Maria Fernandes Pedroso

OPINION MINING FRAMEWORK

1 2 9 0 UNIVERSIDADE D COIMBRA

UNIVERSIDADE D
COIMBRA

Maria Fernandes Pedroso

OPINION MINING FRAMEWORK

Internship Report in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Paulo Pereira de Carvalho (DEI) and Alex Santos (Talkdesk) and presented to
Faculty of Sciences and Technology / Department of Informatics Engineering.

October 2020

This page is intentionally left blank.

Abstract

Nowadays, the amount of information each person produces is astounding. This results in the accumulation of large amounts of data. Quite a relevant amount of this data can be useful for companies if analyzed properly. Currently, when we purchase or use a product, it is common for us to leave feedback in some form, either by surveys, blog or forum posts, or just a small text on the web page we purchased the article from.

For a company, knowing what their potential customers are looking for, what they like or dislike is invaluable, as it would enable them to develop products that could become more appealing. Assuming we are talking about a contact center associated with a company, if conversations are stored in text format, we could obtain extremely relevant information about the company's products and image. However, given the usually large number of calls that are handled in a contact center, it is extremely costly both in terms of time and effort to actually assess the overall opinions expressed on the existing data.

The creation of a tool that would allow us to automatically assess the opinions of customers, concerning different products and their respective features, would go a long way to efficiently collect information on what aspects customers like and dislike on products. This would allow companies to improve their products based on that feedback.

One of the main problems of developing such a tool is the lack of annotated dialogue datasets for opinion mining. As such, one of the objectives of this internship is to create an opinion mining dialogue dataset. Due to the scarcity of dialogue datasets for opinion mining, state-of-the-art approaches for opinion mining are mostly tested on datasets from reviews. These datasets are very different in structure to what we would ideally want. Therefore, we propose to test some state-of-the-art opinion mining approaches on the new dataset that will be developed and thoroughly analyze them. Due to the different intended use case of the approaches, lackluster results are expected. As an optional goal, we will be attempting to improve the performance of the most promising analyzed approaches.

Keywords

Natural Language Processing, Sentiment Analysis, Opinion Mining

This page is intentionally left blank.

Resumo

Nos dias correntes, a quantidade de informação que cada pessoa produz é estonteante. Este facto resulta numa acumulação de grandes volumes de informação. Uma parte relevante dessa informação pode ser extremamente útil para empresas, se for analisada corretamente. Atualmente, quando adquirimos ou utilizamos um produto, é comum deixar alguma forma de crítica, tanto na forma de questionários, *blogs* ou *posts* num fórum, ou até só um pequeno texto na página da plataforma de onde adquirimos o produto.

Para uma empresa, saber o que os seus potenciais clientes procuram, o que gostam ou o que não gostam é informação extremamente valiosa, uma vez que permitiria à empresa desenvolver produtos que seriam mais apelativos. Se estivermos a falar de *call centers* associados a empresas, e se as conversas telefónicas forem guardadas em formato de texto, seria possível obtermos informações relevantes sobre os produtos da empresa ou da sua imagem. No entanto, dado o grande número the chamadas que são tratadas num *call center*, é extremamente dispendioso tanto em termos de tempo e esforço analisar as opiniões existentes nos dados.

A criação de uma ferramenta que nos permita avaliar automaticamente a opinião de clientes, sobre diferentes produtos e as suas respectivas características, seria uma grande contribuição na recolha de informação sobre que aspetos os clientes gostam ou não nos produtos. Isto iria permitir que as empresas possam melhorar os seus produtos com base nesses comentários.

Um dos principais problemas no desenvolvimento de tal ferramenta é a escassez de dados anotados de diálogo para opinion mining. Assim sendo, um dos objetivos deste estágio é criar um *dataset* de opinion mining com diálogos. A falta de dados anotados de opinion mining com diálogos resulta em abordagens que apresentam bons resultados maioritariamente para **datasets** de *reviews*. Estes *datasets* s ão muito diferentes do que nós desejaríamos ter idealmente. Assim sendo, propomos testar algumas abordagens com bons resultados em opinion mining no novo dataset que iremos criar e realizaremos uma análise das mesmas. Devido à diferença entre o caso de uso pretendido das abordagens e o caso de uso a testar, é de esperar que os resultados sejam inferiores para esta situação. Como objetivo opcional, iremos tentar melhorar a *performance* das abordagens mais promissoras.

Palavras-Chave

Processamento de Linguagem Natural, Análise de Sentimento, Extração de Opinião

This page is intentionally left blank.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Internship Objectives . . . 2
- 1.3 Document Structure . . . 3

2 Background Knowledge **5**
- 2.1 Machine Learning . . . 5
  - 2.1.1 Approaches . . . 5
  - 2.1.2 Problems . . . 6
  - 2.1.3 Algorithms . . . 7
- 2.2 Natural Language Processing . . . 9
  - 2.2.1 Linguistics and Natural Language Processing . . . 10
  - 2.2.2 NLP Tasks . . . 12
  - 2.2.3 Information Extraction . . . 19
  - 2.2.4 Topic Modeling . . . 20
- 2.3 Sentiment Analysis . . . 20
  - 2.3.1 Levels of Sentiment Analysis . . . 21
  - 2.3.2 Approaches for Sentiment Analysis . . . 22
  - 2.3.3 Challenges in Sentiment Analysis . . . 23
- 2.4 Opinion Mining . . . 25
  - 2.4.1 Defining Opinion . . . 25
  - 2.4.2 Aspect Extraction . . . 25
  - 2.4.3 Aspect Sentiment Classification . . . 27
- 2.5 Evaluation Metrics . . . 27
  - 2.5.1 Performance Evaluation Metrics . . . 28
  - 2.5.2 Inter-Annotator Metrics . . . 30
  - 2.5.3 Text Similarity Metrics . . . 32

3 Related Work **33**
- 3.1 Sentiment Analysis . . . 34
- 3.2 Opinion Mining . . . 35
- 3.3 Competitors Analysis . . . 38
  - 3.3.1 Main Competitors . . . 38
  - 3.3.2 APIs . . . 39
- 3.4 Resources and Tools . . . 41
  - 3.4.1 Libraries . . . 41
  - 3.4.2 Datasets . . . 42
  - 3.4.3 Other Resources . . . 44

4 Approach **45**
- 4.1 Specification . . . 45

4.1.1 Datasets . . . 45
4.1.2 Dataset Validation . . . 48
4.1.3 Selection of Approaches . . . 48
4.1.4 Evaluation of Approaches on the New Dataset . . . 49
4.1.5 Improvement of the Approaches on the New Dataset . . . 50
4.1.6 Implementation of a BERT aspect-based sentiment classifier . . . 50
4.2 Requirements Analysis . . . 50
4.2.1 Functional Requirements . . . 51
4.2.2 Non-functional Requirements . . . 51
4.2.3 Technological Requirements . . . 52
4.2.4 Language Requirements . . . 52
4.3 Risk Analysis . . . 52
4.3.1 Risk Description . . . 52
4.3.2 Risk Presentation . . . 53
4.4 Planning . . . 54
4.4.1 Planning: 1st Semester . . . 55
4.4.2 Planning: 2nd Semester . . . 56

5 Implementation **58**
5.1 Dataset Development . . . 58
5.1.1 Dataset Format . . . 58
5.1.2 Dataset Construction . . . 59
5.1.3 Dataset Improvement . . . 60
5.2 Dataset Annotation Validation . . . 63
5.2.1 1st Dataset Annotation Validation . . . 63
5.2.2 2nd Dataset Annotation Validation . . . 63
5.3 Adaptation of Selected Approaches . . . 63
5.4 Improvement of Selected Approaches . . . 64
5.5 Implementation of an aspect-based sentiment classification BERT model . . . 65

6 Results **66**
6.1 Dataset Validation Results . . . 66
6.1.1 First Validation Results . . . 66
6.1.2 Second Validation Results . . . 70
6.2 Approaches' Evaluation Results . . . 78
6.2.1 RINANTE Evaluation Results . . . 78
6.2.2 DOER Evaluation Results . . . 80
6.3 Approach Improvement Results . . . 81
6.3.1 RINANTE Improvement Results . . . 81
6.3.2 DOER Improvement Results . . . 82
6.4 BERT Models Validation Results . . . 83

7 Conclusion **92**

Acronyms

AI Artificial Intelligence. 1, 2, 5, 9

ML Machine Learning. 2, 3, 5, 7, 42

NER Named Entity Recognition. 14, 19, 42

NLP Natural Language Processing. vii, 1–3, 5, 9, 11, 12, 19, 20, 26, 33, 41, 42, 44, 50, 54, 55, 84

OM Opinion Mining. 3, 5

POS Part of Speech. 9, 17, 18, 26, 34, 38, 42, 60

SA Sentiment Analysis. 3, 5

This page is intentionally left blank.

List of Figures

2.1 Machine learning approaches and problems. 7
2.2 Main levels of knowledge in linguistics. 10
2.3 Example of a tokenization task. 12
2.4 Example of a sentence segmentation task. 13
2.5 Example of text segmentation into n-grams. 13
2.6 Example of spell checking. 13
2.7 Example of word lemmatization. 14
2.8 Example of word stemming. 14
2.9 Example of named entity recognition. 15
2.10 Example of a stop word identification task. 15
2.11 Example of a dependency parsing task. 16
2.12 Example of a constituency parsing task. 17
2.13 Example of a POS tagging task. 17
2.14 Example of a chunking task. 18
2.15 Example of a chinking task. 19
2.16 Levels of Sentiment Analysis. 21

4.1 Risk matrix. 55
4.2 Internship plan for the 1st semester. 56
4.3 Internship plan for the 2nd semester. 57

This page is intentionally left blank.

List of Tables

2.1 Table of possible opinion mining models' prediction outcomes 28

3.1 APIs functionalities table 41
3.2 Libraries functionalities table 43

5.1 Problems detected in dataset 61
5.2 Solutions presented for corrections in dataset annotation process 61

6.1 Description of the dataset sample generated for the first stage of validation. 67
6.2 Results of analysis of agreement between all annotators from group 1 for the labeling of the new dataset in the first validation phase. 67
6.3 Results of a detailed analysis of the agreement between all annotators for the labeling of aspects in the new dataset in the first validation phase. . . . 67
6.4 Results of a detailed analysis of the agreement between all annotators for the labeling of sentiments in the new dataset in the first validation phase. . 68
6.5 Results of analysis of agreement between different annotator pairings for the only group in the first validation phase. 68
6.6 Results of a detailed analysis of the agreement between all annotator pairs for the labeling of aspects in the new dataset in the first validation phase. . 69
6.7 Results of a detailed analysis of the agreement between the annotator pairs for the labeling of sentiments in the new dataset in the first validation phase. 69
6.8 Overall description of the dataset sample gathered for the first group of raters from the second validation phase. 70
6.9 Results of analysis of agreement between all annotators from group 1 for the labeling of the new dataset in the second validation phase. 70
6.10 Results of a detailed analysis of the agreement between all annotators from group 1 for the labeling of aspects in the new dataset in the second validation phase. 71
6.11 Results of a detailed analysis of the agreement between all annotators of group 1 for the labeling of sentiments in the new dataset in the second validation phase. 71
6.12 Results of analysis of agreement between the different annotator pairings of group 1 in the second validation phase. 71
6.14 Results of a detailed analysis of the agreement between all annotator pairs in group 1 for the labeling of aspects in the new dataset in the second validation phase. 72
6.15 Results of a detailed analysis of the agreement between the annotator pairs in group 1 for the labeling of sentiments in the new dataset in the second validation phase. 72
6.16 Overall description of the dataset sample gathered for the second group of raters from the second validation phase. 73

6.17 Results of analysis of agreement between all annotators from group 2 for the labeling of the new dataset in the second validation phase. 73
6.18 Results of a detailed analysis of the agreement between all annotators from group 2 for the labeling of aspects in the new dataset in the second validation phase. 73
6.19 Results of a detailed analysis of the agreement between all annotators of group 2 for the labeling of sentiments in the new dataset in the second validation phase. 73
6.20 Results of analysis of agreement between different annotator pairings for the second group in the second validation phase. 74
6.21 Results of a detailed analysis of the agreement between all annotator pairs in group 2 for the labeling of aspects in the new dataset in the second validation phase. 74
6.22 Results of a detailed analysis of the agreement between the annotator pairs in group 2 for the labeling of sentiments in the new dataset in the second validation phase. 74
6.23 Overall description of the dataset sample gathered for the third group of raters from the second validation phase. 75
6.24 Results of analysis of agreement between all annotators from group 3 for the labeling of the new dataset in the second validation phase. 75
6.25 Results of a detailed analysis of the agreement between all annotators from group 3 for the labeling of aspects in the new dataset in the second validation phase. 75
6.26 Results of a detailed analysis of the agreement between all annotators of group 3 for the labeling of sentiments in the new dataset in the second validation phase. 75
6.27 Results of analysis of agreement between different annotator pairings for the third group in the second validation phase. 76
6.28 Results of a detailed analysis of the agreement between all annotator pairs in group 3 for the labeling of aspects in the new dataset in the second validation phase. 76
6.29 Results of a detailed analysis of the agreement between the annotator pairs in group 3 for the labeling of sentiments in the new dataset in the second validation phase. 76
6.30 Results of precision, recall and f-score metrics for RINANTE approach with baseline parameters and training using the Semeval 2014 datasets for Levenshtein distance$<$2 and cosine similarity$\geq$75% as the evaluation functions . 80
6.31 Results of precision, recall and f-score metrics for DOER approach with baseline parameters and training using the Semeval 2014 datasets for Levenshtein distance$<$2 and cosine similarity$\geq$75% as the evaluation functions . 81
6.32 Results of precision, recall and f-score metrics for sentiment for DOER approach with baseline parameters and training using the Semeval 2014 datasets for Levenshtein distance$<$2 and cosine similarity$\geq$75% as the evaluation functions 82
6.33 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=140, learn rate=0.0008 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance$<$2 and cosine similarity$\geq$75% as the evaluation functions. 83

6.34 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=140, learn rate=0.001 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 84
6.35 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=140, learn rate=0.0012 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 85
6.36 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=170, learn rate=0.0008 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 86
6.37 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=170, learn rate=0.001 and batch size=[32, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 86
6.38 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=170, learn rate=0.0012 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 87
6.39 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=200, learn rate=0.0008 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 88
6.40 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=200, learn rate=0.001 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 89
6.41 Results of precision, recall and f1-score metrics for RINANTE approach with epoch=200, learn rate=0.0012 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions. 90
6.42 Results of aspect and polarity detection using a BERT model trained with the Semeval 2014 Laptops dataset. 90
6.43 Results of aspect and polarity using a BERT model trained with the Semeval 2014 Restaurants dataset. 91

This page is intentionally left blank.

Chapter 1

Introduction

This document reports the author's internship at Talkdesk, which focuses on the creation of a dataset and the comparison of a few approaches used for *Opinion Mining*. Next, we explain the motivation for this work and define the objectives we want to accomplish. We finish with a brief description of the document structure.

1.1 Motivation

Currently, large volumes of data are produced daily by individuals that use the *Internet*. Companies have realized, for quite some time, that if they manage to analyze the data and extract useful information from it, they can make a lot of money by developing their products according to the feedback they managed to extract from said data, and thus increase interest in their products.

Places where there is an abundance of data are contact centers. Most companies have them and it is an important point of contact with their clients. Due to the current competitive market, knowing what people feel about something is invaluable. With so many existing digital channels, there are multiple ways for people to express their feelings. However, this data will be useless if the businesses cannot process and understand it. Understanding the data will go a long way towards improving their products, know about the customers and even make better company decisions. As mentioned previously, a contact center is a point of contact between companies and their customers. From a company standpoint, it can be a very important channel that enables businesses to understand how their customers feel about them. The reason why it is important to know the sentiment of the customers that reach a contact center is that it can help the agents deal with situations or help company management to know how the company is viewed by their customers.

In contact centers, most people convey their opinion using natural language. Quite a significant amount of that review data is in text form, hence the need of applying Natural Language Processing (NLP) (Jurafsky and Martin, 2019) and Artificial Intelligence (AI) to process all the available information and automatically extract the opinions expressed by customers about entities, regardless of whether they are products, companies or brands and identify what aspects from those entities give origin to that sentiment.

This is where Talkdesk comes in. Talkdesk is a company that provides cloud-based contact center infused with Artificial Intelligence software. Talkdesk focuses on improving contact center solutions for businesses by giving them access to multiple modules that they can

integrate with their cloud contact center solution. Talkdesk aims to help companies make customer experience a competitive advantage by allowing them to adapt their contact centers according to their needs. As these needs may change due to a business's evolution, the adaptation of the contact center solution is a great selling point to attract potential clients (along with the fact that the solution being cloud-based has no requirement for physical setup and system updates supported by the company itself).

One such module is the recently presented *Agent Assist*[1]. The opinion mining service developed in this internship will be integrated in it. *Agent Assist*[2] is a module that uses AI to give guidance to agents, help them resolve customer issues and extract important information from the exchange between customers and agents (such as the opinion of customers concerning the company or their products). Due to the current competitive market and large amounts of information existing in many businesses' contact centers, Talkdesk decided that they wanted to develop a opinion mining module that would help companies obtain feedback on what their customers think of them and of their products. This could potentially allow these enterprises to improve themselves and their products based on the collected information and increase customer satisfaction (and could even lead to an expansion of their customer base).

Consider the following example: in a certain communication services company, an agent is in a call with a client. In that call, the agent asks the client what services the client is subscribed to and what are his thoughts on the services he is currently using. The client answers the questions asked by the agent, who then would have to store the relevant information somewhere where other people from the company would be able to access and analyze the information. Using an opinion mining module, *Agent Assist* would go over the text generated from the call and automatically extract the key aspects that were mentioned during the conversation and what were the client's opinions on them. This process, repeated for multiple calls would save large amounts of time and bring the additional advantage of rendering all the data collected uniform, making it more readable for anyone who would later access it. However, in order to implement a state-of-the-art Machine Learning (ML) solution, large quantities of annotated data are required. There is a severe lack of datasets available to train solutions for such problems because most opinion mining datasets are created from review data, and not actual conversations. As such, the creation of an opinion mining dataset for a similar context such as the one previously mentioned, would prove to be invaluable for the further development of this branch of NLP problems.

Most state-of-the-art opinion mining solutions are trained on datasets built with review data. However there is a difference between the intended use case for these solutions and the use case we intend to apply *Agent Assist* on. Because of this difference, before we can actually develop a solution for our use case, we must first study how some previously developed opinion mining solutions would work on datasets more aligned to our use case. Which leads us to a critical goal, the analysis and comparison of state-of-the-art approaches on a dataset more fit for our use case.

1.2 Internship Objectives

The main objective of this internship is the creation of a new Opinion Mining dataset and perform a comparison of the current state-of-the-art approaches on the newly developed dataset. If there is enough time remaining before the end of the internship, we will attempt

[1] https://www.nojitter.com/customer-experience/agent-assist-powered-talkdesk-iq
[2] https://www.talkdesk.com/call-center-software/ai-automation/agent-assistance/

to improve one of the tested implementations. Summarizing, our internship objectives can be broken down as the follow:

- **Analysis of state-of-the-art approaches, existing technologies and competitors** - This goal is important in order for us to understand what is being currently used in terms of opinion mining, what are the most used approaches.
- **Selection, study and comparison of different promising implementations** - This is one of our main objectives. We will perform a selection between some promising approaches and based on their performance, advantages and limitations, select the ones that would be more adequate for us to compare in a thorough level, in a context more similar to our use case.
- **Development of a new Opinion Mining dataset** - The main use case of opinion mining in Talkdesk would be extracting customer opinions from dialogue transcripts in phone conversations. However, to the best of our knowledge there are no Opinion Mining datasets currently available for dialogues, at least to the public. Therefore, the creation of an Opinion Mining dialogue dataset is also one of the main objectives of this internship.
- **Comparison and testing of the approaches selected** - The approaches must be tested and the necessary adjustments must be made in the code of each approach in order to be able to test them on the new dataset. We will also experiment to see if adjusting the parameters of the approaches can yield better results. Since the dataset developed is in an early stage, we will also perform a detailed analysis of the false positives detected by the approaches. They may be relevant for opinion mining but not originally annotated in the dataset because of they method used to build it.
- **Building BERT models for aspect-based sentiment classification** - Optionally, if time allows it, we will attempt to develop a BERT model for aspect-based sentiment classification using a pre-trained BERT and fine-tuning. Due to the fact that only one of the approaches performs sentiment classification and that the approaches selected to evaluate all resort to the long process of training neural networks, we decided to try implement this model using a BERT model and fine-tuning it for our purpose. The fine-tuning was performed with the train *Semeval 2014 laptops* and *restaurants* datasets and the evaluation of the models trained with each dataset will then be evaluated using their respective test datasets.
- **Writing the internship report** - Since this is an internship for a Master's degree, the writing of the internship report is also a critical objective for the conclusion of this internship.

1.3 Document Structure

This document is divided into seven chapters. The content of each chapter is described further down.

Chapter 1 - This is the introductory chapter.

Chapter 2 - This chapter details the state-of-the art. It starts with the background knowledge on Machine Learning (ML), Natural Language Processing, Sentiment Analysis, Opinion Mining and evaluation metrics.

Chapter 3 - This chapter details the related work on sentiment analysis and opinion. Afterwards, we have a competitor analysis, and we conclude this chapter with the currently available resources and tools.

Chapter 4 - This chapter is about the proposed approach and planning being followed for the internship. We start with the proposed approach, where we present the details of the approach we selected. We continue with the requirements analysis. Following that, we present the possible risks of this project.

Chapter 5 - This chapter details the work performed during the internship. We start by presenting the dataset we created and all the steps taken in its development and we follow with the presentation of the approaches selected to test with the newly created dataset and relevant changes performed on the approaches.

Chapter 6 - This chapter presents the validation results of the dataset created and the test results of the approaches tested on our dataset.

Chapter 7 - This is the last chapter. It presents the overall conclusions of this document and possible future work.

Chapter 2

Background Knowledge

The following section will present the most important concepts on the subjects most associated with the theme of this thesis. We start by introducing concepts in the area of Machine Learning (ML), then we approach Natural Language Processing (NLP), followed by Sentiment Analysis (SA), and we finally reach the theme of this thesis, Opinion Mining (OM). We finish this section with the Evaluation Metrics.

2.1 Machine Learning

Machine Learning (ML), according to (Marsland, 2014), is a branch of Artificial Intelligence (AI) that provides systems with the ability to learn and improve automatically without human intervention. This process requires data in order for the systems to learn and increase their accuracy, it is, therefore, very dependent of the data provided and without sufficient data this approach is not usable.

2.1.1 Approaches

Machine learning methods can be classified into supervised learning, unsupervised, semi-supervised and reinforcement learning categories. Each of these methods have different situations in which they should be used upon, depending on the problem we are trying to solve.

- **Supervised Learning** - In the context of ML, supervised learning (Jurafsky and Martin, 2019) is a type of learning approach where a system utilizes labeled data, in the form of input and output pairs, that are provided as a learning base. Using the data provided for training, the system is then expected to perform validation with different data. This type of learning approach is used for classification and regression problems, and these problems will be presented in section 2.1.2.
- **Unsupervised Learning** - Unsupervised learning (Jurafsky and Martin, 2019) is a learning approach where only input data is provided to the system. In this case, there is no labeled data and the system instead attempts to detect patterns in the provided data to make estimations on future data. Unlike supervised learning, unsupervised learning is not a valid approach for regression or classification problems, as there are no labeled outputs. It is instead used to detect patterns in data and gather it

based on similarities detected (clustering). The types of problems that are solved with unsupervised learning approaches will be presented further into this chapter, in section 2.1.2.

- **Semi-supervised Learning** - Semi-supervised learning (Chapelle et al., 2010) is a machine learning method halfway between supervised and unsupervised learning. It is a recent approach to machine learning that is mostly used in fields where we have large amounts of data and only some of it is labeled. One example of an application where semi-supervised learning is useful is bootstrapping. Bootstrapping (Jurafsky and Martin, 2019) takes a small set of data and tries to generalize the context around the data points from a larger dataset that is not annotated.

- **Reinforcement Learning** - Reinforcement Learning (Sutton and Barto, 2018) is a learning approach based on taking the correct actions to maximize the possible rewards of a given situation. Simply put, this method aims to find the best possible behavior/path it should take in a given situation. Similarly to the unsupervised approach, the data supplied to models that apply reinforcement learning consists only of input data, with no correct classification provided. The reinforcement learning system decides which actions to perform based on the data it is provided and the experience it obtained from processing previous data.

2.1.2 Problems

Machine Learning methods solve a multitude of problems (Nisbet et al., 2009) , with each problem being classified into a different type, depending on what the prediction task is. We provide an introduction to the most popular types of problems that Machine Learning is applied in. These problems are also presented in figure 2.1.

- **Classification Problems** - Classification Problems are a type of problem where a machine learning model must make a prediction from a given input into a set of categories, or discrete values, as output. These categories are classes and not numeric values. Examples of classification problems are: predicting the gender of a person based on their handwriting, or classification of types of crops. These types of problems are solved with supervised learning approaches.

- **Regression Problems** - Regression Problems are very similar to classification problem, the difference being the output of the system. Whereas in Classification Problems the system output is a class/category, in Regression Problems, the output is a real value. Some simple examples of regression problems include: predicting the age of a person, or predicting the salary of a given company employee. Similarly to classification problems, it is applied supervised machine learning approaches to solve these types of problems.

- **Clustering Problems** - Clustering problems have as a final objective for the system to find different groups within the data, gathering the data objects in groups, each with similarities between the different fields, while remaining separate from other data points with different characteristics that are placed in different groups (otherwise named clusters, hence the naming of this type of problems). Clustering problems are solved with unsupervised machine learning approaches, as the data provided for these problems does not come labeled with a correct classification and instead we are grouping data objects based on their similarities and separating them based on their differences.

- **Association Problems** - Association Problems are a type of problem in which we aim to discover rules that will describe the majority of our data. One such example would be: if a customer buys product X it is likely that he will buy product Y. These types of problems are also solved used unsupervised learning and are usually found in the area of commerce, in order to obtain more information on customer habits.

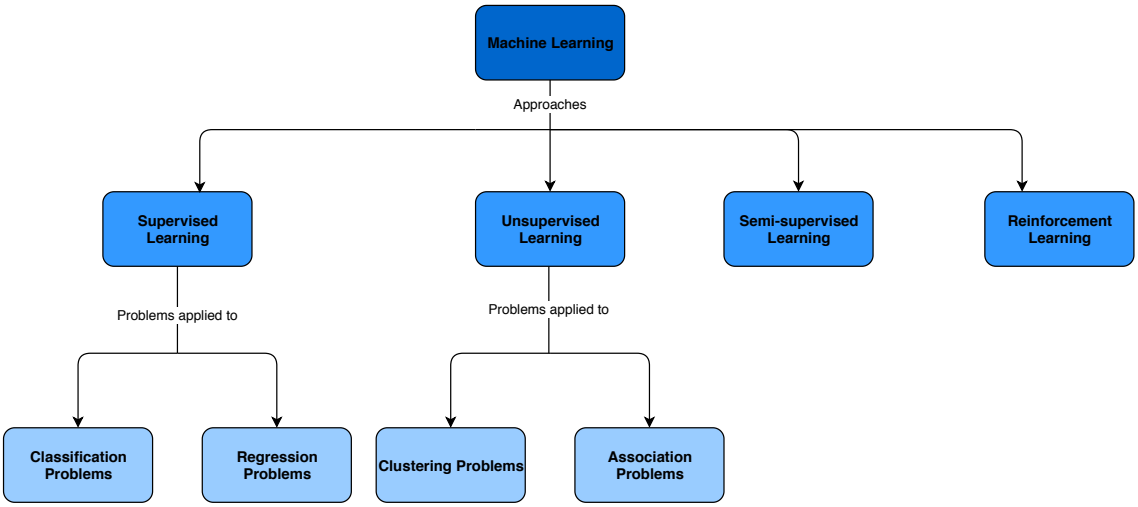

Figure 2.1: Machine learning approaches and problems.

2.1.3 Algorithms

In this section we will proceed to present some frequently used Machine Learning (ML) algorithms.

- **Logistic Regression** - Logistic regression (Nisbet et al., 2009) is a supervised learning approach that is used as binary class predictive model. It models the relationship between independent variables and uses a sigmoidal function to determine an outcome. The values returned from this function are probabilities values between 0 and 1, and the sum of the probabilities must be one. It is mostly used for categorical classification, rather than numerical.

- **Decision Trees** - Also a supervised learning approach, Decision Trees (Nisbet et al., 2009) are used as a predictive model in Machine Learning. In Decision Trees, each node represents a test on a data attribute, a branch represents an outcome of a test in a node and, each terminal node of a decision tree represents a classification label. Decision Trees are a model, built upon two stages: a growth stage and a pruning stage. In the growth stage, the decision tree algorithm finds at each tree node the best feature that discriminates between the different classes, and splits the data into two new nodes based on that feature. This process is recursive, and it keeps occurring until a leaf node classifying the data is achieved. The second stage, pruning, occurs when we destroy the least significant branches of a Decision Tree. This process is used to increase the accuracy of the model, while simplifying it at the same time. The two major strengths of Decision Trees are the effective handling of missing data and the ability to work with both numerical and categorical data.

- **Support Vector Machine** - The Support Vector Machine (Nisbet et al., 2009) approach is based on defining decision planes that define decision boundaries. A

decision plane separates data objects that are classified in different classes and it is built by supplying the model with training data. This particular model supports both regression and classification tasks and is capable of handling continuous and categorical values. We use kernels to rearrange the data objects in different dimension spaces, and we try to achieve the ideal decision plane with a linear function.

- **Naive Bayes** - The Naive Bayesian (Nisbet et al., 2009) classification approach takes into consideration past classifications in order to perform classification on new cases as they appear, using prior probabilities. Prior probabilities are based on evidence obtained from previous classifications. In order words, Naive Bayes performs class labeling based on the currently existing labeled object data. If we have more data objects of a given data classification, it is reasonable to assume that the next data object to classify is also likely to be classified with that classification label. This is the core principle behind Naive Bayes. A major shortcoming that is implied by its name, as it is built upon a naive assumption, is that the predictor variables of the model are independent in their effects on the classification tasks. Regardless of this, and despite its simplicity, this classifier performs well with different datasets, and accepts both continuous or categorical data.

- **Random Forest** - The Random Forest (Nisbet et al., 2009) approach to machine learning trains different trees on slightly different subsets of data (bootstrap samples). Each decision tree in the group of trees votes for the classification of the input case to be classified. This approach has many advantages, the ones most worth mentioning being the ability to handle large amounts of data and the ability to handle missing data values.

- **Deep Learning: Neural Networks** Neural networks (Nisbet et al., 2009) approaches are based on the function and structure of the human neurons. The architecture of this approach is constituted by input nodes that are connected to output nodes (one or more). The connection between these two types of nodes is normally affected by a function and an activation function. In Neural Networks it is possible to insert more nodes in the middle of the network, obtaining a middle layer of neurons. We can then assign weights to the connections between the input and middle layer nodes, and between the middle layer and output layer nodes. These nodes in the middle layer provide the ability to model the relationships between the input and output nodes. One of the most common problems of this approach is over-training a model, in which case, the model performs really well on training data, but his performance is lacking in another dataset. As there is no absolute set of rules to model neural networks, experimentation with different parameters is the only way to search for more efficient results. Neural Networks, unlike Decision Trees, are not a simple algorithm to understand and cannot be interpreted in a logical way in the context of a problem.

  - **Hyperparameters** (Goodfellow et al., 2016) are the settings that are used to control the behavior of learning algorithms used in deep learning. Some examples of hyperparameters are: *number of epochs*, *batch size* and *learning rate*.

    1. **Number of epochs**: The number of epochs Goodfellow et al., 2016 is the number of times the dataset is iterated upon by the network during training.
    2. **Batch size**: Batch size Goodfellow et al., 2016 is the number of samples that are given to a neural network to analyze before it updates its param-

eters. Smaller batch sizes make training slower, while the opposite, fastens the training process.

3. **Learning rate**: The learning rate (Mitchell, 1997) is the parameter used to control how much the weights are changed during each step of the learning process. It tends to be a small value, so as to not cause great shifts in the weights of a network and result in a chaotic weight value adjustments. Large values result in an almost random learning process. However, too small a learning rate will cause the neural network models to take longer to train. That is why we must choose this parameter carefully.

Recurrent neural networks (Vaswani et al., 2017) are a deep learning model that possesses its own memory. However that memory is dependent on the information it has gained from the previous network input and it is linearly dependent on immediately related inputs.

Unlike the recurrent neural networks which were the state-of-the-art approach until a while back, the **transformers** (Vaswani et al., 2017) are new deep learning models that have been proven to be effective in language understanding tasks. They have their own attention mechanism that allows them to observe more inputs than the ones immediately next to the current network, and their decision making process to select the best candidate from the ones he is connected to.

- **Hidden Markov Model** - The Hidden Markov Model (Rabiner and Juang, 1986), is a doubly stochastic model. This model contains unobservable (hidden) states and is followed by another stochastic process that depends on the previous unobservable state. This model is used to deal with situations where we have Markov models, but we do not know which state of the Markov model we are currently in. An example depicting a Hidden Markov Model is presented in (Marsland, 2014), where a teacher tries to guess what activity a student had been involved in a previous night, basing himself only on a observation from the current state of the student. These models are often employed in part-of-speech tagging and speech recognition tasks.

- **Conditional Random Fields** - Conditional Random Fields are probabilistic graphical models, often used in sequence labeling tasks such as Part of Speech (POS) tagging (Lafferty et al., 2001). Conditional Random Fields are commonly used for similar tasks because they are capable of taking context into account when performing labeling tasks. One advantage these models have over Hidden Markov Models is that they use more relaxed strong independence assumptions, which are often a source of error when we are trying to simulate a problem with models. This characteristic is quite useful in order to minimize errors when modeling a problem.

2.2 Natural Language Processing

Natural Language Processing (NLP) (Richards and Schmidt, 2013) is the analysis of human language by a computer. It is a subfield of computer science and computational linguistics and AI. NLP is used to process unstructured data. However, this method is not yet perfected. From simple Decision Trees and Support Vector Machine approaches, to the more complex Neural Networks, NLP has evolved. However, even with all these new approaches, the major problem of NLP is still the same: ambiguity. Ambiguity refers to the problem where a word has more than one different interpretation in a different context. For humans, that is a simple task, as we can understand certain aspects that enable us

to infer the context where the words/expressions are used. However, machines cannot understand context as we do, so they cannot handle ambiguity. Ambiguity is a problem that can occur in the different levels of knowledge found in Linguistics.

2.2.1 Linguistics and Natural Language Processing

Linguistics (Richards and Schmidt, 2013) is the study of language. It studies how language is put together and how it functions. Linguistics studies the different aspects that form a language, including the study of grammar, syntax, and phonetics. We will know present the main levels of knowledge in linguistics (Richards and Schmidt, 2013), as depicted in figure 2.2:

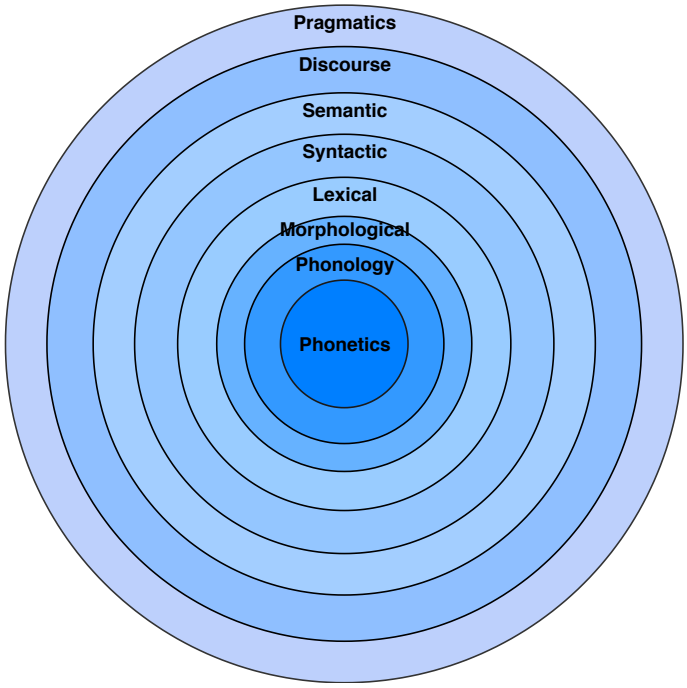

Figure 2.2: Main levels of knowledge in linguistics.

1. **Phonetics Level** - Phonetics is the study of speech sounds. It overlaps with phonology, however these two fields are not the same thing.

2. **Phonology Level** - Phonology is the study of the sound patterns that occur within a given language. Right from the start, we can find sources of ambiguity in this level. If we use the English language as an example, we know that some words, while meaning different things, are pronounced similarly. Some examples of this ambiguity are: *see* and *sea*.

3. **Morphological Level** - The morphological level is the level that deals with parts of words that carry significant meaning, such as suffixes and prefixes. In other words, this level focuses on the study of the rules for word formation.

4. **Lexical Level** - The lexical level is identification of words. In order to perform a lexical analysis of the words in a text, it is first necessary to perform a series of natural language processing operations on the text, which will be presented on section 2.2.2. Ambiguity can also occur at this level when a word has more than one meaning in the language used by the speaker.

5. **Syntactic Level** - Syntax analysis is the study of how sentences are constructed in languages. For the purpose of opinion mining and performing NLP tasks (see section 2.2.2), we will mostly focus how the arrangement of words in a sentence will allow us to determine the relationships between words. Another case of ambiguity may occur at this level. This happens when a sentence can have multiple meaning due to its structure. For example: "*He read the book on his desk*" could be interpreted as someone read a book that was located on a desk or someone read a book while sitting on their desk.

6. **Semantic Level** - Semantics is the linguistics branch that studies the meaning of words. Some words, while written and pronounced in the same way, have very different meanings and often belong to very different grammatical classes. In short, this type of ambiguity occurs when a word or utterance, without the extra information of context of use, possesses multiple interpretations. The simplest example illustrating this ambiguity are homonyms. Homonyms are words that have the same spelling and are pronounced the same way, but have different meanings. This level is what allows us to clear that ambiguity. For the purpose of opinion mining, we wish to extract characteristics of a given product brand or service and a person's opinion about said characteristics. This information is invaluable for companies, and it is no wonder that many big corporations invest in opinion mining to better determine customer reactions to products in the market and enable them to develop new products that would attract a great number of potential clients. In the domain of opinion mining we can define the opinion tuple as:

    - **Opinion tuple** - The tuple is composed by the characteristics of the product, with each characteristic having the associated opinion a customer expressed towards it, and the polarity of the customer towards that characteristic.

7. **Discourse Level** - The discourse level encompasses how sentences relate to one another. Discourse is very context-dependent, as it frequently requires all individuals involved in a conversation to possess some background knowledge in order to understand the entirety of a conversation. Simply transcribing a speech will not allow us to understand the entire idea from a conversation. The way the sentences are organized affects the ideas being conveyed and how others will interpret them. This level is also very important, especially since we are focusing on dialogues between different parties. The analysis of forms of address or coherence between utterances is extremely important when we are discussing spoken text. How a person addresses another, how they organize their sentences/speech can reveal in-depth information about the state of mind of the speaker.

8. **Pragmatics Level** - Pragmatics studies how natural language is used in communication. It looks at how both the literal and non-literal meanings of expressions are used in a given context to convey a message. It is different from semantics because it does not study the literal meaning of sentences, but instead focuses on studying what people actually mean when they use language to communicate, the context of use of the language. Just like semantic and discourse, this level is also important. How often have we heard someone say something that interpreted literally would mean one thing, but taking into account the way it is said the context in which it was uttered is different? One such example would involve the usage of sarcasm (when a speaker means the opposite of what he said). Taking into account a more concrete example: whether or not we liked a certain movie and we utter a sarcastic reply such as "*Oh yes, I loved it*", one would be led to believe that my opinion of the movie was positive based solely on the written text presented. However, the person who asked

us the question and heard our reply would be able to infer the true meaning behind our words and conclude that we did not like the movie.

2.2.2 NLP Tasks

Before we can extract information from a text, we must first process the text. In this section we will proceed to present the most important NLP tasks (Bird et al., 2009; Jurafsky and Martin, 2019) that are frequently used in text processing.

Text Normalization

Text Normalization (Jurafsky and Martin, 2019) consists of a set of operations used to convert a text into a more convenient standard form. The tasks included in text normalization are: **tokenization**, **sentence segmentation**, **lemmatization**, **stemming** and **spell checking** (these tasks will be defined below).

Tokenization

Tokenization (Jurafsky and Martin, 2019) is the task of separating a text into tokens. Tokens are strings of contiguous characters that are, usually, separated by whitespaces. However, that is not always the case. For example, when we have names of locations, such as *Las Vegas*, we do not treat it as two different tokens, instead we treat it as one. In figure 2.3, we have an example of tokenization. We need a text as input to execute tokenization. Once finished, we obtain a tuple with the tokens within the text (output).

(Input) Text: *The weather prediction for today is sunny.*

(Output) Tokenized text: *["The", "weather", "prediction", "for", "today", "is", "sunny", "."]*

Figure 2.3: Example of a tokenization task.

Sentence Segmentation

Similarly to tokenization, sentence segmentation (Jurafsky and Martin, 2019) involves separating a text. However, the separation is not performed token by token. Instead it is the task of separating a text where the tokens are sentences (see example in figure 2.4). In place of whitespaces, the cues to separate a text into tokens are periods, exclamation points and other punctuation marks.

N-grams

N-grams (Jurafsky and Martin, 2019) are a group of n sequenced items from a given text or speech. Their size can vary, however the most frequent sizes are unigrams (n-grams of one token), bigrams (n-grams with two tokens) and trigrams (n-grams with three tokens). Figure 2.5 shows some example sentences split into n-grams.

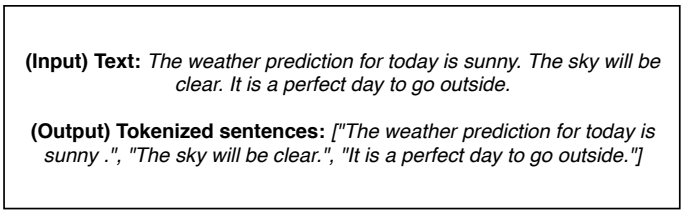

Figure 2.4: Example of a sentence segmentation task.

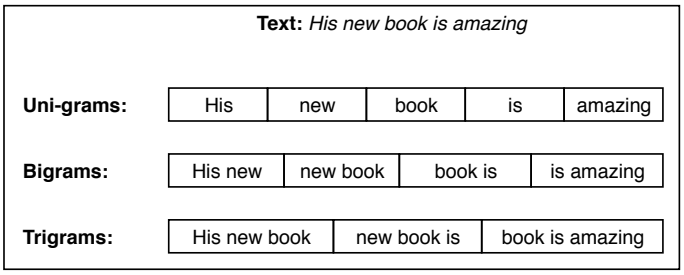

Figure 2.5: Example of text segmentation into n-grams.

Spell Checking

Spell checking (Jurafsky and Martin, 2019) is used to identify incorrectly spelled words and correct them. In our example (figure 2.6), our sentence contains a *typo* in the word *turles*. A spell checker will analyze the closest words to the one we typed and will present them to the person who is typing as suggestions for correction. Spell checking is important because typing errors may lead to problems when analyzing a text.

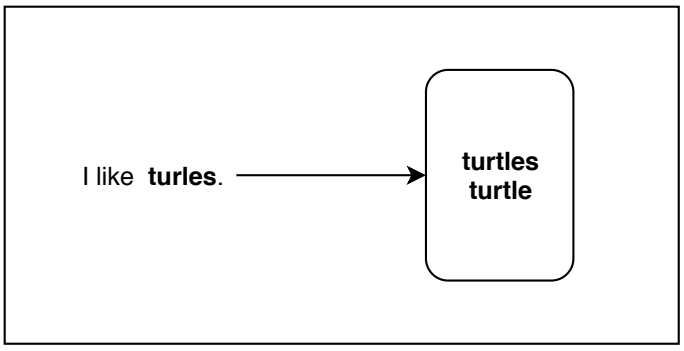

Figure 2.6: Example of spell checking.

Lemmatization[1]

Lemmatization (Jurafsky and Martin, 2019) is the task of determining whether two words have the same root regardless of their superficial differences. In figure 2.7, we have such an example. The words *study*, *studies* share a common lemma *study*.

[1]Results obtained with WordNet lemmatizer: https://text-processing.com/demo/stem/
[2]Results obtained with Lancaster stemmer: https://text-processing.com/demo/stem/

Word	Lemma
study	study
studies	study
studying	studying

Figure 2.7: Example of word lemmatization.

Stemming[2]

Stemming (Jurafsky and Martin, 2019) is referred to as the simpler version of lemmatization, in which instead of searching for the root of a word, we simply cut out a word's suffix. Due to its simplicity and disregard for whether or not a word after cut retains its meaning, it is usually preferable to use lemmatization over stemming. If you compare figures 2.7 and 2.8, you will see that the word *studying* was handled differently with lemmatization and stemming. When *studying* was subjected to lemmatization, the lemma was *studying* while while subjected to stemming it became *study*.

Word	Stem
study	study
studies	study
studying	study

Figure 2.8: Example of word stemming.

Named Entity Recognition

Named Entity Recognition (NER) (Jurafsky and Martin, 2019) is the task where we identify the names of people, organizations, locations or other relevant entities in a text. Classifying types of entities can be quite complicated, as some words can be represented by multiple types of entities. A simple example would be **JFK**. This entity could be both a person and an airport located in New York. In figure 2.9, we have 3 types of entities: *Organization*, *People* and *Location*. *Tesla* is an organization, *Elon Musk* is a person and *California* is a location. If configured to do so, NER can also capture numbers, currencies, dates and time. Take figure 2.9, if we wanted to, we could have configured NER to extract numbers, and we would have captured *150.000*.

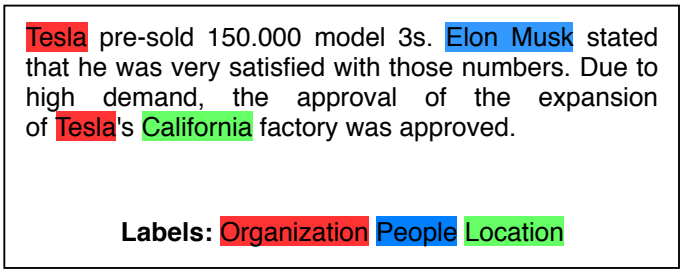

Figure 2.9: Example of named entity recognition.

Stop Words

Stop words (Jurafsky and Martin, 2019) are words that are considered to bring little to no value of information in a text. Words such as "a" and "the" are considered as Stop Words. One task normally performed when analyzing a text is the removal of stop words. Due to their frequent occurrence and little information gain, the removal of stop words can significantly reduce the size of a text. An example of a text with highlighted stop words is presented in figure 2.10.

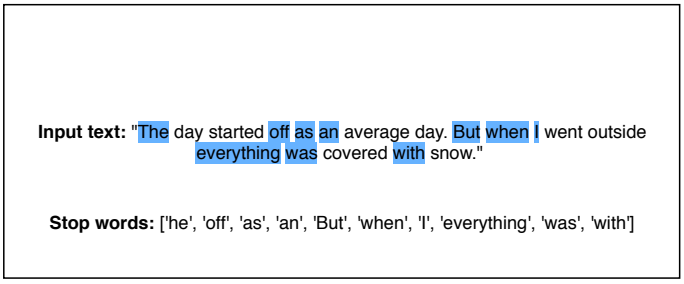

Figure 2.10: Example of a stop word identification task.

Dependency Parsing[3]

Dependency Parsing (Jurafsky and Martin, 2019) is the task performed in order to extract a dependency parsing tree[4], in which the words in this parsing tree are connected according to their relations. In figure 2.11, you have the sentence *The day started off as an average day.* Upon performing dependency parsing, we obtain the tree in figure 2.11: *The* is a determiner related to *day*. *day* is a nominal subject related to the verb *started*. The word *off* is a phrasal verb particle associated with the verb *started* (*started off* is a phrasal verb). *as* is the case marking of *day*. Simply put, this *case* notation is used for prepositions in English. *an* is also a determiner related to *day* (the last one in the sentence). *average* is an adjective modifier related with *day* (second one in the sentence). Its purpose in this sentence is to describe *day*. *day* (the second instance of the word in the sentence) is also a nominal modifier of the verb *started*. It is used here because it serves as a prepositional complement. The last relation present in this dependency parsing tree is between *started* and *"."*. While normally omitted, this relation, *punct* tells us that we reached a piece of punctuation in our current clause (in this case, it is *"."*). To summarize, dependency

[3]Example results generated with: https://corenlp.run/
[4]Syntactic relations used in Stanford CoreNLP: https://universaldependencies.org/en/dep/index.html

parsing is used to only display the relationships between words that are related to one another.

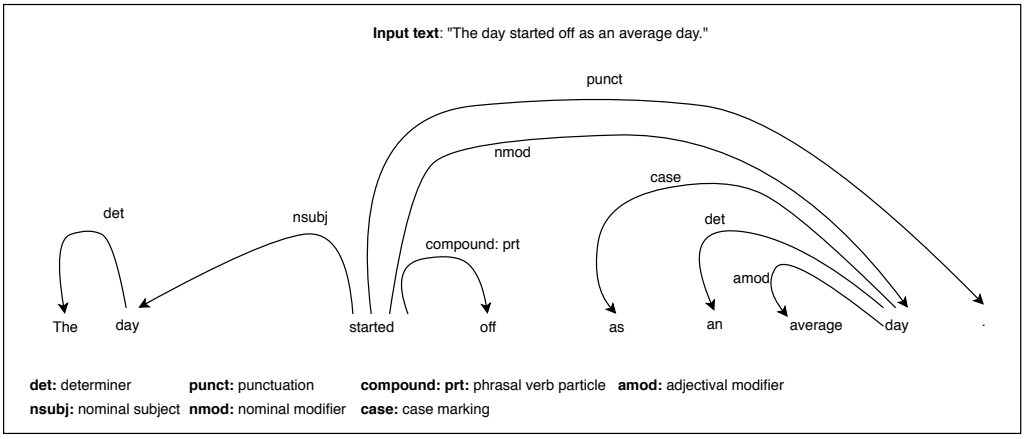

Figure 2.11: Example of a dependency parsing task.

Constituency Parsing[5]

Constituency Parsing (Jurafsky and Martin, 2019) is a task performed in order to extract a constituency-based parsing tree from a sentence. That parsing tree aims to represent the syntactic structure of the phrase it was extracted from. As you can see from figure 2.12, the sentence in the example remains the same as the previous: *The day started off as an average day.*

The sentence itself is simply a declarative sentence (hence the S label). Once we decompose the sentence: *The day* becomes a noun phrase (NP in the figure) and *started of as an average day* the verb phrase (VP). These two labels are assigned at the sentence level. We must continue to decompose these sentences until we reach the word level. In conclusion, constituency parsing is used to study the entire utterance structure and the relationships between those parts of a utterance.

Decomposing the sentences *The day* and *started off as an average day* even more, we get:

- *The day* - *The* becomes a determiner (DT) and *day* a noun of the phrase (at the word level).
- *started of as an average day* - *started* becomes the verb in the past tense of the phrase (VBD) and *off* becomes a particle (both at the word level). *as an average day* is a prepositional phrase (still at the sentence level). However, we still have not reached the word level in this phrase (see figure 2.12).

  Decomposing *an average day*, *an* becomes a determiner (DT), *average* an adjective (JJ) and *day* a noun (NN) of the sentence.

Part-of-Speech Tagging[6]

[5]Example results generated with: https://corenlp.run/
[6]Example results generated with POS tagger from: https://spacy.io/usage/linguistic-features

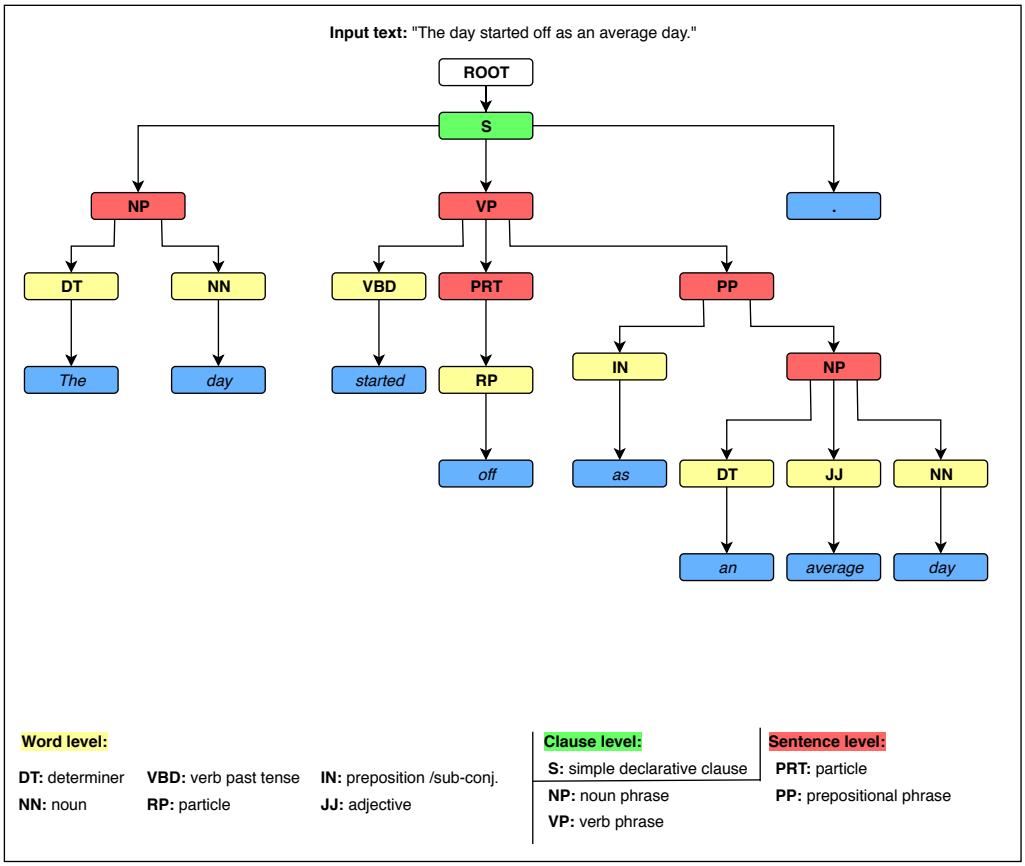

Figure 2.12: Example of a constituency parsing task.

Part of Speech (POS) tagging (Jurafsky and Martin, 2019) is the task that must be completed in order to assign part-of-speech markers for all the words in a text. POS tagging is used to explain how words are used in sentences. As you can see from the example in figure 2.13, the sentence tokens were assigned a category depending on their respective function on the sentence: *The* is a determiner (DT) in the sentence, *day* is as a noun (NN) and *started* is a verb in the past tense (VBD). *off* is a particle (RP) and *as* is a preposition (IN). Finally, *average* is an adjective (JJ).

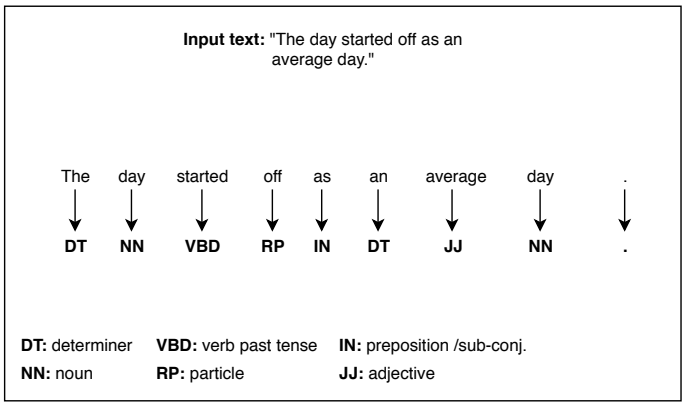

Figure 2.13: Example of a POS tagging task.

[7]Tool used to execute the example: https://spacy.io/usage/linguistic-features#noun-chunks

Chunking[7]

Chunking (Jurafsky and Martin, 2019) is the process used to identify and classify the segments of a sentence by their word-groups (be it noun phrases, verb, prepositional or adjective). Since we get the segments of a sentence classified by their word-groups when we perform POS tagging, this task must be executed before proceeding to chunking.

In figure 2.14, we have an example of noun chunking. Basically, we will extract all the nouns in the sentence as chunks. To do so, we must first perform POS tagging on the sentence in order to obtain the tags that we will use to create the rules for the chunker. In figure 2.14, our grammar will extract chunks that are composed by one or multiple consecutive nouns (words with POS tags 'NN'). In our example, there are only chunks composed by one noun (the word *day*, that appears twice in the sentence that we used as an example).

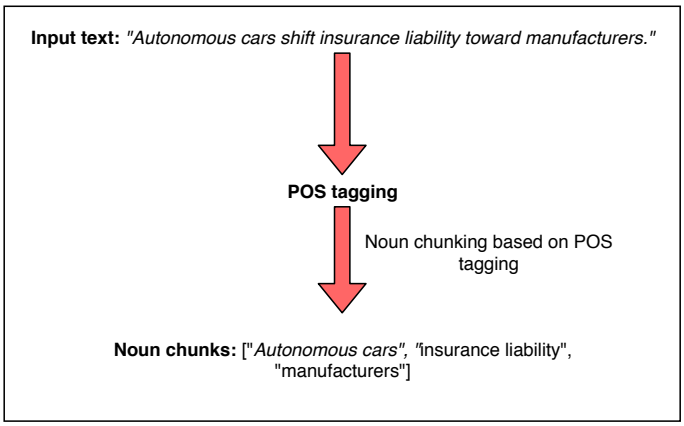

Figure 2.14: Example of a chunking task.

Chinking

Chinking (Bird et al., 2009) is the process used to remove a sequence of tokens (chinks) from chunks obtained from chunking tasks. When we perform chunking we define the rules to include certain patterns from a text. When we do chinking, we define certain patterns for the types of words we wish to exclude from a chunk. Chinking is not a mandatory task in NLP but can sometimes be useful. The removal of pieces of a chunk becomes beneficial to us when we have extra words in a chunk that are not necessary to understand the content of the text. For example, in figure 2.15, we have the chunk *The little house*. Now, imagine we simply want the noun *house*. We create the chinking rule $\}DT\ JJ\{$, which states that we will remove all determiners (DT) followed by an adjective (JJ) from a chunk. In our example, we remove *The little*. After chinking we get a shorter chunk (*house*.

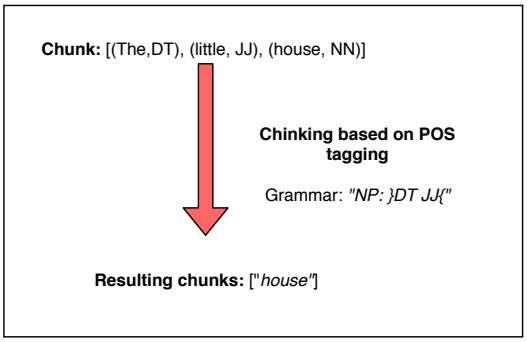

Figure 2.15: Example of a chinking task.

2.2.3 Information Extraction

Information extraction (Jurafsky and Martin, 2019) is the name assigned to the task of automatically extracting information from semi-structured or unstructured data and turns it into a structured format. The most widely known information extraction tasks are (Jurafsky and Martin, 2019):

- **Named Entity Recognition -** Previously mentioned in 2.2.2, NER is the NLP task that is used to detect entities in texts. In the context of opinion mining, it would be useful to know a person's opinion towards a certain entity. Currently, the state-of-the-art approaches for this task are implemented with supervised learning. In the example: "*He does not like Donald Trump*", the entity extracted would be *Donald Trump* and the sentiment associated with it would be negative.

- **Relation Extraction -** Relation extraction is the NLP task used to capture the relations between entities. Take the following sentence as an example: "*The London Eye is located in London*". The relation between the entities "*London Eye*" and "*London*" is "*location*", where the first entity is defined as located within the second entity.

- **Event Extraction -** Event extraction is the task used to identify occurrences of mentions of event in a text. In order to extract an event, we must fulfill two conditions: Firstly, we must have an expression mentioning the event (naming it) or presenting a state. Lastly, we need a time for the occurrence of the event. In the sentence: "*Avengers: Endgame made the most revenue upon its first week in the box*

office". In this sentence, the event is *made* and the time associated with it is *first week*.

- **Template Filling -** Template filling is a task where we parse through documents and attempt to extract information from their text. Then, we would fill in the slots of templates with the text segments extracted from the documents. Take an example of a booking of a flight for an airline company. In order to make a reservation, the client had to call an agent to book a ticket for the flight. Among the information that was exchanged was the destination, the data of departure, the price of the ticket and, of course, the the airline company the client will fly on. Assuming the template contains fields such as: *airline company name*, *price of ticket*, *date of departure* and the flight's *destination*.

2.2.4 Topic Modeling

Topic modeling (Jurafsky and Martin, 2019) is the name attributed to the task that discovers the topics that appear in a collection of documents. It functions on the principle that, given a document about any topic, it should be within reason that some sets of words related to the topic would appear more frequently in the documents. The topics that are generated from topic modeling approaches (the output of these approaches) are usually clusters of words. These clusters of words go a long way towards enlightening us on the main topics of a group of documents, making it a suitable approach for the exploratory analysis of text data. Topic modeling is classified as an unsupervised learning method. Topic modeling can be used for both aspect extraction and sentiment word extraction. The most widely topic model currently used is *Latent Dirichlet Allocation* (Blei et al., 2003a).

- **Latent Dirichlet Allocation -** Is a probabilistic model based on Bayesian networks that is used to generate clusters of data topics from a set of text documents. Each item from a list of topics has a set of probabilities assigned to it for each text that is being analyzed. This characteristic makes this model a reliable method to explore and summarize data within large amounts of text documents.

2.3 Sentiment Analysis

Sentiment Analysis (Jurafsky and Martin, 2019) applied to texts is a text classification task that classifies a given text as reflecting the positive, negative or neutral orientation (also called sentiment) that a writer/reviewer expresses. While NLP and linguistics are fields with a long history, sentiment analysis is a fairly recent area of study, as not much research was done about it before the year 2000, due to the lack of data (texts) in digital format. This does not mean that no research was done at all before that. The earliest works that are related to this field appeared a decade earlier in the 1990's. Such works include (Wiebe et al., 1999), focusing on analyzing and improving reliability in discourse tagging, (Hatzivassiloglou and McKeown, 1997) focusing on predicting the semantic orientation of adjectives and (Hearst, 1992) proposing a method on acquiring the hyponymy lexical relations from texts. After the year 2000 came the widespread use of the internet. The amounts of data generated by users worldwide provided enough data for researchers to work on, making it possible to perform very varied and detailed types of research. Great investments were made to further study sentiment analysis after analyzing the benefits

that this field could provide. Acquiring public and consumer opinions is invaluable for areas such as marketing and politics. Even the information acquired on social media is a very valuable asset organizations can use to study the public's perspective of them.

2.3.1 Levels of Sentiment Analysis

Sentiment analysis is a very complex field of study. However, it is also a very thoroughly studied area. This field was studied mainly at three different levels (presented in figure 2.16), each of which studies a different set of research problems: (B. Liu, 2012)

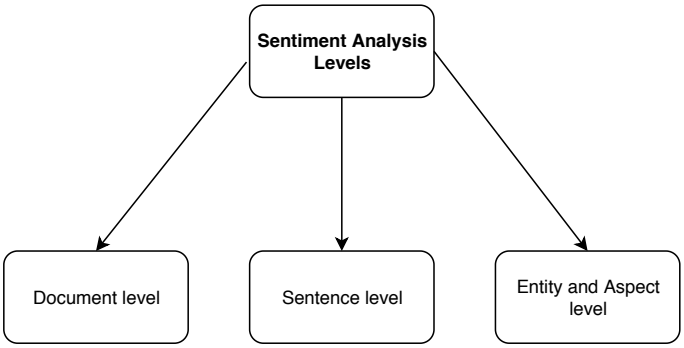

Figure 2.16: Levels of Sentiment Analysis.

- **Document Level** - At the document level, an assumption is made that an entire text expresses an opinion on a single entity. The main task associated with this level is the *document-level sentiment classification* task, in which, given a review about a given product, a system must determine whether the text expresses a positive or negative opinion about the product.

- **Sentence Level** - At the sentence level, we focus on determining whether the sentence expresses a positive, negative or neutral opinion. It is also at this level that we perform an analysis to determine whether a sentence is subjective or objective.

- **Entity and Aspect Level** - This level analyzes the opinions that are expressed by the writer. It is also at this level that we search for and define the opinion target. Defining the target of an opinion tells us what is the entity the writer expressed his opinion about. This information is extremely valuable. Take the following example: *"The iPhone's camera quality is great but the battery life is short"*. In this sentence we have two different opinion targets, the IPhone's *camera quality* and *battery life*. For the former, the opinion is positive however for the latter, the opinion is negative. If you were the company that developed the product, you could learn what parts of your product your customers liked and which they did not, and design a new product which would take your customer preferences into account and create a more successful product.

2.3.2 Approaches for Sentiment Analysis

There are multiple methods and algorithms that can be implemented in sentiment analysis systems, however it is possible to classify them in the following categories:

- **Lexicon-based** - The lexicon-based approach is mostly used when we do not have enough training data for a supervised machine learning approach. Lexicons (Richards and Schmidt, 2013) are lists of words, which in the context of sentiment analysis would include sentiment words, can be used with rule-based algorithms (algorithms that extract knowledge in the form of rules which are easy to understand) and where a certain polarity is assigned based on the detected rule. But lexicon-based approaches have well known weaknesses. For example, there are some situations in which it is hard to define sets of rules, and sometimes, even when we manage to define rules, as humans, it is certain that we will not be able to write down all the rules in a system. Another case where a rule-based algorithm would fail would be if the data the system analysis changed its structure. Due to these problems, another approach was used: machine learning.

- **Machine Learning-based** - Machine learning approaches are efficient when we have large amounts of data. The most used used machine learning approaches for sentiment analysis use supervised learning, in which labeled data is necessary (to a given data input there is a corresponding output label). The most commonly used machine learning algorithms used in sentiment analysis include:

  - **Naive Bayes** - This statistical method (Nisbet et al., 2009) is used to build models that given data observations return the class most likely to have generated the observation (in this case, if the polarity is positive or negative). It is a simple approach, and rather naive, however it produces significant accuracy of sentiment classification.

  - **Logistic Regression** - This regression (Nisbet et al., 2009) model learns what features from input can be useful to discriminate between the different possible classes (which in the context of sentiment analysis is whether the polarity is positive or negative). This approach is useful for sentiment analysis as there is no need to craft features by hand.

  - **Support Vector Machines** - Support vector machines (Nisbet et al., 2009) were one of the most popularly used approaches to train transition-based dependency parsers. Transition-based dependency parsers are parsers that start at an initial configuration and then, by making choices between possible transitions ate each step, reach a terminal configuration. Once a terminal configuration has been reached, the parser returns the dependency tree associated with that terminal configuration. As for the logistic regression approach, it is useful for sentiment analysis, because there is no need to craft features by hand.

  - **Neural Networks** - The neural network (Nisbet et al., 2009) approach is the most recent machine learning approach and it became popular very quickly. While it is not a very intuitive model for a human to understand, it allows for very good accuracy for polarity prediction of a text.

- **Hybrid** - The hybrid approach makes use of both lexicon and machine learning approaches. One of the most effective applications of this approach is to use neural networks to extract rules that would then enable a model to predict the polarity of an opinion.

2.3.3 Challenges in Sentiment Analysis

Sentiment analysis is not just a simple classification problem. If we go deeper into the topic we will realize that there are many challenging problems that can greatly damage the accuracy of sentiment analysis models, if not handled properly. We will now present the most important challenging problems (B. Liu, 2012) that appear in sentiment analysis:

- **Domain Dependence** - We are confronted with the challenge of domain dependency when we train a sentiment analysis model with training data from one domain and then we try to test the same model in a different domain. If we do not try to adapt our approach to handle different domains, then the accuracy of the system will decrease when the domain changes. In (Blitzer et al., 2007), the authors extended their algorithm to handle the change between different review domains with rather successful results, however it is possible to observe that changing the domain from kitchen appliances to book reviews, even though an adaptation strategy was used, still affects the accuracy of a model when it is applied in a different domain from the one the system was trained in.

- **Ambiguity** - Ambiguity is a problem when a given word or sentence has a meaning in one situation context and a different one when used in another context. As machines cannot infer context like humans do, ambiguity is another big challenge for sentiment analysis. (Hindle and Rooth, 1993) tries to solve some ambiguities in prepositional phrase attachments, a type of structural ambiguity, and with their model they managed to improve accuracy, however, they also concluded that the performance of their system still had a worse performance than the human judges that also assessed the attachments.

  *This phone was surprisingly **cheap** for the functionalities it supported.*

  *The fabric of this dress feels **cheap**.*

  In this example, *cheap* allows the readers to deduce the polarity of each expression. In the first one, the user is happy that the phone he bought was *cheap* even though it is a good phone. On the other hand, in the second expression, *cheap* is used to classify the fabric of a dress and it conveys the idea that the reviewer does not like it. Ambiguity can occur in many different circumstances (levels), however we will not go deeper into it and just warn the readers that ambiguity must be handled in sentiment analysis systems in order to achieve good performances.

- **Negation Handling** - Another aspect that poses a challenge in sentiment analysis is negation. When a sentence or expression is expressed in the negative it is obvious for a listener to know what the polarity of the sentiment is. It is not so simple

for machines. If a system overlooks negation, it will fail to detect many important and frequent situations that are necessary to enhance the performance of sentiment analysis systems.

*I am **happy** with the camera quality of this phone, however I am **not satisfied** with its battery life.*

In the first part of the sentence in the example above, the reviewer is *happy* with the camera quality of the phone he purchased. On the other hand, in the second half of the sentence the reviewer is *not satisfied* with the phone's battery life. If sentiment analysis systems are not configured to handle negation (whether it is an obvious negation or not), then in a situation such as this one he will not be able to detect that the reviewer's opinion towards the product's battery life is negative, therefore it is important to handle negation in order to enhance the performance of sentiment analysis systems.

- **Irony and Sarcasm** - Irony and sarcasm are also two very problematic aspects that have a great influence over the sentiment expressed. If in some cases, even for humans it is hard to understand sarcasm then for machines it is nearly an impossible task. Sarcasm and irony are mostly understood due to the way the expression was worded (in the discourse and pragmatic levels) and their understanding relies heavily on the knowledge of the context they were used in. Machines struggle to understand context, so it is substantially harder for sentiment analysis systems to handle irony and sarcasm.

- **Comparative Opinions** - Comparative opinions are slightly tricky to handle, as a reviewer is not expressing his opinion on all the aspects of the product directly, and instead uses comparisons to separate or group different product characteristics based on his positive or negative opinions of the product features. In (Ganapathibhotla and Liu, 2008), the authors focused on mining opinions from comparative sentences and determining which entities in the comparison were the author's preferred ones.

*This computer's **processor is really fast** just like its **graphics card**.*

In the example above, the reviewer presents a direct positive opinion on a computer's processor and he uses a comparison to express that he also likes the graphics card.

- **Implicit Opinions** - Implicit opinions are one of the most challenging aspects to handle in sentiment analysis and opinion mining. It is very complicated to extract opinions that do not follow the structure of opinion target and target polarity. One example of implicit opinion can be found below:

The house is *30 minutes away from the school*, by car.

In this case, the opinion author states that the location of a house is rather far from another place he considers of importance, in this case a school. For a human annotator, it is not complicated to conclude that the opinion author is expressing

a negative opinion on the location of the house he is evaluating (as it is very far from a location with which he desired proximity), however, for a sentiment analysis system, this is not so direct, as there are no terms used to express a direct opinion. Depending on the context, it can be very important to handle implicit opinions.

2.4 Opinion Mining

Opinion Mining (Cambria et al., 2013) is a field that was derived from sentiment analysis that focuses on extracting the opinions reviewers expressed about the characteristics of a certain entity and assigns a polarity to each one. Opinion mining differs from sentiment analysis in the aspect that sentiment analysis tries to assign a polarity to a text, while opinion mining assigns a polarity for all entities extracted from a given text. In a way, opinion mining can be considered more objective than sentiment analysis and a much more restrict natural language processing problem. Using the following sentence as an example, we can verify the objectivity of opinion mining:

"I do not like the battery life of the iPhone as it does not last long, however I love its camera quality."

By definition, opinion mining would extract the entities *"battery life"* and assign a negative polarity to it, as the user does not like that feature, and assign a positive polarity to *camera quality*. However, using sentiment analysis on the review as a whole, even humans would be hard pressed to define its polarity, as we have a positive view of one product feature and a negative view of another, and we do not have any other indicator about which one of these views is more important to define the overall sentiment.

2.4.1 Defining Opinion

The dictionary definition of opinion is "a thought or belief about something or someone" ("OPINION: meaning in the Cambridge English Dictionary", n.d.), however, in the context of this work we must be more inclusive in the information we wish to obtain, what we consider to be an opinion (its format and content) that we must extract with our opinion mining framework.We will thus define opinion as a tuple that will contain the following information:

Opinion: {*Product characteristic* *: *{feature opinion, polarity}**},

in which, *product characteristic or opinion target* is a product feature, *feature opinion or opinion expression* is what the opinion writer said about that feature, and *polarity or sentiment orientation of opinion* is what is the sentiment towards that features (positive, negative or neutral).

2.4.2 Aspect Extraction

Aspect term extraction (B. Liu, 2012) is an information extraction task that extracts aspects that an opinion was cast on. An aspect is a term used to classify activities, events or states described by verbs (Richards and Schmidt, 2013). In the field of opinion mining,

aspect extraction is used to extract the opinion targets from sentences. Opinion targets (B. Liu, 2012) are the entities about which an opinion is made. Opinion expressions (B. Liu, 2012) are the opinions expressed about a given opinion target.

*I **love** the **camera quality** of this phone.*

In the example above, the opinion target of the sentence would be *camera quality* and the opinion expression would be *love*.

Aspect Extraction Approaches
There are four main approaches to aspect extraction (B. Liu, 2012):

- **Extraction based on frequent nouns and noun phrases** - This approach is used to find aspect expressions that are either nouns or noun phrases (B. Liu, 2012). We use POS tagging to get the nouns and noun phrases of sentences and count their occurrence frequencies. Once that is done, we keep the most frequent ones (by defining a threshold for how much should the frequency count be for a aspect to be considered frequent). This approach tends to work because when people talk about the same things they tend to use the same or similar words. If it is an important topic, then it will appear many times. At the same time, if the nouns and noun phrases have low frequency counts, then we can assume that they are not important and discard them. Both (Hu and Liu, 2004) and (Popescu and Etzioni, 2005) use implementations of this approach to extract explicit aspect expressions (just the explicit ones, because it is better to use a different approach to count for implicit aspect expressions).

- **Extraction based on opinion and target relations** - Since by definition an opinion has to have at least one target, we know that the opinion has to be somewhat related to the target. There are two interesting techniques used that are used in this method of aspect extraction (B. Liu, 2012):

  - **Look for "nearest" noun or noun phrase of a sentiment word**: In this method, we normally use a lexicon with a list of opinion words and POS tagging. Firstly, we perform POS tagging on the text. Then we search the text for sentiment words. Once we find a sentiment word, we look for the closest noun or noun phrase (using the POS tags obtained from the NLP task) to that sentiment word and extract it. (Hu and Liu, 2004) used this technique to extract infrequent aspects (to complement his extraction of explicit aspects based on frequent nouns or noun phrases).
  - **Extraction based on dependency parsing**: In this approach we perform the NLP tasks POS tagging and dependency parsing. Since dependency parsing extracts the relations between words in a sentence, we can use those relations obtained from that NLP task to extract aspects (normally by applying rules based on POS tagging for the extraction).

- **Extraction using supervised learning**: As you must remember, supervised learning requires labeled for the training of the models. This data normally has to be annotated by hand, which is quite a arduous and time-consuming task. The two state-of-the-art approaches of this category are (B. Liu, 2012):

  - **Hidden Markov models**: Hidden Markov models (Marsland, 2014) are also a graphical model. In this model, we make measurements at regular intervals, which are the observations of a state. These observations are then used to predict he most likely state. In the context of aspect extraction, it would be if a word is an aspect or not.
  - **Conditional Random Fields**: Conditional random fields (Marsland, 2014) is a graphical model (Marsland, 2014) that gets his name because it uses graph theory to explain the model. The conditional random fields takes into account the context of the predictions (what was predicted in the past, usually the prediction performed immediately before) when performing a new prediction. (Jakob and Gurevych, 2010) applied this approach.

- **Extraction using topic modeling** - Topic modeling is also a graphical model. The result of applying topic modeling is a set of word clusters that would become our topics. This can be applied in aspect extraction. In this case, our topics would be the aspects. However, we would need to separate the opinion target and the opinion words. One approach implementing this technique that has been used is an extension of LDA (Latent Dirichlet allocation). LDA (Blei et al., 2003b) is a probabilistic model (Bayesian model) that is used to collect discrete data. Text corpora is an example of such data.

2.4.3 Aspect Sentiment Classification

Aspect sentiment classification (B. Liu, 2012) is the task performed to determine whether an opinion on an aspect is positive, neutral or negative. Currently, there are two main approaches used in this task:

- **Supervised approach** - Supervised learning is an approach that depends on the training data. For example, if we try to train a machine learning model with training data from one domain, then test it in a different one, we should expect the model to perform badly. The most commonly used supervised learning approaches used for aspect sentiment classification are actually the same as the machine learning-based methods presented in 2.3.2 (see sections 2.1.3 and 2.3.2 for more details): naive Bayes, logistic regression, support vector machines (SVM) and neural networks.

- **Lexicon-based** - Just as it was previously mentioned in section 2.3.2, a lexicon-based approach is mostly used when we do not have enough annotated training data to opt for a supervised machine learning approach. In the context of opinion mining, we use lexicons (list of words) containing sentiment words, phrases and idioms. These lexicons tend to be used with rules and sometimes, parsing trees to determine sentiment orientation. Of course, this approach has its flaws: It can be quite a laborious task to elaborate rules and even then we might not have listed down all the important rules. Which ca cause our system to become less efficient at performing aspect sentiment classification.

2.5 Evaluation Metrics

In this section we will present the metrics most commonly used for opinion mining systems. These metrics can be of two different types: performance evaluation metrics, used to study

how efficient the system is at classifying opinions; and inter-annotator metrics, used to evaluate the consensus between annotators of the data used for the evaluation of the performance of the system.

2.5.1 Performance Evaluation Metrics

Performance Evaluation Metrics are used to assess the performance of a system. Before going through the most widely used system performance assessment metrics, we must introduce some key concepts: Outcomes. Outcomes are the possible results opinion mining models can achieve when trying to predict what is the sentiment of a given customer or reviewer towards a certain product feature. The possible outcomes obtained by these models are presented on table 2.1.

Table 2.1: Table of possible opinion mining models' prediction outcomes

Types of Outcomes	**Definition**	**Example**
True Positives	Outcomes where a model correctly predicts a positive class	**Ground truth:** The customer liked the product feature. **Model predicted:** The customer liked the product feature. **Outcome:** The model is correct and we learned that the client likes the product feature.
True Negatives	Outcomes where a model correctly predicts a negative class	**Ground truth:** The customer did not like the product feature. **Model predicted:** The customer did not like the product feature. **Outcome:** The model is correct and we learned that the client does not like the product feature.
False Positives	Outcomes where a model incorrectly predicts a positive class	**Ground truth:** The customer did not like the product feature. **Model predicted:** The customer liked the product feature. **Outcome:** The model is incorrect and we think the client likes the product feature when, in reality, he does not.
False Negatives	Outcomes where a model incorrectly predicts a negative class	**Ground truth:** The customer liked the product feature. **Model predicted:** The customer did not like the product feature. **Outcome:** The model is incorrect and we think the client does not like the product feature when, in reality, he does.

Since we wish to evaluate the performance of an opinion mining system, we have to use the most suitable evaluation metrics for these types of systems. These metrics were selected for being the most widely used in papers studying these types of models, and are described in more detail below (Eisenstein, 2019).

- **Accuracy** - Accuracy is defined as the ratio of correctly predicted observations to the total number of observations. Accuracy is commonly used as an evaluation metric however, by itself, it is insufficient and often misleading. For example, suppose that the majority of your observations is of one of two possible types, and that a system you developed is very efficient at predicting that particular class, but terrible at the predicting the other class. Due to the fact that the majority of observations is composed of observations of the class your system detects better with, your accuracy level will still be high, even though your system is actually bad at predicting another class, and in cases where the prediction of both classes is important (for example, whether a patient has cancer or not), we need to be sure that we have a way to evaluate the efficiency of a system for all possible types of observation results.

$$Accuracy = \frac{TruePositives}{TruePositives + TrueNegatives} \tag{2.1}$$

- **Precision** - Precision (also called true predictive value) is the ratio of correctly predicted positive class observations over the the total number of predicted positive class observations (which includes both the wrongly and correctly classified positive classes). With Precision, we have a more in-depth understanding on how the system performs in predicting positive class observation, but once again, we should remember that alone, this metric is not enough, therefore next we will present another metric that is always present side-by-side with Precision: the Recall metric.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.2}$$

- **Recall** - Recall (also called sensitivity) calculates the proportion of actual positives that were correctly predicted. To put it simply, Recall is the ratio of the correctly predicted positive observations over the total of positive class observations.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.3}$$

- **F-score** - The F-score (also called f1-score) takes into account both Precision and Recall metrics. The F-score is the weighted harmonic mean of the values of Precision and Recall of a test.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.4}$$

- **Microaveraging** - Also named weighted average, is it a metric used when performing a classification task with more than two labels. It is applied to metrics such as precision, recall and f-score. For each available classification class, metrics are collected and the weight of each class depends on its frequency. The weight of each class then is taken into consideration for the averaging of the metric studied.

- **Macroaveraging** - Macroaveraging is another metric used when performing a classification task with more than two labels. It is also applied to other metrics, such as precision, recall and f-score. Unlike with microaveraging, each class is assigned the same weight, regardless of their respective frequencies. This makes the final averaged result of a metric to be the sum of its value calculated individually for each class label divided by the number of existing class labels.

2.5.2 Inter-Annotator Metrics

Inter-annotator metrics are used the measure the agreement between multiple annotators. These metrics are also a form of test validation. When we have large consensus between annotators we can conclude that the data annotated is more reliable, as it is more likely that other people who might try to annotate the data will do so with the same resulting labels. With our data annotation validated with high consensus, we can conclude that the data is reliable, and we can use it on our system. Should the data annotations lack consensus between annotators, the data would be considered unreliable, as we could not declare with certainty whether the data is correctly labeled due to the disparity between the annotators classifications, which in turn would lead us to question the annotators reliability, or even the validity of the annotation system used. If the data is unreliable, then we could not possibly use it to train or test a system, hence the importance of these metrics. We will now present some of the most widely used inter-annotator agreement metrics:

- **Cohen's Kappa** - Cohen's kappa (Cohen, 1960; Eisenstein, 2019) is a widely used metric for quantifying the agreement of discrete labeling tasks when there are two annotators performing annotation tasks. In order to calculate Cohen's Kappa we first need to know the chance agreement. In this formula, k is the number of labels and Pr(Y = k) is the empirical probability of label k across all annotations,

$$E[agreement] = \sum_{k} \widehat{Pr}(Y = k)^2. \tag{2.5}$$

  The formula used to calculate Cohen's Kappa is given below: the numerator of the equation is the difference between the agreement of the annotator minus the chance agreement and the denominator is the difference between a perfect agreement between annotators and the chance agreement.

$$\kappa = \frac{agreement - E[agreement]}{1 - E[agreement]}, \tag{2.6}$$

  with *kappa* ranging from 0 (when annotators agree randomly) to 1 (when annotators agree in all annotations).

  While not originally done by the authors, extensions for Cohen's Kappa for multiple-rater annotation tasks have been defined, however we will not go further than mentioning their existence.

- **Fleiss' Kappa** - Fleiss' kappa Fleiss, 1971 is also a metric for quantifying annotator agreement between annotators for categorical data. However, this metric can be used for any fixed number of raters, unlike Cohen's kappa, which is usable only when we wish to measure the agreement between two annotators. To calculate Fleiss' kappa,

we start by calculating the proportion of all assignments made to the jth category. Take into consideration that N is the number of subjects, n is the number of ratings per subject and k is the number of categories to which we can assign a subject to. n_{ij} is the the number of raters who assigned category j to a subject i.

$$p_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij}, \tag{2.7}$$

Afterwards, we calculate the agreement between the n raters for subject i. This should be done for the N subjects.

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij} \left(n_{ij} - 1\right) \tag{2.8}$$

Next, we compute the mean of the of the P_i values calculated.

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i \tag{2.9}$$

Another aspect which we have to calculate is the probability of the raters agreeing with each other by chance.

$$\bar{P}_e = \sum_{j=1}^{k} p_j^2 \tag{2.10}$$

Now that we have all the necessary elements calculated, we can finally obtain Fleiss' kappa.

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{2.11}$$

- **Krippendorff's** α - Krippendorff's α (Krippendorff, 2011) is a reliability coefficient that is also used to measure inter-annotator reliability. Unlike Cohen's kappa, Krippendorff's α is applicable to any number of annotators greater or equal than two. Another advantage of this metric is that it also deals well with missing values. The formula for Krippendorff's α is as follows:

$$\alpha = 1 - \frac{D_o}{D_e}, \tag{2.12}$$

and α ranges from $0 <= \alpha <= 1$. D_o is the observed disagreement,

$$D_o = \frac{1}{n} \sum_{c} \sum_{k} o_{c_k} \delta_{c_k}^2, \tag{2.13}$$

and D_e is the expected chance disagreement,

$$D_e = \frac{1}{n(n-1)} \sum_{c} \sum_{k} n_c . n_k \delta_{c_k}^2. \tag{2.14}$$

2.5.3 Text Similarity Metrics

Text similarity metrics, also known as string similarity metrics, are used to measure how similar two strings are (Jurafsky and Martin, 2019). Word similarity is an important aspect of natural language processing tasks. Knowing that two words are similar can be helpful to understand whether two sentences are similar. One natural language understanding task that results from the application of this concept is question answering. We now present some word similarity metrics:

- **Cosine Similarity** - Cosine similarity (Jurafsky and Martin, 2019) is a metric that compares two words vectors representing words to assess how similar they are. The cosine similarity metric values range from 0 to 1. Values close to 0 signify that there is no relation between the two word vectors being compared. For values closer to 1 we can assume that the two words being compared are related. The computation of the cosine similarity between two word vectors is presented in equation 2.15.

$$\cos(\mathbf{w}, \mathbf{v}) = \frac{\mathbf{wv}}{\|\mathbf{w}\|\|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} \mathbf{w}_i \mathbf{v}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{w}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{v}_i)^2}} \tag{2.15}$$

- **Levenshtein distance** - Different from cosine similarity, the Levenshtein metric (Jurafsky and Martin, 2019) calculates the minimum distance between two string sequences. The minimum distance is actually the minimum number of a select array of operations we can perform so that one of the strings we are comparing becomes equal to the other. The operations supported for the calculation of the Levenshtein distance are: the insertion of a character, the removal of a character (deletion) and the substitution of a character. Each of these operations carries a cost. Equation 2.16 contains the formula for the calculation of the Levenshtein distance.

$$D[i,j] = \min \begin{cases} D[i-1,j] + \text{del}_{\text{cost}}(\text{source}[i]) \\ D[i,j-1] + \text{ins}_{\text{cost}}(\text{target}[j]) \\ D[i-1,j-1] + \text{sub}_{\text{cost}}(\text{source}[i], \text{target}[j]) \end{cases} \tag{2.16}$$

Chapter 3

Related Work

In this section we will talk about relevant work done on the fields of sentiment analysis and opinion mining. While the theme of our thesis is opinion mining, this particular field came to existence after sentiment analysis had been sufficiently studied to the point when sentiment analysis and opinion mining became terms that refer to different studies, when previously they were treated as the same. The articles cited in this document were selected based on the following criteria:

- **Citation count:** We expect a highly cited article to be a reliable source of information. We defined at least 50 citations as an acceptable number of citations for a paper or other reference source to have been referenced per year. Of course, papers published less than a year ago won't be held by such standard, as it would be very hard to fulfill this criteria. Other criteria will be used to differentiate more recent papers: **conferences article was presented in** and **author citation count** (explained below).

- **Conferences article was presented in:** Another metric that we are considering to filter through all the published papers is the conferences they were featured in. Some conferences have specific focus themes. There are conferences which focus theme is NLP. Since sentiment analysis and opinion mining are problems of this field, it would make sense that we should look for quality work in such conferences. Another aspect that is important concerning this metric is that some conferences have more prestige than others (and exhibit lower acceptance rates). In conferences where only papers with higher quality are admitted, we can expect to find some relevant works. If a paper was only recently published and has yet to acquire a significant number of citations, we can assume that it has substantial relevance if it was accepted and published in a reputable conference. Some of the conferences favored were: *ACL*, *EMNLP*, *NAACL*, *EACL*, *SEMEVAL*, *COLING* and *LREC*. These conferences focus on the topic of NLP. While not focusing on NLP or opinion mining specifically, *SIGKDD* was also considered a relevant conference for article searching due to its focus on data mining and knowledge discovery. These topics are often related with opinion mining, making it reasonably safe to assume that some papers on opinion mining or sentiment analysis may be published in this conference.

- **Author citation count:** The last quality metric we considered is an author's citation count. We expect an author that is highly cited to produce high quality content.

3.1 Sentiment Analysis

With sentiment analysis being the predecessor of opinion mining, we couldn't leave out important advances made in this field. These advances were used very often on opinion mining as well, which is why this section deserves its place in this document. Being able to extract the sentiment of an individual from a text is a great potential advantage for companies in order to better understand their clients. In fact, sentiment analysis was deemed so useful that studies were made with different sources of text data: Pak and Paroubek, 2010 uses text from twitter to train a document-level (each *tweet* was considered a document) sentiment classifier with a Naive Bayes machine learning approach that uses n-grams and POS tags as features. The authors also tried using SVM and CRF (conditional random fields) models, but the Naive Bayes classifier achieved superior results. They also compared the model's performance when using unigrams, bigrams or trigrams as features and concluded that bigrams obtained the best results. Pak and Paroubek, 2010 also contributed with a method for an automatic collection of a corpus (composed of *tweets*) to train sentiment classifiers.

Similarly to Pak and Paroubek, 2010, Pang et al., 2002 also presented a supervised learning approach to document-level sentiment classification, but for a dataset of movie reviews. They also compared different algorithms: Naive Bayes, Maximum Entropy and SVM. In their experiments, SVM achieved the best results (although the difference between the results was not very large). The authors also compared the results obtained in their document-level sentiment analysis with the values typically obtained in a topic modeling problem, and verified that topic modeling results were superior, which led them to the conclusion that sentiment analysis was a more complex problem than topic modeling.

Taboada et al., 2011 also performed work in document-level sentiment classification, but instead opted for a lexicon-based approach, used together with a sentiment dictionary containing sentiment words and phrases. In their work, unlike Pak and Paroubek, 2010 and Pang et al., 2002, they performed tests with different domains (cross-domain) and the performance was solid throughout the different domains. The authors also handled negation, an important aspect of sentiment analysis, and introduced the concept of intensification (differentiating the strength of a sentiment such as *good* from *very good*, as some opinions are expressed more strongly then others). On another note, but still related to the topic of sentiment analysis, two of the authors of this article, Pang and Lee, were responsible for the development of an annotated movie review dataset that they later made public and that has become a benchmark for many studies associated with sentiment analysis and opinion mining.

On sentence-level sentiment classification, Wiebe et al., 1999 differentiate sentences in two categories: objective and subjective sentences. In this case, the authors stated that since objective sentences just state facts and not sentiments, sentiment classification should just be performed on subjective sentences. They used a supervised learning approach using a Naive Bayes classifier to perform sentence-level sentiment classification and obtained satisfying results. They also compared their system's performance with a system consisting of human voters, and concluded that their approach still had room for improvement, as it did not perform as well as human annotators. Gamon et al., 2005 on the other hand, presented a semi-supervised learning approach for a dataset of car reviews. This approach had the advantage that the labeled data required was much less than for a supervised learning approach. The authors trained their system with a small labeled dataset and then a large one composed of unlabeled sentences. Using a Naive Bayes classifier and a learning algorithm based on expectation maximization, they classified sentences as positive,

negative and "other", with "other" being assigned to sentences that don't have sentiments associated to them (objective sentences). The metrics used for performance evaluation were precision and recall. For the positive review class, precision and recall values were quite high. However for the negative and "other" classes, precision lowered when the authors tried to improve recall for these classes.

Nigam and Hurst, 2004 also presents an approach to sentence-level sentiment analysis, however their proposed approach was lexicon-based with chunking (otherwise called shallow parsing). The authors also compared their system's sentence sentiment classification system with human agreement and the conclusion was that human agreement had a superior performance (in both precision and recall metrics).

3.2 Opinion Mining

We have now reached this internship's theme. As mentioned previously, sentiment analysis and opinion mining were once referred to like the same concept, but now, we know that they are distinct. Since we have already presented some interesting works on sentiment analysis, we will now proceed to present some interesting work developed in opinion mining.

In Hu and Liu, 2004, the authors propose a lexicon-based approach to opinion mining. During the development of their system, the authors also developed their own lexicon with positive and negative adjectives that they used with their system. The lexicon was constructed with an initial seed of 30 adjectives that were manually classified as positive or negative. Then, other adjectives were added, using *WordNet* synonym and antonym, starting from the adjectives in the seed. One disadvantage of the system developed in this paper is that it only considers features that appear explicitly in a review and only nouns or noun phrases. Popescu and Etzioni, 2005 also employed a lexicon-based approach where they improved upon Hu and Liu, 2004 by employing relaxed labeling to determine opinion orientation. Another aspect that further differentiates the two works is that Popescu and Etzioni, 2005 also finds implicit product features. To conclude, Popescu and Etzioni, 2005 also compared the performance of their approach with the one presented in Hu and Liu, 2004 and the results obtained show that their approach was indeed an improvement over Hu and Liu, 2004.

Ding et al., 2008 proposes a lexicon-based approach to predict the semantic orientations of opinions on product features. The datasets used are from a benchmark crafted from *Amazon* reviews on different products. There are two major advantages to the system presented in this paper:

- Firstly, is its ability to deal with opinion words that have semantic orientations that are context-dependent by proposing an approach that can infer semantic orientations of opinions words based on the context of the review.
- Lastly is its capability of aggregating multiple multiple opinion words in a same sentence (as a sentence might have multiple different opinions expressed on it). This is done when the authors proposed a function that gathers the multiple opinion words of a sentence.

Another aspect that is worth mentioning of the approach presented in Ding et al., 2008 is that it also is able to deal with both explicit and implicit opinions (dealing with implicit opinions is rather complicated and not many proposed approaches in other papers tackle

this issue). The authors evaluated their work with the metrics precision, recall and f-score. They then compare their results with two other state-of-the art approaches presented on papers Hu and Liu, 2004 (extended so that it considers all types of features and not just with explicit noun features) and Popescu and Etzioni, 2005. Upon observing the results obtained, the authors concluded that their system outperforms both the approaches presented in Hu and Liu, 2004 and Popescu and Etzioni, 2005, distinguishing it as superior compared to these state-of-the-art approaches.

In Jakob and Gurevych, 2010, the main focus of the article is on the opinion target extraction task, with special focus on discourse-type text. One of the most important contributions of this paper is that it explores opinion target extraction on both single (in single-domain, the domain for the training set is the same as the testing set) and cross-domain (the training domain is different from testing domain) settings. The authors proposed an approach based on Conditional Random Fields (CRF). For testing purposes, the authors decided to compare their approach with Zhuang et al., 2006, using this article's approach as a baseline. For single-domain setting, Jakob and Gurevych, 2010 outperforms Zhuang et al., 2006 for all different domain datasets tested. For cross-domain settings, Jakob and Gurevych, 2010 was still the best performing approach. It should be noted that the performance of Zhuang et al., 2006 for this setting was very very poor, and that the discrepancy between the obtained results for Jakob and Gurevych, 2010 and Zhuang et al., 2006 is quite accentuated, with Jakob and Gurevych, 2010 proving to be a successful improvement over Jakob and Gurevych, 2010.

Q. Liu et al., 2015 also contributed greatly to aspect extraction in opinion mining (which also makes it slightly different from Hu and Liu, 2004, Popescu and Etzioni, 2005 and Ding et al., 2008 as it does not try to classify opinions). When we are talking about using rules to extract aspects, both the tasks of rule creation and selection of rules to use together are costly in terms of time and effort. In this article, the authors propose an automated supervised approach for rule selection. The existing rules were based on syntactical dependency parsing. Their rule selection technique can was capable of selecting rules that performed better when used together. Their work was also closely related with some state-of-the-art approaches presented in Qiu et al., 2011 (that uses a double propagation approach, named as such due to the fact that the extraction of rules is performed using the identified relations between between opinion words and targets and the opinion words and targets themselves) and B. Liu et al., 1998 (where the authors also focused on finding a subset of rules that would then be used to create classifiers that could be employed in many fields of data mining). The authors also compared their approach with Qiu et al., 2011 (because similarly to the authors of this article, they also use rules as input for their proposed algorithm) and Jakob and Gurevych, 2010 (due to the fact that their approach in this article is also supervised and was a state-of-the-art approach). Upon experimentation the authors concluded that their technique was more effective, proving that hybrid approaches can be relevant in opinion mining.

An interesting deep learning approach for opinion expression extraction was presented in Irsoy and Cardie, 2014. In this paper the authors compared the performance between deep and and shallow *Recurrent Neural Networks* (*RNN*), with *Conditional Random Fields (CRF)* and the current state-of-the-art approach for opinion word extraction, *semi-Conditional Random Field (semi-CRF)*. Once testing was performed to compare the two neural network approaches, the authors reached the conclusion that deep *RNNs* outperformed shallow *RNNs*. When the authors compared deep *RNNs* with CRF and semi-CRF approaches (semi-CRF being the current state-of-the-art), the results showed that their approach had the best performance. Of course, these results do not counter the fact that in order to use this approach we would need large volumes of data (which is not always

easy to procure).

In Dai and Song, 2019, the authors try to tackle the lack of labeled training data for neural network based aspect and opinion term extraction. Considering the advantages of rule-based and learning-based approaches for aspect and opinion word extraction, they decided to propose an approach (named RINANTE) that would make combine both of the previous approaches:

- Firstly, a rule-based approach (based on dependency parsing) to automatically extract aspect and opinions from an auxiliary dataset of product reviews.
- Afterwards a Bi-directional LSTM conditional random field based neural model (BiLSTM-CRF for short) is trained with a small human annotated dataset as ground-truth supervision while utilizing the larger rule annotated data obtained from the step before as weak supervision. Different versions of this neural network were developed, but the most promising approach was the one where the authors applied two separate BiLSTM for aspect term and opinion term extraction respectively.

This work ends with a comparison between the performances of the author's system with another 11 different approaches (among them Qiu et al., 2011 and Wang et al., 2016, referred previously). The effectiveness of the proposed approach is confirmed through the tests (with highest F1-score for aspect extraction reaching 86.76% for aspect extraction and 86.34% for opinion extraction). Last interesting aspect of this paper is that the authors made the data that they used for their system (models, rules and datasets) public and stored it in a Github repository[1].

In (Xu et al., 2018), the authors performed fine-grained aspect-based sentiment analysis on product review data from the SemEval 2014 laptops and SemEval 2016 restaurant datasets, more specifically, for the task of aspect extraction. Unlike more complex deep learning approaches, the approach developed by the authors relies on a rather simple *Conventional Neural Network* model while using two types of pre-trained word embeddings (double embeddings): a general domain word embedding (*Glove*) and some domain specific embeddings. The domain specific embeddings were generated using the *Amazon Review Dataset* for the laptop domain and the *Yelp Review Dataset* for the restaurant domain. Among other approaches the authors tested, they also compared the results from their approach while using only the general domain word embedding, or simply just the domain embedding. The results achieved with this approach were compared with other state-of-the-art approaches and achieved outstanding results, outperforming them all: f1-score of 81.59% for the laptops and 74.37% for the restaurant domains.

Another approach that demonstrated good results in aspect-based sentiment analysis was (Luo et al., 2019). Unlike most approaches that simply focused on one of the sub-tasks of the SemEval aspect-based sentiment analysis task, this approach uses two cross-shared *Recurrent Neural Networks* to perform both the task of aspect extraction and aspect sentiment polarity classification. Experiments were performed on three different datasets: for the laptop domain, the dataset used was SemEval's 2014 laptop domain. For the restaurant domain, a merge of the SemEval restaurant dataset from editions 2014 to 2016 was generated. The last dataset is an English Twitter dataset. Compared to other state-of-the-art approaches, the current approach outperforms them in the task of aspect extraction, even beating the previous state-of-the-art approach (Xu et al., 2018). As for the aspect-based sentiment classification task the results were slightly poorer. However, they still were better than the results presented in other competitive approaches.

[1] https://github.com/HKUST-KnowComp/RINANTE

Lastly, another interesting work performed on the subject of opinion mining is about the handling of opinion sentences. In Ganapathibhotla and Liu, 2008, we determine which entities are preferred by analyzing comparative sentences where two entities are compared. The approach proposed by the authors is lexicon-based, using the syntactic function of words obtained through POS tagging, a lexicon of opinion words (developed by one of the authors on a previous article) and WordNet to obtain the comparative words (mostly adverbs and adjectives) that then enabled the authors to detect which entity is preferred. Since there were no other similar works performed at the time, the authors could only assess their own system's performance. Once testing was completed, the results showed that the proposed approach presented accurate results.

In conclusion, as far as opinion mining and aspect-based sentiment analysis goes, recent trends show that deep learning approaches are being favored over machine learning or rule-based approaches, consistently improving over previous state-of-the-art approaches for these tasks, and constantly exhibiting high precision, recall and f1-scores. However, one thing that should be noted is that there isn't a wide variety of datasets for this area of study, as the majority of papers explain that they tested their approaches on the SemEval datasets. These datasets consist of entries of review data, not really dialogues, which is what we would ideally wish for to better fit with our depicted use case. Therefore, the creation of a new dataset with a question-answer interaction between parties where opinions are expressed would go a great way towards enriching the amount of resources available to study this area. Furthermore, it could even be improved on by other parties. Overall, the best performing approach studied so far was (Luo et al., 2019).

3.3 Competitors Analysis

3.3.1 Main Competitors

In this subsection we present the main competitors of Talkdesk. These competitors provide similar services to those offered by Talkdesk, thus they are considered as direct competitors.

- **Gavagai: Explorer** - Gavagai ("Explorer", n.d.) is a Swedish language technology company. While Gavagai doesn't offer a cloud call center service (and thus is not considered a direct competitor), the product it offers, Gavagai Explorer, provides a sentiment analysis tool that determines what are the major sentiments customers have about some related topics. Even though our internship objective is to develop an opinion mining framework that extracts the opinions of customers related to products or brands instead of just determining a sentiment towards a topic, we consider this company a competitor (albeit not a direct one) because of the similarity between our solutions and use cases. Gavagai: Explorer's most relevant characteristics are its extensive language support for their sentiment analysis tool (46 languages) and its support of sentiment classification in 8 different sentiments (positivity, negativity, skepticism, love, hate, fear, desire and violence) instead of the most common 3 (positive, neutral and negative). Their tool employs a heuristic approach instead of a machine learning or deep learning approach.

- **Ameyo: Fusion CX** - Ameyo's ("Sentiment Analysis: Sentiment Analysis Customer Service", n.d.) main product, like Talkdesk's, is a cloud call center solution, as such we classify it as a direct competitor. While their framework differs from ours in the aspect that it is a sentiment analysis tool that is used to gauge the mood of

their customers instead of extracting their feedback about products or brands, we still consider it a competitor. Fusion CX analyzes customer interactions from emails and chat transcriptions and determines the customer's emotional state and identifies when a customer is dissatisfied, allowing agents to redirect that customer to a more experienced agent.

- **Bright Pattern** - Bright Pattern ("Call Center Contact Center Software Features: Bright Pattern", 2020) is also a company that offers cloud-based call center services. It also offers sentiment analysis tools. Bright Pattern's product is built upon the API of IBM' Watson's Natural Language Understanding and it determines a customer's emotions and displays it to an agent, that will then know whether a customer is happy or unhappy.

- **Xdroid: TextAnalytics** - TextAnalytics (Sysadmin@isobarbudapest.com, 2020) is a cloud-based text analysis solution proposed by Xdroid, that analyzes text and extracts keywords and topics and identify the number of times each topic is mentioned with its corresponding polarity. Due to the similarity between the features of our framework and Xdroid's TextAnalytics and the fact that both are to be used in cloud call centers, we consider this company and their service a direct competitor.

- **Hitachi: Sentiment Analysis Service** - Hitachi's Sentiment Analysis Service (Hitachi, n.d.) analyzes text data from social media, customer reviews, mass media and business data such as questionnaires and call centers. Just like our framework, it can be used in contact centers to obtain customer feedback about certain products or brands and can be applied in marketing and product development. While the company isn't a direct competitor with Talkdesk, we still thought it was worth mentioning because they have a product with similar functionalities to the one we will develop during the internship. Hitachi's Sentiment Analysis Service performs data analysis on text data, and it can be performed on data from different media, such as customer reviews, business data (from call centers or questionnaires), newspapers, television or even social media. It also provides a visual representation of customers' sentiments and opinions about a company or products.

Upon analysis, we are able to conclude that the majority of the competing services presented in this subsection are supplied by companies who also offer cloud call center solutions. Most of these solutions are to be used together with other services from those companies. The services supplied by the competitors are also more oriented towards sentiment analysis for topics, rather than fine-grained aspects. An advantage Talkdesk has over these competitors, is that their clients would need to adhere to other services offered by these companies in order to fully integrate them. This would make the adoption of any of these services next to impossible, due to the fact that changing the base where their contact center is built would require a tremendous amount of effort, when a client simply wished to adhere to another service.

3.3.2 APIs

In this subsection we will present the most popularly used APIs for sentiment analysis and opinion mining. These APIs also supports operations similar to the ones performed by our framework, however, unlike our framework and our direct rivals which are services that are straight out ready for use, their functionalities must be implemented into a tool by the developers within the customer company.

- **Google Cloud: Natural Language API** - Natural Language ("Cloud Natural Language nbsp;|nbsp; Google Cloud", n.d.) is an AI and Machine Learning product supplied by Google Cloud. Google Cloud is a service oriented to businesses however it has to be configured by developers in order to become readily accessible, as it is an API. This service also supports sentiment analysis to obtain the overall opinion of customers about a company's products or services. Even though Google calls it sentiment analysis, if their API really does as advertised in their API page, then what they truly offer is opinion mining.

- **IBM Watson: Natural Language Understanding API** - Natural Language Understanding ("Watson Natural Language Understanding - Overview", n.d.) API is one of IBM Watson's modules. It is used by developers to analyze text and extract emotions, entities and most importantly for our context, sentiment. The sentiment analysis made available by IBM Watson for this module is rather simple, assigning a sentiment score to sentences or whole texts. While not a direct competitor, as a company that has a very broad array of areas of business, it is a worthy mention when it comes to sentiment analysis frameworks. From what we managed to establish based on the advertisement of the product, Natural Language Understanding API actually offers a sentiment analysis service (since they only claim to assign a positivety score to a text).

- **Amazon AWS: Comprehend API** - Amazon Comprehend Russo, 2003 API is a service that uses natural language processing to extract insights about the contents of documents. Most importantly, it also features a sentiment analysis tool that companies can use to determine what their customers think about their products. From what we managed to understand from their web page, we believe that they support opinion mining and not sentiment analysis, as per their claim of supplying their users with a tool that can be used to determine what customers think about the users' products.

- **Microsoft Azure: Text Analytics API** - Microsoft's Azure Text Analytics ("Text Analytics", n.d.) API is a cloud-based service that provides natural language processing over text. The main function that this service provides that makes it worth mentioning in the context of our project is sentiment analysis. The sentiment analysis provided by this platform assigns a score between 0 and 1 to a text document. According to how they describe their sentiment analysis service, we can conclude that it is actually a sentiment analysis service and not an opinion mining one.

In this subsection, the APIs presented are supplied by world's leading technological companies. The services offered are also not quite what we would desire in an opinion mining module. Instead, they also focus more on sentiment analysis towards topics, instead of a more detailed analysis of aspects. The pricing also makes the usage of such tools impractical, as the pricing is dependent on the number of requests using the API. After the short introduction about our competitors' services, we will now perform a comparative analysis of the services they provide, which features each service provides and other relevant characteristics. These aspects will be organized in a table to make the comparison of features more clear to the readers (see table 3.1).

Table 3.1: APIs functionalities table

Features	**Google Cloud: Natural Language**	**IBM Watson: Natural Language Understanding**	**Amazon AWS: Comprehend**	**Microsoft Azure: Text Analytics**
Licence type	Paid	Paid	Paid	Paid
Languages Supported	Multiple	Multiple	Multiple	Multiple
Language Detection	Yes	Yes	Yes	Yes
Transliteration	No	No	No	No
Last Released	2019	2019	2019	2019
Tokenization	Yes	Yes	Yes	No
Sentence Segmentation	Yes	Yes	No	No
POS Tagging	Yes	Yes	Yes	No
NER	Yes	Yes	Yes	Yes
Lemmalization	Yes	Yes	No	No
Stemming	No	No	No	No
Chunking	No	No	No	No
Chinking	No	No	No	No
Dependency Parsing	Yes	Yes	No	No
Constituency Parsing	No	No	No	No
Spelling Correction	No	No	No	No
Embeddings	No	No	No	No
Morphological Analysis	No	No	No	No
Sentiment Analysis	No	Yes	No	Yes
Opinion Mining	Yes	No	Yes	No

3.4 Resources and Tools

3.4.1 Libraries

With the growing amount of information available essentially due to the widespread use of the internet, companies have extremely large amounts of customer data available that they have to analyze. This is a colossal task, taking into account the amount of data, so some companies specialize in supplying other companies with tools that enable them to detect customer opinions pertaining their products or services. For this internship, whose goal is to develop an opinion mining framework for the English language, we may use existing open-source Python libraries to perform essential NLP tasks, as most of the existing NLP libraries are actually implemented in this programming language. Below is a list of the most popular open-source NLP Python libraries. After the list that introduces these libraries we present a table with the functionalities supported by each library (see table 3.2).

- **NLTK**: NLTK ("Natural Language Toolkit", n.d.) is one of the oldest and sturdiest NLP libraries for Python. It is also one of the most-known and studied. It allows users to execute many NLP tasks, however it is rather slow and difficult to use, making it impractical to use when we have large amounts of text to analyze, which means that it shouldn't be used for development of commercial applications. It is instead, heavily used in the academic environment, as a learning tool and a main component of systems created in research.

- **SpaCy**: SpaCy ("spaCy · Industrial-strength Natural Language Processing in Python", n.d.) and NLTK are used for the same tasks, however, compared to NLTK, SpaCy is much faster at executing these tasks with larger amount of data. It is also very easy to use, making it a very widely used library in opinion mining systems. Another important aspect that should be mentioned about SpaCy is that it provides models that are trained with neural networks (and can be trained by the users and updated accordingly), which is a feature NLTK does not support. It is also less flexible than NLTK and supports less languages (although it has multi-language models). SpaCy also supports different sizes of language models for each language. As an example, for the English language, SpaCy supports both a lighter and a heavier model. The heavier models offer more functionalities, such as word vectors and higher model accuracy, but the lighter models are also very reliable.

- **Gensim**: Gensim ("gensim", n.d.) is a library mostly used for document similarity or when we want to use word vectors. It does not support most essential NLP tasks (such as POS tagging, NER or dependency parsing) and is mostly used together with other NLP libraries.

- **CoreNLP**: CoreNLP (n.d.) is a library that is supported in the Java programming language, however it can also be used with Python. It supports many languages, however the language models are quite heavy. It is also a rather complex library to use in the sense that you have to write a large amount of code to make use of the NLP features.

- **Pattern**: Pattern ("Pattern", n.d.) is a multipurpose library capable of executing data mining, NLP and machine learning tasks. A feature that is also supplied by this library is a sentence subjectivity/objectivity analyzer. This is very useful, as normally, when we express an opinion, the sentence in which it is expressed in is subjective. While not as well-known as NLTK and SpaCy, it is a very useful library for more complex systems.

- **Polyglot**: Polyglot ("polyglot", n.d.) is a library that supports many languages for different NLP tasks, however due to lack of popularity, it does not benefit from a lot of community support, making issue solving a very slow process.

- **TextBlob**: TextBlob ("Simplified Text Processing", n.d.) provides an intuitive interface for NLTK. It is easy to use and even provides language detection and translation through the Google Translate service. Due to the fact that it uses the NLTK library, this library is also slow, and not recommended for large-scale systems.

3.4.2 Datasets

In order to develop an approach for sentiment analysis or opinion mining, we inevitably need data to test our solutions on. The recent emphasis on ML solutions led to the need of

Table 3.2: Libraries functionalities table

Features	**NLTK**	**SpaCy**	**Gensim**	**CoreNLP**	**Pattern**	**Polyglot**	**TextBlob**
Licence type	Free	Free	Free	Free	Free	Free	Free
Languages Supported	Multiple	Multiple	English	Multiple	Multiple	Multiple	Multiple
Language Detection	No	No	No	No	No	Yes	Yes
Transliteration	No	No	No	No	No	Yes	Yes
Last Released	2019	2019	2019	2018	2018	2016	2019
Tokenization	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sentence Segmentation	Yes	Yes	No	Yes	Yes	Yes	Yes
POS Tagging	Yes	Yes	No	Yes	Yes	Yes	Yes
NER	Yes	Yes	No	Yes	Yes	Yes	Yes
Lemmalization	No	Yes	Yes	No	Yes	No	Yes
Stemming	No	No	Yes	No	Yes	No	Yes
Chunking	No	Yes	Yes	No	Yes	No	No
Chinking	No	No	No	No	No	No	No
Dependency Parsing	No	Yes	No	Yes	No	No	No
Constituency Parsing	No	No	No	No	No	No	No
Spelling Correction	No	No	No	No	Yes	No	Yes
Embeddings	No	Yes	No	No	No	Yes	No
Morphological Analysis	No	No	No	No	No	Yes	No
Sentiment Analysis	No	No	No	No	Yes	Yes	Yes

increasingly larger datasets in order to train those approaches. We will now present some of the most frequently used sentiment analysis datasets for the English language:

- **Large Movie Review Dataset** : Presented in (A. L. Maas et al., 2011), this dataset (A. Maas, n.d.) contains a total of 50.000 movie reviews with either a positive or negative sentiment associated to it (with an even distribution of positive and negative reviews). 25.000 balanced reviews were split for training and testing purposes. This dataset is often used in papers on sentiment analysis where we simply with to test a binary sentiment classification approach.
- **Amazon product data**: Another widely used dataset for sentiment analysis is the Amazon product data dataset (McAuley, n.d.). The complete dataset contains 142.8 million reviews, with each review getting assigned a rating from 1 to 5 (1 being the worst rating and 5 the best), from different types of products sold on Amazon (there are other fields on the dataset, but they aren't relevant to our topic). Smaller subsets of this dataset are often used in sentiment analysis, with the ratings serving as the variable used to define the sentiment of the review.

As for opinion mining datasets (also in English), the list of available resources is substantially shorter:

- **SemEval ABSA Datasets**: SemEval ("SemEval-2016 Task 5", n.d.) is the workshop where aspect-based sentiment analysis was studied the most. The datasets used for this task were slightly modified along the editions of this workshop, but all originate from the aspect-based sentiment analysis task dataset from the 2014 edition of this workshop. This task appeared during the 2014, 15 and 16 editions of this workshop. The datasets encompass laptops and restaurants domains from reviews. Each entry

in the dataset contains the review text and a list of aspect/polarity pairs for each review. The polarity of each aspect can be labeled as either positive or negative, neutral (if it is simply an objective description an aspect) or conflict (in case both a positive and negative sentiment is expressed about an aspect). These datasets and subsequent iterations became the benchmark for this area of study.

- **Sentihood**: This dataset was presented in the conference Coling 2016 ("YRSNLP", n.d.) and it is the work of (Saeidi et al., 2016). This dataset was built using texts taken from the platform *Yahoo*, particularly about locations in London. Each entry in the dataset contains its text and a list of tuples with 3 different fields. Those fields are the target entity of an aspect (the name of a location), the aspect (for example, price or safety of a location), and the sentiment expressed (which is labeled as either positive or negative). Differently from the SemEval datasets, the text extracted from the platform were replies from question-answer pairs, instead of simply being reviews. However, the questions themselves are not present in the dataset, just the replies.

Other datasets we will also mention are related with dialogues. Specifically question-answer interactions. In the context of expressing preferences with a question answering format, the following dataset is presented:

- **Google CCPE**: Google's Coached Conversational Preference Elicitation (Google CCPE for short) ("Coached Conversational Preference Elicitation", n.d.) is a dataset with over 500 annotated conversations and around 12.000 utterances. In this dataset, an individual playing the role of "user" expresses his preferences about the movies domain to another individual representing an "assistant". The main task of the "assistant" is inquiring about the "user's" movie preferences, in an attempt to ensure that the user expresses his preferences without digressing.

3.4.3 Other Resources

Other resources that are often used when performing tasks of NLP, sentiment analysis and opinion mining are presented below:

- **WordNet**: WordNet ("What is WordNet?", n.d.), developed by researchers from Princeton University, is a lexical database of English. It bundles adjectives, adverbs, nouns, and verbs into groups of cognitive synonyms, each of which expresses a different concept, however, it does not contain conjunctions, determiners, particles, prepositions and pronouns. WordNet is accessible to use with the Python library NLTK. WordNet is usable for commercial use ("License and Commercial Use of WordNet | WordNet", n.d.) as long as credit is given to its creators, by including their disclaimer.
- **SentiWordNet**: SentiWordNet (Aesuli, n.d.) is another lexical resource, which, as the name suggests, is based on WordNet. The differentiating factor between these two resources is that SentiWordNet is a resource for opinion mining, because it assigns to each cognitive synonym (or synset) of WordNet three different scores: positivity, negativity and objectivity indexes. SentiWordNet is just like WordNet, free to use ("Creative Commons Legal Code", n.d.) for the production of applications with future commercial use in mind as long as the application produced from it gives due credit to SentiWordNet.

Chapter 4

Approach

In this chapter we will talk about the important aspects of our approach for the accomplishment of the objectives defined for this internship. We will start with the approach's specification. We then proceed to the requirement and risk analysis. Following that is the architecture description. We finish this chapter with the methodology.

4.1 Specification

In this section we will talk in detail about our approach: what dataset we have developed and the criteria for the selection of the approaches for further study. Next, we will also describe the validation process used on the new dataset. Finally, we describe how the performance of the selected approaches on the new dataset was evaluated.

4.1.1 Datasets

The dataset developed used Google's CCPE dataset (Radlinski et al., 2019) as its basis. This dataset covers the movie and series domain. It is available in JSON format. As mentioned previously in section 3.4.2, the dataset contains over 500 conversations and 12.000 utterances. This dataset was selected as a basis because it was a dataset that contains dialogues where opinions and preferences are expressed about movies or series. This coupled with the fact that these preferences are elicited in natural dialogue makes this dataset suitable for our context of use. To our knowledge, there were no other available datasets, for different domains, where we have similar expressions of preferences in a dialogue. The dialogues in the dataset are in English.

Each *conversation* contains the following fields:

- ***conversation_id***: *conversation_id* is an unique identifier assigned to each conversation in the dataset.
- ***utterances***: It is an array of utterances exchanged by the speakers in the conversation.

Next in the hierarchy are the *utterances*. Each *utterance* contains these fields:

- ***index***: It is an index that indicates the order of utterances in a conversation.

- ***speaker***: This field is assigned to the role the person who spoke the utterance. This role can be either *USER* or *ASSISTANT*.
- ***text***: This is the written text from the utterances.
- ***segments***: It is an array containing annotations of certain spans in the utterance's text.

Presently, we proceed to the composition of a *segment*:

- ***startIndex***: It is the position in the utterance, where the annotated part of the text starts.
- ***endIndex***: It is the position in the utterance, where the annotated part of the text ends.
- ***text***: It is the raw text that was annotated.
- ***annotations***: This is an array of annotations for a segment.

Continuing, we have the annotations. Each *annotation* is composed with the following fields:

- ***annotationType***: This field is assigned an *annotation* class. These classes can be: *ENTITY_NAME*, *ENTITY_PREFERENCE*, *ENTITY_DESCRIPTION* AND *ENTITY_OTHER*.
- ***entityType***: This field is assigned an *entity class*. The categories this field can be assigned are: *MOVIE_GENRE_OR_CATEGORY*, *MOVIE_OR_SERIES*, *PERSON* OR *SOMETHING_ELSE*.

We now describe the possible classes for the field *annotationType*:

- ***ENTITY_NAME***: This category is used to mark important entities mentioned in an utterance.
- ***ENTITY_PREFERENCE***: This classification is assigned when a part of a utterance spoken contains an expression of like or dislike towards an entity or some aspect of it. In summary, we can think of the types of sentences labeled with this category as sentences where sentiment or opinions are expressed.
- ***ENTITY_DESCRIPTION***: This label is used when we have a sentence where we are performing an objective description of an entity without displaying any particular like or dislike towards it.
- ***ENTITY_OTHER***: This is the label assigned to other relevant sentences that do not fit in the above conditions. For example, whether a conversation's *USER* has seen a certain movie or not.

Finally, we will describe the possible classes for the field *entityType*:

- ***MOVIE_GENRE_OR_CATEGORY***: This label is assigned when movie genres or some types of movies are present in an utterance.

- ***MOVIE_OR_SERIES***: This category is used when we have a movie or series name in a utterance.
- ***PERSON***: This label is used when we have an actual person's name mentioned in a sentence.
- ***SOMETHING_ELSE***: This last category is used for other important details, such as proper nouns for names of characters in movies or series, or locations present in these media.

Other datasets that were used are the SemEval (Pontiki et al., 2016) datasets for aspect-based sentiment analysis, for the English language. Particularly for the laptops and restaurant domains. Each dataset contains around 3000 sentences extracted from customer reviews of these domains. These two datasets are considered benchmark datasets in aspect-based sentiment analysis since this task was first presented in the SemEval workshop. While these datasets pertain the review domain and are not representative of our desired use case, they are the datasets used for the training and assessment of aspect-based sentiment analysis tasks, present in SemEval workshops from 2014 to 2016. The approaches that we have evaluated are all trained with this data and the models resulting from this training will be used on the newly developed dataset and their performance will be evaluated with the new dataset. The SemEval datasets all possess the same structure:

- ***sentence id***: This is an unique identifier for each review in the dataset.
- ***text***: This field contains the review text.
- ***aspectTerms***: This field contains an array of aspects (if there are any aspects present in a review).
- ***aspectCategories***: This field is unique to the SemEval restaurant dataset. It contains the categories associated with the aspects mentioned in the review and the overall polarity associated to that category (such as price or service). We will not explain in more detail because this field is not going to be used by the aspect-based sentiment analysis approaches selected.

For each *aspectTerm* present in the list *aspectTerms*, we have the following fields:

- ***term***: The field *term* contains the string with the name of the aspect mentioned in a review.
- ***polarity***: This field contains a string with the sentiment associated with the aspect it is paired to. The sentiments can be: *positive*, *negative*, *neutral* and *conflict*. *Positive* and *negative* sentiments are used when the opinion in the review concerning a given aspect is either positive or negative. *Neutral* is assigned when there is an aspect but it is only described and has no sentiment assigned to it. The last label, *Conflict*, is assigned when an aspect receives both a positive and negative evaluation.
- ***from***: This field contains the start position of the aspect in the review text.
- ***to***: This field contains the end position of the aspect in the review text.

Using Google's CCPE dataset as a base, we have created a dataset with aspects and their respective polarities (similar format to the SemEval datasets). Due to the fact that

sometimes in a dialogue, a certain aspect is not explicitly mentioned, and we infer we are talking about it through context, we will not be using a *span* field. *Span* is used when we wish to know where where relevant parts of texts start and end. We are expecting some incorrectness in the annotation process and we will do our best to point them out in the section of the evaluation results of the dataset.

4.1.2 Dataset Validation

Once the dataset was created, we proceeded to perform its validation. For this process, we have resourced other employees from Talkdesk. Their job was to state whether they agree or disagree with the aspects extracted and with the sentiment assigned to each aspect. We asked the annotators whether they agree or disagree (binary classification) with a suggested labeling in order to simplify the validation process. We have built multiple evaluation groups with three elements and have also performed an agreement assessment between pairs of members of the same group. The objective was to compose as many different groups of annotators so that we can cover more of the conversations in the dataset. The metrics that were used are Fleiss' kappa and overall agreement. We have evaluated the results for 10% of the dataset, roughly 50 conversations. We were targeting to achieve 75% agreement rate for our dataset, as it seems a considerable value for our domain (McHugh, 2012).

4.1.3 Selection of Approaches

Once we had a new dataset to test on, we needed to decide which approaches we would test with it. Therefore, we have selected three state-of-the-art approaches for aspect-based sentiment extraction to test with our dataset. In order to select the approaches whose performances we would test with the newly developed dataset, we have used the following criteria:

- **Performance of the approach**: We selected the approaches that obtained the best results in aspect-based sentiment analysis. The metrics we will take into consideration to evaluate performance are: *precision*, *recall* and *f1-score*. Higher values in these metrics will result in the approach being selected.
- **Tasks performed by the model of the approach must be from aspect-based sentiment analysis**: The approach must perform either aspect extraction or aspect sentiment classification.

The approaches that presented the best results in the desired metrics and fulfilled the tasks required were all deep-learning approaches. Those were:

- (Luo et al., 2019): This approach executes both tasks of aspect extraction and aspect sentiment classification. For the former, the results reported in the article were f1-score 82.61% for the SemEval 2014 laptops dataset and 81.06% for the dataset created from the merging of SemEval 2014, 2015 and 2016 restaurant domains datasets. For the latter, the f1-scores for the laptops domain were 60.35% and 72.78% for the restaurant dataset. Another aspect that led into the selection of this approach is that this approach performs both tasks necessary for opinion mining (see section 3.2 for more details on the approaches). While in the article, the authors only report

the f1-score results, their approach also calculates precision, recall and accuracy for both aspect extraction and aspect sentiment classification tasks.

- (Dai and Song, 2019): This approach employs deep-learning to generate rules for aspect extraction. It presented state-of-the-art results for this task, with the SemEval 2014 datasets as train and test datasets respectively. The f1-scores reported in the paper for the 2014 laptops dataset was 86.76% for aspect extraction and 77.92% for the restaurant domain, for that same task. This approach performed better that Luo et al., 2019 on the laptop dataset, but displayed lower results on the restaurant dataset (for more details, see section 3.2). That was already expected, as the training dataset for this domain is smaller that the one in (Luo et al., 2019). Once again, while in the article, the authors only present the f1-score results, however their approach also calculates precision and recall.

All of these approaches had to be adapted so that they could be tested on the new dataset. Furthermore, we slightly altered the validation functions presented in these approaches, as they made use of the *span* component of the aspects in a review in order to evaluate whether an aspect was correct or not. Due to the fact that the dataset developed did not have such a field, we altered altered the approaches' code so that they did not take it that field directly into consideration. We also be added another validation function. While the original ones considered an aspect correct if the Levenshtein distance between the aspect in the dataset with the predicted one was less than 2, we also added one condition where we consider an aspect predicted as correct if its cosine similarity with the actual correct aspect is 75% or more. This value was obtained through parameter adjustment. We selected some threshold values and upon looking at some of the included and excluded true positives and false negatives cases that we had in our dataset, we decided for the 75% threshold. We also performed an analysis of the false positive cases detected by the approaches. We did so because we think these approaches were be able to capture some more fine-grained aspects that were not annotated as aspects in the dataset because of how the dataset was annotated.

4.1.4 Evaluation of Approaches on the New Dataset

For the evaluation of the approaches we used the metrics: precision, recall and f1-score. We considered two different acceptance conditions for true positives: the first one was if the predicted aspect had a Levenshtein distance lesser than 2 with the actual correct aspect, then the aspect candidate was considered an aspect corrcetly identified. The second one was the cosine similarity. We considered a predicted aspect as correct if its cosine similarity was 75% or more similar to the actual aspect in the dataset. This was relevant in situations such as "*I liked the movie Deadpool*". In this case, the dataset might have *Deadpool* only listed as aspect and the predicted aspect as "*movie Deadpool*". However, with cosine similarity we could consider such cases true positives.

The remainder of the aspects predicted were considered false positives and the aspects that were not detected (predicted as aspects by the system) but were annotated in the dataset are labeled as false negatives. Another step we performed, was a more thorough analysis of the false positives detected by the approaches. This was due to the fact that the dataset might not have some aspects annotated that would have been considered as such in the SemEval datasets. An example of such a situation was "I liked the characters in Deadpool". In that case, characters should have been considered an aspect. However, due to the method by which the dataset was built, these types of aspects (specific parts or components of a movie or series entity) were not annotated as aspects in the dataset.

For the detection of aspect polarity, we also used the previous metrics. However, contrary to how they were used before in one of the selected approaches, the metrics were applied individually to each sentiment category (a positive aspect sentiment had a precision associated to it, as well as a recall and f1-score). To summarize, each of the four different sentiments (*positive*, *negative*, *neutral* and *conflict*) had 3 metrics associated to it. The false positives for the different aspect polarities were the number of times an aspect sentiment was predicted as, for example, positive, when it was negative. In this case, our false positive count for the positive sentiment was increased. For the calculation of the false negatives, we considered that the count of false negatives was increased for a specific aspect sentiment category when the aspect sentiment of that category is the correct sentiment annotated in the dataset but it is not matched with the sentiment predicted by the system.

We also attempted to denote problems with the approaches and propose solutions that we could perhaps later implement to solve those problems. However we left the execution of such solutions as a possible task for future work.

4.1.5 Improvement of the Approaches on the New Dataset

As an optional goal, we attempted to improve the results of the approaches. Due to the complexity of some of the architectures of the approaches, we decided to study whether or not there could be any improvement in the results by varying some of the training parameters of the approaches. For that we experimented with some different values for combinations of the following parameters: number of epochs, learning rate and batch size. These parameters are important values for the learning process of the approaches and at the same time required less complex changes in the model configurations.

4.1.6 Implementation of a BERT aspect-based sentiment classifier

Due to a recent trend of utilizing transformers (Vaswani et al., 2017) to solve Natural Language Processing (NLP) tasks in lieu of the previous proposed neural network models, we decided to attempt building a simple aspect-based sentiment classifier using these models. Specifically, we used a pre-trained BERT Devlin et al., 2019 model and fine-tuned it with the data from the *Semeval 2014 restaurants* and *laptops* datasets to train and verify its effectiveness.

4.2 Requirements Analysis

Like in all software systems, we must ascertain its requirements prior to the development. The requirement specification is necessary to ascertain the success or failure of a project as, if all requirements are implemented, we conclude that we finished everything that we set out to accomplish.

The requirements presented below will all have the following characteristics:

- **Requirement Identifier:** The unique identifier for each requirement. Each different type of requirement has a different requirement id prefix, that will be introduced in the respective requirement-type section.
- **Name:** The name of the requirement.

- **Description:** Text containing a brief description of the requirement.

4.2.1 Functional Requirements

Functional requirements are the requirements that specify what the main functionalities of the system are. The Requirement Identifier of this type of requirements is composed by the string "FR" + "number of requirement", such as "FR1", for the first functional requirement. These requirements are more complex than the other requirements presented further down, therefore they require additional labels to be better described. Those labels are:

- **Requirement Priority:** This label is used to classify the priority level of a requirement. A requirement possesses four levels of priority, from 1 to 4, where 1 is the highest priority and 4 the lowest.
- **Requirement Dependencies:** Some requirements can only be implemented when others were previously implemented (for example, we can only use dependency parsing once we have the tokens), thus requirement dependencies is a list of other requirements that must be implemented before we can implement this one.

Now that we have finished describing these requirements' format, we will now list them.

- **Requirement Identifier:** FR1
  Name: Extraction of precision, recall and f1-score metrics
  Description: The selected approaches must be able to calculate the metrics: precision, recall and f1-score.
  Requirement Priority: 1
  Requirement Dependencies: None.
- **Requirement Identifier:** FR1
  Name: Extraction of false positives
  Description: In order to further study the false positives detected by the approaches, each approach must save its detected false positives.
  Requirement Priority: 1
  Requirement Dependencies: None.
- **Requirement Identifier:** FR1
  Name: Extraction of true positives
  Description: The analysis of the true positives detected by each approach and their respective comparison generates a requirement that makes it possible to view the extracted true positive aspects.
  Requirement Priority: 1
  Requirement Dependencies: None.

4.2.2 Non-functional Requirements

The non-functional requirements (identifier code NFR) are requirements that do not represent functionalities of the system but aspects that are mandatory for our system to comply with.

The list of non-functional requirements that must be supported is presented below.

- **Requirement Identifier:** NFR1
  Name: Annotator agreement with the annotation
  Description: The overall agreement between the annotators who perform the validation and the annotation performed by the intern must be over 75%.

4.2.3 Technological Requirements

Technological requirements (identifier code TR) are requirements that are related with the technologies that are going to be used in the project.

- **Requirement Identifier:** TR1
  Name: Programming language
  Description: The approaches to be tested must be implemented either with the Python or Java programming language.
- **Requirement Identifier:** TR2
  Name: Tools and resources licenses
  Description: The tools used for the system must be available for commercial-use (dataset excluded). However, due the academic nature of this internship, the approaches themselves do not have to be available for commercial-use.
- **Requirement Identifier:** TR3
  Name: Tools and resources cost
  Description: The tools used for the system must be available for use free of cost.

4.2.4 Language Requirements

Language requirements (identifier code LR) are the requirements that involve the language that the system must be configured to handle.

- **Requirement Identifier:** LR1
  Name: Dataset text language
  Description: The dataset created must contain dialogue in the English language.
- **Requirement Identifier:** LR2
  Name: Approaches' Text language
  Description: The approaches selected must be able to deal with and analyze a text in English.

4.3 Risk Analysis

In this section we will present the risks that might potentially arise during the project.

4.3.1 Risk Description

Each risk will possess the following components to describe it:

- **Risk Identifier:** This is the risk identifier. All risks will have an identifier which we can use to address it.

- **Name:** As the name suggests, each risk will also have a name. The name makes it easier for the reader to understand what the risk entails.
- **Description:** Following the name, there will be a short description of the risk, to help the readers understand the context of the risk occurrence.
- **Probability:** The probability of a risk occurring. It can be classified in the following classes:
  - **High**: Probability assigned to a risk that has a very high probability of happening, which we define at around]70%, 100%].
  - **Medium**: Probability assigned if a risk has an probability of occurring between]30%, 70%].
  - **Low**: Probability assigned to a risk that has a small probability of occurring. Interval defined between [0%, 30%].
- **Impact:** Impact basically describes the threat level that the risk has to endanger the completion of the project. The levels assigned to this aspect are:
  - **Maximum**: This classification is assigned if the risk becomes an actual problem and the project schedule or the developed system's performance is significantly affected.
  - **Medium**: This classification is assigned if the risk becomes an actual problem and the project schedule or the developed system's performance is slightly affected.
  - **Minimum**: This classification is assigned if the risk becomes an actual problem and the project schedule or the developed system's performance is only affected very lightly.
- **Handling Time Limit:** This aspect represents the time required to handle a risk should it become an actual problem, for example, a risk that has a very high impact on our project must be swiftly dealt with. Below we present its levels:
  - **Immediate**: The problem must be handled within 1 week.
  - **Short**: The problem has to be dealt with at maximum within 2 weeks.
  - **Medium**: A problem with this Handling Time Limit must be solved between 2 to 4 weeks.
  - **Long**: This is the longest interval granted to solve a problem. The time limit defined to solve a problem with this classification is up to 6 weeks.
- **Mitigation Strategy:** When talking about risks, it becomes very important to minimize their chance of occurrence, hence the need to define mitigation strategies that we will have to apply in order to ensure the smoothness of the internship.

4.3.2 Risk Presentation

In this section, we will present the risks we identified for this internship. In order to make the comparison between the risks more understandable, we also created a risk matrix (see figure 4.1), where we placed our risks (based on the probability of the occurring and their impact).

- **Risk Identifier:** R1
  Name: Lack of annotated opinion mining datasets
  Description: Due to the fact that opinion mining is a fairly recent area, there is a lack of freely available opinion mining datasets.
  Probability: Medium
  Impact: Maximum
  Handling Time Limit: Medium
  Mitigation Strategy: One possible solution would be to create a manually annotated opinion mining dataset (which is one of the objectives of this internship). This task may take too long and affect the project schedule. Another solution would be to further look for more datasets, being careful to read the type of license of the found datasets, as datasets with non-commercial licenses can't be used for the training of a model that would be later used for commercial use.

- **Risk Identifier:** R2
  Name: Lack of familiarization with NLP and opinion mining
  Description: At the start of the project, the intern might not be familiar with NLP or opinion mining, such as the lack of knowledge of the tasks performed in NLP and on how to implement such tasks and existent approaches.
  Probability: Medium
  Impact: Medium
  Handling Time Limit: Medium
  Mitigation Strategy: Reading about NLP and opinion mining. Completing tutorials online will also help the intern to gain practical knowledge of the techniques used in these areas.

- **Risk Identifier:** R3
  Name: Use of new technologies and tools
  Description: Initially, the intern might not be familiar with the technologies and tools he might need to use further down the project timeline.
  Probability: Medium
  Impact: Medium
  Handling Time Limit: Medium
  Mitigation Strategy: Reading about the existing technologies and tools and completing tutorials that make use of these technologies and tools in order to acquire practical knowledge to later apply in the project.

- **Risk Identifier:** R4
  Name: Failure to validate dataset
  Description: It could be complicated getting enough people to collaborate on the validation of the dataset
  Probability: Medium
  Impact: Maximum
  Handling Time Limit: Immediate
  Mitigation Strategy: Securing for volunteers early, so that we do not have have delays in the validation attributed to a lack of people for the process.

4.4 Planning

In this section we will talk about the planning that was developed for the first and second semesters of this internship.

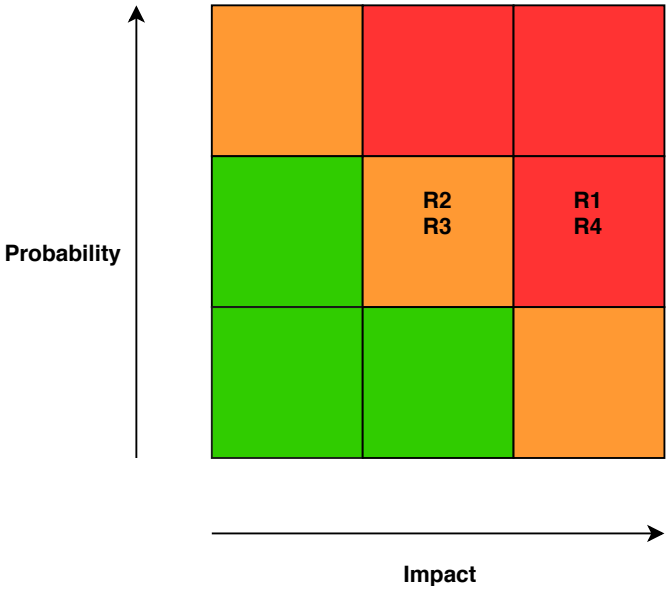

Figure 4.1: Risk matrix.

4.4.1 Planning: 1st Semester

During the first semester, there were no deviations from the schedule defined at the beginning of the internship. The first two weeks of the internship were dedicated to Talkdesk Academy, which is a program to integrate new members in the company and give an insight on how the company works and the technologies used.

After Talkdesk Academy, we had an Up-to-Speed exercise to get the new members used to the libraries and technologies they will later on need to perform NLP tasks.

The remainder time of the first semester was used to learn about natural language processing, opinion mining and writing the thesis. We started with the Background Knowledge (section 2), followed by a research of the resources and tools available that are useful for the tasks we would need to execute later during the second semester, and, we also performed a competitor analysis to assess who are our business rivals (which companies have similar products to the one we are developing).

Afterwards, we started detailing the Proposed Approach and detailed the methodology used during the internship and the timeline predicted and actually followed for this semester and we finish the planning section with the expected plan for the second semester.

Once the introduction and conclusion sections were concluded, we entered the review period, as per scheduled, where we proceeded to perform the necessary modifications on the document as per the feedback obtained from the advisors from the company and university.

While in the work plan, delivery was expected to occur on the 17th of January, the actual delivery occurred on the 20th. This was due to the need of performing more corrections before the delivery.

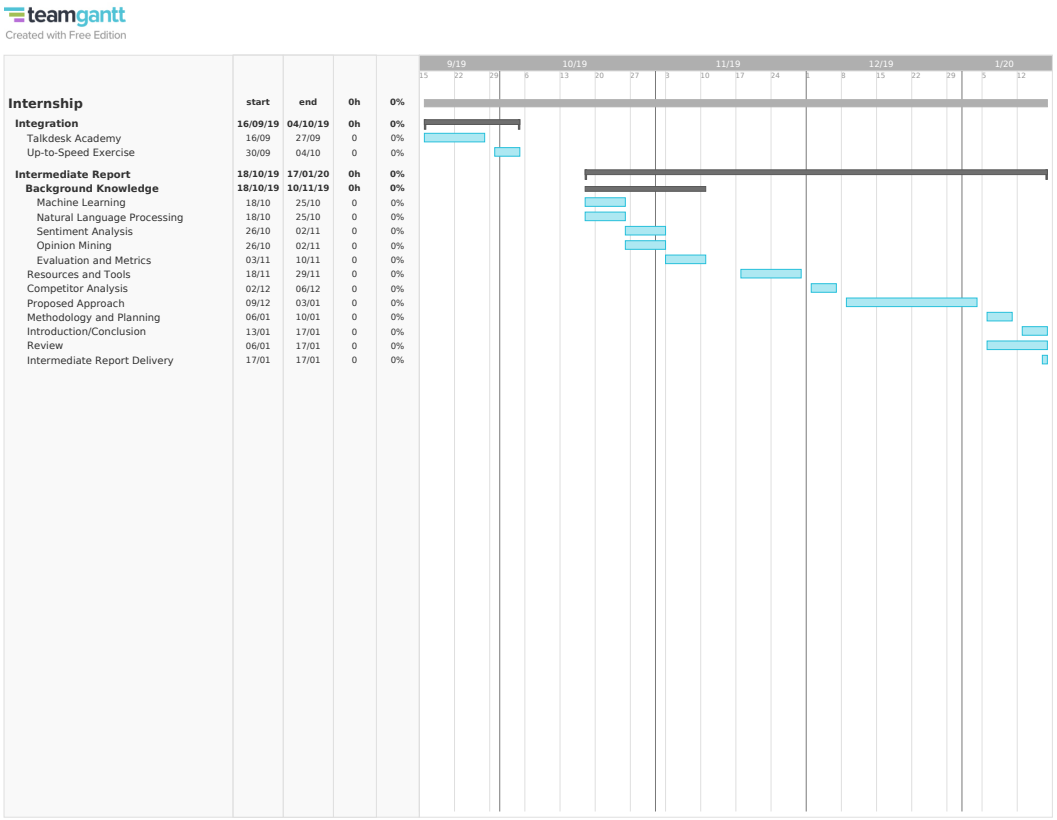

Figure 4.2: Internship plan for the 1st semester.

4.4.2 Planning: 2nd Semester

Due to changes in the internship's objectives, the proposed planning for the second semester presented in the intermediate defense was changed. We started by defining new objectives based on the feedback of the intermediate defense. Once that task was completed, we searched for a dialogue dataset that could be adapted to the context of our use. Afterwards, we defined the approaches to be thoroughly compared. Then we began the construction of the dataset, so that it suits our needs. After creating the dataset, we performed a small assessment of its quality and applied some necessary fixes. Once that task was completed we proceeded to the evaluation of the annotation by other parties. At the same time that the annotators performed the validation, we proceeded to the implementation of the necessary changes in the approaches so that we could test them on the new dataset. After assessing the performance of the approaches we decided to add the common false positives obtained from the approaches to enrich the dataset. Once finished, we attempted to improve the results of the approaches and we also built a simple BERT aspect-based sentiment classifier model, and we started writing the final report while submitting chapters for review as they were finished. In the last week, we reviewed and rewrote the necessary aspects of the report before submitting it.

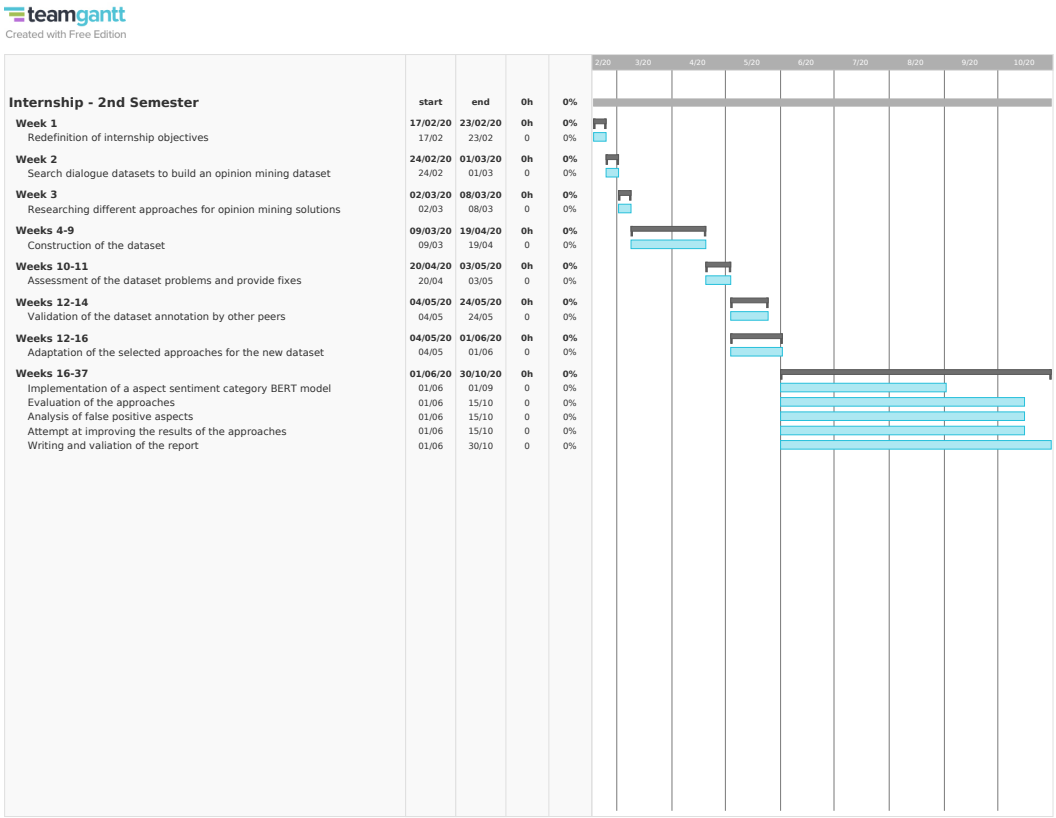

Figure 4.3: Internship plan for the 2nd semester.

Chapter 5

Implementation

As previously described, this project's main goals are the development of a new opinion mining dataset and a comparison of performances between approaches on the new dataset. In order to fulfill these goals, certain tasks had to be undertaken. In this chapter we will present the tasks executed for the fulfillment of these objectives and describe how they were completed.

5.1 Dataset Development

As mentioned in section 4, the dataset developed was built upon Google's CCPE dataset. Now we will present the tasks performed in order to obtain the final dataset.

5.1.1 Dataset Format

We started by defining what fields we wanted the new dataset to contain. Each dataset entry corresponds to an utterance taken from the Google CCPE dataset.

- ***id***: *id* is the field that contains the identifier of an entry in the dataset. This identifier is actually defined by the order in which a utterance is in the dataset. For example, the first utterance/entry in the dataset would have the *id:"0"*.
- ***conv_ id***: *conv_ id* is the field that contains the *conversation id* to which an utterance belongs to. These values are taken from the Google dataset *conversation_ id* field.
- ***text***: *text* contains the utterance text. This field is also filled in with data taken directly from the original Google dataset.
- ***terms***: This field contains a list aspects that are present or discussed in a utterance. This list can sometimes be empty.
- ***term***: This field is one of the fields that help compose an aspect present in a utterance. This field contains the name of the aspect that is being discussed. In order to fill in this field, we will resort to using the annotation from the Google dataset.
- ***polarity***: This field is similar to the polarity fields in *SemEval's* aspect-based sentiment analysis datasets. This field can be either one of these options: *Positive*, *Negative*, *Neutral* and *Conflict*.

All fields in the dataset, with the exception of *aspects*, are filled with *string* type values. However, fields within *aspects* that comprise an aspect are also filled in with *string* values.

We opted not to add a *span* field to the dataset due to the fact that sometimes in a utterance we are referring to an aspect that is not explicitly mentioned in the current utterance. However, that could be added to a list of future tasks. We could implement *span* by assigning it two identifying values: The first would be an index of the utterance where he was explicitly mentioned (which is currently also represented by *id*). The last would be obtained using the *startIndex* and *endIndex* fields from the Google Dataset. These fields contain the start and end positions of specific strings in a utterance.

5.1.2 Dataset Construction

In order to obtain the dataset in the format presented in section 5.1.1, we developed an automatic approach that takes advantage of Google's dataset annotations. It should be noted that, the field names used below are the same ones presented in section 4.1.1. They are matched to the description of Google's dataset.

We defined that each *utterance* in the Google dataset would be considered an entry in our dataset, and that its *id* will be the order in which it appears in the original dataset. *conv_id* would contain the *conversationId* value that the utterance belongs to in Google's dataset.

For aspect extraction, we assume that all *entity_names* from the dataset have the potential to be aspects. We also look into the *entity_preferences* and *entity_description* fields, as these are the fields where in a utterances where we talk about relevant characteristics or sentiment about *entity_names*.

We search from the *entity_preferences* and *entity_description* list of *segments* (from the utterances of the Google dataset) for all literal mentions of elements in the list of *entity_name elements*. This method solves the cases where the *entity_name* is mentioned in the *"USER"* preference utterances and we obtain entity name and entity preferences pairs (pairs that contain the *entity_name*, the entity_preference, the polarity assigned to that preference and the utterance id).

Another situation that had to be handled is when a *entity_preference* or *entity_description* does not contain an explicitly indicated *entity_name*:

To solve that problem, we saved the last *entity_names* for each *entityType* and then we match the *entity_preference* or *entity_description's entity_type* to the last saved *entity_name* with the same *entityType* and we established the connection between them. We obtain entity name and entity preferences pairs (pairs that contain the *entity_name*, the *entity_preference*, the polarity assigned to that preference and the utterance id). We also assume that utterances from the *"ASSISTANT"* may contain aspects. They might also contain information that is necessary for us to know what a *"USER"* is commenting on.

For sentiment polarity, we have 4 possible different classifications: "Positive", "Negative", "Neutral" and "Conflict". From the pairs formed with a mapping between (*entity_name*, *entity_preferences*) and (*entity_name*, *entity_description*), we performed the following operations:

- For (*entity_name*, *entity_description*) pairs, we assume that the polarity is "Neutral". This because *entity_description* is used for entity description purposes, thus having no sentiment associated to it.

- For (*entity_ name*, *entity_ preferences*) pairs, we use the *NLTK's* library sentiment analysis tool *Vader* to define the aspect polarity. We used *Vader's compound* field as a threshold[1] to define which polarities would be assigned from the remaining categories, *"Positive"*, *"Negative"* and *"Conflict"*. When we use send a string from an *entity_ preference* to *Vader* we remove the *entity_ name* because they might negatively influence *Vader*. Take the following sentence: *"I liked Die Hard"*. Sending the entire utterance to *Vader* returns a *"Negative"* polarity. However, by removing the *entity_ name* and replacing *entity_ names* with the pronoun *it*, we obtain a *"Positive"* polarity, and our aspect / polarity pair would be (*"Die Hard": "Positive"*).

There was also another aspect that we attempted to handle. Short replies to queries. Sometimes, a speaker would ask a question and the reply would come in the form of a *"yes or no"* answer. Other similar situations would include questions such as *"what is your favorite movie?"* and the reply would be just an *entity_ name*. It would be complicated to know what aspects we talking about in these utterances and deduce polarities with just that reply.

In an attempt to improve the annotation in these cases, we decided to analyze the utterances to see if they possess certain combinations of POS[2] elements in their constitution. We make use of the POS tagger and *dependency parser* of *Spacy*. Take this query and answer: *"Did you like Deadpool?"* and *"No"*. We search the reply for certain POS elements. If we find that there aren't enough informative elements, we use information from the previous utterances. In this case, we establish that we are talking about *"Deadpool"* and that we are asking whether someone "liked it". We extract *"Did you like"* from the utterance and send it to *Vader*. Then depending on the reply *"yes or no"*, we would leave as it is or have the polarity returned negated. In the previous example, the question elements would be returned with a *"Positive* polarity. However, because the reply is a negation, we negate the polarity to *"Negative*.

The entity pairs constructed are then mapped to their respective utterances and used to build the dataset. One weakness we are aware that our dataset possesses is that only the annotated entities from the original dataset can become aspects. Compared to **absa** benchmark datasets, we are lacking the annotation of more fine-grained aspects. Take the following example: *"I liked the actors in the movie."*. Because *actors* wasn't a labeled entity in original dataset, it does not get annotated as an aspect in our dataset. We are aware of this, and as future work, we will consider the addition of these types of aspects to the dataset.

5.1.3 Dataset Improvement

Before submitting the dataset for evaluation by third parties, we performed a evaluation on a sample of the dataset to detect problems in the creation process of the dataset. We also proposed and attempted some fixes. For this small evaluation, we randomly selected ten conversations and attempted to verify whether the aspects we expected to be extracted were actually detected and if the polarities assigned to them were also what we were expecting. From that analysis, we detected some problems (see table 5.1):

[1]Threshold for the remaining polarities are: (Conflict: abs(Vader(compound)) <= 0.2); (Positive: Vader(compound) > 0.2); (Negative: Vader(compound) < -0.2). Threshold achieved by experimenting with different values.

[2]The POS tags we searched for in utterances were: "VERB" (for verbs), "AUX" (for auxiliary verb), "PRON" (pronouns), "NOUN" (nouns), "ADJ" (adjectives) and "ADV" (adverbs). The dependency relation we also searched for was: "neg" (negation).

Table 5.1: Problems detected in dataset

Problem Identifier	**Problem Description**
#1	Short yes or no reply utterances
#2	Short reply utterance, normally with just the liked / disliked entity
#3	"Positive" and "Negative" cases get labeled as "Conflict"
#4	Confusion between "Negative" and "Positive" sentiment labels
#5	Missing expected entity_names and entity_preferences in Google dataset utterance
#6	Incorrect mapping of multiple implicitly related entity_names in one preference

Table 5.2: Solutions presented for corrections in dataset annotation process

Problem Identifier	**Solution Description**	**Status**
#1	We can use a parser to check what elements are in the reply. For example: When replying with a negative or positive answer (negation: yes or no) we can assume that we need the sentence of the question before. If the fragment of the question utterance turns out positive and the reply is positive, then we conclude that it is positive, otherwise we assign it as negative.	Done
#2	Similarly to the previous problem, we use elements from the previous utterance to determine the polarity of the aspect.	Done
#3	We use the "compound" element of Vader with a threshold to define the difference between: "Positive, "Negative" and "Conflict"	Done
#4	We use a similar solution approach as for #3	Done
#5	We would need to manually search for all missing annotation	Not done
#6	We would need to see if the sentence refers to multiple entities and look for a group of matching entities in the previous utterance	Not done

Problems #1 and #2 concern short answers to queries. When asking a question, an *"Assistant"* may ask if an *"User"* liked or disliked a certain movie or if he had a favorite or least favorite movie. The replies to these types of questions can come short on information. The shortest answers only include an affirmative or negative reply (#1) or an *entity_ name* (#2).

Problems #3 and #4 concern polarity assignment. *"Conflict"* polarity cases tend to result in confusions between *"Positive"* or *"Negative"* labels (#3). Other times, there are some cases where *Vader* wrongfully assigns a *"Positive"* or *"Negative"* when it should have been the other (#4).

Problem #5 is related to missing annotations in the original dataset. In some entries of the Google dataset we are missing *entity_ names* or *entity_ preferences* annotations which cause our approach to fail forming the aspect / polarity pairs. Finally, problem #6 is related when we have multiple entities present in an utterance and in a following utterance we talk about them without explicitly mentioning them. The sentence *"Yes I loved them"*, preceded by *"Did you like The Matrix and Terminator?"* makes us infer that the speaker liked these two movies. The approach only picks on the last matching *entity_ name*, meaning it only matches (*"Terminator": "Positive"*).

In an attempt to fix these problems, we proposed the following fixes (see table 5.2):

The solutions for problems #1 to #4 were implemented in the final process to generate the dataset (described in section 5.1.2). The last two weren't due to time constraints. We can consider these placing these tasks as future work.

Each annotator had a Google Spreadsheet document, with 11 pages corresponding to the *conversation_id* that they belong to. The first page contains a README file to help the raters understand what is required from them in this task. For the remaining ten pages, each page contains data from the conversation whose identifier is on the name of the page. This spreadsheet possesses the following fields:

- ***Utterance***: it is a field that contains the utterance from a speaker from the dataset.
- ***Aspect-Sentiment pair***: This field contains the aspect extracted from a utterance with its respective annotated polarity. When a *utterance* does not contain an aspect, its respective *Aspect-Sentiment pair* will appear as empty (*"[]"*). When an utterance contains multiple aspects we added another line to the Google Sheets page, this still containing the same *utterance Validation:Aspects* and *Validation:Sentiment*. The only change is the entry field *Aspect-Sentiment pair*, which contains the other aspect / polarity annotation for that *utterance*.
- ***Validation:Aspects***: This field was one of the fields the raters had to fill in. The values a rater could assign were either 1 to agree with the aspect suggested or 0 to disagree.
- ***Validation:Sentiment***: Also one of the fields the raters had to fill in. The values the raters had to choose from were either 1 to agree with the sentiment assigned to the aspect of the same entry or 0 to disagree.

The metrics we used to study agreement between the raters and whether or not our dataset could had been well labeled were:

- **Fleiss' kappa:** in order to study the agreement between the grouping consisting of three annotators.
- **Cohen's kappa:** to assess the agreement between the the pairs of annotator groups we could form with the 3 annotators participating in the evaluation.
- **Agreement between the annotators and the labeling suggested for the dataset:** it records the frequency of times when the specified annotator groups (both three person and pair annotator groups) agreed among themselves and with the proposed aspect or sentiment field.
- **Disagreement between the annotators and the labeling suggested for the dataset:** which inversely specifies the frequency of times when the specified annotator groups agreed among themselves that the labels proposed for the utterances of the dataset were incorrectly assigned (unfit for the situation).
- **Agreement between the annotators:** finally, to verify whether the annotators had a high agreement among themselves and were being consistent.

We used these metrics in both the assessment of agreement for aspect extraction and sentiment analysis. Furthermore, we also used these metrics when we separately analyzed the different possible categories that were assigned for both the aspect (whether an utterance was lacking in aspects or we had suggested a valid aspect) and sentiment fields (among the four different sentiment categories what was the distribution of agreement for each one).

5.2 Dataset Annotation Validation

Once we believed that the dataset has achieved a reasonable quality for the sample we tested, we decided to proceed to the validation step. In this step we wish to know whether the other parties who are going to perform the role of raters agree with our annotations. The validation of our dataset occurred in two phases: a starting phase and a main validation phase.

5.2.1 1st Dataset Annotation Validation

Before attempting to request multiple groups of We started by performing a smaller validation with three peers who volunteered for the task. For the staring phase, we proceeded with the following actions:

- We searched for 3 volunteers for the task of annotation validation.
- From the 502 conversations available in the dataset, we selected 10 at random for the raters to evaluate.
- We created Google Spreadsheets for each rater to fill in.
- Once every person finished validating the dataset, we proceeded to study the agreement between the annotators and our suggested fields for the dataset.

5.2.2 2nd Dataset Annotation Validation

After analyzing the results from the first validation and taking into account feedback, some modifications were made to the README file, with the intent to help clear doubts that the annotators may have.

Once gain we searched for volunteers. This time we hoped to establish more than one group, so we gathered 9 volunteers. Each group of 3 would annotate a different set of conversations. The structure of the Google Sheets document was the same as for the previous validation. However, the number of conversations each annotator would rate in this phase was 30, not 10 as it was previously. This brings the total of conversations that were rated in this phase to 90.

The metrics extracted and the tools used were the same as for the previous validation phase. In total, 100 conversations were evaluated by raters.

5.3 Adaptation of Selected Approaches

While we waited for the raters to finish evaluating the annotations for the conversations assigned to them, we started to perform the necessary adaptations for the three approaches we selected to compare. Among the content that needed to be changed was the format of our new dataset (which we had matched with the input from the other datasets used in the approach). We also decided to add another evaluation function to assess the performance of the approaches. While the standard would be to consider as a true positive an aspect that has an edit distance 2, we also decided to alternatively consider as a true aspect a predicted aspect that has a cosine similarity of 75% or over with the actual true aspect.

We also added functions to extract the false positives and write them to text files, as we believed that some false positives that were not labeled as aspects in the dataset could still be considered aspects. Among other functions that had to be altered were the input reading functions. Other adaptations were made to the code of each approach to make the insertion of input parameters an easier task (such as the evaluation function to be used and the dataset to evaluate the approach on) and the storing of the trained models. Lastly, we also had to perform some changes on the format of test dataset (Google dataset) and added some components such as a list of tokens per utterance in the dataset, so that it would be understood by our models. The approaches evaluated were:

- Rule Incorporated Neural Aspect and Opinion Term Extraction, shortened to RINANTE (Dai and Song, 2019). See section 3.2 for more details on the approach.

- Dual Cross-Shared RNN, known as DOER (Luo et al., 2019). Please read section 3.2 for more details.

These approaches have code repositories which we will use and adapt to perform the performance evaluation.

For RINANTE, we did not have to change much of the dataset. The dataset was written in *json* format so that the model could read the test data. Nothing too different from what we had before. We simply created the necessary utterance token file that contains a list of utterance tokens per dataset utterance (entry). In order to create this additional file we compared different tokenizer tools from the following libraries: split from the *string* library and the *tokenizers* from *NLTK* and *spacy's* small model. We compared the tokenizers on the training dataset utterances (to which we have access to) and select the one that most closely resembled the original *tokenization*, as there wasn't any specification in the RINANTE approach code or paper about how they generated the auxiliary token files. We used *spacy's* tokenizer tool to build these file, as it was the one whose *tokenization* method most closely resembled the tokenization shown in the *laptops* and *restaurants* train datasets. We also checked for any previously overlooked invalid characters ("such as u2000 to replace punctuation on some words).

5.4 Improvement of Selected Approaches

Once we assessed the performance of the approaches we also attempted to improve them. Due to time constraints, we opted to try and change some parameters from each approach and observer whether we obtained better results or not. The changes on the values of these parameters are based on their baseline parameters (which we considered to be the parameters presented in the approaches' articles). For example: taking into account that the number of epochs of the baseline DOER approach is 50, we defined that we would test other number of epochs such as 30 and 70. We started by selecting an inferior and superior value for the new value of the parameters and we proceeded from there. Among the parameters that we decided to experiment on were: the batch size, number of epochs and the learning rate. The metrics used to evaluate whether there was an improvement in the approaches were the same as the ones used to assess the performance of the approaches on the new dataset.

5.5 Implementation of an aspect-based sentiment classification BERT model

Upon reading (Sun et al., 2019), and finding a simple example of an implementation of sentiment analysis using BERT ("Google Colaboratory", n.d.), we decided to see if we could fine-tune a BERT model to see if we could successfully predict whether a certain target could be considered as an aspect, and, if so, deduce what is the sentiment expressed.

While the original code presented in (**google**) was used to read an input string (*input_ a*) to perform a classification, we took advantage of BERT's flexibility that allowed us to use a second input string (*input_ b*) with little effort and associate it with the first input and establishing a relation between them that would be represented in the output (as a*classification class*). We started by altering the *Semeval 2014 laptops* and *restaurant* datasets to a format that BERT could read. Each dataset was already divided into a train and a test portion. For each entry of the dataset, if the utterance/review had an aspect, then we would modify the dataset so that an entry in it would include the utterance text, the aspect text and the classification of the relation. Due to the need to also have some negative class samples (where there isn't a relationship between the utterance and the aspect we are binding the dataset entry with), we decided that for entries in the datasets where the utterances don't have a single aspect, we will randomly add a non-related aspect and label it with as not related. In the cases where an aspect was actually present in a utterance, we assigned one of the following values to the *utterance text* and *aspect* pair:

- **1** -The second string element is an aspect of the utterance, and it is contains a positive polarity.
- **2** -The second string element is an aspect of the utterance, and it is contains a negative polarity.
- **3** -The second string element is an aspect of the utterance, and it is contains a neutral polarity.
- **4** -The second string element is an aspect of the utterance, and it is contains a conflict polarity.

This was done to both *Semeval 2014* datasets, for both the training and testing sets.

We then proceeded to do the fine-tuning of two BERT models, each one using one of the different domain train datasets. The validation of the models was done using the test datasets of each domain.

Chapter 6

Results

In this section we will present the results obtained during the validation of the developed dataset and during the evaluation of the approaches.

6.1 Dataset Validation Results

In this section we will be focusing on the evaluation of the dataset developed during the internship. The evaluation process was divided into two phases: the first and second validation phases. For each validation phase and group we also present relevant information about the dataset assembled and assigned to each group of annotators participating in the validation tasks.

6.1.1 First Validation Results

The first validation phase was our first attempt at defining the validation protocol. We started by gathering three volunteers that composed a group. We then randomly selected 10 conversations from the dataset to build our validation sample. We then gave a copy of the validation sample to each of the three volunteer raters to analyze. We asked them whether they agreed with the aspect and sentiment labels assigned to each utterance of a conversation. Each rater was given an *excel* document to fill out with their classification. This document also contained a *README* sheet which contained some guidelines for filling out the document. Such guidelines included an overview of the concepts of aspect and the different available sentiments that we asked the volunteers to classify. As previously stated in 5.2, we used agreement, Cohen's kappa and Fleiss' kappa as our evaluation metrics.

In table 6.1, we organized relevant information about the sample dataset used for validation in this phase. To summarize, the most important characteristics presented in the table are: the number of utterances, the number of labeled aspects (*Number of aspects*) and the frequency count of each sentiment of an aspect.

In table 6.2 we compare the selections of the group of three annotators, for the aspect and sentiment separators presented in the *excel* file. Firstly, the reported Fleiss' kappa value is not very high for either the aspect and sentiment fields. Looking at the second row of values in the table, we can also observe that the annotators do not have a high agreement between themselves and the proposed labels for each field and that they did not frequently disagree with our proposed annotations. Looking at the last row, the overall

	Dataset sample description
Validation stage	1st
Number of groups in validation stage	1
Validation group #	1st
Number of raters per group	3
Number of conversations	10
Number of utterances	233
Number of aspects	101
Positive sentiment aspects	64
Negative sentiment aspects	21
Neutral sentiment aspects	9
Conflict sentiment aspects	7

Table 6.1: Description of the dataset sample generated for the first stage of validation.

Validation phase 1 | Group 1 | All raters	**Aspect**	**Sentiment**
Fleiss' kappa	0.156	0.279
Agreement between annotators and the labeling	0.622	0.541
Disagreement between annotators and the labeling	0.034	0.077
Agreement between annotators	0.657	0.618

Table 6.2: Results of analysis of agreement between all annotators from group 1 for the labeling of the new dataset in the first validation phase.

agreement between annotators is not very high, which meant that there were diverging opinions within the group. We also observed with more detail each possible labels of the aspect and sentiments fields.

For the aspect field, the annotators were presented with either an aspect suggestion or an empty field symbolizing the lack of aspects present in a utterance. In table 6.3), we observed that Fleiss' kappa value was higher for the aspects that were suggested. However, kappa for the utterances with no aspects suggested recorded lower values. We also remarked that the annotators had a high agreement between themselves in the cases where we supplied suggestions of aspects for utterances. In the cases where we suggested an absence of aspects, there was less agreement among the annotators, not quite reaching 50%. The low agreement between the annotators and the proposition of an utterance being empty of an aspect coupled with the fact that the disagreement between the annotators and the proposed label led us to conclude that there were relevant divergences of opinion within the group of annotators and that some of this group's members believed that utterances that we had marked as lacking in aspects actually contained some aspects.

Validation phase 1 | Group 1 | All raters	**Non-empty asp.**	**Empty asp.**
Fleiss' kappa	0.433	0.021
Agreement between annotators and the labeling	0.842	0.455
Disagreement between annotators and the labeling	0.030	0.038
Agreement between annotators	0.872	0.493

Table 6.3: Results of a detailed analysis of the agreement between all annotators for the labeling of aspects in the new dataset in the first validation phase.

In table 6.4 we studied the sentiment field. We studied the field by further analyzing each of its possible categories. Fleiss' kappa for the positive and negative sentiment fields

Validation phase 1 Group 1 All raters	Pos. asp. sent.	Neg. asp. sent.	Neu. asp. sent.	Conf. asp. sent
Fleiss' kappa	0.667	0.339	-0.071	-0.050
Agreement between annotators and the labeling	0.842	0.812	0.00	0.00
Disagreement between annotators and the labeling	0.030	0.078	0.444	0.429
Agreement between annotators	0.872	0.890	0.444	0.429

Table 6.4: Results of a detailed analysis of the agreement between all annotators for the labeling of sentiments in the new dataset in the first validation phase.

were higher than for the neutral and conflict sentiments. In fact, for the neutral and conflict sentiments the value is negative, meaning that, had the annotators rated these sentiments randomly, they would have had a higher agreement. This aspect is further backed by the fact that the agreement between the annotators and the positive and negative sentiment labels suggested is high and that the agreement for the neutral and conflict labels is 0, meaning that the annotators never agreed when we proposed these labels. However, looking at the disagreement values between them and the neutral and conflict labels we can see that they believed around 40% of the time that these labels were not the correctly assigned. Overall, glancing at the agreement between the annotators of the group we can see that they do not have the same opinions when it comes to the neutral and conflict class sentiments. Since we finished studying the group of raters as a whole we proceeded to study the agreement between the pairs of annotators that we can form within that annotator group.

Validation phase: 1 Group 1		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Asp.	Sent.	Asp.	Sent.	Asp.	Sent.	Asp.	Sent.
Annotator pairs	**#1 and #2**	0.124	0.226	0.635	0.567	0.052	0.103	0.687	0.670
	#1 and #3	0.166	0.240	0.657	0.762	0.052	0.112	0.708	0.674
	#2 and #3	0.443	0.564	0.880	0.803	0.039	0.090	0.918	0.893

Table 6.5: Results of analysis of agreement between different annotator pairings for the only group in the first validation phase.

Upon inspecting the table 6.5, we can see that the both the Cohen's kappa and the agreement between annotators #1 and #2 and #1 and #3 is significantly lower than the same values for the pairings of annotators #2 and #3. This data is leading us towards the idea that rater #1 has diverging opinions compared to raters #2 and #3. The information in table 6.6 also tells us that rater #1 is more critical in the situations where we annotated entries of the dataset as lacking in presence of an aspect.

Finally, table 6.7 shows us more details about the study of the agreement pertaining the sentiments among the pairs of annotators. Both Cohen's kappa and agreement between annotators and the labeling lead us to believe that there is consistency among the annotator

Validation phase: 1 Group 1		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.
Annotator pairs	#1 and #2	0.431	0.090	0.871	0.455	0.040	0.061	0.911	0.516
	#1 and #3	0.590	0.105	0.911	0.462	0.040	0.061	0.951	0.523
	#2 and #3	0.337	0.560	0.842	0.909	0.040	0.038	0.882	0.947

Table 6.6: Results of a detailed analysis of the agreement between all annotator pairs for the labeling of aspects in the new dataset in the first validation phase.

Validation phase 1 Group 1		Pos. asp. sent.	Neg. asp. sent.	Neu. asp. sent.	Conf. asp. sent
Raters 1 and 2	**Cohen's kappa**	0.857	0.323	-0.200	-0.400
	Agreement between annotators and the labeling	0.859	0.810	0.000	0.000
	Disagreement between annotators and the labeling	0.109	0.048	0.556	0.429
	Agreement between annotators	0.968	0.858	0.556	0.429
Raters 1 and 3	**Cohen's kappa**	0.623	0.417	-0.174	0.588
	Agreement between annotators and the labeling	0.844	0.714	0.000	0.143
	Disagreement between annotators and the labeling	0.078	0.095	0.667	0.714
	Agreement between annotators	0.922	0.809	0.667	0.857
Raters 2 and 3	**Cohen's kappa**	0.525	0.314	0.182	-0.235
	Agreement between annotators and the labeling	0.813	0.667	0.111	0.000
	Disagreement between annotators and the labeling	0.078	0.095	0.556	0.571
	Agreement between annotators	0.891	0.762	0.667	0.571

Table 6.7: Results of a detailed analysis of the agreement between the annotator pairs for the labeling of sentiments in the new dataset in the first validation phase.

pairs and that they mostly agreed with the annotation when we are talking about the positive and negative sentiments present in the dataset sample. However, the same cannot be said for the neutral and conflict labels. Not only is the agreement between annotators for these classes lower, but also those are the classes where they consistently disagree the most with the proposed annotation.

Overall, we can conclude that during this phase of validation, there were some aspects where the raters and the dataset annotations were in agreement: the identification of aspects in utterances (when the annotated dataset marks that there is one in an utterance) and the positive and negative sentiments associated with aspects are the areas where there is more certainty among the annotators. On the other hand, in the instances of the dataset that are labeled as lacking the presence of aspects or in the cases where the annotated sentiment associated with an aspect is either neutral or conflict causes greater disagreement between the group annotators and also between the annotators and the annotations of the dataset.

6.1.2 Second Validation Results

The second validation phase was more elaborated. We used the feedback obtained during the first validation phase to fine-tune the process for the second. We updated the *README* sheet within all *excel* documents handed to all the volunteers participating in this validation phase, so that it present a more clear explanation of concepts and we also added concrete examples of application of the aforementioned concepts, hoping to clear as many doubts as the annotators could potentially get. We also increased the number validation groups. While in the first phase we only had one group of three raters, we now have three different groups with three different raters each. The volunteers performing the validation in this phase are different from the ones who performed this task in the previous phase. We also increased the number of conversations each rater had to classify to 30. Each group had a different set of conversations randomly selected from the dataset after we excluded the 10 conversations used in the previous phase.

	Dataset sample description
Validation stage	2nd
Number of groups in validation stage	3
Validation group #	1st
Number of raters per group	3
Number of conversations	30
Number of utterances	768
Number of aspects	320
Positive sentiment aspects	212
Negative sentiment aspects	53
Neutral sentiment aspects	23
Conflict sentiment aspects	32

Table 6.8: Overall description of the dataset sample gathered for the first group of raters from the second validation phase.

Upon observing tables 6.9, 6.17 and 6.24 we can see that both the Fleiss' kappa and the agreement between the annotators in these groups and the dataset annotations are already higher than the values collected during the first validation phase.

Validation phase 2 | Group 1 | All raters	**Aspect**	**Sentiment**
Fleiss' kappa	0.585	0.712
Agreement between annotators and the labeling	0.885	0.806
Disagreement between annotators and the labeling	0.035	0.092
Agreement between annotators	0.920	0.898

Table 6.9: Results of analysis of agreement between all annotators from group 1 for the labeling of the new dataset in the second validation phase.

The values presented in tables 6.10, 6.18 and 6.25 also lead us to believe that the annotators seem to be more in agreement with each other and the annotation when it comes to the identification of aspects (see *non-empty asp.* column, particularly, the rows about *fleiss' kappa*, *the agreement between the annotators and the dataset labeling* and the overall *agreement between the annotators*). However, when we look at annotations in the dataset where an utterance is marked as lacking aspects, the group 2 seems to agree less.

As for the sentiments annotated in the dataset samples, according to the values observed in 6.11, 6.19 and 6.26, it seems that group 2 was more in agreement with their members

Validation phase 1 | Group 1 | All raters	Non-empty asp.	Empty asp.
Fleiss' kappa	0.630	0.530
Agreement between annotators and the labeling	0.863	0.902
Disagreement between annotators and the labeling	0.047	0.027
Agreement between annotators	0.909	0.929

Table 6.10: Results of a detailed analysis of the agreement between all annotators from group 1 for the labeling of aspects in the new dataset in the second validation phase.

Validation phase 2 Group 1 All raters	Pos. asp. sent.	Neg. asp. sent.	Neu. asp. sent.	Conf. asp. sent
Fleiss' kappa	0.787	0.618	0.321	0.385
Agreement between annotators and the labeling	0.816	0.698	0.130	0.062
Disagreement between annotators and the labeling	0.108	0.113	0.391	0.656
Agreement between annotators	0.925	0.811	0.522	0.719

Table 6.11: Results of a detailed analysis of the agreement between all annotators of group 1 for the labeling of sentiments in the new dataset in the second validation phase.

and the sentiment annotations in the dataset. Group 1 members as a whole seems not to be in agreement with each other, and both neutral and conflict class sentiments seem to generate agreement between the annotators when in which they argue that the dataset annotations for these two classes could be incorrect. Group 3's results resemble group 2's, and they are quite stable. They even seem to agree with the instances where the we are studying the neutral and conflict sentiments.

Validation phase: 2 Group 1		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Asp.	Sent.	Asp.	Sent.	Asp.	Sent.	Asp.	Sent.
Annotator pairs	1# and #2	0.685	0.741	0.923	0.845	0.042	0.098	0.965	0.943
	1# and #3	0.535	0.667	0.892	0.815	0.043	0.103	0.935	0.918
	2# and #3	0.557	0.731	0.900	0.831	0.042	0.105	0.941	0.936

Table 6.12: Results of analysis of agreement between the different annotator pairings of group 1 in the second validation phase.

As far as the pairs of annotators in each group are concerned (see tables 6.12, 6.20 and 6.27), we can see that they are also reasonably in agreement with each other and even with the proposed dataset annotations, especially if we consider the results observed in the previous validation phase.

Studying further the aspect field in the pairs of each validation groups (the results are present in tables 6.14, 6.21, 6.28), we can also see that the annotators seem to be in agreement with each other and agree with the suggested dataset annotations. The less

Validation phase: 2 Group 1		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.
Annotator pairs	#1 and #2	0.702	0.653	0.891	0.946	0.063	0.027	0.954	0.973
	#1 and #3	0.595	0.470	0.872	0.906	0.059	0.031	0.931	0.937
	#2 and #3	0.596	0.514	0.878	0.915	0.056	0.031	0.934	0.946

Table 6.14: Results of a detailed analysis of the agreement between all annotator pairs in group 1 for the labeling of aspects in the new dataset in the second validation phase.

homogeneous annotator in this large pool of people would be the rater #3 from group 2. However, the disagreement between him and the other members of his group is limited to the situations where utterances are annotated as they did not have an aspect. While others seem to agree with the empty aspect label, this annotator disagrees.

Validation phase 1 Group 1		Pos. asp. sent.	Neg. asp. sent.	Neu. asp. sent.	Conf. asp. sent
Raters 1 and 2	Cohen's kappa	0.778	0.638	0.240	0.349
	Agreement between annotators and the labeling	0.840	0.755	0.174	0.094
	Disagreement between annotators and the labeling	0.108	0.132	0.478	0.688
	Agreement between annotators	0.948	0.887	0.652	0.782
Raters 1 and 3	Cohen's kappa	0.698	0.694	0.374	0.464
	Agreement between annotators and the labeling	0.816	0.698	0.261	0.125
	Disagreement between annotators and the labeling	0.108	0.189	0.435	0.688
	Agreement between annotators	0.924	0.887	0.696	0.813
Raters 2 and 3	Cohen's kappa	0.896	0.517	0.359	0.355
	Agreement between annotators and the labeling	0.858	0.736	0.217	0.063
	Disagreement between annotators and the labeling	0.118	0.113	0.478	0.781
	Agreement between annotators	0.976	0.849	0.695	0.844

Table 6.15: Results of a detailed analysis of the agreement between the annotator pairs in group 1 for the labeling of sentiments in the new dataset in the second validation phase.

In tables 6.15, 6.22 and 6.29, we can once again see that annotators have a greater agreement when they are analyzing positive and negative sentiments associated with aspects. In all three groups, it is more common to see the annotators disagree with the neutral and conflict sentiments annotated in the dataset. However they often agree among themselves with that assessment, so as future work we should consider investing more time in finding a method to label the sentiments that are getting criticized. It should be noted that rater #1 in group 2, seems to have diverging opinions when it comes to the conflict sentiment, when compared with his colleagues, annotators #2 and #3.

To summarize the whole dataset evaluation process, it was important to perform two validation phases, with the first one being a small test, in order to improve the methodology and to provide better support to the annotators that volunteered for the next phase. Overall about the dataset, some of the information we were able to gleam from the data

	Dataset sample description
Validation stage	2nd
Number of groups in validation stage	3
Validation group #	2nd
Number of raters per group	3
Number of conversations	30
Number of utterances	782
Number of aspects	323
Positive sentiment aspects	194
Negative sentiment aspects	61
Neutral sentiment aspects	28
Conflict sentiment aspects	40

Table 6.16: Overall description of the dataset sample gathered for the second group of raters from the second validation phase.

Validation phase 2 | Group 2 | All raters	**Aspect**	**Sentiment**
Fleiss' kappa	0.264	0.459
Agreement between annotators and the labeling	0.720	0.581
Disagreement between annotators and the labeling	0.035	0.121
Agreement between annotators	0.755	0.702

Table 6.17: Results of analysis of agreement between all annotators from group 2 for the labeling of the new dataset in the second validation phase.

Validation phase 2 | Group 2 | All raters	**Non-empty asp.**	**Empty asp.**
Fleiss' kappa	0.576	0.146
Agreement between annotators and the labeling	0.892	0.599
Disagreement between annotators and the labeling	0.031	0.037
Agreement between annotators	0.923	0.636

Table 6.18: Results of a detailed analysis of the agreement between all annotators from group 2 for the labeling of aspects in the new dataset in the second validation phase.

Validation phase 2 Group 2 All raters	**Pos. asp. sent.**	**Neg. asp. sent.**	**Neu. asp. sent.**	**Conf. asp. sent**
Fleiss' kappa	0.550	0.807	0.586	0.399
Agreement between annotators and the labeling	0.902	0.918	0.857	0.825
Disagreement between annotators and the labeling	0.026	0.049	0.036	0.025
Agreement between annotators	0.928	0.967	0.893	0.850

Table 6.19: Results of a detailed analysis of the agreement between all annotators of group 2 for the labeling of sentiments in the new dataset in the second validation phase.

collected was:

- Sometimes annotators will argue that the dataset has some missing aspects which are not identified as such. This may stem from the fact that the method used to build the annotation for this dataset relied on the labels of the original dataset, and because

Validation phase: 2 Group 2		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Asp.	Sent.	Asp.	Sent.	Asp.	Sent.	Asp.	Sent.
Annotator pairs	1# and #2	0.555	0.674	0.904	0.776	0.040	0.129	0.944	0.905
	1# and #3	0.185	0.366	0.726	0.590	0.043	0.145	0.770	0.734
	2# and #3	0.269	0.435	0.742	0.610	0.054	0.155	0.796	0.765

Table 6.20: Results of analysis of agreement between different annotator pairings for the second group in the second validation phase.

Validation phase: 2 Group 2		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.
Annotator pairs	#1 and #2	0.552	0.556	0.916	0.895	0.034	0.044	0.950	0.939
	#1 and #3	0.492	0.102	0.904	0.601	0.034	0.050	0.938	0.651
	#2 and #3	0.673	0.171	0.907	0.625	0.050	0.057	0.957	0.682

Table 6.21: Results of a detailed analysis of the agreement between all annotator pairs in group 2 for the labeling of aspects in the new dataset in the second validation phase.

Validation phase 1 Group 2		Pos. asp. sent.	Neg. asp. sent.	Neu. asp. sent.	Conf. asp. sent
Raters 1 and 2	Cohen's kappa	0.757	0.843	0.381	-0.129
	Agreement between annotators and the labeling	0.762	0.672	0.215	0.050
	Disagreement between annotators and the labeling	0.160	0.262	0.500	0.500
	Agreement between annotators	0.922	0.934	0.715	0.550
Raters 1 and 3	Cohen's kappa	0.767	0.608	0.345	-0.016
	Agreement between annotators and the labeling	0.753	0.557	0.179	0.025
	Disagreement between annotators and the labeling	0.170	0.262	0.536	0.650
	Agreement between annotators	0923	0.819	0.715	0.675
Raters 2 and 3	Cohen's kappa	0.821	0.747	0.227	0.297
	Agreement between annotators and the labeling	0.747	0.607	0.179	0.075
	Disagreement between annotators and the labeling	0.191	0.279	0.464	0.700
	Agreement between annotators	0.938	0.886	0.643	0.775

Table 6.22: Results of a detailed analysis of the agreement between the annotator pairs in group 2 for the labeling of sentiments in the new dataset in the second validation phase.

those components weren't highlighted in that dataset, they did not carry over to our own (previously, the dataset only had annotated movie or television genres, names of people or locations and movie or television series names). If we were talking,

Dataset sample description	
Validation stage	2nd
Number of groups in validation stage	3
Validation group #	3rd
Number of raters per group	3
Number of conversations	30
Number of utterances	732
Number of aspects	327
Positive sentiment aspects	195
Negative sentiment aspects	59
Neutral sentiment aspects	35
Conflict sentiment aspects	38

Table 6.23: Overall description of the dataset sample gathered for the third group of raters from the second validation phase.

Validation phase 2 | Group 3 | All raters	**Aspect**	**Sentiment**
Fleiss' kappa	0.461	0.672
Agreement between annotators and the labeling	0.878	0.760
Disagreement between annotators and the labeling	0.022	0.102
Agreement between annotators	0.900	0.862

Table 6.24: Results of analysis of agreement between all annotators from group 3 for the labeling of the new dataset in the second validation phase.

Validation phase 2 | Group 3 | All raters	**Non-empty asp.**	**Empty asp.**
Fleiss' kappa	0.469	0.447
Agreement between annotators and the labeling	0.847	0.904
Disagreement between annotators and the labeling	0.034	0.012
Agreement between annotators	0.881	0.916

Table 6.25: Results of a detailed analysis of the agreement between all annotators from group 3 for the labeling of aspects in the new dataset in the second validation phase.

Validation phase 2 Group 3 All raters	**Pos. asp. sent.**	**Neg. asp. sent.**	**Neu. asp. sent.**	**Conf. asp. sent**
Fleiss' kappa	0.571	0.479	0.153	0.396
Agreement between annotators and the labeling	0.882	0.814	0.743	0.816
Disagreement between annotators and the labeling	0.036	0.051	0.000	0.026
Agreement between annotators	0.918	0.864	0.743	0.842

Table 6.26: Results of a detailed analysis of the agreement between all annotators of group 3 for the labeling of sentiments in the new dataset in the second validation phase.

for example of how much we liked the plot of a certain movie, the movie would be identified and would have a positive sentiment associated to it. However, the plot, which is one of the movies components, the direct target wouldn't be picked up by our annotation approach.

- Annotators seem to have mixed opinions about aspects labeled with the neutral senti-

Validation phase: 2 Group 3		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Asp.	Sent.	Asp.	Sent.	Asp.	Sent.	Asp.	Sent.
Annotator pairs	1# and #2	0.636	0.775	0.914	0.801	0.042	0.137	0.956	0.937
	1# and #3	0.346	0.634	0.898	0.784	0.025	0.115	0.922	0.899
	2# and #3	0.405	0.607	0.891	0.772	0.031	0.116	0.922	0.888

Table 6.27: Results of analysis of agreement between different annotator pairings for the third group in the second validation phase.

Validation phase: 2 Group 3		Cohen's kappa		Agreement between annotators and labeling		Disagreement between annotators and labeling		Agreement between annotators	
		Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.	Non-empty asp.	Empty asp.
Annotator pairs	#1 and #2	0.574	0.683	0.914	0.913	0.037	0.047	0.951	0.960
	#1 and #3	0.403	0.258	0.862	0.926	0.040	0.012	0.902	0.938
	#2 and #3	0.470	0.314	0.859	0.916	0.049	0.017	0.908	0.933

Table 6.28: Results of a detailed analysis of the agreement between all annotator pairs in group 3 for the labeling of aspects in the new dataset in the second validation phase.

Validation phase 1 Group 3		Pos. asp. sent.	Neg. asp. sent.	Neu. asp. sent.	Conf. asp. sent
Raters 1 and 2	Cohen's kappa	0.874	0.848	0.302	0.652
	Agreement between annotators and the labeling	0.810	0.763	0.257	0.184
	Disagreement between annotators and the labeling	0.154	0.186	0.400	0.684
	Agreement between annotators	0.964	0.949	0.657	0.868
Raters 1 and 3	Cohen's kappa	0.626	0.559	0.364	0.345
	Agreement between annotators and the labeling	0.759	0.661	0.286	0.053
	Disagreement between annotators and the labeling	0.128	0.095	0.400	0.789
	Agreement between annotators	0.884	0.756	0.686	0.842
Raters 2 and 3	Cohen's kappa	0.646	0.436	0.471	0.240
	Agreement between annotators and the labeling	0.759	0.627	0.286	0.053
	Disagreement between annotators and the labeling	0.133	0.153	0.457	0.711
	Agreement between annotators	0.892	0.780	0.743	0.764

Table 6.29: Results of a detailed analysis of the agreement between the annotator pairs in group 3 for the labeling of sentiments in the new dataset in the second validation phase.

ment. Sometimes they agree with each other and the annotation assigned. However, it is more common for them to collectively decide that they do not agree with the annotated sentiment that was assigned to the aspect.

- The conflict sentiment seems to be the most complicated one to analyze. Some of the results above can be justified by perhaps the inadequacy of the conditions to which we decided that would trigger the annotation of an aspect as associated with the conflict sentiment (which causes some sentiments to be labeled as conflict when they would not be otherwise interpreted as such). Some of those cases should come from there, but others can be because when in a utterance we express both positive and negative opinions about something, the annotators will decide on the polarity based on the more intense feeling they capture from the utterance, instead of simply calling assigning to an aspect a conflict sentiment.

6.2 Approaches' Evaluation Results

We performed the evaluation of the performances of the approaches in our newly-created dataset. We tested the approaches while training with the laptops and restaurants domains for all the approaches. For aspect extraction tasks, we used precision, recall and f1-score as performance evaluation metrics. For aspect sentiment classification we also used precision, recall and f1-score. However we decided to measure these metrics for each individual sentiment category (*positive*, *negative*, *neutral* or *conflict*).

In this section we present the results obtained from the testing of the selected approaches on the new dataset. The results presented here use the same values for the parameters as the papers where these approaches were published. However, there were some small changes that were made for the approaches. The training data for all the approaches were in some cases reduced to only include the *Semeval 2014* Laptops and Restaurants datasets. The combination of parameters that were used in these approaches and presented in the papers were named as *Baseline*. The metrics collected were precision, recall and f1-score and we used both *Levenshtein distance* < 2 and *Cosine similarity* $\geq 75\%$ as aspect validation functions.

The first validation function was the original function used in the *Semeval* conference to evaluate whether an aspect had been correctly predicted. The second aspect validation function was selected because we thought that it was excluding relevant aspects due to possible error in delimiting the position of the aspect in an utterance. For example, if in a utterance we talked about a ″Die Hard movie", the aspect would be *Die Hard*, but it would not be incorrect to consider *Die Hard movie* a valid aspect as well. The threshold for the similarity was obtained by experimenting with different values. Upon collecting some examples with different thresholds we decided that 75% was a reasonable threshold that was not to low and not to complicated to match aspects with.

The most important questions or aspects that we hoped to be able to answer or address once we performed tests on each of the selected approach's models were:

1. How would we describe the performance of the different approaches' models on our created dataset?
2. Which of the selected approaches had the best performance? And, if possible to determine, which training dataset yielded the most favorable results?
3. Should the model performances be lackluster for both implementations, find plausible justification for the results obtained.

After both approaches were tested, we added the false positives commonly obtained from the studied approaches to the dataset. This idea had merit because these approaches can catch what even we could consider as aspects but were not captured by the annotation process during the creation of the dataset.

6.2.1 RINANTE Evaluation Results

The first approach selected, RINANTE, was a neural network that also used a group of mined sentence rules to extract potential aspects. The datasets used to train the models whose results we obtained were the *Semeval 2014 restaurants* and *laptops* datasets. Each of these datasets were used to train 20 models so that we could could collect our desired

metrics: precision, recall and f1-score metrics (each of these metrics ranging from 0 to 1 in the result tables presented below). The necessity of training multiple models on the same data stems from the stochastic nature of the training process of the models. A model trained multiple times with either one of these datasets displayed different results even if the validation dataset used to assess the model had been the same.

The parameters selected for this approach were the same ones as presented in the paper (Dai and Song, 2019). We used the configuration in this paper as the staring point to evaluate the RINANTE approach performance on our dataset.

The questions we hoped to address with the tests performed on this approach were:

- Which dataset would allow for better results when evaluating the performance of a model on our created dataset?
- What possible reasons could be named to answer for the performance of the models shown?

The results of the tests were presented on table 6.30. The results obtained from training models using either the restaurant or laptops datasets were rather poor. Using the *Levenshtein distance* < 2 metric, the precision, recall and f1-score values were around 2%, 0.7% and 2%, respectively for models trained with the two different datasets. Even for a less strict aspect validation function such as *Cosine similarity* $\geq 75\%$ only ended showing results in the order of 4% for precision, 1.5% for recall and 2% for f1-score. While the results obtained with the different models were very close, the laptops dataset trained model showed a slight edge over the restaurant dataset. The difference was of a very marginal magnitude (less than a 0.5% gain all metrics, using both evaluation functions).

The lackluster results were not unexpected. RINANTE was an approach created to solve the problem of aspect extraction on review texts for laptop and restaurant domains. On the other hand, we decided to test it on a dialogue dataset. This dialogue dataset also had a different domain (movie and television series). Not only were the topics of the training and evaluation datasets unrelated, but the actual discourse used in them were very different. In a review, all the relevant aspect information was explicitly present in its text. Each entry in the *Semeval 2014* datasets was the full text of a review and all the aspects were present in the text. In our dataset, because it was a dialogue and each entry of the dataset was a dialogue utterance (which could have also had been as short as a *yes* or *no* reply to a query about whether a speaker liked something or not or a noun to indicate what was viewed favorably), sometimes the target of the opinion might was not explicitly mentioned in the utterance where the preference was stated. for example, in one of the entries of the dataset, the different parties were discussing the horror movie genre and one of them asks the other what movie of that genre they liked. The reply was simply the name of the movie. The utterance where we had the information about the target was different than the one where the target is actually called upon. This resulted in many cases where the dataset annotators considered that an aspect was present in a utterance and the model was not able to pick up on it, as he did not have knowledge of prior exchange of information, which would have necessarily been present in a review text.

To summarize, the approach is designed to work on review texts and its effectiveness is deeply affected by the domain and even more so in our case, the type of discourse used in the dataset. A dialogue was simply very different from a review.

One potential avenue to exploit in an attempt to improve performance would be to perform some manner of post-processing that would allow the system to acknowledge that there is

a specified entity target in a utterance and that very close to that utterance, a preference is expressed about it. This can be considered a task to explore for future work.

Baseline - RINANTE **epochs = 170** **batch size = 64** **learning rate = 0.001**		**Precision** **Levenshtein<2** **Cos. sim≥75%**	**Recall** **Levenshtein<2** **Cos. sim≥75%**	**F1-score** **Levenshtein<2** **Cos. sim≥75%**
Train: laptops	**Mean**	0.022 0.047	0.007 0.015	0.011 0.023
	Variance	3.056×10^{-5} 1.249×10^{-4}	8.484×10^{-6} 4.020×10^{-5}	1.511×10^{-5} 7.035×10^{-5}
	Standard deviation	0.006 0.011	0.003 0.006	0.004 0.008
Train: restaurants	**Mean**	0.020 0.044	0.007 0.015	0.010 0.022
	Variance	1.893×10^{-4} 1.030×10^{-4}	5.642×10^{-6} 2.872×10^{-5}	1.028×10^{-5} 5.301×10^{-5}
	Standard deviation	0.004 0.010	0.002 0.005	0.003 0.007

Table 6.30: Results of precision, recall and f-score metrics for RINANTE approach with baseline parameters and training using the Semeval 2014 datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions

6.2.2 DOER Evaluation Results

Similarly to RINANTE, due to the stochastic nature of the DOER approach, we performed training multiple times using the same datasets (once again, 20 times for each dataset) and evaluated the performance of each model on our dataset. Differently from RINANTE Dai and Song, 2019, the DOER implementation presented in (Luo et al., 2019) was constructed based on the paper (Xu et al., 2018), using two types of word embeddings: a general domain embedding that is common for both *laptop* and *restaurant* trained models and a domain-specific. The domain specific embedding depended on the training dataset used. For the training of a model with the *Semeval laptops 2014* dataset, the domain specific embedding used was constructed using Amazon laptop reviews. For the restaurant domain, the embeddings were built with *Yelp* restaurant reviews. The main difference between this approach and RINANTE is that RINANTE relies on the mined rules to know when to expect an aspect whereas DOER relies on the embeddings.

Similarly to RINANTE, the performances of the DOER models trained with the *laptops* and *restaurants* datasets were poor. Table 6.31 contained the results of the metrics used for performance assessment of the DOER models trained with these datasets and tested on our developed dataset. Here the results are substantially better compared to the results in 6.30. Precision, recall and f1-score values for both evaluation functions are around twice the values obtained with the RINANTE approach. Using the *cosine similarity* function, the precision reaches 7.9%, recall 2.2% and f1-score 3.7%. While these values are still not great, we can

As for the polarity evaluation results they weren't great either. The system does somewhat a decent job at predicting positive and negative sentiments, achieving around 70% precision

Baseline - DOER **epochs = 50** **batch size = 16** **learning rate = 0.001**		**Precision** **Levenshtein<2** **Cos. sim≥75%**	**Recall** **Levenshtein<2** **Cos. sim≥75%**	**F1-score** **Levenshtein<2** **Cos. sim≥75%**
Train: laptops	**Mean**	0.040 0.063	0.013 0.021	0.020 0.031
	Variance	1.233×10^{-4} 2.0168×10^{-4}	2.047×10^{-5} 3.572×10^{-5}	3.942×10^{-5} 6.527×10^{-5}
	Standard deviation	0.011 0.014	0.005 0.006	0.006 0.008
Train: restaurants	**Mean**	0.079 0.113	0.016 0.022	0.026 0.037
	Variance	1.6177×10^{-4} 1.3373×10^{-4}	2.277×10^{-5} 3.324×10^{-5}	5.138×10^{-5} 7.219×10^{-4}
	Standard deviation	0.013 0.012	0.005 0.006	0.007 0.008

Table 6.31: Results of precision, recall and f-score metrics for DOER approach with baseline parameters and training using the Semeval 2014 datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions

for positive aspects and 60% for negative aspects. However, it is very bad at detecting neutral or conflict aspects, such that the precision for these two sentiments is 0%. This is not really the fault of the model, but it is more of a problem of lack of enough neutral and sentiment polarity examples.

6.3 Approach Improvement Results

Due to time constraints and the complicated nature of the approaches we selected to adapt, it was decided that we would attempt to improve the performance of the RINANTE approach by adjusting some hyperparameters. The hyperparameters selected were: the number of epochs, the batch size and the learning rate. Each of these test configurations presented in the tables below were repeated 5 times. In this section we will proceed to discuss our results and share our conclusions.

6.3.1 RINANTE Improvement Results

Varying the hyperparameters for this approach did not improve its performance in a substancial way. Most likely for the same reason as the approach presents lackluster results even with the *baseline* configuration. Which further cements the point that this approach, as it is, is simply not made to be used for dialogue datasets. RINANTE needs some form of auxiliary mechanism to handle a different type of discourse. As stated in 6.2.1, some post-processing could allow for an improvement on the current results but otherwise, the way the approach works is not ideal for our problem.

In our dataset, we do not have all the necessary information in a utterance being analyzed by the model. The dialogue in the dataset had often situations where information was

Table 6.32: Results of precision, recall and f-score metrics for sentiment for DOER approach with baseline parameters and training using the Semeval 2014 datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions

Baseline - DOER **epochs =) 50** **batch size =) 16** **learning rate =) 0.001**		**Precision** **Levenshtein2** **Cos. sim≥)75%**				**Recall** **Levenshtein2** **Cos. sim≥)75%**				**F1-score** **Levenshtein2** **Cos. sim≥)75%**			
		Pos	**Neg**	**Neu**	**Conf**	**Pos**	**Neg**	**Neu**	**Conf**	**Pos**	**Neg**	**Neu**	**Conf**
Train: laptops	**Mean**	0.835	0.159	0.173	0.000	0.338	0.143	0.336	0.000	0.473	0.128	0.222	0.000
		0.587	0.950	0.158	0.050	0.328	0.212	0.320	0.007	0.414	0.336	0.206	0.013
	Variance	0.007	0.025	0.009	0.000	0.012	0.025	0.048	0.000	0.013	0.013	0.015	0.000
		0.013	0.050	0.009	0.050	0.011	0.014	0.034	0.001	0.0108	0.024	0.014	0.003
	Standard deviation	0.083	0.159	0.094	0.000	0.109	0.157	0.218	0.000	0116	0.111	0.123	0.000
		0.113	0.224	0.097	0.224	0.106	0.117	0.185	0.032	0.104	0.156	0.117	0.056
Train: restaurants	**Mean**	0.773	0.529	0.000	0.000	0.687	0.470	0.000	0.000	0.723	0.460	0.000	0.000
		0.771	1.000	0.000	0.067	0.711	0.448	0.000	0.004	0.738	0.613	0.000	0.007
	Variance	0.000	0.075	0.000	0.000	0.012	0.103	0.000	0.000	0.005	0.074	0.000	0.000
		0.001	0.000	0.000	0.067	0.009	0.010	0.000	0.000	0.004	0.010	0.000	0.000
	Standard deviation	0.020	0.274	0.000	0.000	0.112	0.320	0.000	0.000	0.070	0.273	0.000	0.000
		0.036	0.000	0.000	0.258	0.093	0.100	0.000	0.014	0.061	0.100	0.000	0.027

scattered in question and answer exchanges and cannot be found as easily as it would in a review format text. Take this query as an example: one speaker asks: *"What was your favorite comedy movie this year?"*. The next utterance in a dataset will be a reply with the name of the favorite movie of the person being questioned and nothing else but its name. By itself, this approach would not be able to understand from just that one reply that we are talking about an aspect. On the other hand, if the dataset entry was something like *"My favorite movie was Taxi 4"*, the approach would have caught the aspect because of its straightforward structure where the target and other relevant text marks that usually accompany an aspect, such as *"favorite"* which relates to a *"movie"* that is called *"Taxi 4"*. Since the way the information reaches the model is not shaped in the format that the approach is prepared to handle, it will sadly fail to catch many aspects. If we had paired question and answer utterances and reshaped their content into one single utterance with all the key aspects, then the system would be capable of catching them.

"Yes" or *"no"* are also some of the entries that constitute an entire utterance in a dataset. Once again, on its own the utterance fails to supply the model with the necessary indications that a preference is being expressed.

6.3.2 DOER Improvement Results

Similarly to RINANTE, the variation of the learning hyperparameters did not result in a visible difference in increase of the performance of the models. The data that this approach tested is simply not what this model was made to handle and that implies that further work needs to be done on a more complex level (pre-processing or post-processing), but simply changing parameters will not yield any substantial results. This too was also to be expected as the purpose of usage of this model was the same as RINANTE. Which means that in order to increase the performance of this model we had better invest time in contemplating how to get utterances from the dataset to make sense for the system.

RINANTE - Laptops		Precision Levenshtein<2 Cos. sim≥75%	Recall Levenshtein<2 Cos. sim≥75%	F1-score Levenshtein<2 Cos. sim≥75%
epochs: 140 batch size = 32 learning rate = 0.0008	Mean	0.019 0.046	0.007 0.017	0.010 0.025
	Variance	1.098e-5 6.608e-5	1.771e-6 1.584e-5	3.116e-6 2.737e-5
	Standard deviation	0.003 0.008	0.001 0.004	0.002 0.005
epochs: 140 batch size = 64 learning rate = 0.0008	Mean	0.017 0.035	0.004 0.009	0.007 0.014
	Variance	9.476e-6 9.252e-5	6.106e-7 5.693e-6	1.507e-6 1.447e-5
	Standard deviation	0.003 0.010	0.001 0.002	0.001 0.004
epochs: 140 batch size = 128 learning rate = 0.0008	Mean	0.010 0.019	0.001 0.003	0.002 0.004
	Variance	3.895e-6 3.020e-5	1.174e-7 8.361e-7	3.457e-7 2.484e-6
	Standard deviation	0.002 0.005	0.000 0.001	0.001 0.002

Table 6.33: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=140, learn rate=0.0008 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

6.4 BERT Models Validation Results

As previously mentioned in section 5.5, we performed the evaluation of each model on its respective test domain dataset (that we had previously modified to be understood by BERT). We then collected precision, recall and f1-score metrics for each one of the 5 possible categories in the dataset. We then also calculated the accuracy, macro average and weighted average of each model. Each model was trained 5 times and its results are displayed in tables 6.42 and 6.43 as an average of the values obtained in each evaluation.

In table 6.42 we can see that the model has quite a high precision for the label *0* (there is no aspect) and the same can be said for the recall and f1-score of that class. For class *1* (there is a positive aspect in the utterance), the precision, recall and f-scores are also fairly high, between 85% to 89%. The *negative* (*class 2*) label had a lower precision, but still presented a high recall. Neutral's (*class 3*) precision is quite high, but the recall metric suffered greatly. As for the *conflict* aspect class, the precision, recall and f1-score are 0, meaning that the model did not manage to correctly pinpoint any conflict aspects. Although, that is not completely unexpected as the number of occurrences of conflict aspects in the *Semeval 2014 laptops* test dataset was only 16, compared to the 1032 existing entries in the dataset. While the weighted average shows us that our system performs well if we take into account the proportion of class distributions in the dataset, if we discard that notion, then our metrics only reach values around 60%. In order to increase this value we would need to add more examples to the training dataset that would include more of the less frequent classes.

RINANTE - Laptops		Precision Levenshtein2 Cos. sim≥)75%	Recall Levenshtein2 Cos. sim≥)75%	F1-score Levenshtein2 Cos. sim≥)75%
epochs: 140 batch size = 32 learning rate = 0.001	**Mean**	0.021 0.045	0.007 0.015	0.010 0.022
	Variance	2.694e-5 1.348e-4	2.386e-6 8.924e-6	5.179e-6 2.121e-5
	Standard deviation	0.005 0.012	0.002 0.003	0.002 0.005
epochs: 140 batch size = 64 learning rate = 0.001	**Mean**	0.021 0.047	0.006 0.013	0.009 0.020
	Variance	7.133e-6 8.017e-5	2.349e-6 1.173e-5	64.113e-4 2.220e-5
	Standard deviation	0.003 0.009	0.002 0.003	0.002 0.005
epochs: 140 batch size = 128 learning rate = 0.001	**Mean**	0.015 0.023	0.002 0.003	0.004 0.006
	Variance	1.253e-5 1.867e-5	5.167e-5 9.535e-7	1.465e-6 2.655e-6
	Standard deviation	0.004 0.004	0.001 0.001	0.001 0.002

Table 6.34: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=140, learn rate=0.001 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

Similarly, in table 6.43 we can see that the model also has quite a high precision, recall and f1-score for the labels *0* (there is no aspect) and *1* (there is a positive aspect in the utterance). The negative class also had a more balanced performance in their metrics for the training with the restaurant datasets (precision, recall and f1-scores rounded at around 70%). However, just like in 6.42, the precision for the neutral class is higher than the respective recall and f1-score. Once again, the conflict class had a bad performance, with precision, recall and f1-score of that class being 0. The weighted average tells us that the model performed well taking into account the number of instances of each class that exist in the dataset. However, ignoring the class distribution ratio, the macro average of the collected metrics is quite similar to 6.42.

Overall, the results were quite good, considering that we had a small training sample and it is understandable that people are turning to these models for NLP tasks, as they are easy to use and show reasonable results for not very large training times.

RINANTE - Laptops		**Precision** **Levenshtein2** **Cos. sim≥)75%**	**Recall** **Levenshtein2** **Cos. sim≥)75%**	**F1-score** **Levenshtein2** **Cos. sim≥)75%**
epochs: 140 **batch size = 32** **learning rate = 0.0012**	**Mean**	0.021 0.044	0.01 0.021	0.013 0.028
	Variance	5.341e-5 1.313e-4	5.050e-6 2.756e-5	1.059e-5 3.879e-5
	Standard deviation	0.007 0.011	0.002 0.005	0.003 0.006
epochs: 140 **batch size = 64** **learning rate = 0.0012**	**Mean**	0.02 0.044	0.007 0.015	0.010 0.022
	Variance	3.665e-5 1.402e-4	9.005e-6 3.726e-5	1.700e-5 6.939e-5
	Standard deviation	0.006 0.012	0.003 0.006	0.004 0.008
epochs: 140 **batch size = 128** **learning rate = 0.0012**	**Mean**	0.020 0.034	0.004 0.006	0.006 0.011
	Variance	1.745e-6 2.372e-5	1.738e-7 3.654e-6	2.954e-7 8.508e-6
	Standard deviation	0.001 0.005	0.0 0.002	0.001 0.003

Table 6.35: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=140, learn rate=0.0012 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

RINANTE - Laptops		Precision Levenshtein2 Cos. sim≥)75%	Recall Levenshtein2 Cos. sim≥)75%	F1-score Levenshtein2 Cos. sim≥)75%
epochs: 170 **batch size = 32** **learning rate = 0.0008**	**Mean**	0.022 0.049	0.010 0.023	0.014 0.031
	Variance	1.735e-5 8.854e-5	1.236e-5 7.169e-5	1.588e-5 9.248e-5
	Standard deviation	0.004 0.009	0.004 0.008	0.004 0.010
epochs: 170 **batch size = 64** **learning rate = 0.0008**	**Mean**	0.022 0.049	0.007 0.015	0.010 0.023
	Variance	1.442e-5 1.0371e-4	2.757e-6 1.915e-5	5.340e-6 3.792e-5
	Standard deviation	0.004 0.010	0.002 0.004	0.002 0.006
epochs: 170 **batch size = 128** **learning rate = 0.0008**	**Mean**	0.018 0.029	0.003 0.005	0.005 0.008
	Variance	2.944e-6 7.747e-6	1.973e-7 5.496e-7	5.078e-7 1.415e-6
	Standard deviation	0.002 0.003	0.000 0.001	0.001 0.001

Table 6.36: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=170, learn rate=0.0008 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

RINANTE - Laptops		Precision Levenshtein2 Cos. sim≥)75%	Recall Levenshtein2 Cos. sim≥)75%	F1-score Levenshtein2 Cos. sim≥)75%
epochs: 170 **batch size = 32** **learning rate = 0.001**	**Mean**	0.026 0.060	0.011 0.025	0.015 0.035
	Variance	1.956e-5 9.255e-5	2.973e-6 1.241e-5	6.043e-6 2.628e-5
	Standard deviation	0.004 0.010	0.002 0.004	0.002 0.005
epochs: 170 **batch size = 128** **learning rate = 0.001**	**Mean**	0.021 0.038	0.004 0.008	0.007 0.013
	Variance	1.442e-5 1.0371e-4	2.757e-6 1.915e-5	5.340e-6 3.792e-5
	Standard deviation	0.002 0.006	0.000 0.002	0.001 0.003

Table 6.37: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=170, learn rate=0.001 and batch size=[32, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

RINANTE - Laptops		**Precision** **Levenshtein2** **Cos. sim≥)75%**	**Recall** **Levenshtein2** **Cos. sim≥)75%**	**F1-score** **Levenshtein2** **Cos. sim≥)75%**
epochs: 170 **batch size = 32** **learning rate = 0.0012**	**Mean**	0.025 0.051	0.011 0.023	0.015 0.032
	Variance	1.735e-5 8.854e-5	1.236e-5 7.169e-5	1.588e-5 9.248e-5
	Standard deviation	0.008 0.011	0.002 0.004	0.003 0.004
epochs: 170 **batch size = 64** **learning rate = 0.0012**	**Mean**	0.024 0.046	0.009 0.017	0.013 0.024
	Variance	1.442e-5 1.0371e-4	2.757e-6 1.915e-5	5.340e-6 3.792e-5
	Standard deviation	0.008 0.014	0.003 0.007	0.005 0.009
epochs: 170 **batch size = 128** **learning rate = 0.0012**	**Mean**	0.021 0.038	0.004 0.008	0.007 0.013
	Variance	2.944e-6 7.747e-6	1.973e-7 5.496e-7	5.078e-7 1.415e-6
	Standard deviation	0.002 0.006	0.000 0.002	0.001 0.003

Table 6.38: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=170, learn rate=0.0012 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

RINANTE - Laptops		Precision Levenshtein2 Cos. sim≥)75%	Recall Levenshtein2 Cos. sim≥)75%	F1-score Levenshtein2 Cos. sim≥)75%
epochs: 200 batch size = 32 learning rate = 0.0008	Mean	0.023 0.053	0.011 0.027	0.015 0.035
	Variance	2.141e-5 2.139e-4	1.326e-5 1.290e-4	1.786e-5 1.761e-4
	Standard deviation	0.005 0.015	0.004 0.011	0.004 0.013
epochs: 200 batch size = 64 learning rate = 0.0008	Mean	0.019 0.046	0.007 0.017	0.010 0.024
	Variance	2.853e-6 8.377e-5	2.598e-6 3.480e-5	3.857e-6 5.865e-5
	Standard deviation	0.002 0.009	0.002 0.006	0.002 0.008
epochs: 200 batch size = 128 learning rate = 0.0008	Mean	0.009 0.031	0.006 0.006	0.008 0.010
	Variance	2.087e-5 2.404e-5	9.958e-7 1.376e-6	2.686e-6 3.613e-6
	Standard deviation	0.005 0.005	0.001 0.001	0.002 0.002

Table 6.39: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=200, learn rate=0.0008 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

RINANTE - Laptops		**Precision** **Levenshtein2** **Cos. sim≥)75%**	**Recall** **Levenshtein2** **Cos. sim≥)75%**	**F1-score** **Levenshtein2** **Cos. sim≥)75%**
epochs: 200 **batch size = 32** **learning rate = 0.001**	**Mean**	0.022 0.048	0.012 0.026	0.015 0.033
	Variance	2.599e-5 9.452e-5	2.991e-5 1.254e-4	3.289e-5 1.329e-4
	Standard deviation	0.005 0.010	0.005 0.011	0.006 0.012
epochs: 200 **batch size = 64** **learning rate = 0.001**	**Mean**	0.018 0.038	0.006 0.013	0.009 0.019
	Variance	7.100e-6 2.987e-5	3.251e-6 1.159e-5	5.010e-6 1.792e-5
	Standard deviation	0.003 0.005	0.002 0.003	0.002 0.004
epochs: 200 **batch size = 128** **learning rate = 0.001**	**Mean**	0.018 0.034	0.004 0.007	0.006 0.012
	Variance	2.654e-6 1.539e-5	6.482e-7 3.960e-6	1.271e-6 8.077e-6
	Standard deviation	0.002 0.004	0.001 0.002	0.001 0.003

Table 6.40: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=200, learn rate=0.001 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

RINANTE - Laptops		Precision Levenshtein2 Cos. sim≥)75%	Recall Levenshtein2 Cos. sim≥)75%	F1-score Levenshtein2 Cos. sim≥)75%
epochs: 200 batch size = 32 learning rate = 0.0012	Mean	0.027 0.055	0.014 0.028	0.018 0.037
	Variance	1.168e-5 5.039e-5	9.897e-6 5.564e-5	9.416e-6 5.280e-5
	Standard deviation	0.003 0.007	0.003 0.007	0.003 0.007
epochs: 200 batch size = 64 learning rate = 0.0012	Mean	0.025 0.050	0.012 0.024	0.016 0.032
	Variance	1.415e-5 7.811e-5	1.808e-5 7.874e-5	1.721e-5 7.939e-5
	Standard deviation	0.004 0.009	0.004 0.009	0.004 0.009
epochs: 200 batch size = 128 learning rate = 0.0012	Mean	0.022 0.046	0.006 0.012	0.009 0.019
	Variance	1.015e-5 7.579e-5	1.630e-6 6.938e-6	3.486e-6 1.616e-5
	Standard deviation	0.003 0.009	0.001 0.003	0.002 0.004

Table 6.41: Results of precision, recall and f1-score metrics for RINANTE approach with epoch=200, learn rate=0.0012 and batch size=[32, 64, 128] parameters with training using the Semeval 2014 laptops datasets for Levenshtein distance<2 and cosine similarity≥75% as the evaluation functions.

BERT: Training with laptops dataset (avg. of 5 models)				
Labels	**Precision**	**Recall**	**F1-score**	**Label frequency**
0 - Not an aspect	0.999	0.998	0.999	378
1 - Positive	0.847	0.884	0.865	341
2 - Negative	0.540	0.886	0.670	128
3 - Neutral	0.766	0.388	0.507	169
4 - Conflict	0.000	0.000	0.000	16
Total				1032
accuracy	-	-	0.831	-
macro avg.	0.630	0.631	0.608	-
weighted avg.	0.838	0.831	0.818	-

Table 6.42: Results of aspect and polarity detection using a BERT model trained with the Semeval 2014 Laptops dataset.

BERT: Training with restaurants dataset (avg. of 5 models)				
Labels	**Precision**	**Recall**	**F1-score**	**Label frequency**
0 - Not an aspect	0.999	0.999	0.999	194
1 - Positive	0.849	0.962	0.902	728
2 - Negative	0.723	0.761	0.741	196
3 - Neutral	0.682	0.355	0.464	196
4 - Conflict	0.000	0.000	0.000	14
Total				1328
accuracy	-	-	0.838	-
macro avg.	0.650	0.615	0.621	-
weighted avg.	0.819	0.838	0.818	-

Table 6.43: Results of aspect and polarity using a BERT model trained with the Semeval 2014 Restaurants dataset.

Chapter 7

Conclusion

Currently, the importance of customer service cannot be ignored. In the business branch, it is common knowledge that it is more expensive to acquire new customers (due to advertising) than to retain customers. It is very important to ensure that customers are satisfied with the customer service provided to them. If not, with the current competitive market, they can just leave to another company that provides better services. Contact centers are where the communication between a company and its client occurs, therefore it is essential that the customer service provided there be excellent in order to maintain customer satisfaction and retain customers.

For Talkdesk, one way of ensuring that the communication between agents and customers is as fluid as possible came through the creation of a module to help the agents (*Agent Assist*). The primary goal of this internship is to develop an opinion mining framework that would allow us to extract the opinions of product features expressed by reviewers. That framework will then be used in the *Agent Assist* module.

During the first semester, we started by studying the currently existing state-of-the-art approaches for performing opinion mining. We also researched technologies (such as libraries) and other resources (like datasets and lexicons) that could prove to be useful for the implementation of the framework during the second semester. We also made a competitor analysis to see if Talkdesk's competitors were offering similar products. We also wrote down the requirements for the opinion mining framework that we will later implement on the second semester. Once that was concluded, we defined our approach based on the constraints we had. Which turned out to be a linguistic approach, as we do not have enough labeled data to make a valid attempt at a machine learning approach.

In the second semester, we changed our internship objectives and we decided to create a dialogue opinion mining dataset. Once created and its validation performed, we took note of some important aspects that could constitute as future work. Overall, the dataset annotation was mostly agreed on by the annotators. We also tested some state-of-the art approaches on the new dataset, and the results were, as expected, lackluster because they weren't developed for use in dialogue datasets. We also attempted to improve the results of the approaches by testing different hyper-parameters, but the results weren't improved. We also used the common false positives extracted from both the approaches studied and decided to add them as aspects to the dataset. Finally, we concluded by attempting to build a aspect-sentiment classification BERT model, by fine-tuning them with the *Semeval 2014* datasets. The results were very good for the detection of not aspects, of positive sentiment aspects and negative sentiment aspects. The neutral aspects were also detected reasonably, although not comparable to the two sentiments before.

To conclude, we elaborate on some points that could be further studied be labeled as future work: first, about the created dataset. The results of the dataset validation point out that there are some problems with the conflict label that is assigned to aspects in the dataset. One point to improve on would be to find a different method or criteria to assign the conflict label to an aspect. Another thing that could be improved on at a later date would be to attempt to improve the performance of the approaches studied by attempting to do some post-processing.

Bibliography

(n.d.). https://stanfordnlp.github.io/stanfordnlp/

Aesuli. (n.d.). Aesuli/sentiwordnet. https://github.com/aesuli/sentiwordnet

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: Analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003a). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003b). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 440–447. https://www.aclweb.org/anthology/P07-1056

Call center contact center software features: Bright pattern. (2020). https://www.brightpattern.com/call-center-software-features/

Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). New avenues in opinion mining and sentiment analysis. *IEEE Intelligent systems*, *28*(2), 15–21.

Chapelle, O., Bernhard, S., & Zien, A. (2010). *Semi-supervised learning.* MIT Press.

Cloud natural language nbsp;|nbsp; google cloud. (n.d.). https://cloud.google.com/natural-language/

Coached conversational preference elicitation. (n.d.). https://research.google/tools/datasets/coached-conversational-preference-elicitation/

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37–46.

Creative commons legal code. (n.d.). https://creativecommons.org/licenses/by-sa/4.0/legalcode

Dai, H., & Song, Y. (2019). Neural aspect and opinion term extraction with mined rules as weak supervision. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5268–5277. https://doi.org/10.18653/v1/P19-1520

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *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)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Ding, X., Liu, B., & Yu, P. S. (2008). A holistic lexicon-based approach to opinion mining. *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 231–240. https://doi.org/10.1145/1341531.1341561

Eisenstein, J. (2019). *Introduction to natural language processing.* The MIT Press.

Explorer. (n.d.). https://www.gavagai.io/products/explorer/

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, *76*(5), 378.

Gamon, M., Aue, A., Corston-Oliver, S., & Ringger, E. (2005). Pulse: Mining customer opinions from free text. *international symposium on intelligent data analysis*, 121–132.

Ganapathibhotla, M., & Liu, B. (2008). Mining opinions in comparative sentences. *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, 241–248. http://dl.acm.org/citation.cfm?id=1599081.1599112

Gensim. (n.d.). https://pypi.org/project/gensim/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Google colaboratory. (n.d.). https://colab.research.google.com/github/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb

Hatzivassiloglou, V., & McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, 174–181.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. *Proceedings of the 14th conference on Computational linguistics-Volume 2*, 539–545.

Hindle, D., & Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, *19*(1), 103–120. https://www.aclweb.org/anthology/J93-1005

Hitachi, L. (n.d.). Sentiment analysis service : Hitachi enterprise application service. https://www.hitachi.com/products/it/appsvdiv/service/sentiment-analysis/index.html

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 168–177.

Irsoy, O., & Cardie, C. (2014). Opinion mining with deep recurrent neural networks. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 720–728.

Jakob, N., & Gurevych, I. (2010). Extracting opinion targets in a single-and cross-domain setting with conditional random fields. *Proceedings of the 2010 conference on empirical methods in natural language processing*, 1035–1045.

Jurafsky, D., & Martin, J. H. (2019). *Speech and language processing* [Unpublished Manuscript]. https://web.stanford.edu/~jurafsky/slp3/

Krippendorff, K. (2011). Computing krippendorff's alpha-reliability.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning*, 282–289.

License and commercial use of wordnet | wordnet. (n.d.). https://wordnet.princeton.edu/license-and-commercial-use

Liu, B. (2012). *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers.

Liu, B., Hsu, W., Ma, Y., et al. (1998). Integrating classification and association rule mining. *KDD*, *98*, 80–86.

Liu, Q., Gao, Z., Liu, B., & Zhang, Y. (2015). Automated rule selection for aspect extraction in opinion mining. *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Luo, H., Li, T., Liu, B., & Zhang, J. (2019). DOER: dual cross-shared RNN for aspect term-polarity co-extraction. *CoRR*, *abs/1906.01794*. http://arxiv.org/abs/1906.01794

Maas, A. (n.d.). Large movie review dataset. https://ai.stanford.edu/~amaas/data/sentiment/

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. http://www.aclweb.org/anthology/P11-1015

Marsland, S. (2014). *Machine learning: An algorithmic perspective, second edition* (2nd). Chapman & Hall/CRC.

McAuley, J. (n.d.). Amazon product data. http://jmcauley.ucsd.edu/data/amazon/

McHugh, M. (2012). Interrater reliability: The kappa statistic. *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, *22*, 276–82. https://doi.org/10.11613/BM.2012.031

Mitchell, T. M. (1997). *Machine learning* (1st ed.). McGraw-Hill, Inc.

Natural language toolkit. (n.d.). https://www.nltk.org/

Nigam, K., & Hurst, M. (2004). Towards a robust metric of opinion. *AAAI spring symposium on exploring attitude and affect in text*, *598603*.

Nisbet, R., Elder, J., & Miner, G. (2009). *Handbook of statistical analysis and data mining applications*. Academic Press, Inc.

Opinion: Meaning in the cambridge english dictionary. (n.d.). https://dictionary.cambridge.org/dictionary/english/opinion

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. *Proceedings of LREC*, *10*.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 79–86.

Pattern. (n.d.). https://pypi.org/project/Pattern/

Polyglot. (n.d.). https://pypi.org/project/polyglot/

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., et al. (2016). Semeval-2016 task 5: Aspect based sentiment analysis. *10th International Workshop on Semantic Evaluation (SemEval 2016)*.

Popescu, A.-M., & Etzioni, O. (2005). Extracting product features and opinions from reviews. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 339–346. https://www.aclweb.org/anthology/H05-1043

Qiu, G., Liu, B., Bu, J., & Chen, C. (2011). Opinion word expansion and target extraction through double propagation. *Computational linguistics*, *37*(1), 9–27.

Rabiner, L. R., & Juang, B. H. (1986). An introduction to hidden markov models. *IEEE ASSp Magazine*.

Radlinski, F., Balog, K., Byrne, B., & Krishnamoorthi, K. (2019). Coached conversational preference elicitation: A case study in understanding movie preferences. *Proceedings of the Annual SIGdial Meeting on Discourse and Dialogue*.

Richards, J. C., & Schmidt, R. R. (2013). Longman dictionary of language teaching and applied linguistics.

Russo, A. J. (2003). Pt. https://aws.amazon.com/pt/comprehend/

Saeidi, M., Bouchard, G., Liakata, M., & Riedel, S. (2016). SentiHood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 1546–1556. https://www.aclweb.org/anthology/C16-1146

Semeval-2016 task 5. (n.d.). http://alt.qcri.org/semeval2016/task5/

Sentiment analysis: Sentiment analysis customer service. (n.d.). https://www.ameyo.com/fusion-cx/features/sentiment-analysis

Simplified text processing. (n.d.). https://textblob.readthedocs.io/en/dev/

Spacy · industrial-strength natural language processing in python. (n.d.). https://spacy.io/

Sun, C., Huang, L., & Qiu, X. (2019). Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Hu-*

man Language Technologies, Volume 1 (Long and Short Papers), 380–385. https://doi.org/10.18653/v1/N19-1035

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. A Bradford Book.

Sysadmin@isobarbudapest.com. (2020). Textanalytics. https://xdroid.com/solutions/text-analytics/

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational linguistics*, *37*(2), 267–307.

Text analytics. (n.d.). https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://arxiv.org/pdf/1706.03762.pdf

Wang, W., Pan, S. J., Dahlmeier, D., & Xiao, X. (2016). Recursive neural conditional random fields for aspect-based sentiment analysis. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 616–626. https://doi.org/10.18653/v1/D16-1059

Watson natural language understanding - overview. (n.d.). https://www.ibm.com/cloud/watson-natural-language-understanding

What is wordnet? (n.d.). https://wordnet.princeton.edu/

Wiebe, J. M., Bruce, R. F., & O'Hara, T. P. (1999). Development and use of a gold-standard data set for subjectivity classifications. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 246–253. https://doi.org/10.3115/1034678.1034721

Xu, H., Liu, B., Shu, L., & Yu, P. S. (2018). Double embeddings and CNN-based sequence labeling for aspect extraction. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 592–598. https://doi.org/10.18653/v1/P18-2094

Yrsnlp. (n.d.). https://coling2016.anlp.jp/

Zhuang, L., Jing, F., & Zhu, X.-Y. (2006). Movie review mining and summarization. *Proceedings of the 15th ACM international conference on Information and knowledge management*, 43–50.