_RELAY",
"utterance_relevance": 0,
"triples": [
    {
        "ClausIE": [
            {
                "triple": "Triple(index='1', subject='I', predicate='need', object='three tickets to Portland for the March 9th')",
                "triple_relevance": 0
            },
            {
                "triple": "Triple(index='1', subject='I', predicate='need', object='three tickets')",
                "triple_relevance": 1
            }
        ]
    }
]
```


Figure 5.1: Dataset entry example.

## 5.2 Triple Extraction

After the initial pre-processing phase, the next step was to find a way of representing an utterance in the form of a subject-predicate-object triple. The existing triple extraction approaches were studied and scrutinized, and in this domain, four relevant approaches stood out - StanfordOpenIE, Open IE 5.0, MinIE, and ClausIE.

[1] https://github.com/huggingface/neuralcoref

- StanfordOpenIE: a classifier that learns to extract self-contained clauses from longer sentences by traverse a dependency parse tree recursively. With natural logic inference over these clauses, it is capable of determining specific arguments for each candidate triple by shortening these utterances while maintaining the necessary context maximally. This approach is a case where there is a need for a labeled dataset [8].

- Open IE 5.0: a combination of the analyzers CALMIE, responsible for the extraction from conjunctive sentences, BONIE, bound with the extraction from numerical sentences, RelNoun for noun relations extraction, and SRLIE which is accountable for identifying n-ary extractions [52].

- MinIE: built on top of ClausIE, along with the triples, the approach confers information about polarity, modality, and quantities with semantic annotations. The polarity indicates the factuality of each extraction, while modality refers to the triple certainty/possibility. Finally, a quantity is a phrase that denotes an amount of something [28].

Since all of these extraction methods introduced own a license that makes commercial use impossible without having to make the code publicly available, the chosen path was a ClausIE Python Wrapper, an approach with a viable utilization license. Regarding our dataset, the field "triple" is filled with the output resulted from performing him.

ClausIE, as the name suggests, is a clause-based approach that extracts relations, and their respective arguments, from natural language text, by exploiting linguistic knowledge according to the grammatical function of the sentence constituents. ClausIE is based on dependency parsing and a subset of domain-independent lexica and requires no training data [23].

Due to the strategy followed by this approach, it manages to take a group of sentences and obtain seven different types of clauses depending on the constituents, as illustrated in figure 5.2.

| | Pattern | Clause type | Example | Derived clauses |
|---|---|---|---|---|
| | | | **Basic patterns** | |
| $S_1$: | $SV_i$ | SV | AE died. | (AE, died) |
| $S_2$: | $SV_eA$ | SVA | AE remained in Princeton. | (AE, remained, in Princeton) |
| $S_3$: | $SV_cC$ | SVC | AE is smart. | (AE, is, smart) |
| $S_4$: | $SV_{mt}O$ | SVO | AE has won the Nobel Prize. | (AE, has won, the Nobel Prize) |
| $S_5$: | $SV_{dt}O_iO$ | SVOO | RSAS gave AE the Nobel Prize. | (RSAS, gave, AE, the Nobel Prize) |
| $S_6$: | $SV_{ct}OA$ | SVOA | The doorman showed AE to his office. | (The doorman, showed, AE, to his office) |
| $S_7$: | $SV_{ct}OC$ | SVOC | AE declared the meeting open. | (AE, declared, the meeting, open) |
| | | | **Some extended patterns** | |
| $S_8$: | $SV_iAA$ | SV | AE died in Princeton in 1955. | (AE, died)<br>(AE, died, in Princeton)<br>(AE, died, in 1955)<br>(AE, died, in Princeton, in 1955) |
| $S_9$: | $SV_eAA$ | SVA | AE remained in Princeton until his death. | (AE, remained, in Princeton)<br>(AE, remained, in Princeton, until his death) |
| $S_{10}$: | $SV_cCA$ | SVC | AE is a scientist of the 20th century. | (AE, is, a scientist)<br>(AE, is, a scientist, of the 20th century) |
| $S_{11}$: | $SV_{mt}OA$ | SVO | AE has won the Nobel Prize in 1921. | (AE, has won, the Nobel Prize)<br>(AE, has won, the Nobel Prize, in 1921) |
| $S_{12}$: | $ASV_{mt}O$ | SVO | In 1921, AE has won the Nobel Prize. | (AE, has won, the Nobel Prize)<br>(AE, has won, the Nobel Prize, in 1921) |

S: Subject, V: Verb, C: Complement, O: Direct object, $O_i$: Indirect object, A: Adverbial, $V_i$: Intransitive verb, $V_c$: Copular verb, $V_c$: Extended-copular verb, $V_{mt}$: Monotransitive verb, $V_{dt}$: Ditransitive verb, $V_{ct}$: Complex-transitive verb

Figure 5.2: Example of identified patterns and corresponding grammatical functions.

This research allowed, simultaneously, for a performance analysis focused on the effectiveness of the approach. More precisely, on the number of sentences in which a triple(s) was generated or not. Based on the moments that characterize our dataset, the following results were collected: **35.6%** of the utterances were not associated with at least one triple. Two reasons can justify or mitigate this percentage: short utterances composed by a couple of words, such as "Okay great"; and utterances lacking a predicate or an object grammatical component.

## 5.3 Utterance Relevance

Before performing clustering and grouping together related utterances/triples, it was time to differentiate which sentences contained redundant information and consequently, not relevant for our approach. Two strategies were conducted to solve this problem: one focused on the sentence intent, and the other based on the triples that constitute an utterance.

Regarding the first one, as previously stated, the module Sentence Classification correlates an utterance with specific intent. Instantly, we can discard those with intents that we know, from the start, that they never represent value or help us to obtain the final template. The following image 5.3 displays the non-relevant intents and the corresponding percentage of appearance on our dataset:

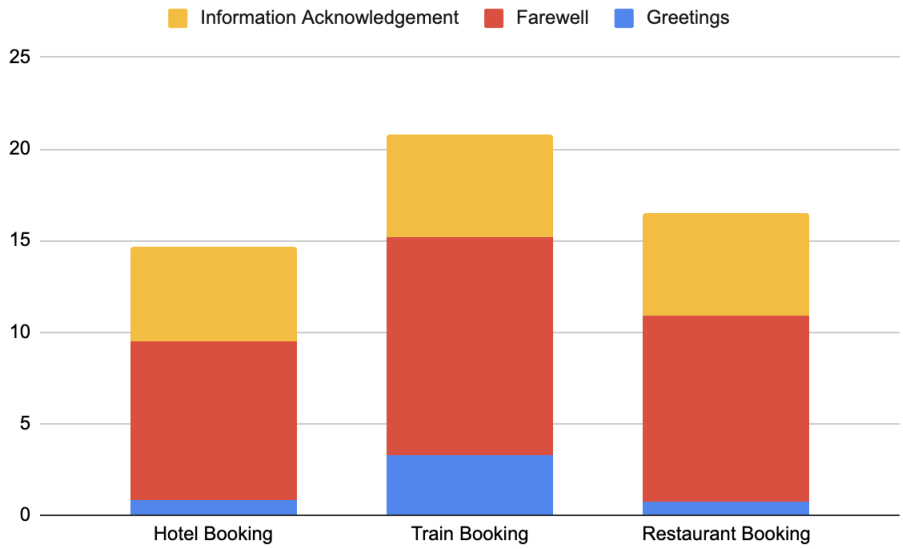


Figure 5.3: Non-relevant intents.

It is possible to see that these three types of intent combined translate in a weight of 15 to 20 percent, which is a significant step for excluding utterances considered irrelevant on the approach pipeline.

The second approach heavily relies on the triples of each utterance. Intending to distinguish if a triple is relevant, it was necessary to achieve two subtasks. Firstly, perform entity extraction. By entity extraction, we mean extracting information from which component of the triple. A couple of different strategies were used:

- Extract data present in the triple based on **patterns** defined by the intern: spaCy features a rule-matching engine, Matcher [2], which allows us to write rules to find tokens, words, and phrases in a text. When compared with regular expressions, mentioned in section 2.3.1, the advantage of this matcher is the capacity to work over Doc and Token objects instead of only strings. Also, it reveals to be more flexible by searching for lexical attributes.

  Matcher patterns are lists of dictionaries, one for each token. The keys are the names of token attributes, mapped to their expected values. For instance, figure 5.4 exemplifies a pattern that includes two token attributes "TEXT" mapped to string values in the form of REGEX rules.

```
{[{"TEXT": {"REGEX": "^[0-9]$"}}, {"TEXT": {"REGEX": "^star(s?)$"}}]
```

Figure 5.4: Example of a Matcher pattern.

[2]https://spacy.io/usage/rule-based-matching

To use a pattern we need to load a model ("en_core_web_sm") and create the NLP object, initialize the Matcher with the shared vocab from the model, and add a pattern to that vocab. For instance,

More specifically for this project, we used these patterns to extract, for instance, schedules ("11:18", "11 AM"), ratings ("4 stars"), postcodes, durations ("5 nights"), persons ("three people"), and addresses. This last was also detected with Pyap, an MIT Licensed text processing library for parsing addresses.

- Extract data present in the triple based on the **module NER** of the libraries SpaCy and Duckling [3]: we used Docker to run a server and therefore connect to a Duckling container to perform requests to this library. Furthermore, since both do not have the full capability to extract locations or restaurant names, we opted to match the analyzed text to a predefined list of entities of this kind obtained from public APIs.

However, extracting information is not enough to measure the triple relevance. The second subtask consists of making use of the Part-of-speech tagging with the help again of the SpaCy library. Because we need to consider the context, we grouped the triple components into a sentence. We know at the outset that if a sentence contains a noun, adjective, number, or a proper noun, the likelihood of providing valuable information is high. With the displaCy visualizer, for example, figure 5.5 exemplifies how a noun tag can represent a suitable extractable feature.

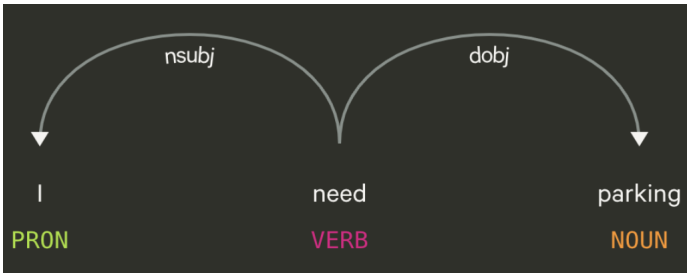


Figure 5.5: Example of Part-of-speech tagging.

With the aggregation of these subtasks, the framework forthwith is competent to infer the triple relevance and concurrently capture duplicate triples. Briefly, if a triple does not contain any entity extracted or include a relevant part-of-speech tag, the triple under analysis is considered not admissible. Accordingly, the dataset field "triple_relevance" is assigned with a boolean flag. If all the triples that compose an utterance are assigned as not relevant, we conclude that utterance can be **discarded** and removed from the system. From these combined methods, we can discard, on average, **17%** of utterances.

[3] https://duckling.wit.ai/

## 5.4 Clustering

To obtain better results, before proceeding to perform clustering, we tried to reduce the variance among utterances and respective triples, and consequently, increasing their similarity. The first strategy consisted of modifying the utterances with generic tags obtained from the previous entity extraction method. For instance, for the example:

*I want a reservation for four persons and four nights.*

this approach will produce the following output:

*I want a reservation for* ***INFORMATION_PERSONS*** *and* ***INFORMATION_NIGHTS****.*

The second strategy englobes some steps: collect the tokens from an input utterance with a Word Tokenizer and remove those who are stop words. Both these operations are implemented by applying the NLTK library.

Moving on to the clustering method itself, the utterances entering this operation are represented in a numeric vector: sentence embeddings. We tested four approaches:

- Sentence Embeddings with Word2Vec
- Sentence Embeddings with GloVe
- Sentence Embeddings with FastText
- Sentence-BERT

The first three calculate the embeddings for an utterance based on each word's embedding and the average weight obtained. Word2Vec, trained on Google News data, is provided by Google and based on 100 million words with 300 dimensions [55]. Global Vectors for Word Representation (GloVe) rendered by Stanford NLP with various models from 25 to 300 dimensions base on a maximum of 840 billion tokens. The word embedding, in this case, is built with word co-occurrence probability [62]. Finally, FastText, which is released by Facebook and provided by three models with 300 dimensions each, includes character n-grams that enable computing word representations for words that did not emerge in the training data [14].

However, there are some issues when employing these type of approaches:

- Ignore word ordering. For instance, the utterance "Talkdesk product is straightforward to use, I do not need any help" in this situation is similar to "I do need help, Talkdesk product is not straightforward to use";

- Incapability to capture the semantic meaning of a sentence. The word "crash" has multiple meanings depending on the context, e.g., "I crashed my motorcycle", or "I crashed a party";

- Problems when dealing with long utterances length.

These issues justified our decision to opt with Sentence-BERT with a siamese or triplet network structure to generate semantically meaningful sentence embeddings that can be applied in unsupervised scenarios as clustering. We used an already trained Sentence Transformer model to embed sentences, which was *bert-large-nli-stsb-mean-tokens*, and substantially outperform the previous approaches [69].

With the approach chosen for sentence embeddings, next, we required to define the most appropriate clustering algorithm considering the domain of this project. Therefore we impose specific conditions such as the algorithm that does not need to specify in advance what is the number of clusters to generate. Immediately methods like K-Means and Spectral Clustering were rejected. The second restraint was the algorithm's capability to deal with large-sized datasets without compromising the time and memory complexity. Again, we could cut the Affinity Propagation method of the candidate's list. At this point, there were two possible solutions. The pipeline will either include a density-based clustering approach, capable of identifying arbitrarily shaped clusters where clusters are dense contiguous regions and outliers low dense regions data points [72], or a hierarchical approach that seeks to build a hierarchy of groups. The third condition was related to the input parameters needed to be defined. With the module *sklearn.cluster*, the density-based method, DBSCAN, obligates us to determine the maximum distance between two samples for one to be considered in the other neighborhood of the other. Coupled with this parameter, and the fact that sometimes tend to generate a high number of clusters, it allowed us to reach a decision.

Thus, the clustering task was implemented using Hierarchical Clustering, more precisely the Agglomerative Clustering, with the module *scipy.cluster.hierarchy.fcluster*. Inherent to this method, the algorithm is straightforward. Initially, a proximity matrix is computed, and each existing data point is considered an individual cluster settled in the ground level. At each iteration, the most similar clusters merge to the next level, and the matrix updated. This process continues until a level with just one cluster is formed. A cluster tree, referred to as a dendrogram, is built to represent the clusters and levels. Figure 5.6 displays an example of a dendrogram.

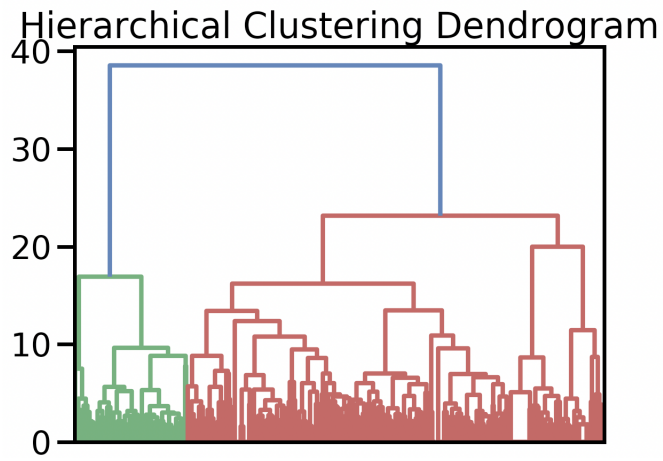


Figure 5.6: Example of a cluster tree.

The following are methods for calculating the proximity matrix:

- Simple: the distance between cluster $c$ and cluster $j$ is set as the shortest distance between two points in each cluster.
- Average: the distance between cluster $c$ and cluster $j$ is set as the average distance between each point in one cluster to every point in the other cluster.
- Complete: the distance between cluster $c$ and cluster $j$ is set as the longest distance between two points in each cluster.
- Ward: criterion for choosing the pair of clusters to merge at each step is based on the Ward variance minimization algorithm.

All these clusters linkage methods generate a linkage matrix $Z$ as their output. Combined with this linkage matrix, the hierarchical process further needs the maximum number of clusters to perform. For this project, based on the average size of each type of dataset dialogue, the intern established as 35 the maximum number of clusters possible.

The attendant step of the pipeline is to determine the optimal number of clusters. To obtain this number, we used the silhouette score method from the sklearn library. The silhouette coefficient of data measures how well data are assigned to its cluster and how far they are from the remaining. For instance, for the Hotel Booking dataset, we calculate the silhouette score with the parameter "maxcluster" ranging between 25 and 35 and select the one with the best score. The following table 5.3 displays an illustration of this selection.

| Number Clusters | Silhouette Score |
|---|---|
| 25 | 0.2884754253686447 |
| 26 | 0.29288795685468155 |
| 27 | 0.3015619161337606 |
| 28 | 0.30273834527858295 |
| **29** | **0.30343268056828454** |
| 30 | 0.2949018043657406 |
| 31 | 0.29786598597689024 |
| 32 | 0.2991436244804131 |
| 33 | 0.2995893658512116 |
| 34 | 0.28891683310076394 |

Table 5.3: Selection of the optimal number of clusters.

It is possible to infer that the optimal number of clusters for this dataset is 29, with a higher silhouette score. Conclude this task, the clusters containing the dialog utterances are associated with the respective triples.

## 5.5 Fact Extraction

Received the clusters with the utterances and respective triples, the next task on the approach pipeline consists of extracting factoids and relevant data. As explained in the Approach chapter, factoids are nothing more than unique features that merely characterize a single dialog of a given moment, instead of a standard feature between several dialogs. Extracting these facts includes analyzing the triples representing the utterances contained in the clusters and, based on patterns, defining those that stand out before others and constitute non-redundant information. The question that arose here was how to extract essential facts based on the structure of the triple? The answer relied on the type of triple present. Exist two types which are:

- **Association Triples** with a component(s) already extracted from the previous entity extraction step associated with some feature(s). Given the triple:

  Subject: *HOTEL_NAME*; Predicate: *have*; Object: *parking*

  The facts extracted in this example will be the subject (HOTEL_NAME), representing a piece of information already replaced by a defined tag and the object ("parking").

- **Attribute Triples** with a particular attribute characterized or designated and with no component(s) replaced with a defined tag. For instance:

  Subject: *Price*; Predicate: *is*; Object: *cheap*

In this case, the facts extracted will be the subject ("price") and the object ("cheap") as well.

Triple components are considered relevant, depending on their corresponding part-of-speech tag. If either the subject or the object, have a "NOUN", an "ADJ", or a "PROPN", and in turn, the object has a "AUX", a "VERB", or a "NOUN", which are the cases displayed previously, there are facts to be selected and extracted.

Additionally, we need to deal with some special cases. When an object component has a sequence composed of an adjective and a noun, e.g., "free parking", we maintain only the word associated with the noun tag — "parking". Another situation is the presence of adjacent noun substrings such as "phone number", where we have to extract both of them as a unique fact.

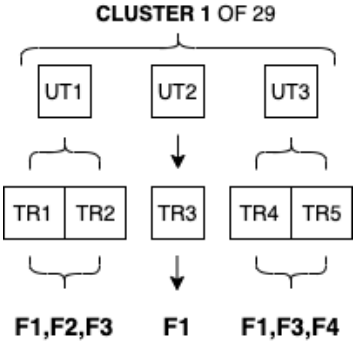


Figure 5.7: Facts associated with a cluster.

Once obtained all the facts associated with the utterances composing a cluster, as represented in figure 5.7, the following stage was to define and distinguish which of those are factoids and which will be candidates to enter in the final moment template. An approach with four steps is performed to achieve this:

1. Singularize plural facts. If a particular fact has the tag "NNS" or "NNPS," it will be transformed into his base form. For instance, the possible fact "Bedrooms" will be converted to "Bedroom".

2. Averiguate if there are synonyms in the list of facts. This substep is implemented with NLTK WordNet. WordNet stores synonyms in the form of synsets, where each word in the synset shares the same meaning. However, linguistic data is very contextual, and therefore, it is required to do word sense disambiguation. One hypothesis was using the Lesk algorithm to estimate the word's sense in the given context sentence [45], but in this situation, we only have access to the fact itself. The second and chosen approach was to use the first sense of the word synsets.

3. Search for facts that are substrings of others. For instance, consider we have two facts — "phone number" and "phone." The second one is a substring of the first, so these two facts must be considered the same.

4. The final action for determining if a fact is or is not a factoid was counting the fact occurrences on each utterance of the cluster list. With the three previous steps completed, we just needed to compare fact by fact and reach an occurrence percentage. We imposed that if that percentage is above or equal to 50%, the fact in analyzing is indeed relevant and represents the cluster. Considering the example displayed in figure 5.7, both fact **F1**, with a 100% occurrence percentage, and fact **F3**, with 66.7%, are the final cluster facts. In the opposite, facts F2 and F4 are assigned as factoids.

The output of this algorithm resembles what exhibit in figure 5.8 with the relevant and patterned facts associated with each cluster.

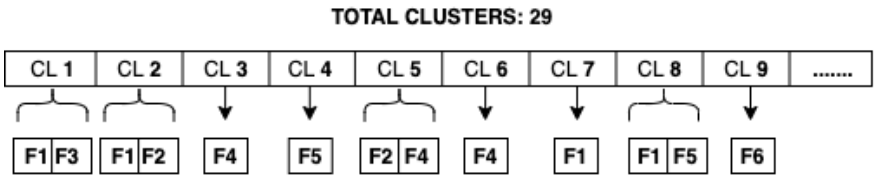


Figure 5.8: Candidate facts to enter the moment template.

## 5.6 Template Generation

To determine which of these facts enter the template, we defined three features:

- The number of cluster occurrences and relative percentages.
- The combined size length of the clusters that the fact appears and corresponding relative proportions.
- The portion of template dialogs that include the fact. Considering the 59 dialogs that compose the Hotel Booking moment, with each having a manually set template, we match the fact with those templates and extract the number of appearances.

Besides all of these features, the next step was to compute and assign a weight to each of them. This weight measures feature importance. This task was implemented using the Sklearn library, and the logistic regression model, usually used when the dependent variable(target) is categorical. In this case, the target will be one, meaning the fact is included in the final template and zero, if otherwise. After receiving the values of the features and the target, the optimization function returned the following weights, presented in table 5.4, for the established features:

| Feature | Feature Weight |
|---|---|
| % Cluster Occurrences | 14.78% |
| % Size Cluster | 23.81% |
| % Appearances in Dialogs | 61.42% |

Table 5.4: Feature Importance.

Ultimately, as shown in table 5.5, with the feature values and feature weights combined and computed, it is possible to obtain the decisive target that imposes which facts will be appended in the moment template. This target is associated with a threshold carefully dictated, based on which value the framework grants the most reliable performance, set to 30%. Regarding this table, it is an authentic example tested with our framework, and it represents the domain type Hotel Booking.

| Fact | Cluster Occurrences | Size Clusters | Appearance Dialogs | Final Value |
|---|---|---|---|---|
| **Hotel Name** | 8.82% | 2.15% | 96% | 60.73% |
| **Wifi** | 2.94% | 2.38% | 56% | 35.41% |
| **Parking** | 2.94% | 3.34% | 58% | 36.84% |
| **Rating** | 5.88% | 5.49% | 65% | 42.11% |
| **Price** | 2.94% | 7.64% | 81% | 52% |
| **Date** | 8.82% | 12.65% | 62% | 42.40% |
| **Number Nights** | 2.94% | 9.55% | 65% | 42.61% |
| **Booking Status** | 5.88% | 6.21% | 44% | 29.38% |
| **Postcode** | 2.94% | 0.72% | 13% | 8.59% |
| **Region Name** | 5.88% | 2.86% | 15% | 10.79% |
| **Reference ID** | 5.88% | 7.16% | 67% | 43.74% |
| **Address** | 5.88% | 3.82% | 31% | 20.82% |
| **Phone Number** | 2.94% | 2.15% | 12% | 8.31% |

Table 5.5: Template generation process.

By analyzing this table, we can conclude that the template for this domain type will include the facts: hotel name, wifi, parking, rating, price, date, number of nights stayed, and reference ID. On the opposite side, booking status, postcode, location, address, and phone number are considered factoids regarding this domain.

# Chapter 6

# Validation and Results

Process Validation represents a significant component of the project pipeline. It evaluates and establishes evidence whether the developed framework meets all the requirements and obeys all the constraints imposed throughout the planning phase. Succinctly, it stages the final step towards the fulfillment of this project.

This chapter at the outset describes the test environment, introduces the validation datasets handled, laid out the validation plan, and posterior the results will be critically analyzed and discussed.

## 6.1 Test Environment

Both development and test environment are related to the local environment in which the implementation was carried out, more specifically, the intern's machine. Table 6.1 showcases its specification:

| **Specification** | **Test and Validation Machine** |
|---|---|
| Operating System | macOS Mojave Version 10.14.6 |
| CPU | 1.4 GHz Intel Core i5 64-bit |
| Memory | 8 GB 2133 MHz LPDDR3 |

Table 6.1: Test Machine Specification.

## 6.2 Validation Dataset

Before reaching this stage, we were using for implementation support the MultiWOZ dataset divided by only three-moment types. Therefore we needed to encounter and incorporate another dataset to

obtain a reliable validation and simultaneously respect the prerequisites imposed on the MultiWOZ, which were dataset size and the dataset domain. Furthermore, we decided that the dataset applied through implementation would be involved during the validation phase.

After a research component analyzing different public datasets, the choice fell on the recent Schema-Guided Dialogue (SGD) dataset [68], consisting of task-oriented conversations amongst a user and a virtual assistant. Compared to the implementation dataset, SGD achieves greater diversity and covers a wider variety of tasks expected of automated dialogue agents, as presented on the following table 6.2.

| **Moment Dataset** | **Number of Utterances** | **Number of Dialogs** |
|---|---|---|
| Flight Booking | 391 | 40 |
| Car Rental | 420 | 40 |
| Bus Booking | 444 | 40 |
| Medical Services | 445 | 40 |
| Weather Forecast | 291 | 40 |
| Property Visit | 450 | 40 |

Table 6.2: Summary of the validation dataset.

This dataset to be included in the approach pipeline also required to undergo a pre-processing phase, as mentioned in section 5.1. With this process concluded and the datasets defined, only the approach to validate our approach was lacking.

## 6.3 Validation Approach

It is relevant to refer that at this stage, the requirements, functional and non-functional, assessment strategies should be already carefully defined. Although the approach components are independent and capable of being reused, testing them separately was not possible for reasons of time consumption and training resources, i.e., building datasets for each component of the pipeline was impracticable and unachievable to us. Therefore the closing decision was to opt to test only the final result, the full pipeline combined.

With this stated, to test and confirm the success of our approach, we conducted 32 user-testing sessions. As a quality assurance measure and requirements assessment, these participants and their results would be essential for this project. Regarding the selection of the participants, no strict rule was held.

Two pipeline validation hypotheses were examined. In the first case, the participants would receive previously defined templates from the various dialogue moments, and their job was reporting if those templates were, or not, accurate. However, and with the likelihood, when displaying the templates, that the users will be directly influenced, we discarded this hypothesis and elected the other. Concerning the chosen case, the participants had to assume that they worked as agents in a call center. Their associated respon-

sibility was to answer incoming calls from possible clients and fill structural call-summaries after the end of the respective calls. We tried to simulate this situation by presenting to the participants a collection of different transcripted dialogues, divided by domain types. Their goal was to perceive which information is considered relevant and, finally, elaborate a template, for each domain type, with the features to be included in a call-summary. Figure 6.1 demonstrates an example of these transcripted dialogues [1].

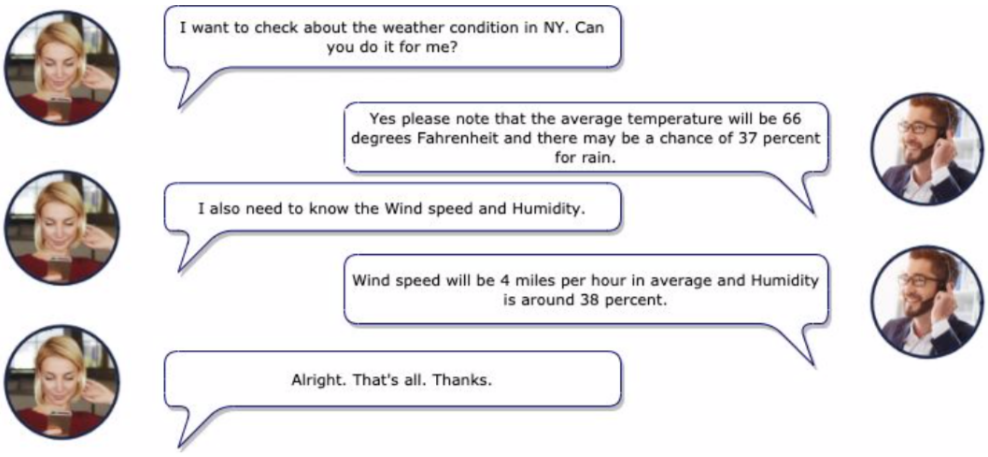


Figure 6.1: Example of a transcripted dialogue from the Weather domain type.

To apply variance and randomization to the validation data, we divided the participants into three groups. Each group would receive for each type of domain, a collection of different dialogues. This way, we would conceivably obtain a wider spectrum of responses that would benefit this validation.

In turn, the intern had to evaluate each participant's response. As stated, for each dialogues collection of the same domain type, the users would annotate a list of features manually. Those lists would then be compared with the lists that the approach automatically identified. This way, we evaluate the system based on its ability to replicate the templates that humans have created. For performing this comparison, we manually matched the list of features selected by the users and the list of features defined by the system and extracted specific metrics. To understand the computation of these metrics, we assume for our project scope:

*True Positive*: number of features correctly classified as included in the template of a domain type

*False Positive*: number of features incorrectly classified as included in the template of a domain type

*True Negative*: number of features correctly classified as not included in the template of a domain type

*False Negative*: number of features incorrectly classified as not included in the template of a domain type

[1] The attachments section of this report shows the supporting and filing documents that participants were entitled to.

Accordingly, the following metrics are applied:

- **Accuracy**: the ratio of observations correctly predicted with the total number of observations.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \tag{6.1}$$

- **Precision**: the ratio of positive observations correctly predicted with the total predicted positive observations.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{6.2}$$

- **Recall**: the ratio of positive observations correctly predicted with the total observations in the actual class.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{6.3}$$

- **F-Measure**: explained in equation 2.6.

The next section describes the tests performed to validate the approach pipeline and the results obtained from them.

## 6.4 Test Results

When validating a project, the first step is to calculate the concordance between participants to comprehend if the task imposed was well defined and refined. If the output value is low, it may indicate that they did not understand well what was asked or that the task revealed to be complex to perform. This value is computed by pairing up all the participants and then perform an average of all the result samples. This value is computed by pairing up all the participants and then perform an average of all the result samples. For the different domain types, we obtained a value interval between 65 and 90%.

$$Concordance = \frac{Concordances}{Concordances + Disagreements} * 100 \tag{6.4}$$

Concluding that the task required on the participants was adequately established, the subsequent step was to compare the templates defined by the users with the templates obtained by the approach. In this scenario, we experimented different variants of the results:

- Considering the average when compared to each annotator. For the same domain type, we calculate the accuracy, precision, recall, and f-measure for the template of the annotator and then average the metrics among all of them.

Table 6.3 shows the obtained results for the model, considering the average between participants. It exhibits the values of the precision, accuracy, recall, and f-measure of the model.

| **Moment Dataset** | **Precision** | **Accuracy** | **Recall** | **F-Measure** |
|---|---|---|---|---|
| Hotel Booking | 75.25% | 64.50% | 82.06% | 76.97% |
| Train Booking | 84.71% | 72.70% | 83.24% | 82.54% |
| Restaurant Booking | 86.02% | 66.18% | 74.16% | 77.56% |
| Flight Booking | 75.92% | 65.09% | 84% | 77.85% |
| Car Rental | 91% | 71.83% | 77.96% | 82.33% |
| Bus Booking | 78.06% | 67.42% | 82.79% | 79.26% |
| Medical Services | 83.78% | 55.18% | 60.83% | 68.88% |
| Weather Forecast | 93% | 81.89% | 88.83% | 89.50% |
| Property Visit | 63.73% | 53.07% | 77% | 67.93% |
| **Average** | 81.27% | 66.43% | 78.99% | **78.09%** |

Table 6.3: Results of the first validation approach.

Having into account the non-functional performance requirement, the obtained measures consolidate the success of the approach.

- Considering, for each set of dialogues of the same type, a single set with all the features suggested by the participants at least two times (n=2).

| **Moment Dataset** | **Precision** | **Accuracy** | **Recall** | **F-Measure** |
|---|---|---|---|---|
| Hotel Booking | 50% | 65% | 100% | 66.67% |
| Train Booking | 58.30% | 73.68% | 100% | 73.65% |
| Restaurant Booking | 61.51% | 72.22% | 100% | 76.19% |
| Flight Booking | 46.15% | 61.11% | 100% | 63.15% |
| Car Rental | 80% | 87% | 100% | 88.89% |
| Bus Booking | 54.54% | 72.22% | 100% | 70.58% |
| Medical Services | 72.72% | 83.33% | 100% | 84.21% |
| Weather Forecast | 85.71% | 90.91% | 100% | 92.35% |
| Property Visit | 38.46% | 68% | 100% | 55.55% |
| **Average** | 60.82% | 74.83% | 100% | **74.58%** |

Table 6.4: Results of the second validation approach with n=2.

- Same as the previous one, but only with the features suggested by the participants at least three times (n=3).

| Moment Dataset | Precision | Accuracy | Recall | F-Measure |
|---|---|---|---|---|
| Hotel Booking | 88.90% | 95% | 100% | 94.12% |
| Train Booking | 63.63% | 80.95% | 100% | 77.77% |
| Restaurant Booking | 72.72% | 83.33% | 100% | 84.21% |
| Flight Booking | 75% | 89% | 100% | 85.71% |
| Car Rental | 88.90% | 93.33% | 100% | 94.12% |
| Bus Booking | 60% | 76% | 100% | 75% |
| Medical Services | 88.90% | 94.73% | 100% | 94.12% |
| Weather Forecast | 100% | 100% | 100% | 100% |
| Property Visit | 45.45% | 75% | 100% | 62.50% |
| **Average** | 75.94% | 87.48% | 100% | **85.28%** |

Table 6.5: Results of the third validation approach with n=3.

- Same as the previous one, but only with the features suggested by the participants at least five times (n=5).

| Moment Dataset | Precision | Accuracy | Recall | F-Measure |
|---|---|---|---|---|
| Hotel Booking | 100% | 100% | 100% | 100% |
| Train Booking | 77.80% | 89.47% | 100% | 87.51% |
| Restaurant Booking | 80% | 89% | 100% | 88.89% |
| Flight Booking | 85.71% | 94.44% | 100% | 92.31% |
| Car Rental | 100% | 100% | 100% | 100% |
| Bus Booking | 75% | 88.23% | 100% | 85.71% |
| Medical Services | 100% | 100% | 100% | 100% |
| Weather Forecast | 100% | 100% | 100% | 100% |
| Property Visit | 45.45% | 75% | 100% | 62.50% |
| **Average** | 84.88% | 92.90% | 100% | **90.77%** |

Table 6.6: Results of the fourth validation approach with n=5.

## 6.5 Results Analysis

Since the planning phase of this internship, one of the risks associated with this approach was the high degree of novelty that we faced. Consequently, the lack of ideal results or validation strategies confronted configured a tremendous challenge in terms of complexity and planning, and both implementation and validation phases were directly influenced. Additionally and as previously stated, due to reasons of time and resources, the incapability of testing the performance of each separately component of the approach pipeline affected our task even more. Despite these hurdles, a validation strategy focused on the comparison

of templates automatically created by the developed framework with templates manually annotated by participants was applied. Four variants constituted this strategy, with the aggregate results enabling us to infer some conclusions and judgments why we reached certain values.

Taking into account that we throughout the requirement assessments imposed an ambitious objective that was implementing a pipeline with plenty of identified risks and simultaneously inflicted a minimum performance value set as 70%, validating our approach and obtaining for each variation an f-measure superior translated in a successful approach by our part. A successful approach means that it is in accordance with the functional and non-functional requirements in terms of quality.

However, there are still some details to discuss. As it is possible to see, for the last three variants, as the $n$ target increases, the results exponentially improve. This improvement was due to the high concordance value between participants. Participants do not tend to vary their response much, so the most common features tend to have many occurrences. They end up being focused on what is most obvious, which is what the model gets best, and therefore the high results are justified. Since the number of occurrences is almost homogeneous, and without significant discrepancies, the calculation of False Negatives tends to be zero. Consequently, the recall metric tends to the maximum percentage.

Moreover, the first validation approach based on the average results between annotators, despite having obtained an f-measure value of 78%, was hampered by some components of the pipeline. For instance, mainly problems in the extraction of triples when utterances hold poorly grammatical forms or spurious words considerably affect the results.

This page is intentionally left blank.

# Chapter 7

# Conclusion

In this chapter, the main contributions produced during the internship will be stated as well as recapping the approach implemented and the obtained results from it. We will also mention the challenges faced and how we managed to surpass them. Ultimately, we will address, in the context of dialogue-conversations, future, and potential work, that deserves further investigation.

The main objective of this internship was the development of a framework proficient of automatically generating structural summaries, or as we refer templates, from dialogue transcripts between a call center agent and a potential customer. With the capability to optimize the experience of customers and improve the assistance process by reducing after call-work, we went into this experience with great ambition and enthusiasm. From my point of view, this objective was accomplished to the extent that we indeed obtained templates capable of reaching a level of concordance acceptable between users and different domain types.

Moreover, this internship had a strong investigative component that revealed it to be essential to overcome risks associated and steep paths ahead. Was addressed the challenges when dealing with the amount of information generated in the form of unstructured information, especially in dialogues. The lack of context, the ambiguity, misspellings, and phenomena cases are just a few characteristics that express the struggle and complexity when trying to processing automatically natural language and extract relevant information of a text. It required an in-depth research investment to acquire background knowledge about the existent techniques in NLP and Information Extraction (IE) areas that try to solve these challenges.

At an early stage of our work on the State of the Art, we familiarized and discussed the approach of works that extract content from dialogues, works that summarize dialogues, and even works that summarize based on information in the form of entities and relations. This exploratory work set an indispensable step to understanding which are the principal vulnerabilities, challenges, and strengths related to the purpose of this internship. This knowledge was vital for choosing a promising and appropriate implementation.

Another common significant challenge when dealing with innovative projects is to gather ground-truth data to use as input as well as a way of validating the results obtained, and the meantime without high

costs. After relentless searching and analyzing, we finally get consent on which dataset met the goals of this project.

Other complex challenges already stated during this thesis consisted of developing a framework that uses the minimum supervision or needs for specific configurations, the intern's experience with frameworks and libraries, the lack of validation options, among others. In short, there were some complex problems, but in general, the most part was mitigated.

## 7.1 Contributions

The main contributions resulting from this work were:

- The Background Knowledge and State of the Art in the field of NLP and Automatic Summarization.
- The creation of different datasets ranging from several domain types and with different types of approaches regarding triple extraction.
- The development of modules for tasks related to Anaphora Resolution, Clustering, Named Entity Recognition, Utterance, and Triple Relevance, Fact Extraction, and Feature Importance.
- The development of the Template Generation framework, which obtained interesting results and simulated adequately what we were trying to test.

## 7.2 Future Work

The results obtained with the developed framework are promising and reveal that a future product capable of generating automated templates from dialogues is possible. However, implement a full and generic system that guarantees exceptional results is still associated with some difficulties. For instance, assuming that the sentences are grammatically well-formed and well written, which never happens in a context of dialogue, translates into a huge performance discrepancy. It is necessary to deal with a set of complex linguistic phenomena that correlate great complexity. Ellipses, anaphoras, associate a question with a response are just some examples.

In the case of opting for a solution with triple extraction, this also needs some improvements, especially when dealing with utterances lacking grammatical forms. ClausIE exposed some challenges with these utterance types.

# References


[1] *Deep Learning Academy*, Accessed January 10, 2020. https://www.deeplearning-academy.com/p/ai-wiki-machine-learning-vs-deep-learning.

[2] *Avoma*, Accessed January 3, 2020. https://www.avoma.com/.

[3] *Dialpad*, Accessed January 3, 2020. https://www.dialpad.com/features/voice-intelligence/.

[4] *RaeNotes*, Accessed January 3, 2020. https://www.raenotes.com/.

[5] *Reason8 AI*, Accessed January 3, 2020. https://reason8.ai/ph/?ref=producthunt.

[6] *Sonia AI*, Accessed January 3, 2020. https://sonia.com/features#.

[7] *Voicea*, Accessed January 3, 2020. https://www.voicea.com.

[8] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July 2015. Association for Computational Linguistics.

[9] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

[10] Francisco Azuaje, Ian Witten, and Frank E. Witten ih, frank e: Data mining: Practical machine learning tools and techniques. *Biomedical Engineering Online - BIOMED ENG ONLINE*, 5:1–2, 01 2006.

[11] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. 1997.

[12] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. Scrum: An extension pattern language for hyperproductive software development, 2000.

[13] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 2009.

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[15] Giuseppe Bonaccorso. *Machine learning algorithms: popular algorithms for data science and machine learning*. Packt, 2018.

[16] Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, and Antonio Feraco. *A Practical Guide to Sentiment Analysis*. Springer Publishing Company, Incorporated, 1st edition, 2017.

[17] Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 829–833, Beijing, China, July 2015. Association for Computational Linguistics.

[18] Goutam Chakraborty. Analysis of unstructured data: Applications of text analytics and sentiment mining. 2014.

[19] Christopher Cherpas. Natural language processing, pragmatics, and verbal behavior. *The Analysis of verbal behavior*, 10:135–47, 04 1992.

[20] R. Chopra. *Artificial Intelligence*. S CHAND & Company Limited, 2012.

[21] Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 2000.

[22] John M. Conroy and Dianne P. O'Leary. Text summarization via hidden markov models and pivoted qr matrix decomposition. 2001.

[23] Luciano Del Corro and Rainer Gemulla. Clausie: Clause-based open information extraction. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 355–366, New York, NY, USA, 2013. Association for Computing Machinery.

[24] H. P. Edmundson. New methods in automatic extracting. *J. ACM*, 16(2):264–285, April 1969.

[25] Jacob Eisenstein. *Introduction to natural language processing*. The MIT Press, 2019.

[26] Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*, 2019.

[27] Alexander Franz. *Automatic Ambiguity Resolution in Natural Language Processing: An Empirical Approach*. Springer-Verlag, Berlin, Heidelberg, 1996.

[28] Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. MinIE: Minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[29] George Giannakopoulos and Vangelis Karkaletsis. Summary evaluation: Together we stand npowered. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, CICLing'13, pages 436–450, Berlin, Heidelberg, 2013. Springer-Verlag.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[31] Nitin Hardeniya. *Nltk essentials: build cool NLP and machine learning applications using NLTK and other Python libraries*. Packt Publishing, 2015.

[32] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):607–616, June 1996.

[33] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[34] Jerry R. Hobbs. Coherence and coreference*. *Cognitive Science*, 3(1):67–90, 1979.

[35] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, October 2009.

[36] Karen Sparck Jones. Automatic summarising: factors and directions. *CoRR*, cmp-lg/9805011, 1998.

[37] Daniel Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson, 2009.

[38] Daniel Jurafsky and James H. Martin. *Speech and language processing*, volume 3. Pearson, 2014.

[39] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[40] Davis King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 07 2009.

[41] David Kriesel. *A Brief Introduction to Neural Networks*. 2007.

[42] Ela Kumar. *Natural language processing*. I. K. International Publishing House Pvt. Ltd., 2012.

[43] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 68–73, New York, NY, USA, 1995. ACM.

[44] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.

[45] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, page 24–26, New York, NY, USA, 1986. Association for Computing Machinery.

[46] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[47] Yang Liu, Kun Han, Zhao Tan, and Yun Lei. Using context information for dialog act classification in DNN framework. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[48] H. P. Luhn. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, April 1958.

[49] B. Marr. A short history of machine learning - every manager should read. *Forbes*, February 2016.

[50] Stephen R. Marsland. Machine learning: An algorithmic perspective, second edition. 2014.

[51] Ryo Masumura, Setsuo Yamada, Tomohiro Tanaka, Atsushi Ando, Hosana Kamiyama, and Yushi Aono. Online call scene segmentation of contact center dialogues based on role aware hierarchical lstm-rnns. pages 811–815, 11 2018.

[52] Mausam Mausam. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 4074–4077. AAAI Press, 2016.

[53] Donald Michie, D. J. Spiegelhalter, C. C. Taylor, and John Campbell, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, USA, 1995.

[54] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[55] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[56] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[57] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *CoRR*, abs/1611.04230, 2016.

[58] Ani Nenkova and Kathleen McKeown. *Automatic summarization*. Now Publ., 2011.

[59] Miles Osborne. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, AS '02, pages 1–8, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[60] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[61] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[62] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[63] James L. Peterson. Computer programs for detecting and correcting spelling errors. *Commun. ACM*, 23(12):676–687, December 1980.

[64] Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. Gecor: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue, 2019.

[65] Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing  Management*, 40(6):919 – 938, 2004.

[66] Vipul Raheja and Joel Tetreault. Dialogue Act Classification with Context-Aware Self-Attention. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3727–3733, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[67] Sebastian Raschka and Vahid Mirjalili. *Python machine learning machine learning and deep learning with Python, scikit-learn, and TensorFlow*. Packt Publishing, 2018.

[68] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*, 2019.

[69] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[70] Linda Rising and Norman S. Janoff. The scrum software development process for small teams. *IEEE Softw.*, 17(4):26–32, July 2000.

[71] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[72] Joerg Sander. *Density-Based Clustering*, pages 270–273. Springer US, Boston, MA, 2010.

[73] Sunita Sarawagi. Information extraction. *Found. Trends Databases*, 1(3):261–377, March 2008.

[74] Kristie Seymore, Andrew McCallum, and Ronald Rosenfeld. Learning hidden markov model structure for information extraction. January 1999.

[75] Ian Sommerville. *Software engineering*. Addison-Wesley, 2011.

[76] Josef Steinberger and Karel Jezek. Evaluation measures for text summarization. *Computing and Informatics*, 28:251–275, 2009.

[77] Philippe Thomas. Semi-supervised learning by olivier chapelle, bernhard schölkopf, and alexander zien (review). *IEEE Trans. Neural Networks*, 20(3):542, 2009.

[78] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March 2002.

[79] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[80] Chien-Sheng Wu, Andrea Madotto, Zhaojiang Lin, Peng Xu, and Pascale Fung. Getting to know you: User attribute extraction from dialogues, 2019.

[81] Koichiro Yoshino, Shinsuke Mori, and Tatsuya Kawahara. Spoken dialogue system based on information extraction using similarity of predicate argument structures. In *Proceedings of the SIGDIAL 2011 Conference*, pages 59–66, Portland, Oregon, June 2011. Association for Computational Linguistics.

[82] Dong Yu and Li Deng. *Automatic Speech Recognition*. Springer London, 2015.

# Appendices

This page is intentionally left blank.

# Appendix A

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sed efficitur magna, et laoreet elit. Praesent sed nulla est. Cras in sem nec sem posuere hendrerit eget eu ligula. Duis tincidunt, lacus quis sollicitudin porttitor, nibh lorem tempor elit, nec tempus nunc ante id felis.

This page is intentionally left blank.

## Appendix B

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum sed efficitur magna, et laoreet elit. Praesent sed nulla est. Cras in sem nec sem posuere hendrerit eget eu ligula. Duis tincidunt, lacus quis sollicitudin porttitor, nibh lorem tempor elit, nec tempus nunc ante id felis.