UNIVERSITY OF PADOVA

DEPARTMENT OF MANAGEMENT AND ENGINEERING DTG

MASTER'S THESIS IN MANAGEMENT ENGINEERING

APPLICATION OF NATURAL LANGUAGE PROCESSING TECHNIQUES IN A BUSINESS CONTEXT FOR INSIGHT EXTRACTION

SUPERVISOR
PROF.SSA MARTA DISEGNA

MASTER CANDIDATE
THAYSA FERNANDA SILVESTRIN

CO-SUPERVISOR
NICOLÒ BIASETTON

STUDENT ID
2043195

Academic Year
2023-2024

Abstract

In the current scenario, businesses are dealing with increasing levels of volatility and uncertainty. It highlights the necessity for identifying oncoming change and adapting to leverage an advantage. Few technologies are as powerful or as versatile to do so as Machine Learning and its associated domains. This work aims to evaluate the current playing field and the NLP methodologies businesses apply to gather insights through Sentiment Analysis. After comprehensive review of current practices for both business and industrial sectors, data composed of customer reviews on a selected product was extracted from the web. Three models widely recognized for their suitability to NLP tasks were trained on four varying datasets derived from the initial extraction, Support Vector Machine, Random Forest and Naive Bayes, testing both the hypothesis of techniques like stemming or not, and the effectiveness of using either customer reviews or their titles for the task. The result confirmed the use of review titles for better classification results, in both stemmed and non-stemmed datasets.

Keywords: Machine Learning; Natural Language Processing; Customer Reviews; Sentiment Analysis.

Sommario

Nel contesto attuale, le imprese stanno affrontando crescenti livelli di volatilità e incertezza. Ciò sottolinea la necessità di individuare i cambiamenti imminenti e adattarsi per sfruttare un vantaggio. Poche tecnologie sono tanto potenti e versatili quanto il Machine Learning e i suoi ambiti correlati. Questo lavoro si propone di valutare lo stato attuale del campo di gioco e le metodologie di NLP (Natural Language Processing) che le imprese applicano per raccogliere informazioni tramite l'Analisi del Sentimento. Dopo una revisione completa delle pratiche attuali sia nel settore commerciale che in quello industriale, sono stati estratti dati dalle recensioni dei clienti su un prodotto selezionato dal web. Tre modelli ampiamente riconosciuti per la loro idoneità alle attività di NLP, Support Vector Machine, Random Forest e Naive Bayes, sono stati addestrati su quattro set di dati diversi derivati dall'estrazione iniziale. Sono stati testati sia l'ipotesi di utilizzare tecniche come lo stemming o meno, sia l'efficacia di utilizzare recensioni dei clienti o i loro titoli per l'attività. I risultati hanno confermato l'uso dei titoli delle recensioni per ottenere migliori risultati nella classificazione, sia nei dataset con stemming che senza stemming.

Keywords: Machine Learning; Natural Language Processing; Customer Reviews; Sentiment Analysis.

Contents

LIST OF FIGURES

LIST OF TABLES

1

Introduction

With the current scenario of high volatility, risk and uncertainty, the continued investment in innovation becomes a survival mechanism for modern businesses (Obradovic, 2016). At its core, it represents their ability of noticing oncoming change and leveraging it to get ahead of competitors (Obradovic, 2016). Few other technologies have as much to contribute to this end today as Machine Learning and its associated fields, both for prediction and for gaining an advantage. Likewise, few other technologies are as versatile: the applications range from business intelligence, to project development and management, from marketing, to production and maintenance, to briefly mention a few.

Traditionally the process of data analysis has been conducted through the use of numerical data, instead of focusing on the synthesis of knowledge from text - especially considering the industrial sector and the fast advancement of its digitization (May et al, 2022). Only more recently has Machine Learning found a strong development in the Business sector, a growth mainly related to the maturity achieved on Natural Language Processing (NLP) technologies and the availability of big data (Van der Aa, 2022). Due to its extensive use of documentation and textual data (Van der Aa, 2022), the sector allows for a more accessible implementation of the technology than others.

This more recent growth and maturity of NLP technologies can be traced to three deciding factors: “the application of innovative or rarely exploited machine learning algorithms, the availability of datasets and the availability of appropriate infrastructures to store data and run computationally intensive algorithms on large datasets” (Quarteroni, 2021, p. 6). Meaning that now, more so than in times past, businesses have easy access to both the technology – be it tangible or intangible – and the necessary data to power it.

The ramifications are all-encompassing: from the possibility to ease communication , not only towards their clients, but also towards their entire supply chain while surpassing even language barriers (Bahja, 2021) to investing in the collection and prediction of public opinion

on their products and services to gather insights (Cambria, 2016). The former brings implications to the enhancement of trade relations and global commerce (Bahja, 2021), while the latter, the focus of this research, lends itself to the evolution of an era in which the future of products and services is defined by the opinion and needs of the communities who use them (Cambria, 2016).

On a wider scale, the collection of user's opinions, mainly through social networks, provides information about their context and individual behavior (Algefes et al, 2022). Today, all of us have the possibility of expressing opinions on "different subjects related to politics, education, travel, culture, commercial products, or subjects of general interest" (Goularas & Kamis, 2019, p. 9), meaning that the knowledge available for extraction is of great significance and can then be combined and used to "effectively predict, forecast, and infer outcomes of real-world events" (Algefes et al, 2022, p. 2). The author still mentions that the reach of such social media representation, for example, goes beyond 3.6 billion active users, and this number seems to be growing over time - not reducing.

Going even further, Alpaydin (2016) points out the symbiotic relationship between consumers generating data, while also consuming it. Given that everytime they buy a product, access a website, post on social media or simply use online maps to get around, they generate large amounts of data, consumers have come to expect products and services that understand and predict their needs and interests - that are specialized for them.

Among the technologies companies have at their disposal to parse through consumer data, Sentiment Analysis represents the collection of techniques that allows for sentiment retrieval from various sources, even surpassing the challenges of the more widely available unstructured data, like volume, velocity and variety (Vargas, 2017). These techniques are "numerous and are based on different methods of natural language processing and machine learning techniques for extracting adequate features and classifying text in appropriate polarity labels" (Goularas & Kamis, 2019, p. 9). Also called Opinion Mining, "studies usually follow a workflow consisting of two steps: identify (topics, opinionative sentences), and classify (sentences, documents)" (Tsytsarau & Palpanas, 2011, p. 4).

However, the quality of the outcome obtained through the use of Opinion Mining is essential for any following analysis, and can be quite challenging. The identification of a given textual opinion and its polarity evaluation are the basis for further research on the subjective themes contained in the text, for example, which are essential elements for deepening the insight acquired from the data (Tsytsarau & Palpanas, 2011).

In sum, the application of Machine Learning, along with NLP and associated technologies to understand public context and opinion, finds applications for both predicting change and preparing to take advantage of it. Especially in the business environment, which is the motivation for this work: gathering user generated data (UGD) on a product and parsing it for opinions and insights with NLP and Sentiment Analysis techniques. The product chosen needed to fit two main prerequisites: have user generated data available for the analysis – meaning a B2B product, and be of Italian manufacture.

The product chosen for analysis was the La Specialista Prestigio Espresso Machine, produced by the italian company De'Longhi, as if fit in both criteria. The data was gathered through the web scraping of websites such as Amazon, as it had the highest number of

reviews in English. For this purpose, the language becomes a deciding factor as the number of resources and libraries available vary a lot per language, being English the most common one and so the one with the larger amount of material – both for data gathering as for the tools to analyze it.

This work contributes to the ever growing literature on the applications of tools such as NLP and SA in the business environment, by providing the evaluation of a potential methodology for gathering and analyzing publicly available customer data using Supervised Machine Learning. The end goal is to validate the approach in regards to the extraction of useful information and insights through widely available tools and resources.

This means that, while this work does not claim to be an exhaustive overview of all models that can be applied for analysis, it should capture the most widely accepted ones for the particular task it aims to develop.

This research is divided into 6 sections, following a logical progression of arguments and steps, organized in the following manner: Section 2 provides an overview of the current applications of NLP in the business sector and a brief mention of recent developments on the industrial sector, Sections 3 to 5 provide further context to the opportunity tackled by this paper and a deeper review of the concepts therein. Lastly, Section 6 presents the case study and the conclusion reached.

1.1 Related Works and Current Research

This section aims to provide insight about the current state of the art in the fields of NLP and SA applications, for both research and business/industrial purposes.

1.1.1 Applications of NLP

Natural Language Processing is a subfield of artificial intelligence, focused on teaching computers how to understand human language – this means inferring contextual information and mining valuable insights from human written text (Global Risk Institute, 2022).

While NLP is definitely not a new field of research, with ELIZA simulating simple conversations (Bahja, 2021) and SMART allowing for the use of natural language to research a textual database (Quarteroni, 2018) already in the 1960s, it is still more commonplace inside non-industrial sectors than it is outside of them (May et al, 2022). This section provides a brief overview of the solutions in development for the use of companies, from the point of view of both business and industrial applications, with a greater focus on the former.

Starting with a brief outlook of industrial environments, although many of the procedures can be available in text form, the use of NLP “is not capable of end to end prediction based on textual knowledge but rather focused on very well described, unique tasks” (May et al, 2022, p. 186). Going further, there are so many possible influencing factors that cannot be foreseen in such environments, the result is oftentimes an increasing amount of unstructured, natural text data (Ziora, 2021; May et al, 2022), and “pre-trained models [...] cannot be used because the production vocabulary is different from everyday language”

(May et al, 2022, p. 186). In sum, training industrial models can be highly task-specific and in general needs to be developed from the ground up.

To avoid the high development and management costs of building a structured semantic context for a specific manufacturing field – such as an industry-specific ontology, Ziora (2021) proposes a textual similarity approach for the retrieval of troubleshooting options for new problems, based on past similar solutions found in technical assistance reports. Similarly, Single et al (2020) propose the use of NLP to build a database of chemical accidents – ontology-based – so that companies can consult to investigate possible abnormal operations and the effectiveness of the safety measures implemented.

If these issues can be solved, the use of NLP and associated methods can increase efficiency and save time for a wide range of manufacturing scenarios. This represents the possibility of long-term advantages within a company, and is the reason why May et al (2022), for example, propose a pipeline to gather, pre-process and use maintenance logs and written commentary on failures and downtimes in a manufacturing plant to predict machine downtime - an as of yet novel application on NLP and ML.

Another field starting to develop further regards human-machine communication and interaction - which is usually done through a software using input such as a keyboard entry or a touch screen command (Gui & Harth, 2021). The application of NLP could simplify the process by suppressing the need for professional knowledge of the system to interact with it, accepting natural language input as suggested in.

Now, regarding the business side of NLP applications, the field becomes a little wider. With extensive applications “in many branches such as Finance, Banking, Telecommunication, E-commerce, Medicine, or Healthcare, it is often implemented in the process of contemporary business organization management” (Ziora, 2021, p. 76). As highlighted by the Global Risk Institute (2022), with its increasing popularity, an ever increasing number of firms are making use of NLP to better understand their clients and the environment they are working in. Even further, they affirm that companies that do not choose to actively use NLP techniques to gain insights on today's widely available data will soon be at a great competitive disadvantage in the market.

As corporate applications tend to be more accessible, they are also more widespread. At the forefront of this movement is the easy availability of large sets of textual data, online and offline, inside and outside of the company. Zhecheva & Nenkov (2022, p. 108) explain that “corporate unstructured textual data could be found in emails and social media posts, but also in clients’ reviews, web pages, blogs, news, CRM systems files, survey responses, etc”. They also mention sources such as job performance evaluations, contracts and meeting summaries. All of these have the potential to provide insights for gaining competitive advantage.

The first recent application worth mentioning is the work developed by Vashisht & Dharia (2020), which combines a Business Intelligence tool with a chatbot, with the goal of providing a quick and intuitive manner or engaging with the company’s data. Using NLP, the firm’s managers can query the database and gather insights intuitively. Not only that, this interface completely bypasses the need for technical knowledge on how to use the BI tool, further widening its potential user base and accessibility.

The capacity to empower people when using NLP to analyze company data is further explained by Sergeeva et al (2022, p. 1), when they say that “while technology is the enabler of these improvements, the biggest change is Cultural – Every Authorized Business Person has access to the underlying data to perform their own analysis using their own tools”. On a wider lens, it has the capacity to remove the focus from specific Data Specialist resources and provide more autonomy to the numerous businesspeople in whichever domain of knowledge they specialize in – they can conduct their own analysis to make more informed decisions on their respective areas.

Still related to Business Intelligence, by applying NLP and word embedding algorithms, [23] proposes a system that provides firms a way of generating summary reports on the patenting landscape of their industry automatically. The goal is to provide enterprises with an in depth analysis of growing trends within emerging industries and the state-of-the-art technology within their market. Moreover, it eases the continuous patent research and evaluation during the lifecycle of product development. The insights can then be used to further base not only the overarching company strategy, but also R&D and Project Management, financial and patenting strategy and marketing campaigns.

On the financial side, the attempts to go beyond numerical data started many decades ago: “the earliest attempts to import other predictors employed discourse analysis techniques developed from linguistics and Naïve statistical methods such as word spotting” (Frazier et al. 1984; Brachman and Khabaza 1996, as cited by Xing et al, pg. 50).

Some recent NLP and text mining applications for finance regard the “information retrieval and the classification of financial statement content” (Global Risk Institute, 2022, p. 3). More specifically, tasks like comparison and verification of company reports against financial statements, insight extraction from press releases and new articles, fraud detection and even market prediction (Global Risk Institute, 2022). In the latter case, however, (Xing et al, pg. 50, pg. 57) points out the need for a more interdisciplinary approach, affirming that the task “is positioned at the intersection of linguistics, machine learning, and behavioral economics”.

Another application that has been gathering a considerable amount of interest in the last decade is the chatbot. This is highlighted by saying that “there can be observed growing popularity in its application in different business domains, especially in customer care in banking, financial, telecommunication and healthcare institutions” (Ziora, 2021, p. 78). Especially in the business perspective, the author mentions that chatbots can render companies accessible and available in the eyes of customers, independent of time and location, with high potential for scalability at low cost.

Thinking of Logistics and Supply Chain Management, Quarteroni (2018) provides one application for conformity checks on the documentation of raw goods received: NLP helps to identify entities such as organizations or locations mentioned in the text that may need to be further assessed through background checks before confirming a given transaction. This solution relies on a search-based application, in which the documentation is scanned and then evaluated through the use of a statistical named entity recognizer - the result of which brought the reduction of 80% of the time needed for a human to perform the task.

Still in this field, Blume Global leveraged NLP as explained by Ziora (2021), using it to query complex datasets to optimize their Supply Chain. In the same vein, the company applied Machine Learning and NLP techniques to perform in-depth analysis of social media, news and reports to gather an understanding and implement mitigating solutions to the risks they were subjected to.

Widely used in Marketing, the NLP methods are categorized based on which information is desired and the depth of analysis; “these are Concept and topic extraction, Relationship extraction, and Sentiment and writing style extraction” (Hartmann & Netzer, 2023, p. 206). The first, for example, refers to the identification of single words or themes in the text, and is exemplified by the framework proposed by Tirunillai & Tellis (2014), to understand the market through brand-mapping, the segmentation inside of it and the dynamics over time. The second contains the methods to find the link between words or entities within the text (Hartmann & Netzer, 2023), here represented by a ML and NLP alternative to discovering customer’s needs from User Generated Content proposed by Timoshenko & Hauser (2019).

In companies today, this is still usually done through customer interviews, which can be quite long and expensive. Finally, the last category refers to the classification of the sentiment or writing-style prevalent in the text, oftentimes lexicon-based (Hartmann & Netzer, 2023), and represents the category in which the present research can be found. To exemplify with a related publication, (Chakraborty et al, 2022, pg. 600) developed “a scalable text analysis method to convert open-ended text reviews from online review platforms to produce attribute-level summary ratings”, looking to capture both the valence of sentiment as well as its degree – in a scale of positive to negative.

1.1.2 Applications of SA

Regarded as a subfield of NLP, Sentiment Analysis can be considered a text classification task whose fundamental goal is to discover feelings and ideas from a given collection of written text – be it by a person or an organization (Thada & Shrivastava, 2020; Salmony & Faridi, 2021). This task, also called Opinion Mining, is traditionally meant to classify a given piece of text like a social media post or comment based on its polarity, positive, neutral or negative. (Ziora, 2021)

The main approach for this task is through the use of Machine Learning models, either with Supervised (such as linear classifiers and support vector machines) or Unsupervised Learning models. (Ziora, 2021)

Restricting the evaluation more specifically to the Business domain – where this work is placed – SA provides “in-depth insight into the attitude of buyers' feedback about their product; therefore, they can improve their strategy to meet the customer's expectations and needs and avoid loss.” (Salmony & Faridi, 2021, p. 132) But much beyond that, there are wide applications of SA for tasks such as brand and reputation monitoring, market research and competitor analysis, employee sentiment analysis, or trend analysis just to name a few.

Many of the works cited here are further detailed in section 1.1.3, as they are more closely related to the research developed in this work.

1.1.3 Related Works

In this section, the goal is to understand the recent developments in the NLP and SA fields that are more strictly related to the research undertaken by this work, meaning the sentiment classification of user generated content. Not only that, but the techniques and results are also briefly discussed.

When researching publications for this specific field on scholarly literature websites – such as the widely known and used Scopus – using combinations of the keywords "NLP", "SA" with "customer reviews", "user reviews" or "user generated content", there is a very clear upwards trend over time. From around 2010 onwards, the number of publications has only grown year over year - with more than 400 new works added just in 2022.

As seen previously, this is a direct result of factors such as the increase in data availability and the development of technologies able to support its analysis and use. Methods such as web scraping, in which researchers iteratively extract useful information from freely available web sites, are a common practice. For example, in the work of Waila et al (2012), paired with pre-existing databases, the authors scraped movie reviews websites such as IMDB to gather data and train sentiment classification models using both Supervised Machine Learning – Naive Bayes and SVM — and Unsupervised Semantic Orientation.

The movie review databases didn't go through stop word removal or stemming during pre-processing. For the Supervised ML models, the researchers found that NB had a slightly better performance when compared to SVM, even if the latter is typically regarded as superior. However, more than that, the Unsupervised ML model realized with POS tagging as feature extraction of adjectives found in the reviews had the best performance overall: above 80% Accuracy on all three datasets.

Another research that found some success with the Naive Bayes model was done by Rain (2012), this time in an evaluation of probabilistic ML models looking also into Decision List Classifiers. The goal here was to evaluate 15 products with the highest number of reviews, mostly books and movies, divided into 3 datasets. The author tackled the problem of the ambiguity (lack of distinction) of using a 1 to 5 stars rating system by setting a new system, in which reviews with 1 or 2 stars had a '0' score, while reviews with 5 stars got a score of 1. They offset the fact that the majority of products receive mostly good reviews, so those were more restricted.

All three of the datasets worked better with the Naive Bayes model, based on their Accuracy. However, it is interesting to notice that the worst performance for both algorithms was with the same dataset – which the author concludes happened because the data was not as specific as the others. Ultimately, it leads to the conclusion that training the model and using it on the same specific product or range of products will yield better results than trying to find a generic, widely encompassing model, and provide deeper information on specific features and properties. In sum, a model developed with data on book reviews should be used to classify new book review inputs, a model developed with electronic product reviews should be used to classify new such inputs and so on.

On a more focused look into Supervised ML models, Singla et al (2019) trained not only probabilistic classification models like Naive Bayes, but also deterministics ones like SVM

and Decision Trees. The authors have extracted mobile phone reviews from Amazon.com to form a big database, filtered and pre-processed to remove elements such as stop words, digits and special symbols: the result is over 400.000 reviews for 4.500 phones.

The only features selected for the training were Product Name, Brand and the reviews themselves, as these were the most relevant ones. After determining the sentiment orientation with a sentiment dictionary – including eight different emotions and their valence – the corresponding Positive or Negative polarity tag calculated was added to each review. The prepared data was imbalanced and thus corrected using an undersampling technique for the training of each model. The results were evaluated with the Accuracy measure of each one, and in contrast with research by Waila et al (2012) the widely regarded as superior SVM had the best performance – over 80% – against 65% by the Naive Bayes model, the worst out of all three.

This type of contradicting result may stem from numerous reasons, such as data variability between the experiments and the way in which the text and features are represented – Waila et al (2012) used TF-IDF with no stop word removal or stemming, while Singla et al (2019) performed those steps along with removing punctuation marks and digits. Even the domain can be taken into consideration, as movie reviews may have differences in language and context from phone reviews. These factors are better evaluated in future sections, but highlight the necessity of training more than one single model agreed to be the best one for the task, and compare their performance before selecting the most suited one for the task – and the dataset used.

Still in the Supervised ML area, Thada & Shrivastava (2020) have looked into Logistic Regression for Sentiment Classification. The research employed a publicly available dataset by Kaggle, Unlocked Mobile phone, which after cleaning and pre-processing resulted in just over 300.000 reviews. Before using a 80/20 training and testing data split, the data was converted into numeric representation through the Bag of Words approach – meaning that the position of the words is not a relevant factor, only their frequency in the document is considered.

The performance of the model was evaluated in terms of the Roc AUC curve, which reached a very high score of 0.927. Interestingly, some of the words with the highest contribution to positive sentiments were *excellent, loving and amazing*, while for negative they were *worst, worthless and horrible*.

The same research also looked into the TF-IDF approach, now using not only the frequency but also the weight of the words to a given document and later removing the ones with less importance. This method resulted in a Roc AUC curve of 0.927 – meaning there was no improvement from the BOW approach, save that it used considerably less features to reach the same score.

In an attempt to include the order of the words in the analysis along with their frequency, one last attempt was made: adding bi-grams. This means that the analysis is not conducted only in words such as “not” and “bad” independently, which could indicate negative connotations, but also “not bad”, which can have a more positive connotation. This addition returned a Roc AUC curve of 0.967, the best one. Now, among the terms most indicative of positive feelings were “*no issues”*, while for negative feelings were “*not happy” and “not satisfied”*.

One thing to keep in mind from the comparison of the three methods is that while n-grams can be quite useful for an increased model performance, they should be used carefully as the number of features can increase very quickly.

Aside from strictly business side applications, Kaur (2022) applied the concept of NLP and SA to analyze the trend in political news publication, given the fact that the manner in which media portrays everyday occurrences can shape people's decision-making and predefined notions. This task can be tricky, as news are not typically reported with objectively positive or negative language, the ideas are conveyed through more complex discourse. The author used the Vader Sentiment Analyzer to label the data as positive, neutral or negative at word-level, and TF-IDF for text representation.

Using a 25/75 train-test split, the author trained both Naive Bayes and Logistic Regression models, which obtained performance of 59% and 61% Accuracy, respectively. As these are too low for actual implementation and use, the author suggests improvements starting with POS tagging.

Trupthi et al (2016) considered only two classes, instead of the typical three: positive and negative. The idea here was to classify movie reviews with Naive Bayes and evaluate the impact of varying feature extraction techniques on the final model. Although the baseline model with a simple BOW representation reached an Accuracy of 73%, but Precision and Recall measures were not good: although the model predicted well True Positive values, there were too many False Positives – an occurrence explained by the fact that people tend to use positive words even in negative reviews, after the "not" negation ("not good" or "not great"). Another point to consider is that due to the fact that words could only be classified as positive or negative, without a neutral category, and stop words were not initially removed, they had a bigger impact than necessary on the output.

The second model was then trained two times, the first by removing stopwords and the second by adding meaningful bigrams. In doing so, they have seen the metrics reduced even further due to stopword removal, but increased enormously as the result of the bigrams. Their main takeaway is the fact that when the classification has too many features, many of them are low information and as such may decrease model performance. So removing these means removing noise that affects richer value features, allowing the latter more weight and minimizing the possibility of overfitting the model.

Another path for grasping public sentiment outside of the business field is developed in a study by Ng et al (2022), that evaluated over 300.000 tweets to understand how people saw the monkeypox virus outbreak. The goal was to inform public health decision makers on policy and awareness campaigns. Different from the works seen so far, the researchers started with topic modeling, using BERTopic to segregate the data into thematic groups and generate the main idea behind each one. The research provided very pertinent insights: three out of the five topics identified dealt with fear that monkeypox was to become the next worldwide pandemic, while the remaining two dealt with lack of faith in public institutions seen as inadequate.

Another possible approach to using NLP for SA applications is through the use of lexicons. Gupta et al (2021) calculate the polarity of sentences within tweets, which commonly contain

online slangs that are trickier to classify, using both lexicon-approach and machine learning classification techniques (SVM, Random Forest and Linear Regression).

The researchers started by initializing both Vader and a dictionary that considers over 300 slangs and their meanings as key-value pairs. Then, the data was processed to substitute all of the slangs found by their meaning, so the phrase "*The joke was funny lol*" becomes "*The joke was funny laugh out loud*". Next, the lexicon was used to calculate the polarity of each phrase, set as either positive, negative or neutral as a score between +4 and -4, subsequently normalized to fit within the interval of [-1, +1].

The scores were finally summed up to calculate the final polarity of each tweet, and later used as input for training and testing ML models. This approach outperformed previously existing works using only the Vader lexicon, achieving accuracy of over 90% for all three models trained while maintaining equally high levels of Precision and Recall. It shows the necessity of using both pertinent and updated lexicons in the correct context: tweets are typically filled with informal language such as slang that evolve quickly over time, and should be dealt with as such.

The issues that arose were mainly due to extended slang, such as "*looool*", emojis and images – the last two were initially removed or substituted by a numeric data type NaN. Most of these can be dealt with by expanding the lexicon and keeping it up to date.

As just seen, there is no one unique way of solving the NLP task of sentiment analysis, there can be many approaches more or less successful – each within their own parameters and limits. Many of the topics here presented are further discussed along this research to provide the basis of the project developed.

Finally, a summary of the works presented in this section is seen below, in Table 1. The columns contain information on the authors of each journal, the data used and the NLP techniques applied for each one. Lastly, the models trained and the metrics evaluated are presented. This summary makes it easy to find the common trends of the current research on sentiment analysis for customer generated data in general, such as using Accuracy to judge the quality of the models trained and the ubiquity of Naive Bayes as one of them.

#	Authors	Data gathering	NLP	Models trained	Metrics
60	P. Waila, Marisha, V.K. Singh and M.K. Singh	Pre existing data on movie reviews plus web scraping data	TF-IDF	Naive Bayes, SVM and Unsupervised Semantic Orientation	Accuracy, Precision, Recall and F-measure
48	Rain, C.	Product review from 15 products listed on Amazon	BOW, bigrams	Naive Bayes, Decision List Classifier	Accuracy per number of features
49	Zeenia Singla, Sukhchandan Randhawa, Sushma Jain	Product review from 4500 phones listed on Amazon		Naive Bayes, SVM, Decision Trees	Accuracy
51	Vikas Thada, Utpal Shrivastava, Ruchi	Unlocked Mobile phone by Kaggle	BOW, TF-IDF, bigrams	Logistic Regression	ROC_AUC curve
121	Kaur, P.	Web scraping of live political news data	TF-IDF and count-based methods	Naive Bayes and Logistic Regression	Accuracy, Precision, Recall and F1 score
100	Trupthi, M. Pabboju, S. Narasimha, G.	Movie reviews	BOW plus stop word removal and rich bigrams	Naive Bayes	Accuracy, Precision, Recall
122	Q.X. Ng, C.E. Yau, Y.L. Lim, L.K.T. Wong, T.M. Liew	Twitter posts	Topic modeling (BERTopic)	-	-
123	Gupta, S. (a) Bisht, S Gupta, S (b)	Twitter posts	Vader lexicon and slang lexicon	SVM, Random Forest and Linear Regression	Accuracy, Precision, Recall and F1 score

Table 1. Summary of Related Works.

2

Machine Learning

As a sub-branch of Artificial Intelligence (Mrabet et al, 2021), Machine Learning comprises all activities that try to teach a computer to imitate the way human beings learn, using pattern recognition. This is done in a data-based manner, rather than rule-based, meaning that instead of being given the logical rules to arrive at a conclusion, of having its algorithm programmed directly, the model is trained on available data and learns how to arrive at a conclusion by itself.

The evolution from one to another happened because the traditional inductive reasoning method used is no longer efficient - or even feasible - when done by people (Alpaydin, 2016). The volume of the data collected is too large and the manual analysis related to it is too costly, which means that computer models that can extract information automatically - or learn from it - have become essential. In this manner, Machine Learning “offers new tools to overcome challenges for which traditional statistical methods are not well-suited” (Jiang et al, 2020, p. 675).

Even further, the tasks to be solved have become more complex: comparing, for example, sorting simple numbers to sorting spam emails (Choudhary & Gianey, 2017). While the first has very clear inputs, outputs and the logical procedure to follow from one to the other, the latter requires instructions that are not always clear cut and easy to generalize. Machine learning, then, gives the model the ability to learn and improve with experience, without being given explicit instructions.

Machine Learning is also fundamentally about generalization (Choudhary & Gianey, 2017): as explained by Alpaydin (2016), the mathematical model built is not intended to fully identify the process from input to output, but to serve as a good enough approximation of it, taking into account patterns and regularities that occur in the data. Once ready, the model can be

descriptive and be used to gather insights on the process itself, or *predictive* to make predictions - assuming that the future will not be vastly different from the moment in which the data was gathered. The core goal, then, is to “instruct computers to use data or past experience to solve a given problem.” (Muhammad & Zhu, 2015, p. 1)

In the case of prediction, Machine Learning may even be preferable to traditional statistical tools (Jiang et al, 2020), as it has fewer restrictive statistical assumptions, such as linear relationships and absence of multicollinearity.

The field can be divided into four general categories, and the first step into the process is to choose the one most suited for the analysis to be done: Supervised learning, Unsupervised learning, Semi-Supervised learning and Reinforcement learning. (Kour & Gondhi, 2020; Jiang et al, 2020)

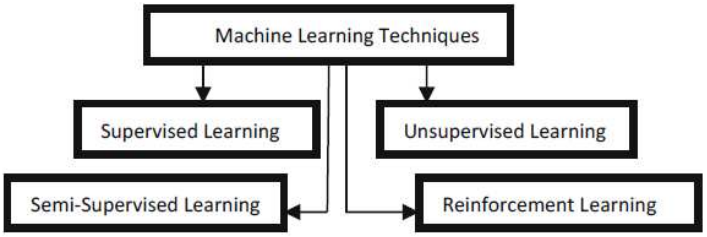


Figure 1. Classification of Machine Learning techniques by Kour & Gondhi, 2020.

2.1 Supervised Learning

The aim of Supervised Learning algorithms is to learn how to map the input into an output using values already corrected by a supervisor. This means that the model essentially receives a set of labeled examples to train and learn from - containing both the input and the output, and then tries to predict the correspondence to unseen points; the results are evaluated and adapted by a human until reaching an accurate enough prediction power using the loss function (Kour & Gondhi, 2020; Jiang et al, 2020). Here, perhaps even more than in other cases, the data used typically undergoes critical preparation and preprocessing (Muhammad & Zhu, 2015).

In general, the idea is to have a large enough labeled dataset so that it can be split into two: one for training and validating the model, and another for testing it. The split should be done randomly, and during training different error values can be achieved depending on how the division was made. It should be noted that when validating the model to find the best possible one, the data points used become part of the training phase and so cannot be then used for testing - in a way, they are now known to the model. (Alpaydin, 2016; Sen et al, 2019).

This subfield of ML has two main types of tasks: Classification, in which the goal is to predict a categorical output, so a discrete value called class label, and Regression, in contrast, which aims to predict a continuous output, a numerical value (Mrabet et al, 2017; Jiang et al, 2020). Some existing algorithms can be used for either task, such as Support Vector Machines and Neural Networks, while others are more specific to only one of them.

In both cases, there may exist a number of sub configurations that will influence the performance of the model trained, how well it will generalize the pattern learned, also called Hyperparameters (Jiang & Rosellini, 2020). These must be chosen and adjusted as needed, according to the model under training, to achieve a good fit.

2.1.1 Classification

Humans are exceedingly good at classification tasks, differentiating between one object and another based on shared features: the most common example given is recognizing dog breeds, as they all have the same sets of characteristics that vary slightly on appearance, color and size but at the end of the day they are all dogs (Faul, 2019). How a person learns how to differentiate them is based on the features of each one.

There may be different types of classification tasks, as explained by Sen et al (2019): there is the binary classification, in which there are only two possible classes for any given input - for example, if an email is spam or not or if a picture contains a dog or not. And there can be multi-label classification, in which three or more labels are possible for any given input - for instance, if the grade achieved by a student is above average, average or below average. Even further, the same input can belong to more than one single label, meaning the classification task is not so clear cut and the boundaries between the classes are quite close to each other.

The goal of the model is to find “a set of instances that share a common property” (Alpaydın, 2021, p. 57), also called a label or more commonly a class. So the task itself is to find a way to formulate the class using these shared features, to find a discriminant that represents the boundary that separates the inputs based on the space defined by their attributes (Alpaydın, 2021). This can mean classifying an email as spam or not, or a potential borrower as likely to default on the loan or not – and provide insights to make real world decisions.

For text classification, for instance, typically “the goal is to determine whether a given document belongs to the given category or not by looking at the words or terms of that category” (Kadhim, 2019, p. 1). The task can be summarized through the words of Muhammad & Zhu (2015), by saying that the classification process of a given dataset D is based on finding the model that best describes the way the values of attributes contained in the set size A [A_1,A_2… A_n] relate to the classes contained in the class label vector C [c_1,c_2… c_n], where $c_n \geq 2$.

There can be three types of attributes, according to Hájek & Barushka (2019): the first is called Boolean and can only assume one of two values, so typically questions answered as yes or no (answers to is “is it blue?” and “does it have hair?” and so on). The second type of feature is called categorical, either ordered or unordered, and can be one out of three or more values (will answer questions such as “what color is it?” and “how many stars does it rate on a scale of 1 to 5?”) – these usually require further handling before use through dummy variables [70, pg 59]. The last type of feature is called continuous, so real-value answers belong here (“what is its height?” and “how much rain fell during January?”). These three types represent characteristics that are shared by the group, the dataset under analysis, and can be used to set them apart into classes – not all features are equally

significant, some not at all. It depends on the researcher which ones should be kept and which ones should be left out of the model.

ML based text classification belongs to the Supervised ML field, meaning that the model "needs to be trained on some labeled training data before it can be applied to actual classification task" (Waila et al, 2012, p. 2). If the learning model is based on a statistical method, such as Naive Bayes, it is called a statistical text classifier – and the same goes for so called vector space based models such as k-Nearest Neighbors and Support Vector Machine algorithms, now representing the documents as high dimensional vectors.

Classification tasks can also be solved through Decision Tree induction models, which allow for an easy understanding of how the model defines the class for each input, and can be quite effective. These models are further discussed in subsequent sections, but for now suffice to know that they typically use labeled data for classification purposes.

For model performance evaluation, Mrabet et al (2017) provide an overview. Starting with Accuracy for binary classification seen in (1), a metric commonly used: it represents the ratio between the sum of the correct predictions over the total number of data points in the testing dataset – so how correct were the model's predictions overall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Given that:

TP represents the number of positive inputs correctly classified.

TN represents the number of negative inputs correctly classified.

FP represents the number of positive inputs classified incorrectly.

FN represents the number of negative inputs classified incorrectly.

However, this measure can be deceiving when dealing with an imbalanced dataset, not reflecting properly the efficiency of the model - datasets in which one class vastly outnumbers another. It can be quite common for real life dataset and is one of the challenges to face when performing classification tasks. In this case, the Accuracy metric can be paired with the model's Precision and Recall rates to have a deeper look on its performance over specific classes. These are widely used in the NLP field for tasks such as document classification and query classification performance, for example.

Choudhary & Gianey (2017, p. 4) point out further that the Accuracy metric lacks other specificities, such as "the relation between the data attributes, the measure of correct distribution of data instances to each and every class possible, the number of the positive outcomes from among all received positive outcomes, and several others".

Seen in (2) and (3), respectively, Precision represents the proportion of values predicted as positive that was actually correct, while Recall represents the proportion of positive samples that were predicted as such overall.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

This means that a model can be great at its given task overall, but predict better to one class or another. One consequence of this is that the model should then be tailored to fit the needs of the task. For example, if the goal is to use scans to predict if a tumor is benign or malignant before being directed to a licensed professional, a researcher may decide to develop a model that performs particularly well in detecting if they are malignant, to make sure that no patient goes on undiagnosed. This notion is important because oftentimes improving one damages the other, so prioritizing one over the other needs to be in accordance with the application.

If one were to try and use only Precision and Recall to evaluate one model against another for example, without looking at the bigger picture, the outcome can also be negative. Say for example, a bank builds models A (15% Precision, 95% Recall) and B (10% Precision, 85% Recall) for classifying borrowers that are likely to default on a loan or not, requiring an Accuracy of at least 90% to be used. Model A can be said to be the better choice, but cannot be used because it does not reach the Accuracy level desired.

One metric that is commonly added for this purpose is the F1 score, and it is based on the harmonic mean of both Precision and Recall – so as to find the best combination of both. This means that this metric evaluates the model through its predictions for each class, instead of simply the overall performance, and as such by maximizing F1, so are the other metrics maximized. It is calculated through (4) and ranges from 0 to 100% or 0 to 1.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

Next, Mrabet et al (2017) introduce the Confusion Matrix: it keeps track of True Positive and True Negative predictions - the ones that were correctly made, along with the False Positive and False Negative predictions - conversely, the ones that were wrongly made. These allow for a more careful evaluation, as seen previously, one can be more important than the other: if the goal is to use the model to detect possible mines, for instance, the ratio of True Positives would be more important and thus prioritized, as the loss for getting a single one wrong would be greater than classifying a True Negative or a False Positive.

Seen in Table 2, the Confusion Matrix is a direct summary of the model's performance, and can also be extended to multi-class classification tasks. In that case, the concepts of True Positives and True Negatives remain the same, as do False Positives and False Negatives; what changes is that these will now be extender for each class the model is predicting for, so the diagonals (where the given class column and row intersect) represent the True Positive value.

	Predicted class		
True class	Positive	Negative	Total
Positive	tp : true positive	fn : false negative	p
Negative	fp : false positive	tn : true negative	n
Total	p'	n'	N

Table 2. Confusion matrix representation by Alpaydin (2016, p. 561)

This level of analysis is very interesting for imbalanced datasets, as mentioned previously. These contain large disparities in the number of samples from one class to the other, and are quite typical of real world data. The authors Shmueli et al (2018) mention the example of models trained to identify fraudulent credit card transactions: the data gathered would result in many non-fraudulent transactions in comparison to the suspicious ones and as such the models would have very little to distinguish between one class and the other. The issue can be mitigated, before building the model, by using techniques of oversampling or undersampling, depending on the model's needs. It should be noted, however, that this applies to binary classification models, as multiclass classification does not lend itself well to the procedure. (Shmueli et al, 2018)

The idea is to reach a similar number of samples for each class, especially because random sampling for a model on imbalanced data may produce even fewer inputs of the rare class – and quite commonly these are the most important ones. Undersampling, simply put, means that "the more plentiful class is undersampled, relative to the rare class" (Shmueli et al, 2018, p. 181).

The metrics used for the evaluation of models trained on data manipulated through undersampling should still, typically, be tested on data that represents the real ratio between one class and another (Shmueli et al, 2018). For example, if the underrepresented class composes 5% of the total inputs, and the other 95%, the model could be trained to learn on undersampled data that moved this ratio to a 50/50 split - and finally tested on a dataset composed as the original one. If that is not possible, the researcher may assign so-called oversampling weights to the inputs of each class represented in the confusion matrix and adjust the metrics to better represent reality.

2.1.2 Regression

Regression tasks are essential in the Machine Learning field and quite prevalent. However, this work provides only a brief overview as a contrast to the main topic, Classification. This is due to the fact that the project does not intend on developing regression tasks in its course.

In the case of Regression tasks, the model is defined and optimized to predict a numerical value output. A non-trivial application example is provided by Alpaydin (2016, p. 4), when mentioning the navigation of "an autonomous car, where the output is the angle by which the steering wheel should be turned at each time, to advance without hitting obstacles and deviating from the route".

Regression models, for optimization, are calculated in such a way so as to minimize the approximation error [30, direct, pg 4-10], meaning that "our estimates are as close as possible to the correct values given in the training set" (Alpaydin, 2016, p. 10). This is usually done starting from the simplest option possible, a linear model, and evolving the complexity as needed, for quadratic or higher order polynomials or nonlinear functions - while at the same time optimizing the parameters (Alpaydin, 2016).

The magnitude of the errors calculated for Regression models depend directly on the difference between real outputs and the predicted ones; this means that they should be analyzed in relations to the data, not as absolute numbers - but also serve as contrast to

Classification models, where it can be difficult to grasp the closeness or lack thereof between the many categories (Mohri et al, 2018).

2.2 Unsupervised Learning

For Unsupervised Learning tasks, there is an absence of a training process: the models are meant to assess by themselves any pattern they may find on the unlabeled data. This means that there is only the input data, and the output will depend on what the model discovers as the most frequently occurring patterns, trajectories, clusters - also known as density estimation in the field of Statistics (Muhammad & Zhu, 2015; Alpaydin, 2016).

These models tend to be particularly good in description tasks, “because they aim to find relationships in a data structure without having a measured outcome” (Jiang et al, 2020, p. 2). However, since there is no labeled data, the quantitative evaluation of the performance of a model can be more complicated to perform (Mohri et al, 2018). This is one of the reasons why they are so important for researchers: unlabeled data is so much easier and cheaper to find, which allows for a higher accessibility (Alpaydın, 2021).

2.2.1 Clustering

One possible model to perform density estimation is Clustering: as the name suggests, the goal is to group the inputs based on their similarities, a “sorting process where the criteria governing the sort are not known” (Faul, 2019, p. 149).

While it is widespread and commonly used, typical applications of clustering are on exploratory data analysis (Alpaydın, 2021) or the analysis of very large datasets (Mohri et al, 2018). In the first case, not much is known about the data set and the goal is to find naturally occurring groups, to be evaluated and classified later on. On the second, however, the data set is known; for example: the Customer Segmentation performed by a company, grouping them based on demographic characteristics and past transactions (Alpaydın, 2021). The information extracted can then be used to devise strategies for each cluster identified, regarding their communications, products and services, now a little more customized than before, or even find deviations and identify a possible niche segment.

One point of attention, however, is the features chosen for the analysis: the emphasis put on the attributes chosen to represent the data points will determine the outcome – and so affect the model's coherence. One way of thinking about this is considering candies (Faul, 2019): some are larger or smaller, have more or less sugar, one shape or another, chocolate or jelly; the distinguishing features can be many, and so the sorting will vary with the purpose of the analysis.

Because in Supervised Learning the data has a label, it makes sense to try and keep only the features that relate to it - however, for Unsupervised Learning the data has no label and so remains the question of what features are relevant to be used, or not used (Dy & Brodley, 2004).

Another application, for example, is document clustering. When combined with NLP, treating documents as bag-of-words allows the algorithm to find similarities between them – like the

number of shared words – and assign groups for classification. This means that a document can be automatically organized and filtered by subject, such as politics, sports, fashion and so on, or have their information extracted quickly and efficiently. The critical variable here is the choice of the lexicon used to give meaning to the words (Alpaydin, 2016).

In more complex applications, a class, a cluster, can even be made up of multiple groups: in optical character recognition, for example, there can be several ways of representing the same number 7, or in speech recognition where the same word may be spoken in different ways. The class these examples are assigned to needs to reflect this mixture (Alpaydin, 2021).

2.3 Semi-supervised Learning

Regarding Semi-supervised learning tasks, it tries to gather the best points of both Supervised and Unsupervised fields by using labeled and unlabeled data together, in different ratios (Khour & Gondi, 2020). This can happen for several reasons, such as in fields where the unlabeled data is very easy to obtain, but labeling it is very expensive, and so “the hope is that the distribution of unlabeled data accessible to the learner can help him achieve a better performance than in the supervised setting” (Mohri et al, 2018, p.6).

2.4 Reinforcement Learning

The fourth and last subcategory of Machine Learning tasks works by interacting with the environment to collect information, sometimes even affecting it, and is then rewarded depending on the effect created (Mohriet al, 2018). By trying to maximize the reward, the model is allowed to determine by itself the best possible behavior within the given context and learn (Kour & Gondhi, 2020).

One possible dilemma to deal with is exploration versus exploitation, as pointed out by Mohriet al (2018, p. 6), “since he must choose between exploring unknown actions to gain more information versus exploiting the information already collected”.

This work does not go into further detail, as the related concepts and models are not used in it.

2.5 Importance of Data

The model calculated to relate inputs to outputs depends directly on the dataset used, making f(x) deterministic (Faul, 2019). This means that independent of the task at hand, ensuring the appropriate data quantity and quality is crucial and should be appropriate to the learning model trained - some of them do well with small amounts of data - like Naive Bayes, while others will only thrive on large amounts of it - such as Neural Networks.

Since these models are being created and used to draw insights to make more informed business decisions, and thus have the potential to cause considerable effects on organizations, their quality will have a direct impact on the decisions taken (Ziora, 2020). At the end of the day, too much data with low quality will make for poor models and predictions.

It is crucial for the success of any model to ensure appropriate data quality and quantity, be it qualitative or quantitative. However, because data acquisition can be very expensive, both financially and time-wise, its use needs to be optimized. This is even more important because typically models require two entirely separate datasets: one for training and one for testing, meaning that data quantity becomes a valuable asset.

For Supervised Learning, for example, one of the most common optimization methods is called Cross Validation (Faul, 2019), and consists of methodologically and iteratively partitioning the available data multiple times into complementary subsets, one for training the model and one for validating its generalization capabilities. In the end, the model trained on the best possible partitioning is chosen based on its given metric - and so moves on to the testing phase, with the testing dataset.

Cross validation essentially measures the quality of the inductive bias of each hypothesis, each model among all those trained, and selects the best one with a predefined parameter. This metric can be, for example, the Accuracy of the models (which should be maximized), or the Mean Square Error (which should be minimized).

Data quality itself can be managed, to a certain point, before training the model. Even if the dataset is usually imperfect and contains noise factors or missing feature values, these can typically be cleaned, or the categories transformed into dummy variables to better adjust and reflect the content. (Sen et al, 2019).

This theme is further discussed in the specific sections for NLP and Sentiment Analysis, as there are challenges and considerations to be made regarding the data that are particular to these application fields.

The last point of attention regarding the data is ethical: both when selecting it and applying their conclusions. In the words of Patti et al (2017, p. 154), “the choice of the data sample will significantly affect the results we will be able to draw by a system trained on them, and the annotation itself must be designed and done in order to avoid biases and to expose the shared knowledge of the speakers’ community”.

2.6 Machine Learning Models

Before discussing the models themselves, there are a few concepts to understand. The most basic one is inductive bias: it represents the set of assumptions taken when building the model, the prior knowledge and beliefs that went into the process, and it influences the learning and the generalization from the available data. One way of introducing inductive bias into the model is assuming a given hypothesis space $\mathcal{H}$ – a space that contains all possible solutions between input and output, constrained by the model selected and its configuration (Alpaydin, 2016; Brownlee, 2020).

For example, by choosing to train a linear model – and so making the assumption that the relationship between the variables is so – the hypothesis space $\mathcal{H}$ is then constrained to find only linear solutions for $f(x)$ when learning the function that maps input to output with the least amount of error. In this way, the inductive bias shapes the learning process, guiding how the model searches for a suitable solution, favoring some hypotheses h over others.

This is important because $f(x)$ is the general function that describes the data and defines the model.

Some hypothesis spaces are larger than others: in regression, for example, as the order of the polynomial increases, so do the capacity and complexity of the space. Then, the next assumption to be made becomes when to stop (Alpaydin, 2016). The objective of making assumptions is to find the hypothesis space $\mathcal{H}$ with the highest probability of containing the solution to $f(x)$. So learning "is not possible without inductive bias, and now the question is how to choose the right bias" (Alpaydin, 2016, p. 38).

Finding the best possible hypothesis space $\mathcal{H}$ is directly related to model selection, but it should be highlighted that the goal of the model is not to find the exact relationship between input and output of past data, but to predict the output of new data – meaning that it needs to have the capacity of generalization (Alpaydin, 2016). Even if the data represents only a small subset of all possible input-output instances, the goal is to learn a model that's capable of predicting well for new and unseen inputs (Alpaydin, 2021).

Regarding the hypothesis space $\mathcal{H}$, the best course to find a model that is capable of generalizing well is to "match its complexity to the complexity of the function underlying the data. As the author Alpaydin (2016, p. 39) explains, "if $\mathcal{H}$ is less complex than the function, we have underfitting. But if we have $\mathcal{H}$ that is too complex, the data is not enough to constrain it and we may end up with a bad hypothesis."

If there is noise present in the dataset used in the training, using a hypothesis that is too complex may cause the model to learn it along with the underlying function – which creates a model with overfitting issues (Alpaydin, 2016). Conversely, complexity is also related to the sample size: if it is small, pairing it with a too complex $\mathcal{H}$ can also lead to overfitting (Mohri et al, 2018). A possible solution for this can be acquiring more data to improve the model, but "if $\mathcal{H}$ is not chosen well, no matter which $h \in \mathcal{H}$ we pick, we will not have good generalization" (Alpaydin, 2016, p. 39).

Another further assumption is made, now regarding the data – one which makes learning possible at all: its regularity, where similar inputs have similar outputs. The model is able to learn patterns because as the value of the inputs changes, so does the value of the output, in the past as in the future (Alpaydın, 2021). Without this regularity, it would not be able to transition between particular cases and a general model - it would simply not be applicable.

What the model needs to do is find the function $f(x)$ that best generalizes the relationship between input and output from the training data to unseen instances. Author Alpaydın (20, p. 40) sums up the combination of the generalization capacity of the model with the data and hypothesis space $\mathcal{H}$ assumptions by saying that "as the amount of training data increases, the generalization error decreases. As the complexity of the model class $\mathcal{H}$ increases, the generalization error decreases first and then starts to increase ". This last increase happens, then, due to overfitting: the model memorized too well the training data or its noise and so lost the capability to generalize well on new and unseen inputs.

Alpaydın (2016) expresses these three concepts as the triple trade-off:

> i) The hypothesis space $\mathcal{H}$ should be just large enough to include the solution that solves the input into the output from the data;

ii) The amount of data used should be enough to allow the model to form an optimal hypothesis h;

iii) A good optimization method must be used to find the hypothesis with a good enough generalization capability.

But it is possible – and necessary – to delve into one more consideration regarding the complexity of the hypothesis space for Supervised Learning algorithms: that of the interplay between model complexity and the bias-variance trade off. It represents the balance between the model's ability to capture the underlying patterns in the data – the bias – and its susceptibility to fluctuations in the training set – the variance.

While the inductive bias seen previously relates to the assumptions made regarding the data distribution and relationship between variables, thus shaping the learning process, the bias refers to the systematic error in a given model's predictions and thus speaks of its performance after the learning phase. The bias represents how much the solution found differs from the true function, from the true relationship between input and output – or, in other words, a small bias signifies that the model fits the process well enough (Faul, 2019).

Variance, on the other hand, is more closely related to how the different possible solutions change from one another based on the changes that occur in the dataset – put simply, how sensitive the model is in relation to changes in the training data (Faul, 2019).

The problem is that "in general, making one of them smaller increases the other. This is called the bias-variance trade-off" (Faul, 2019, p 228). The concept can be better explained by saying that by increasing the complexity of the model when looking for a better fit of $f(x)$ (and lowering the bias), the solution will encompass more of the minor changes in the dataset, thus increasing its variance.

Conversely, if the variance is kept very low, it's possible that the model has a poor goodness of fit, and so a high bias; this means that the ideal model needs to account for the best possible trade-off between its bias and its variance (Alpaydın, 2016).

One way of adjusting the model's learned coefficients to reduce unnecessary complexity is by introducing a penalty term (or entropy measure) when calculating and validating them (Faul, 2019). This is further discussed in the following sections.

In sum, according to Alpaydın (2021), the goal of using machine learning models is to learn the rules that explain and govern the data: by looking at the function calculated, the underlying process of the data can be explained. For example, a model whose goal is to calculate the credit score of a customer can potentially provide information on the factors – and how strong is their influence – of a high-risk or low-risk consumer. Unfortunately, some of the models seen in the next section are better at this task than others; for instance, while Linear Regression and Decision Trees models are very easy to interpret, a Logistical Regression one can be trickier to evaluate the factors and their inherent weight – and a Neural Network model works in a black-box manner and will provide no such insights.

Although there are models that are flexible and can be used for more than one single method of learning, this work will explain the models based on the previously seen

classification, along with a brief explanation provided on the context of the applications of each, as follows below.

2.6.1 Generalized linear models

As seen in previous sections, regression problems are about discovering the relationship between variables of interest, meaning that they are “related to curve fitting, interpolation, and data prediction” (Hájek & Barushka, 2019). The idea at its core is to understand and predict the change of a dependent variable, given the change in one or more independent variables related to it. These models are trained with labeled datasets, thus being classified as Supervised Learning methods.

Regression models are widely used and tend to be easy to interpret, which promotes their applications in a variety of fields like Machine Vision and Engineering, Medicine, Finance, Economy, going as far as predictions for Hydrology and Seismology (Hájek & Barushka, 2019). This means these models can use the same logic to predict variables of such a wide range as machine degradation, market volatility and ground water level.

The performance of regression models depends on how closely they follow the trend of the data, and so how well they predict the output for unseen input – but it hinges on the assumption that the dataset used for training is as diverse and comprehensive as needed to capture the relationship between the different variables (Alpaydın, 2021).

The performance will also depend on the predictors [53, pg 230], and how much they correspond both to each other and to the regressand: the aim is to find a linear model based on few minimally correlated regressors, that are in turn each highly correlated to the output, to explain the data while keeping as much information as possible.

When the first requirement is not true, the issue presented is called Multicollinearity (Faul 2019), and means that at least one predictor is redundant, modeling the same behavior in the output as another; in essence, this means that different input coefficients give the same output, and thus since different models will be equivalent to each other, small changes in the input will cause large changes in the model.

2.6.1.1 Linear Regression

Widely used for being simple and highly interpretable, Linear Regression models are a powerful framework inside of the Supervised ML field for modeling and predicting numerical outcomes. Not only that, it provides a benchmark to compare other, more complex models, and justify the need for more or less complexity. In essence, by fitting a linear equation to a dataset, the relationship between the diverse building blocks and their labels can be estimated.

It is important (Yale University Department of Statistics and Data Science, n.d.) to evaluate and make sure that there exists a relationship; while one does not need to cause the other, but they do need to have some degree of association, of correlation. A very simple verification tool is the scatterplot between input and output: if it does not show any correlated upwards or downwards trend, a linear regression model may not be particularly useful. Conversely, the correlation coefficient can serve as a quantitative indicator of the

relationship: a value in between -1 and 1; the closer it is to the extremes, the stronger the linear relationship, while it is weaker or nonexistent the closer it arrives to 0.

Given a set of observation (x_n, y_n) pairs, the correlation coefficient is calculated through (5).

$$r = \frac{1}{n-1}\Sigma\left(\frac{x-\bar{x}}{s_x}\right)\left(\frac{y-\bar{y}}{s_y}\right) \quad (5)$$

It is a case of polynomial regression, in which the polynomial is a linear one (Faul, 2019). In its simplest form with only one dependent variable, the regression equation (Yale University Department of Statistics and Data Science, n.d.) is quite straightforward, composed of: y as the output to be estimated, a as the intercept – so the output value when all possible inputs are zero, b as the slope and finally, x as the independent variable, as such

$$y = a + b * x \quad (6)$$

However, the models created are typically much more complex, with many more independent variables represented by each respective coefficient β_n, thus transforming the initial representation into

$$y = \beta + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon_n \quad (7)$$

Since it's been developed and applied in so many different fields, an array of terminologies can be used to describe each element. As such, this research will use the following: the output – scalar function value *y* – is called interchangeably regressand or dependent variable, while the input – features or characteristic of the data – may be called either regressor or independent variable. The relationship to the measurements according to Faul (2019) is summed up by

$$y_n = f(x_n) + \epsilon_n \quad (8)$$

The noise present in the data is represented by ϵ, which is "assumed to be independent and identically distributed (i.i.d.) following the normal distribution" (Faul, 2019, p. 217) – also called Homoscedasticity. It is a key assumption for linear regression, and implies that the errors (or residuals) vary uniformly across all levels of the output data; they do not have a

particular pattern influenced by independent variables. The simplest method to evaluate the homoscedasticity of the model is by plotting the errors against the predicted values: if there is no discernible pattern, the assumption holds.

Beyond simply fitting a straight line into the data, Linear Regression is a very dynamic and flexible model: the term linear refers to the relationship between the inputs, not the function of the inputs themselves – which means that these can be non-linear (Faul, 2019). In the end, the model is the linear "function of the input whose slope and intercept are the parameters learned from the data" (Alpaydın, 2016, p. 41).

Once the function $f(x_n)$ is approximated, the coefficients calculated – also called vector weights (Faul, 2019) – can provide an indication of the importance of each independent variable, if they are to be kept or removed, if they add or subtract from the output. Even further, they can provide insights into the underlying relationship in the data.

There are different techniques to calculate the coefficients, the most common of which is called Ordinary Least Squares. This method finds the best-fitting line for the data by "minimizing the sum of the squares of the vertical deviations from each data point to the line" (Yale University Department of Statistics and Data Science, n.d.). It is a very straightforward measure for data that conforms to assumptions of homoscedasticity and absence of collinearity.

To compare the models amongst themselves, between a simpler one and a more complex one, with more parameters, it can be useful to calculate the R^2. It is found with the residuals of each model, and takes into consideration the errors, but also the number of parameters used, for example, which is the reason why this comparison can be done.

Author Kutner (2005) provides two choices to move forward when the simple linear regression model is not the correct choice for the given data set: either move on to another, more appropriate model, or employ a tactic of data transformation, to adapt it to the particular solution. If successful, the latter option may be better: it allows for the continued use of the simpler methods.

The transformation approach tries to linearize, as much as possible, a nonlinear regression function.

2.6.1.2 Polynomial Regression

An extension of the linear regression, here the idea is to fit a more complex function on the data, a polynomial of a higher degree. Meaning that, "when the regression function is not linear, a direct approach is to modify the regression model by altering the nature of the regression function" (Kutner, 2005, p. 128). Thus, by increasing the complexity of the model itself (the order of the polynomial), it then becomes more flexible and capable of capturing more complex relationships within the data.

These models may contain one or more independent variables, each present in various powers (Kutner, 2005). For example, considering a very simple example in Equation (9), a *second order model with one predictor variable*, the independent variable is present to the

first and second powers. Because of that, the typical notation is slightly different to reflect the pattern, as in (10) below.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_n \quad (9)$$

$$y_i = \beta_0 + \beta_1 x_i + \beta_{11} x_i^2 + \epsilon_n \quad (10)$$

Again in this case, β_0 represents the value of the dependent variable y_i when all inputs are zero: the value may or may not have meaning, depending on the problem under evaluation.

It is important to keep in mind, however, that a possible drawback of this approach is that by increasing the order of the polynomial, “small changes in the dataset cause a greater change in the fitted polynomials; thus variance increases” (Alpaydın, 2016, p. 82). This is the direct reflection of the bias-variance trade off discussed at the beginning of this section.

Special caution is needed when considering models with independent variables present above the third power: the interpretation of each coefficient may become difficult and the estimations for unseen values can lose reliability (Kutner, 2005). The higher the order – and better the fit – the more the prediction line will coincide perfectly with the observed values from the training data set. This means that the model may lose the overarching trend underlying the data, which is the main purpose of creating a machine learning model for estimating purposes.

Going further, polynomial regression models “may provide good fits for the data at hand, but may tum in unexpected directions when extrapolated beyond the range of the data”(Kutner, 2005, p. 294). To be sure to find the best possible generalization, the complexity of the learner needs to correspond to the complexity of the data – and so the function that describes it; for polynomial regression, one indicator for the order chosen is the one that best minimizes the generalization error (Alpaydın, 2016).

2.6.1.3 Logistic Regression

By providing a discrete output, compared to the continuous output from the previous models, Logistic Regression is primarily used for Classification problems or predicting the probability of an event. Despite the common name, this is where they differ: logistic regression models mainly aim to find the decision boundary between one class and another, to model the relationship between dependent and independent variables and the probability of the outcome belonging to a particular category, not a continuous numerical output.

It is a quite flexible model from the standpoint of the data types to be used: not just continuous, but the models can be estimated using binary data, or nominal or ordinal – so long as these are transformed into dummy variables during the data preprocessing stage.

Although the framework used has its similarities with linear regression models, also estimating coefficients for independent variables, the underlying statistical technique and

interpretation of the model are different. Above all, it tends to be used for binary classifications, so situations in which there exist two possible outcomes: yes or no, 0 or 1, true or false. The logistical function, for example, is given by

$$f(z) = \frac{1}{1+e^{-z}} \tag{11}$$

This function, also known as a Sigmoid Function, if plotted for intervals between -∞ and +∞ will be represented as

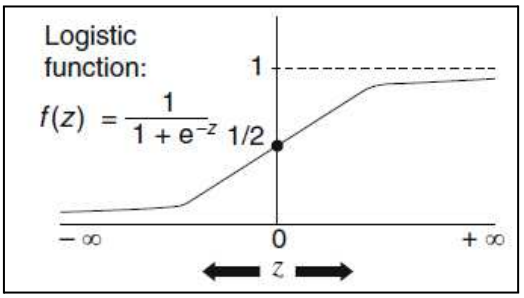


Figure 2. Representation of the sigmoid function for [-∞; +∞] by Kutner (2005).

The total range of the function $f(z)$ is between 0 and 1, independent of the value of z, which is perfect for describing probabilities; even further, the shape of the graph lends itself nicely to simulating a threshold between one class and another, be it sharper or more gradual. These are the two main motives for the model's popularity (Kutner, 2005).

Now, to translate this function into a model the output z_i is defined as the linear sum of the independent variables x_i multiplied by the coefficients β_i, as seen in (12). This linear sum is subsequently solved using $f(z)$, which results in (13) as follows:

$$z_i = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i \tag{12}$$

$$f(z_i) = \frac{1}{1+e^{-(\alpha+\Sigma\beta_i x_i)}} \tag{13}$$

The unknown parameters to be calculated by the algorithm are α and β_i, based on the training data set available (Kutner, 2005), and its end goal is to map the linear combination of the independent variables to a range between 0 and 1. To illustrate these concepts, an example can be proposed:

As for the calculation of the parameters themselves, the Maximum Likelihood method is typically used.

Since it can be trickier to understand the influence of each parameter, if compared to the linear regression models, the results and the relationship among them can be interpreted using the Odds Ratio. This indicator essentially measures the odds of the dependent variable changing from one category to another when influenced by a one-unit change in the independent variable.

It is also less intuitive to try and compare a simpler logistic regression model, with less parameters, to a more complex one. It is simpler to evaluate if a particular parameter is significantly impacting the output, or not – in which case, it can be removed to save space and time.

Because a logistic regression model does not share the same concept for residual as a linear one, for instance, using the R^2 for this task is not possible.

2.6.2 Deterministic models

The models classified as deterministic learn rules from the dataset and apply them to make predictions on new inputs, meaning that the same input will always have the same output – these models then can be very understandable and straightforward, but tend to perform poorly with very complex or nonlinear data (Bhaskaran, 2023; Kothari, 2023).

Some tasks may benefit more than others, especially ones with regular and vast datasets, because this type of model has no role for randomness, as the rules learned in the beginning are then applied equally on all subsequent inputs. The author Mehta (2022) states clearly that the “mathematical characteristics are known in this case. None of them is random, and each problem has just one set of specified values as well as one answer or solution. The unknown components in a deterministic model are external to the model.”

One model in particular, Support Vector Machine, has particularly good performance for natural language processing, and more specifically sentiment analysis, tasks. It is quite prevalent in the published research discussed in this work.

2.6.2.1 Decision Tree-based models

First introduced in the 1960s, the concept of Decision Tree models was meant to represent a structured decision-making process, easy to visualize and interpret. The model itself is considered a Supervised Machine Learning method, and has been extensively used for classification, regression and clustering purposes – mainly as weak learners to boost more effective learners because they can be fast to train, evaluate and interpret (Mohri et al, 2018)

A decision tree can be defined as a hierarchical model that identifies local regions through iteratively splitting the data based on the similarities found. Put simply, the model looks for rules in the data, it proposes questions about it to take decisions to segment it into branches until arriving at the final leaf – the resulting output. Each decision point is called a node and “the tree structure is not fixed a priori but the tree grows, branches and leaves are added,

during learning depending on the complexity of the problem inherent in the data" (Alpaydin, 2016, p. 214).

The concept can be represented as seen in Figure 3, in which x_1 and x_2 are two variables, two inputs that can be used to classify the output into 5 categories. The model starts with a decision node, in this case $x_1 < a_1$, and branches out iteratively until the input reaches the final leaf, the output, which contains its class (for classification tasks) or its value (for regression tasks). Each leaf then represents a region inside the data that contains a particular class or value (Mohri et al, 2018).

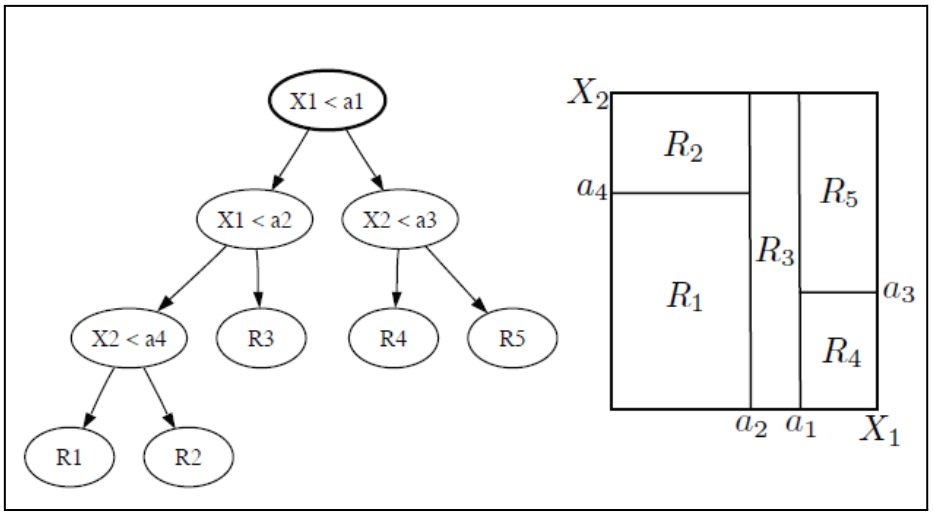


Figure 3. Concept of Decision Tree model logic by Mohri et al (2018).

The tree seen in this figure is called Univariate Decision Tree, as each decision node considers only one of the available input dimensions to branch out. Typically, if the variable is discrete, than the node can be split up to its n-value – for example, if a given attribute is the color ϵ (black, white, green and blue), then the node can be split at most into 4 branches, one for each value (Alpaydin, 2016). If the variable is numerical, then the node is represented by a comparison to split the data as seen above ($x_1 < a_1$).

There can be many trees that split the training data with a minimum amount of error and the goal is usually to find the shortest one, with the smallest possible number of nodes so as to control the complexity of the model – known as recursive binary splitting (Alpaydin, 2016). In this case, the algorithm typically looks for the best possible split at each point based on a given criterion without considering possible future splits for optimal overall outcome. For classification tasks, the nodes of a tree are typically guided by an impurity criterion that evaluates the how pure each branch is after the node: the algorithm then stops branching out when a sufficient level of purity is found or when the number of points per leaf have become too small to divide it further (Alpaydin, 2016, Mohri et al, 2018).

Authors Mohri et al (2018) propose two criterions for measuring impurity for testing nodes in classification tasks, which are quite similar in that both try to find the variable that splits the data into the most homogenous (or pure) child nodes.

$$Entropy = -\sum_{l=1}^{k} p_l(n)\, log_2 p_l(n) \quad (14)$$

$$Gini\ index = \sum_{l=1}^{k} p_l(n)(1 - p_l(n)) \quad (15)$$

In which l represents a possible class out of $l \in [k]$, and so $p_l(n)$ represents the fraction of points that belong to l for each node n.

The Entropy measure seen in (14) varies from 0 to 1, and measuring the disorder or impurity of a node suffers the more variable its composition (Mohri et al, 2018): supposing classes A and B, a node containing 10 points A and 10 points B would have a higher entropy than one containing 5 points A and 5 points B. Similarly, a node containing only points of class A would have zero entropy, which is the highest level of purity achievable.

The Gini Index seen in (15), on the other hand, varies between 0 and 0.5 and tries to calculate the probability that a given input will be misclassified when its class is randomly chosen based on the distribution of the data (Mohri et al, 2018). The lower it is, the better, as it represents a smaller probability of misclassification – having a Gini index value of 0 means all values are properly classified.

For numeric variables, which can potentially contain an infinite number of points, this process of comparison can be simplified by testing out adjacent points between the different classes or halfway between points (Alpaydin, 2016).

The tree may calculate either Entropy or the Gini Index metric for each variable available and its possible splits to recursively define the one with the least impurity in the child nodes created.

The idea is that the purer each subsequent split is, the shorter the tree will be, and Alpaydin (2016) points out that different algorithms may use one metric or the other: the classification and regression tree (CART) developed by University of California professor Leo Breiman and colleagues uses the Gini Index as guide, while the Regression Tree ID3 by researcher Ross Quinlan is based on Entropy – however research has shown that these two metrics can be used interchangeably to yield similar results.

The author goes on to mention that this methodology overall tends to favor variables with many values, as their branches are typically less impure, and this causes undesired complexity. For noisy data, for example, having a larger tree to make sure it is as pure as possible may cause overfitting, so a threshold can be set to build a tree that may not have perfect Entropy or Gini values, but one that is close enough. This value is based on the cost of misclassification, as having a too small threshold can lead to large trees with high variance, and high thresholds lead to models with high bias.

For regression tasks the more appropriate metric is the mean square error (MSE) of the estimated value, as the variables to predict are continuous in nature. Author Prasad (2022)

explains the following equation of MSE to select the best split at each phase of the tree and reduce the error in the child node:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y_i})^2 \quad (16)$$

So if Y_i represents the actual output value, and $\widehat{Y_i}$ represents the predicted one, the idea is to minimize how far the prediction is from the expected output.

Because the fundamental idea behind decision trees is so easy to interpret, they have been extensively used for machine learning purposes, even over more complex and accurate models that do not provide the same simplicity in its interpretation (Alpaydin, 2016). The algorithms can take many data types as input (like numerical, categorical and ordinal) and be quite fast to train (Rathi et al, 2018), being reliably used for recommendation systems and credit scoring, or spam email detection and sentiment analysis in the NLP field.

One point to keep in mind, however, is that despite their straightforward interpretation decision tree models are hierarchical, meaning that each phase depends on the previous one and so can be quite unstable due to small changes in the data (Mohri et al). The resulting splits may be quite different from one model to the other with similar datasets.

The models can be evaluated based on their purpose (Shmueli et al, 2019): as seen in previous sections, classification algorithms are typically chosen or discarded based on their Accuracy, Precision and Recall – or more visually the ROC curve. As for regression tasks, the models are usually evaluated based on their Mean Square Error or Root Mean Square Error.

These values and the acceptable thresholds are also task-dependent.

2.6.2.2 Random Forest models

When the value of a highly interpretable model is not necessary, where visualizing the model's rules is not a requirement, the logic behind Decision Trees can be improved upon to reach better performance: one such example was introduced by Leo Breiman and Adele Cutler, named Random Forest (Shmueli et al, 2019). It can be used to tackle both regression and classification problems as a Supervised Machine Learning approach.

The idea behind this technique is to combine several uncorrelated trees, joining multiple algorithms to improve predictive performance based on the Ensemble Learning concept: merging several classification models to acquire a better, more accurate final one. Author Alpaydin (2016, p. 235) explains it by saying that "if we train not one but many decision trees, each on a random subset of the training set or a random subset of the input features, and combine their predictions, overall accuracy can be significantly increased."

Essentially, Random Forest models are created through Bagging, an ensemble technique that leverages bootstrapped data to create and merge a myriad of slightly different models (Alpaydin, 2016). This is possible because by bootstrapping the data, by randomly picking input/output pairs, several new same size datasets can be created – and the models can commit and learn from different errors on different subsets. This is typically done with

replacement, meaning that the same datapoint can be randomly picked more than once or not at all.

It is worth it to mention that both Decision Tree and Random Forest methods work with the initial division of the original dataset into training and testing subsets.

The trees, each with their own bootstrapped dataset, are trained with a random subset of variables at each stage (Shmueli et al, 2019): this is done to make sure that all trees are not created equal and reduce the correlation between one another, especially in the first split performed; it also means that Random Forests are created through bagging both the data and the predictive features. The process is represented below, in Figure 4.

There are three hyperparameters to be set to create a Random Forest model: node size, number of trees and the number of sampled features (IBM, n.d.). The first serves as a threshold for how far the tree can be split and grown, so if the value set for minimum node size is 10, once the tree reaches a point in which a given node n has that amount of points or less, it cannot be split any further into other branches. The second represents how many decision trees will be trained and this value can vary a lot based on the size of the dataset, its complexity and the computational time and power required. Lastly, the number of variables sampled is the way the model avoid highly correlated trees and a very typical value for classification tasks is the value of $\sqrt{p}$, while for regression tasks it can be $\frac{p}{3}$, in which p is the number of all predictive variables available (Sutton, 2020).

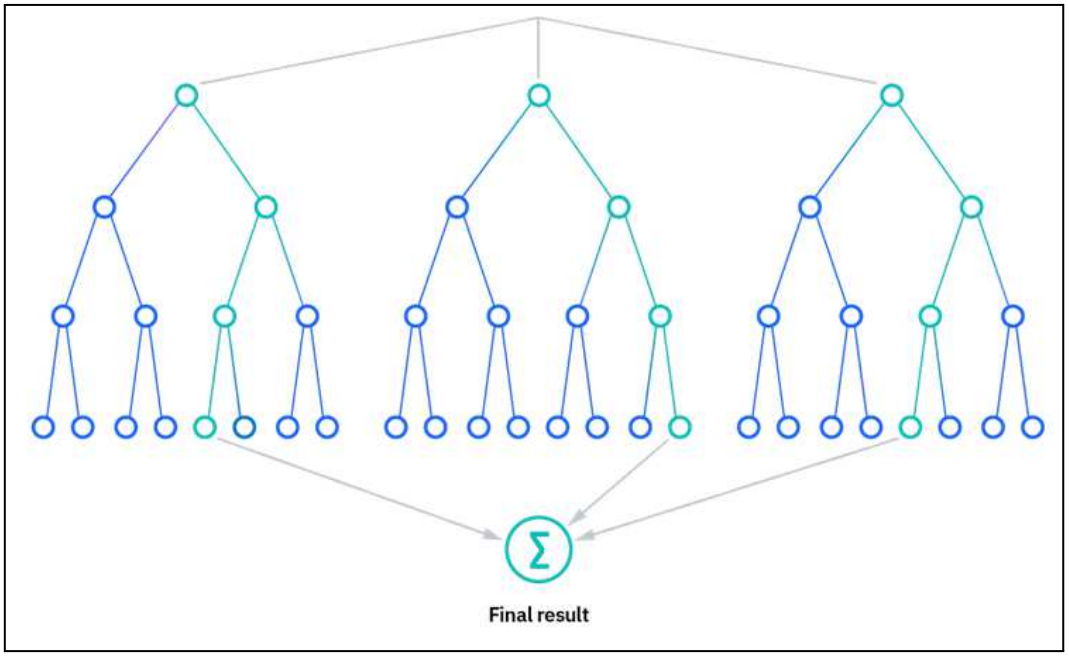


Figure 4. Concept of Random Forest model logic by IBM (n.d).

Still related to the number of sampled features hyperparameter, authors Syam & Kaul (2021, p. 170) point out that “when the analyst suspects that the data has a large number of

correlated predictors variables, then fitting a random forest with a smaller sample of features would be helpful".

The manner in which the decision trees created from the many subsets of the data are joined into one final Random Forest model is different for either regression and classification tasks.

In the case of regression, the predictions set by each individual tree are typically averaged amongst each other to define the prediction of the Random Forest, as they are represented by continuous values (IBM, n.d.; Lohith et al, 2023). For classification, on the other hand, Random Tree models typically aggregate all of the predictions based on the majority rule: "it is just that class which is the most commonly occurring class among the D predictions" (Syam & Kaul, 2021, p. 171).

It is interesting to note that this method does not result in a tree that can be easily displayed in a diagram and interpreted, meaning this type of model is no longer so easily interpretable. However, because of the training performed on so many slightly different datasets and the fact that the features are not always selected equally, Random Forest models can provide scores on the importance of each variable, the contribution of each one (Shmueli et al, 2019). The author also explains that this is done by evaluating the decrease in the Gini Index metric for each predictor over all of the trees trained.

As seen in the previous section, Decision Trees can be prone to overfitting the data, tightly encapsulating the samples used for training; it is not so with Random Forest models (IBM, n.d.). This happens due to the process of averaging the predictions and classifications of the uncorrelated trees. The flipside of this extra processing is the time and power it consumes.

In the same line, the evaluation of the models follows the task it tackles.

2.6.2.3 Support Vector Machine

One of the most widely used supervised machine learning models for both regression and classification tasks, especially in the last couple of decades in the NLP field, is the Support Vector Machine. It is interesting to note that its roots "go to potential functions, linear classifiers, and neighbor-based methods, proposed in the 1950s or 1960s; it is just that we did not have fast computers or large storage then for these algorithms to show their full potential" (Alpaydin, 2016, pg. 16).

It is a discriminant-based method that applies the principle of never solving a more general and complex problem before the problem itself: for example, when solving classification tasks with SVM one only needs to discover where the class boundary lies, instead of also investigating class densities and other unnecessary information (Alpaydin, 2016).

This results in the model's ideology of finding the best hyperplane so that one class is distinguished from another (Chauhan et al, 2021). Not only that, but doing so with the largest margin possible between the classes so as to insure good generalization in case of factors such as noise, meaning that "the optimal separating hyperplane is the one that maximizes the margin" (Alpaydin, 2016, p. 351). This point is clearly represented in Figure 5 below.

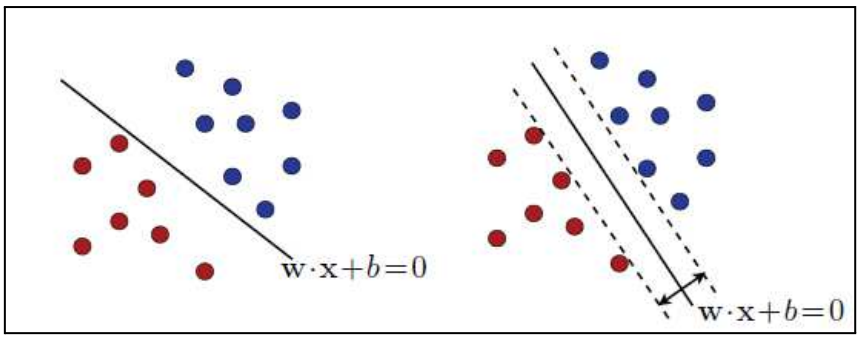


Figure 5. Visual representation of the optimal separating hyperplane by Mohri et al (2018).

It represents a very simple linear classifier under the assumption that a linear hyperplane can perfectly separate the data into two populations, defined by the equation $w.x + b = 0$.

Since there can be infinite hyperplanes that separate the populations, the SVM model works with the idea of a geometric margin, which the authors Mohri et al (2018, p. 80) describes as "the geometric margin of a linear classifier h for a sample S = (x_1..,x_m) is the minimum geometric margin over the points in the sample [...], that is the distance of the hyperplane defining h to the closest sample points."

Put more simply, the geometric margin in a SVM model measures the distance between the decision boundary between the classes and the nearest data point from either one. The larger it is, the higher the confidence of the model's predictions and the less likely it is to make mistakes, which is why a SVM typically looks for the hyperplane with the "maximum geometric margin and is thus known as the maximum-margin hyperplane" (Mohri et al, 2018, p. 80).

That is seen in the right side of the figure, where the hyperplane is maximizing the available margin in relation to the points in each class – these points, the closest ones to the separating hyperplane, are called Support Vectors and are the only ones that carry any information whatsoever to the model.

If, however, the classes are not perfectly linearly separable and as such no hyperplane separates them fully, the model may try to define the one with the least error. This tends to be the more common case, because real world data is usually riddled with noise and outliers. Author Alpaydin (2016) defines two types of deviations that the model can incur: an instance of a given class can lie either on the margin (and as such not far away enough from the hyperplane) or on the other side of it altogether and be misclassified by the model.

To solve this issue, the idea is to define a soft margin that allows for a given amount of misclassification based on the task at hand (Mohri et al, 2018). The hyperparameter used to control the level of misclassification is commonly known as C or slack variable, and the larger it is, the larger the allowed misclassification. It is typically chosen through cross validation based on the best model's Accuracy value and represents a trade-off between the largest margin possible and the smallest error achieved, a bias-variance trade-off: the smaller C is, the more variation the predictions will have. Conversely, the higher C is, the smaller the variance the model will have but the bias may increase along with it.

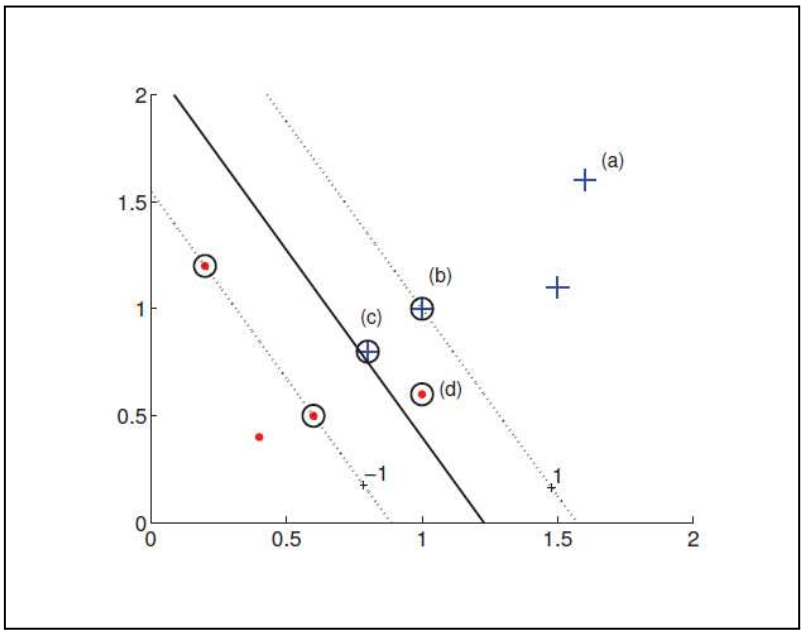


Figure 6. Visual representation of the soft margin concept by Alpaydin (2016).

Author Alpaydin (2016, p. 355) refers to it as a penalty factor to reduce the complexity of the model, noting that "we are penalizing not only the misclassified points but also the ones in the margin for better generalization, though these latter would be correctly classified during testing." They provide Figure 6 as a visual example, explaining that the only instance not considered a support vector is (a), as it is far from the margin on the correct side and as such would be classified well. Sample (b) lies on top of the margin, but on the correct side, while (c) is too close to the hyperplane. Lastly, sample (d) is a misclassification, as it is on the wrong side of the hyperplane.

If the problem cannot be linearly separated, however, one can map the space in which the data is distributed using non-linear basis functions, typically resulting in more dimensions than the original space (Alpaydin , 2016). This solution refers to the use of kernels such as polynomial and gaussian as a manner of "projecting the data into a high dimensional space" (Chauhan et al, 2021, p. 4).

2.6.3 Probabilistic models

Beyond the deterministic view of the models just discussed, the work through more static guiding rules, probabilistic machine learning tries to express the uncertainty related to the predictions to be made. The relevance of this point, in relation not only to the data but to the entire process itself, is emphasized by the words of Ghahramani (2015, p. 452), saying that real word data "can be consistent with many models, and therefore which model is appropriate, given the data, is uncertain. Similarly, predictions about future data and the future consequences of actions are uncertain. Probability theory provides a framework for modeling uncertainty".

Using the mathematics of probability theory to encompass the randomness and ambiguities present in real world data, especially Bayes' Theorem, the field tries to quantify and model the uncertainty related to each possible outcome (Zhu, 2018).

This modeling comes at various levels, from the lowest one (measurement noise, for example) to the highest and more complex – such as the parameters the models uses, how many of them are considered and at which values they are good for predictions, even the structure of the model itself, linear regression or neural networks (and then their hyperparameters) (Ghahramani, 2015).

Probabilistic models try to move away from this type of reasoning, they do not try to provide one single deterministic output, but the probability distribution of possible outcomes. Not only can this approach be more nuanced, it may provide a more comprehensive understanding of the variability contained in the data itself.

Author Murphy (2012, p. 307) further specifies that the core of probabilistic models is answering questions like "how can we use this distribution to infer one set of variables given another in a reasonable amount of computation time? And how can we learn the parameters of this distribution with a reasonable amount of data?".

And this can be done in a very straightforward manner, also known as Bayesian Learning: "probability distributions are used to represent all the uncertain unobserved quantities in a model and how they relate to the data. Then the basic rules of probability theory are used to infer the unobserved quantities given the observed data" (Ghahramani, 2015, p. 453). In essence, the prior probability distributions are transformed into posterior probabilities to predict likely outcomes.

Typically, probabilistic models can be used for both supervised and unsupervised machine learning tasks, but one important distinction to be made is the number of parameters: if it is fixed, the model is considered parametric and is used for supervised ML; on the other hand, if the model is non-parametric, the number of variables to learn from grows with the amount of data used in its training (Murphy, 2012).

2.6.3.1 Multinomial Naive Bayes classifier

The Naive Bayes classifier is a widely known and used Machine Learning model, with extensive applications in the NLP field. First applied to text classification tasks in 1964 (Martin & Jurafsky, 2023), the more recent applications are discussed in section 2.3.

In relation to other models, Naive Bayes classifier tends to work well for smaller or more constrained datasets, when data is not widely available or difficult to retrieve. One possible explanation for this is that Bayesian models are not as prone to overfitting the noise contained in the dataset, because they typically work through averaging the model over the data, instead of fitting the parameters to it (Murphy, 2012; Ghahramani, 2015).

One fundamental assumption of the model – and the reason why it is named *naive* – is that all of the features are conditionally independent and have the same weight upon the final outcome (Murphy, 2012; Alpaydin, 2016; Martin & Jurafsky, 2023). In reality, the features most likely do have dependencies amongst each other, but this assumption allows for a

simpler model that still works well – especially in cases in which the data available is not sufficient to accurately predict dependency.

Authors Martin & Jurafsky (2023, p. 5) provide a clear description of how the multinomial Naive Bayes model works, and as such is used as the basis for the explanation contained in this section. The example described by them of a text classification task is given by the phrase "Naive Bayes is a probabilistic classifier, meaning that for a document d, out of all classes $c \in C$ the classifier returns the class $\hat{c}$ which has the maximum posterior probability given the document."

It is based on the Bayes Theorem, given by (17) in which $P(c|w)$ represents the probability of c given that we know d has happened. In other words, how likely is a text to belong to a given class, given that it contains a particular set of words (features).

$$P(c|w) = \frac{P(d|c)\,P(c)}{P(d)} \tag{17}$$

Because the goal is to find the class that maximizes the probability of $\hat{c}$ given the document d the equation becomes:

$$\hat{c} = argmax_{c \in C} P(c|d) = argmax_{c \in C} \frac{P(d|c)\,P(c)}{P(d)} \tag{18}$$

After transforming the data, the text, into its bag-of-words representation (section 3.3.1) and keeping count of the frequency of each word contained therein, the model starts by calculating the prior probability of each class $P(c)$. This means the probability that any one class will occur within the dataset based on the proportions present in it.

The variable $P(d)$, the probability of the document, can be disregarded because it is constant, it does not change for each class considered.

Next, the model looks for the likelihood of the document given the class $P(d|c)$; this task can be quite difficult and computationally expensive if trying to calculate all possible combinations of features, of words – even impossible if the number of parameters is too high. As such, the model pairs the assumption of conditional independence with the idea that word order can be disregarded, naively simplifying the process: now, the probability of each word $P(w_i|c)$ occurring in a given class can simply be the product of one independent probability by the other as seen in (19).

$$P(d|c) = P(w_1, \ldots, w_i|c) = P(w_1|c) \cdot P(w_2|c) \cdot \ldots \cdot P(w_i|c) \tag{19}$$

Let it be said that this is calculated using all of the words contained in the document in question, so as to include the probability of each and every one of them occurring in that particular class.

The application of this process for the text classification example is then expressed by (20), iterating through all the words in the document – represented by i position.

$$class = argmax_{c \in C} P(c) \cdot \prod_{i \in Positions} P(w_i|c) \quad (20)$$

The final $class$ assigned is calculated for all $c \in C$ and selected as the one that maximizes the calculated probability.

One issue present in this calculation, due to the fact that it is the result of chain multiplication, is the possibility of encountering words that were not present in a given class on the training dataset. These words will be assigned a zero probability of occurrence during the classification process and as such the total probability will result in a zero value. The typical solution for this is adding the Laplace Estimator, also known as add-one smoothing, to make sure the probability is never zero, even when the feature is non-occurring for a given class.

Unknown words – that were not present in the training dataset at all – can be dealt with by ignoring and removing from the test set, and stop words that do not usually add value can be removed when sorting through the BOW (such as sorting the words by their frequency and removing the top 10 -100 entries).

The Naive Bayes model can be adapted to the task. For sentiment analysis, for example, the simple fact that a word occurs seems to be more important than how many times it has occurred, so one may set a word count maximum of 1 per document – even if the word is present more than once in it (Martin & Jurafsky, 2023).

Finally, to evaluate the performance of the model created, typical metrics discussed in section 2.1.1 are used, such as Precision, Recall and F1.

2.7 Model Evaluation and Selection

In the words of Mrabet et al (2021, p.4), “after building any ML model, the need to evaluate and analyze the behavior and performance of this model is primordial”. What they mean by this phrase is that for a ML model to have actual value, it needs to be carefully assessed through characteristic and pertinent metrics. It also means that the influence of the task itself should be taken into account.

A model built typically should be tested through metrics important in themselves, such as Accuracy or MSE seen previously, as these will inform the research on how well it generalizes on real world, unseen data. How the model will perform under the intended circumstances.

However it needs to be tuned, also, with the task in mind: a model built to spot, recognize and classify tumors from a patient’s exam should place a much heavier weight on positive classifications. This happens because the cost associated with a false positive is much higher than the one associated with a false negative, the latter of which will be evaluated and discarded by a medical professional. However, allowing for a tumor to pass by as a false negative can bring serious consequences.

In sum, not only the data needs to be carefully considered for quality and quantity, the model built and tuned for the problem at hand, but also the weight placed on the erros and assumptions de model makes need to be carefully considered.

3

Natural Language Processing

3.1 Introduction to NLP

As established in previous sections, Natural Language Processing is a “tract of Artificial Intelligence and Linguistics, devoted to making computers understand the statements or words written in human languages” (Khurana et al, 2022, p. 3714). The field can be subdivided into two parts: Natural Language Understanding, in which this research is placed, and Natural Language Generation, which comprises the process of teaching machines how to produce language in a meaningful way.

NLU, for its part, is what allows researchers and professionals today to teach machines how to extract concepts and entities, emotions, keywords and more from natural human language – using knowledge from the field of Linguistics to draw on subjective concepts such as meaning and context (Khurana et al, 2022). And it has come a long way.

The main stages of NLP development along the decades has been summarized below by Campesato (2021), starting from the 1950s, highlighting the most common techniques per each period.

1950s-1980s: rule-based systems
1990s-2000s: corpus-based statistics
2000s-2014: machine learning
2014-2020: deep learning

In its initial stages, around the 1950s, the field was coming into existence simultaneously in different parts of the world and had a strong focus on Machine Translation for English and Russian (Khurana et al, 2022). With the further development of computer-based linguistic studies and the beginnings of AI, during the decades between 1950 and 1980s researchers started using conditional logic and rule-based algorithms to include user’s beliefs and

intentions, attempting to include the more subjective nature of human language (Khurana et al, 2022).

However, there were many issues with this approach, two of which are lack of flexibility and range: human language is logical only to a certain extent and thus cannot be fully encompassed as such, and is in constant flux and evolution, meaning that the rules can change over time. So rules that worked – with exceptions – yesterday, may no longer work today or tomorrow.

Even further, for rule-based methods it can be difficult to connect two related sentences: words such as “that” or “this” can be used to reference one to the other, and over time statistical analyses were developed to try and predict the words most likely to follow a given sentence – a tactic that reigned during the 90s, but the correct interpretation is better inferred with modern NLP methods (Campesato, 2021).

The challenge of interpreting, analyzing and manipulating natural language data demands different tools and methods, which are still under development today (Khurana et al, 2022). The current state of the art “generally adopts Machine Learning algorithms or is more generally based on statistical machine learning” (Khan et al, 2016, p. 98), embracing algorithms already discussed such as Decision Trees and Markov chains and still using the strategy of predicting the next word in a given sequence of words (Campesato, 2021).

The latest development, from 2014 onwards, involves combining Neural Networks with NLP tasks – this pairing can be considered a turning point, and is now better able to take into account the patterns within the data itself, instead of the manually constructed rules of previous decades.

This means that beyond an initial machine translation task, the field is now awash with complex applications ranging from Automatic Summarization (in which the computer *understands* a body of text and provides a general summary of its main ideas), to Discourse Analysis (the identification of a discourse structure and the context in which it is written), from Named Entity Recognition (the identification of different entities referred to within a body of text, such as a company or a person) to Optical Character Recognition (the transformation of handwritten text into machine-readable text) and many others (Khan et al, 2016).

In sum, it can be said that in this ever evolving field, “in high-level terms, there are three main approaches to solving NLP tasks: rule-based (oldest), traditional machine learning, and neural networks (most recent)” (Campesato, 2021, p. 142). This does not mean, however, that past complexities no longer exist.

3.1.1 Challenges of NLP

Grammar, as understood by the NLP field, “is not a set of rules that a speaker must follow as its production is considered to be well-formed, but rather a description of the syntactic phenomena used by any linguistic community at a given time” (Goyal et al, 2018, p. 128). This means that the understanding of grammar in NLP is not a pre-established, well defined set of rules about a given language, but rather the amalgamation of patterns and characteristics that exist within it, at any given point in time.

And for that, one of the main issues regards its ambiguity: the possibility that the same sequence of words, the same phrase, can lead to different interpretations (Khan et al, 2016). It is typically related to syntactic, semantic or lexical levels, better broken down by Naeem et al (2020, p. 3719) in saying that in the case of "syntactic level ambiguity, one sentence can be parsed into multiple syntactical forms. Semantic ambiguity occurs when the meaning of words can be misinterpreted. Lexical level ambiguity refers to ambiguity of a single word that can have multiple assertions". All of these tend to be specific per language, and should be dealt with as such.

In the first case, syntax-level refers to the rules to be followed to form a correct sentence in a given language (Khan et al, 2016). This means that the main reason for syntactic ambiguity in the text is its structure: because of a poorly defined expression, a phrase such as "he ate the cookies on the couch" can mean that the cookies eaten by him were previously on the couch, or that they were eaten while he was sitting on the couch (Khan et al, 2016). They are not incorrect, per se, but the structure could be better adapted towards one meaning or the other.

For its part, semantics is the meaning of a given word within a sentence: this suggests that the relationship between the several tokens can infer different, subtle meanings (Khan et al, 2016). An example given by the author is "colorless green ideas sleep furiously", which is correct syntactically, in its structure, but makes no sense due to its contradicting meanings: how can something be green and colorless at the same time?

In general, semantic ambiguity happens because the same word may be equivocal and have a variety of meanings, depending on its context, and so can usually be solved through word sense disambiguation (Khan et al, 2016). In the previous example, perhaps the word *green* can mean fresh, new, instead of the color.

The third level, lexical ambiguity, happens due to context, the lexicon in which it's present. The same word can represent different things when seen in different contexts, to which Khan et al (2016) give the example of the word *bank*: it can belong to a financial lexicon, as the institution, as it can belong to a nature lexicon, as the bank of a river.

These three levels of ambiguity can be solved by knowing the whole sentence (Khurana et al, 2022), which can be challenging as NLP tasks are not typically carried on at the sentence level. As such, several solutions have been proposed, such as "POS (Part of Speech) tagging, NER, SBD, word sense disambiguation and word segmentation that are carried out using machine learning models" (Khan et al, 2016, p. 95). These will be discussed in further detail in subsequent sections.

Beyond ambiguity and contextual understanding, identifying emotions such as irony and sarcasm or contradictory statements can also be quite challenging for NLP tasks (Campesato, 2021). The first is directly linked to Sentiment Analysis tasks, and the main issue is that the connotation of the sentence is the opposite of what is actually written; researchers can try to circumvent this by training models given certain cues that occur frequently with sarcastic or ironic affirmations, such as "*yeah, right*" and "*whatever*", or word embedding techniques (Roldós, 2022).

Beyond discussing the general issues NLP tasks have to deal with, one should keep in mind that these can vary vastly from one language to another, or even be specific within the language itself.

Author Campesato (2021, p. 118) Explains that each spoken language has its own rules, involving "a set of grammar rules of varying degrees of complexity, along with language specific features". They mention that some, like German and Japanese, can better withstand the changing order of words within a sentence, for example, while others like English and Latin languages depend upon a stricter order to convey the proper meaning. Words that exist in one may not exist in another, some may even be completely exclusive to a particular people, not having even a comparable representation outside of it. One can go further and explore genders, verbal conjugation or even the order in which the words are written – left to right or right to left, but the takeaway is that NLP is typically very language specific, and thus may require a deeper linguistic knowledge to be handled correctly.

As for the changes within the language itself, the main challenges refer to accents, slangs and dialects. One thing to keep in mind is that NLP tasks are not only related to written text, but also spoken human language – and may involve contact with people of all classes and backgrounds. This reflects the challenge posed: although accents, slangs and dialects may have some common features, they can represent changes inside the language based on country, region and population groups – such as age groups or "tribes": regional accents vary how the same words are pronounced or even the meaning itself, while slangs are not controlled or regulated by anyone, and can be used for obfuscation, understood only by a specific group, or represent completely new words (Campesato, 2021).

Models intended for broad use need to take into account these colloquialisms, informal phrases and expression and culture-specific lingo – and these tend to be ever evolving within a given language. One possible solution, then, is to use a vast amount of data, and models that are regularly trained and updated (Roldòs, 2020).

Another challenging issue to keep in mind is negation, because it can fully invert the meaning of a sentence, of an opinion – and as such, particularly important for opinion mining tasks. Authors Pang & Lee (2008) explain that bag-of-words method for example, seen in further detail in later sections, would represent phrases such as "*I like this book*" and "*I don't like this book*" in a very similar manner, the only element that transforms the meaning into polar opposites is the negation element "*don't*". Negation also pervades issues of sarcasm and irony, which can be much more subtle and sophisticated – thus difficult to grasp with NLP models.

One way of dealing with it is by prepending "not_" to the words that occur after a token of negation (Martin & Jurafsky, 2023). In effect, it takes the phrase "*I didn't like this product*" and defines the tokens [not_like, not_this, not_product]. As these are more likely to occur in negative contexts, one assumes they will be seen as cues to negative classifications.

Within a more restricted space, NLP tasks can also be related to domain-specific language. This creates specific lexicons that shape the models' understanding of words and their meanings: in business, different industries often use very different language and terms – for example, a model specific for the summarization of legal documents versus a model trained for assessing and solving plant maintenance issues will contain vastly different terms. These

“lexicons” on which models are trained will define their working capacity: a finding brought up previously, when discussing the work of Rain (2012) – in their research, the models were trained on reviews for products such as books, and when used to analyze different products their accuracy was much lower.

Even further, the same document may contain multiple topics. Thinking of product reviews, they may contain comparisons between several products, for example, or it may cite several varying features of the same product as subtopics (Pang & Lee, 2008). One may then need to choose if to disregard all topics but the main one, or try and identify all of them. The author provides insight into Sentiment Analysis tasks, saying that the latter is more common, saying that usually we “try to identify the topics and then determine the opinions regarding each of these topics separately” (Pang & Lee, 2008, p. 27).

Today there are a vast number of analysis tools specifically designed for numerous industries, however more niche fields may have to create and train new models for their own use (Roldòs, 2020). Even further, NLP-based models need to be developed with their final task in mind, be it very general and all encompassing like chat bots or very specific.

3.2 Text Preprocessing and Representation

By now it has been understood that a computer does not understand text as humans do: they interpret a sequence of symbols, related to each other only to the extent of the researcher’s choice. This means that when preparing the body of texts – the corpus – to train a particular model, it needs to be processed in a way that is optimized for this interpretation.

“The initial step involves validating the contents of a dataset, which involves making decisions about missing and incorrect data values” (Campesato, 2021, p. 501), or joining several sources of data into a single body, the dataset should be preprocessed with NLP techniques. This tends to involve very standard procedures, discussed below, such as Text Normalization, Noise Removal and Tokenization – these are mostly good practices to make sure the data is clearer and easier to interpret, and some task dependent ones: Stop word removal, Stemming and Lemmatization, for example.

The way text preprocessing is done depends on the task at hand, as all techniques have their own characteristics, advantages and disadvantages, which means that all steps in the process should answer the question of “is important information being lost or is only irrelevant noise being taken away?” (Yse, 2021). In this way, the data quality is better enforced..

3.2.1 Noise Removal

Due to the widespread availability - and subsequent use – of user generated content from social media and e-commerce platforms, noise removal is gaining more attention in NLP tasks. These tend to be trained on very clean data, and thus may have problems when trying to generalize to interact with User Generated Content (UGC).

In the NLP field, noise is a non-standard content that can generally be categorized as either harmful or meaningful: in the first case it is unwanted and should be normalized (thus removed) because it affects the model negatively, while the second should be kept as it is a

content that enhances how the model will perform on unseen data – how it will generalize (Sharou et al, 2021).

It comes as no surprise that noise can carry meaning or intentions and may be useful to selectively transfer it to the output, “such as emojis in machine translation to preserve sentiment” (Sharou et al, 2021, p. 1). This means that this process is one in which the researcher should take into account the task the NLP system aims to perform and customize their choices as such.

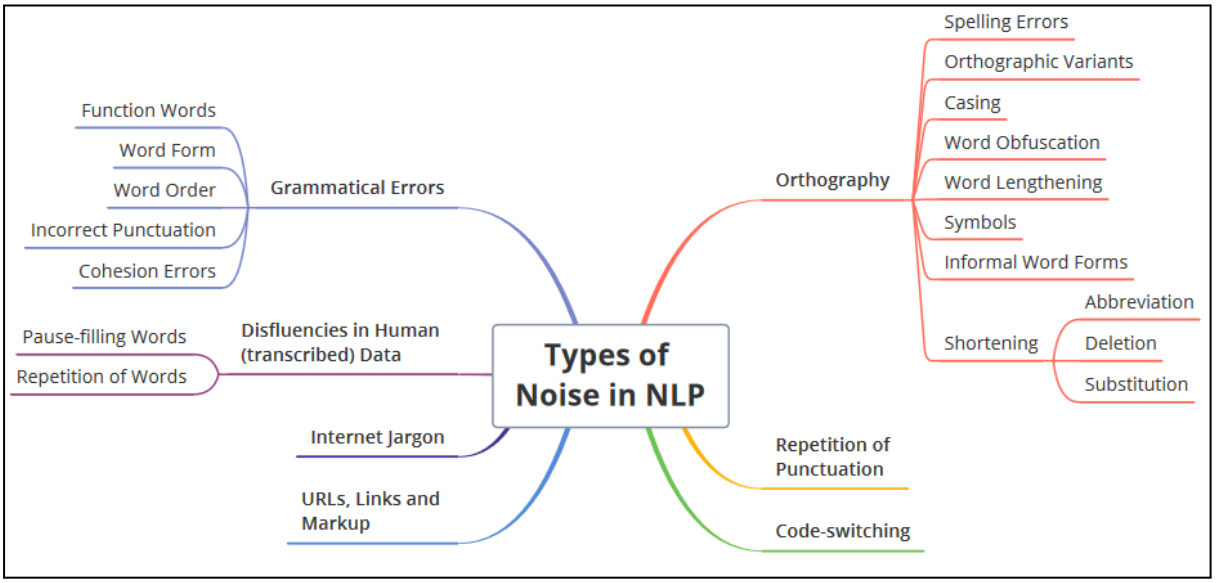


Figure 7. Possible types of noise in Machine Learning processes by Sharou et al (2021)

The author of the noise taxonomy seen in the figure above to provide the following overview:

- **Orthography**: relates to the way in which words are written and can be exemplified by simple, unintentional spelling errors, or the more complex cases of purposeful variation through the use of word lengthening or shortening and symbols.
- **Repetition of punctuation**: typically serve the purpose of emphasizing an emotional state or a meaning, like double exclamation marks. These can be important for sentiment analysis, for example.
- **Code-switching**: encapsulates the use of different languages within the same text sample, either words or entire sentences.
- **Grammatical errors**: related to incorrect composition of a sentence, exemplified through the use of wrong propositions or verbal conjugation.
- **Disfluencies in Human (transcribed) Data**: these are typical issues from spoken language, such as filler or repeating words.
- **Internet jargon**: refer to words not traditionally used in everyday conversations, but specific to the Internet realm, such as *upvote* and *downvote*.

- **URLs, Links and Markup**: can be preserved if referring to meaningful content, for example social media mentions using the @ character may denote an entity (that can be tagged or have content analyzed in regards to it), and hashtags on tweets can bring strong indications in Sentiment Analysis tasks.

All of these sources of noise can fall into a harmful case or a meaningful case, based on the task at hand, and as such may be evaluated individually to define the optimal strategy – there's no universal solution, which may represent an issue for systems that follow a pre-set pipeline.

3.2.2 Text Normalization

By normalizing the textual data, the goal is to reduce its randomness and bring it closer to a given standard – this should improve the efficiency of the process and the model, as it reduces the amount of different information the computer needs to parse through (Yse, 2021).

Noise removal tends to involve several steps, like "the removal of unwanted hashtags, emojis, URLs, special characters such as '&,' '!,' '$,' and so forth" (Campesato, 2021, p. 151). Some characters may typically be removed straight away, such as numbers, while others demand a bit more thought; taking into consideration the period ("."), for example, by removing it altogether from the text elements such as ellipsis (the use of three consecutive periods) will also be removed along with the potential meaning it brings to the text – it could be treated as a token during the tokenization task (Campesato, 2021).

This means punctuation can be treated as separate words and should also be considered based on the task to be undertaken: it is important for setting boundaries between words and sentences (through commas and periods) and identifying meanings (question marks and exclamation marks, for example) – essential concepts for Part-of-Speech tagging or speech synthesis tasks [64, pg 11].

Perhaps a more complex application is through spoken language applications. Oftentimes people use filler words and pauses, repeat or do not finish words – again, to decide whether to remove or keep them in the dataset depends on the application (Martin & Jurafsky, 2023). For example, filler words may indicate the beginning of a new sentence or a change of subject.

In the English language in particular, thought should be given to correct for contractions: these are combinations of words such as "*we'll*" in place of "*we will*" and "*don't*" instead of "*do not*". For standardization purposes, a typical solution to this is the creation of a dictionary of contractions and their full form expressions to correct their instances in the textual body before performing any other step (Yse, 2021).

Lastly, a very standard practice is to convert all characters into their lowercase counterparts, maintaining case uniformity; this task can be quite problematic depending on the language and if it uses accents or not. For Sentiment Analysis tasks, very positive or very negative sentences may be written in uppercase, usually in its entirety, and if removed would love its emphasis either way (Sharou et al, 2021).

As this work uses the English language, in which the use of accent marks is quite rare, the main issue brought forth could be lowercasing names and surnames and thus changing the entity to which it may refer (Campesato, 2021). This has impacts on techniques like Part-of-Speech or Named-Entity tagging (Yse, 2021) but should not be the case, however, of the processing of public product reviews in which the use of names and surnames is not particularly common due to privacy concerns.

3.2.3 Tokenization

The process of segmenting the text in chunks with which the system is built is called Tokenization: “in essence, it’s the task of cutting a text into pieces called *tokens*” (Yse, 2021), so words, subwords or even single characters. Ultimately, NLP systems will run with numerical data to understand patterns and calculate errors and losses, to be understood and processed meaningfully, which means that the textual data needs a guiding rule to be transformed into a numerical counterpart – text is transformed into tokens because that’s the level in which NLP models process it.

They are the basis for the vocabulary in which the models will be trained on, the set of unique elements contained in the data. So the same text can be subdivided in numerous ways, one different from the other, and result in models with varying characteristics.

The complexity of “splitting a sentence, paragraph, or document into its individual words” (Campesato, 2021, p. 152) can vary a lot depending on the language and the lexicon and may demand further processing: working with biomedical data, for instance, may involve dealing with several acronyms, which can the dealt with through named entity recognition (NER).

How the developer chooses to define a token – the building blocks of the system – is not always straightforward and affects how the document is represented, at the computer level, and will have consequences throughout the rest of the process (Dong & Liu, 2020; Khanna, 2022): decision such as if the punctuations were kept and they should now become tokens themselves, preserve hyphenated words in their entirety or not, how contractions were dealt with in previous steps (a common solution is to separate them, in such a way that wouldn't becomes two tokens – “would” and “n’t”), if tokenization should be done at word level or subword level and many others.

The techniques are explained by Khanna (2022) in the following sections:

3.2.3.1 Word-based tokenization

The most common solution used, tends to separate tokens as single words based on delimiters such as the blank spaces between them and punctuation marks (if present). Depending on the choice of delimiter, the tokens change: for example, by considering only blank spaces when punctuation was not removed from the text, a model can learn the same word as several different tokens such as “house”, “house!” and “house?”. This would be considered a problem, as it hinders the model and increases the computational complexity without any additional benefit.

A solution to this problem is setting a limitation to the vocabulary created: instead of using the totality of the tokens found, the developer may choose to keep only a select number of

the most common tokens in the given corpus and discard the rest. However, the number of tokens kept should be representative of the data, as discarding part of it means loss of data and information.

Lastly, misspellings can constitute an issue: if the corpus contains the word “house” and a misspelled version of it, “houes”, they will be seen and considered as different words.

3.2.3.2 Character-based tokenization

Because a given language may have many different words, but a small set of characters (letters, punctuation marks) with which to form all of them, character-based tokenizers were conceptualized. The idea is to separate the text at the level of its individual characters and the result is a very small vocabulary – fewer tokens than if compared to Word-base tokenization.

One of the main advantages of this technique is the possibility to represent even previously unknown words, ones that were not present in the training corpus, and a very small set of Out-of-Vocabulary words. Even further, misspelled words can be understood so as to not lose information so this technique can save on both memory and complexity.

However, a character typically does not carry the same amount of information, or meaning, as an entire word does, and increases the number of tokens used to represent a single element (for example, the word *house* would have five tokens instead of being only one).

3.2.3.3 Sub-word based tokenization

The midway point between the two previous techniques, the main idea of sub-word based tokenization is to solve the problem posed by very large vocabulary sizes (tokenization at word level) and by loss of meaningful tokens (tokenization at character level). The goal is to allow the model a manageable sized vocabulary whilst maintaining a deeper understanding of meaningful representations.

The solution relies on reducing the number of divisions performed on common words, and splitting rare words into smaller, more meaningful subwords. The example given by the author is “boy” and “boys”: the first remains the same, but the second is divided into two tokens – the root word “boy” and its plural indicator “s”.

This technique teaches the model to find similarities between words with the same root and commonalities in the use of given suffixes. Different NLP models may have their own particular way of performing sub-word based tokenization, especially ones that have been noted as State of the Art; one such example is the BERT model, released in 2018 by Jacob Devlin and colleagues at Google: the tokens created receive different notations for root words and suffixes, so the model is better able to understand their place and relationships.

3.2.4 Stop word removal

Not all words are important for conveying the meaning of a given sentence. These are called stopwords: although omitting them creates sentences that are syntactically incorrect, the meaning of the sentence remains unchanged – to the point that even text representation

techniques like Bag of Words and TF-IDF can still work well without them (Campesato, 2021).

These are words that are very commonly used, and as such carry no particular meaning by themselves – as such, they can be removed with minimal loss to the model to better prioritize the important information.

For the English language, for example, these words may include “a”, “an”, “the” and so on, however there’s no one single list of stopwords to use per idiom and the use of different toolkits may render differences in the output (Campesato, 2021).

As is prevalent in NLP tasks, the removal of stopwords is dependent on the application: for Sentiment Analysis, for example, their removal may mean significant loss of information, while text classification tasks tend to not rely too much on stopwords and are supported by other words present in the text (Khanna, 2022b).

3.2.5 Stemming and Lemmatization

There are two typical techniques for reducing words into their quintessential form and meaning, both of which produce slightly different results: Stemming and Lemmatization.

As explained by Bengfort et al (2018, p. 72), “stemming uses a series of rules – or a model – to slice a string into a smaller substring. The goal is to remove word affixes (particularly suffixes) that modify meaning”, and in doing so group related words together, even if the root is not standardized in a dictionary. The example provided is the plural of Latin languages: by removing the “s” or “es” suffixes, the word can be converted into its root form. Also through Stemming, the words “connection”, “connected” and “connecting” would all become the same root “connect” (Yse, 2021).

Stemming may however create words that do not exist as such. This can happen due to Over-stemming, where a too large part of a token is removed and so grouped with other incorrectly (for instance, “universe” and “university” may become “univers”), or due to Under-stemming, where two or more words are reduced to more than one root, when they should have been reduced to a unique root (for example, “data” and “datum” stemmed into “dat” and “datu”, instead of “dat” for both) (Yse, 2021).

Two such Stemming algorithms used for the English language are *Lancaster Stemmer*, *Porter's algorithm* and its improved version, the *Snowball Stemmer*.

Lemmatization, on the other hand, is based on a dictionary (Bengfort et al, 2018): the technique checks each token against an predefined list and returns its lemma, its canonical base word. This means that this technique tends to better handle irregular cases and similar tokens with different POS tags. The author points out, for instance, that “the verb 'gardening' should be lemmatized to 'to garden', while the nouns 'garden' and 'gardener' are both different lemmas.” If done through Stemming, all three tokens would have been turned into “garden”.

The lemma, then, is “a set of lexical forms having the same stem, the same major part-of-speech, and the same word sense” (Martin & Jurafsky, 2023, p. 11), and as such groups tokens together not only based on their composition but also based on their meaning

and uses. This means that very complex languages tend to be better manipulated through Lemmatization, such as Arabic.

A very widespread Lemmatization algorithm is called WordNet.

The performance of the technique can even be improved by providing the context in which the token exists, typically done through POS tagging (Yse, 2021). This method assigns each word in a given sentence the role it represents, such as noun, verb, adjective and so on. It is discussed in further sections.

In sum, each technique has its own advantages and disadvantages. For instance, Lemmatization can be less complex, but much faster, as it requires only the splicing of string variables, while Stemming is slower in all of its effectiveness, as it needs to verify each token against a pre-existing dictionary or database (Bengfort et al, 2018).

3.3 Language Modeling and Text representation

“Converting a piece of text into a feature vector or other representation that makes its most salient and important features available is an important part of data-driven approaches to text processing” (Pang & Lee, 2008, p. 20). This process begins with language modeling and text representation.

The natural awareness a person has about the placement and use of words in a given sentence is called Language Intuition – and is not taught explicitly (Campesato, 2021). This knowledge depends on language being predictable, and is imparted through repeated interaction with a particular language and its grammar rules, vocabulary, synonyms and numerous other factors. In English, for example, this means knowing that the phrase “I live in a…” will most likely be completed by nouns such as “house” or “apartment”.

This is not the case for computers, as they do not see and understand natural language as humans do – and so it needs to be translated and taught through a gamma of Text Representation techniques. The main idea is to find the optimal way to capture the important information and relationships within the textual data and make it accessible to the computer using vectors, matrices or embedding.

This process through which a developer can impart Language Intuition onto an NLP model is called Language Modeling. It describes how one chooses to represent the text data to give the model the capacity to “describe language and make inferences based on that description” (Campesato, 2021, p. 204) and can be done precisely because of the predictability characteristics of human language. [67, pg 9]

There can be several ways to represent the text to bridge the gap between human language and computer algorithms, like single words, small sequences of words (such as n-grams), entire sentences or even paragraphs, and the end result is called a language model (Campesato, 2021). When ready, the model is nothing more than a “probability distribution for sequences of words” able to “take as input an incomplete phrase and infer the subsequent words most likely to complete the utterance” (Bengfort et al, 2018, p 8). In sum, it allows for a machine to predict what comes next in a given sentence.

This capability is directly related to several NLP-based tasks, such as Text Generation – quite notorious now with recent breakthroughs on Large Language Models (LLMs), Language Translation, Text Summarizations and even Sentiment Analysis. In the latter case, developing an SA language model means teaching it to associate words, phrases and contexts with different sentiment categories, such as Positive, Neutral and Negative; it then may use the patterns learned to classify new inputs.

There exist some challenges to this process, however, one of which is data sparsity – if the model learns through examples, it can be quite difficult and computationally expensive to gather a meaningful sized dataset of possible configurations of human language. These issues are typically tackled through the choice of how the text is represented (Campesato, 2021).

The text representation can be done through Discrete text embedding techniques (such as Bag of Words, n-grams and TF-IDF), Distributional text embedding (such as Embedding Layer, Word2Vec and GloVe) or Contextual text embedding (using the transformer architecture), discussed in sequence.

3.3.1 Bag of Words (BOW)

The BOW method is quite simple: it transforms the textual data into a vector "based on a dictionary of unique words that appear in a document, and generates an array with the number of occurrences in the document of each dictionary word." (Campesato, 2021, p. 175). In other words, BOW builds a vector based on the occurrence of single words within a text, and usually associates them with the frequency in which they appear.

To do so, Bengfort et al (2018) explain that every document (so every single body of text, like a single customer review within a corpus of many) is represented as a vector whose length is the same as the total vocabulary contained in the corpus. The spaces within each vector are then used to specify the relationship between their document counterpart and the vocabulary: typically through the frequency of each token, however other representations such as one-hot or TF-IDF (to be discussed further) are also possible.

To better conceptualize this idea, the author Campeato (2021, p. 175) explains the process using "*This is a short sentence*" as an example and marking word frequency: "the corresponding 1x5 vector for the dictionary is (this, is, a, short, sentence). Hence, the phrase 'This sentence' is encoded as the vector (1, 0, 0, 0, 1)." It shows clearly how to build the document vector and its relationship to the original vocabulary, but it also provides evidence of how word order is lost and so is the context in which they are used.

In the end, each document will have its own unique representation.

Regarding the word representation associated within a vector, token frequency is quite common but leaves out grammar and word order, and tends to have a Zipfian distribution that places more significance on tokens that occur more often. This may impact the model, especially if it was built to deal with Normal distributions, such as Generalized Linear Models (Bengfort et al, 2018). For this reason, different representations were defined, one of which is called One-hot Encoding.

It is a “boolean vector encoding method that marks a particular vector index with a value of true (1) if the token exists in the document and false (0) if it does not” (Bengfort et al, 2018, p. 59). Essentially, this representation marks the presence/absence of a given token within a document, and can be important in some applications. For classification purposes, for example, the simple presence or absence of a given predictor may be more important than the number of times it is used in a document (Shmueli et al, 2019).

One-hot encoding tends to be more effective for small documents – such as tweets, simplifying each document into its core components, and finds special application in artificial neural networks due with discrete input activation functions (Bengfort et al, 2018).

The BOW method tends to be used for its simplicity and can be seen as an unigram, an individual token n-gram in which n = 1 (Campesato, 2021). However, it loses important information as it does not keep track of the context in which the words are written or the length of a particular document, and the final vectors generated can become extremely sparse if the vocabulary is too large (Bengfort, 2018; Campesato, 2021). The latter impacts the speed and performance of the model, and can be mitigated. “For very large corpora, it is recommended to use the Scikit-Learn HashingVectorizer, it uses very low memory and scales to large datasets as it does not need to store the entire vocabulary and it is faster to pickle and fit since there is no state” (Bengfort, 2018, p. 59).

An extension of the BOW method, in the interest of capturing word order and context, is the n-gram.

3.3.2 N-grams

Another vector-based text representation method is the n-gram, which aims to prioritize a comprehensive vocabulary that’s still manageable. The idea is still transforming the textual data into a word vector of integer values, however the tokens are no longer evaluated singularly as unigrams, but sequences of n adjacent tokens.

This means that instead of seeing each token – typically words – on their own, they are gathered in an ordered sequence of number n, and so retain word positioning (Campesato, 2021). The choice of n is usually done through cross validation and will determine the size of the sequence. For example: for the same phrase used previously “*This is a short sentence*”, a bigram representation (n = 2) would be “This is” and “is a” and so on. Conversely, a trigram (n = 3) would define “This is a” and “is a short”, and so on.

Within the context of Sentiment Analysis tasks, for example, authors Pang & Lee (2008, p. 32) highlight the importance of token position to the overall sentiment or subjectivity of a text; they explain the focus on discourse structure using the example of customer reviews and its contrast to topic-based summarization tasks in which “the beginnings of articles usually serve as strong baselines in terms of summarizing the objective information in them, the last n sentences of a review have been shown to serve as a much better summary of the overall sentiment of the document than the first n sentences”.

The point, however, is to define the order of the n-gram – an ongoing debate according to the author. They mention two works as example, the first conducted by Pang et al, yielded better results using unigrams over bigrams when representing text when classifying the

polarity of movie reviews, while the second by Dave et al saw better results when pairing bigrams and trigrams for the same task.

Compared to the BOW method, the n-gram vectors tend to be shorter and the model trained is better able to predict associations between words and themes, and is better prepared to predict how a given sentence will be written. One point to keep in mind, however, is that the more n increases, the less repeating n-grams will be found: this drastically reduces the number of equal instances occurring in various contexts the model can train on and recognize.

Another point to be careful with is that the choice of n will affect the bias-variance tradeoff, seen in previous sections. Authors Benggort et al (2018, p. 134) sustain that “a small n leads to a simpler (weaker) model, therefore causing more error due to bias. A larger n leads to a more complex model (a higher-order model), thus causing more error due to variance”. Within the context of supervised machine learning applications, the right balance needs to be observed for the model to have acceptable results on new, unseen data.

Now, because the goal is to consider also the context, punctuation can be used to define boundaries: sentences represent discrete ideas, and as such n-grams can be used to identify combinations of tokens that start or end them (Bengfort et al, 2018).

Some common applications are seen in the auto-completion of phrases – such as the one found in emails, auto spelling verification and even voice-based personal assistant bots: n-grams can teach the difference between a question and a request, for instance (Srinidhi, 2021). But although this method can be simple to implement and yield good results, it has since been surpassed by Neural Language Models for performance as they are better able to capture and approximate information (Li et al, 2022). The study mentioned goes on to evaluate and discuss if n-grams are still relevant today, citing its interpretability and lower training cost as arguments to conclude it is so.

Another common application of the n-gram method is as a tool to embed algorithms (Campesato, 2021), such as Word2Vec discussed ahead in section 3.3.4. This pairing should provide the model with both the short-term word dependencies and context found by n-grams and their semantic relationships provided by Word2Vec. One should keep in mind, however, the complexity added by pairing methods, and so the hyperparameters (such as the choice of n) of each should be taken into consideration very carefully.

3.3.3 TF-IDF

As mentioned previously, the TF-IDF algorithm can be seen as an improved BOW representation, as it “takes into account the number of occurrences of a given word in each document as well as the number of documents that contain that word” (Campesato, 2021, p. 80). It follows the principle that more valuable information is shown by words that appear more rarely in a document, and so beyond just indicating frequency or presence/absence, the method can evaluate the relative importance of a given token inside a document and the corpus. In other words, the TF-IDF technique prioritizes the relevance of a term over its frequency.

TF represents the concept of Term Frequency, so how many instances of a particular token exist inside of a document; this information can be used to compare documents amongst

each other and find which pairs are more similar to one another. IDF, on the other hand, represents the Inverse Document Frequency, so how rare a given token is across all documents inside of a corpus; it is calculated logarithmically so as to avoid bias from longer documents or terms that appear much more frequently than others in an exponential tendency (Campesato, 2021).

The combination of both values is used to calculate the TF-IDF score, which is always greater than or equal to zero. This means that the closer this score is to 1, the more important and informative the given term is in relation to the document. The opposite is true for values closer to zero. A TF-IDF equal to zero, for example, means that the term appears in all documents of the corpus.

Supposing a given Term Frequency $tf(t, d)$ of a token t inside of its given document d, Bengfort et al (2018, p. 65) explain the method through the following formulas:

$$tf(t, d) = 1 + log f_{t,d} \quad (21)$$

$$idf(t, D) = log 1 + \frac{N}{n_t} \quad (22)$$

Where $tf(t, d)$ calculated through (21) is subsequently used to calculate the term's IDF value in regards to the total set of documents D through (22), N is the number of documents and n_t represents the total frequency of the term considering all documents.

Lastly, the TF-IDF score is calculated through the multiplication of both factors, as in:

$$tfidf(t, d, D) = tf(t, d) \,.\, idf(t, D) \quad (23)$$

After all values are calculated for the vocabulary (the tokens) inside the corpus, "the words are sorted in decreasing order, based on their tf-idf value, and then the highest scoring words are selected". (Campesato, 2021, p. 182). The number of terms to be used is a hyperparameter chosen by the researcher.

One of the benefits of using TF-IDF is that "it naturally addresses the problem of stopwords, those words most likely to appear in all documents in the corpus, and thus will accrue very small weights under this encoding scheme" (Bengfort, 2018, p. 65) This works along the lines of setting a higher importance on rare words, which lends itself well to most text analytics activities.

Typically, this method is used for tasks such as Document Ranking or Classification, Clustering and Information Retrieval, and it is important to note that it captures presence and relevance of terms, but not the semantic value brought by them.

The TF-IDF method tends to be quite fast, but one needs to keep in mind that it works better when dealing with single words, instead of a phrase, for instance. In that case, partial matches can be accepted, but again they allow for a large leeway of errors and loss of relevance (Roldòs, 2020). For more effective results, methods such as Word Embedding can work better.

3.3.4 Word Embedding

The applications of this technique are in the same range: Text Classification and Document Clustering, depending on if the data is labeled or unlabeled. That is the main reason for its comparisons to One-hot BOW and TF-IDF. In the first case, for instance, author Campesato (2021) points out that Word Embeddings are able to reduce the vectors into smaller versions, while carrying more valuable information.

As for TF-IDF, the author provides the task of differentiating between the words “tigers” and “lions” inside of a document: both terms are related to the same topic, wild animals, however “tf-idf values for these two documents will not determine that the documents are similar: doing so involves a distributed representation (such as doc2vec) for the word embeddings of the words in the two documents”.

Instead of defining a given word through a value, such as BOW and TF-IDF, Word Embedding represents it through a vector.

Word embedding techniques have revolutionized the field of NLP by allowing models a deeper understanding on meaning and context, and representing words with similar meanings in a similar fashion (Brownlee, 2019). The idea here is to represent words as dense vectors (as they work more efficiently than their sparse counterparts seen previously) in a high-dimensional space. Each of these dimensions should represent given characteristics, and so similar words will have similar vector representations – which provides the model deeper information about the relationship between words and the context in which they are used (Bengfort et al, 2018). Provides the example by saying that the technique “attempts to inherit the semantic properties of words such that ‘red’ and ‘colorful’ are more similar to each other than they are to ‘river’ or ‘governance’”.

The similarity among words is typically captured through the backpropagation method when training the Neural Network: the model will assign similar activation values, similar weights for words used in similar contexts. In the end, this will be reflected in the embedding space created for the dataset, with very dissimilar terms far away from each other, and similar ones, close. Not only that, but the use cases of words can also be captured, meaning that the use of the same term for positive phrases, compliments, can be set apart from its use with sarcastic and ironic phrases – and and so can their connotations.

It can be quite a complex process. However, there are many pre-trained word embeddings that are publicly available for use, meaning that it is not typically developed by scratch every time one develops a language model. There are 3 main possibilities to discuss: Embedding Layer – which foregoes pre-trained algorithms, Word2Vec and GloVe.

3.3.4.1 Embedding Layer

The first, Embedding Layer, is quite straightforward and works when the model to be trained is based on Neural Networks, meaning that the word embedding is not an input given to the model – but learned jointly with the model itself in a one-hot encoded configuration (Brownlee, 2019). It can be quite demanding and slow, requiring very large amounts of clean data, however it will create a word embedding specific to the text at hand.

3.3.4.2 Word2Vec

The second, Word2Vec, is a pre-trained word embedding model, created by a team of researchers at Google under the direction of Tomás Mikolov. It is a very efficient way of creating the vectors in the embedding space for representing a body of text, composed of floating point values, and has since become an industry standard for NLP applications.

Similar to Embedding Layer, Word2Vec "involves a neural network consisting of an input layer, a hidden layer (with no activation function), and an output layer that has the same dimension as the input layer" (Campesato, 2021, p. 191). So the purpose and method are the same, the difference lies in the fact that Embedding Layer is done almost manually by the researcher, as a part of the Neural Network's training itself, a component of if, while Word2Vec is a specific algorithm for creating word embeddings, already pre-trained, that can be tuned to fit a particular task.

The main idea here is to make predictions instead of counting words and, again, capturing the context of a term by the words that happen on either side of it: the preceding and succeeding words (Campesato, 2021). Not only that, it works on vector math to find the relationship among different terms: author Brownlee (2019) provides the example of subtracting "*man-ness*" from the term "*King*" and adding "*women-ness*" to find the term "*Queen*" – signifying the relationship of "*king is to queen as man is to woman*".

The Word2Vec algorithm can be trained through two different word representations techniques: Continuous Bag of Words (CBOW) or skip-grams, both of which learn based on a term's neighboring words. While the CBOW technique model "learns the embedding by predicting the current word based on its context, the continuous skip-gram model learns by predicting the surrounding words given a current word" (Brownlee, 2019). The idea is represented in Figure 8.

The idea is that the CBOW architecture "starts with a set of surrounding words and then attempts to predict the target word which [...] involves a feed forward neural network that determines word embeddings" (Campesato, 2021, p. 195). Because of its configuration, the author explains, with input and output layers having the same size, it creates more compact word embeddings. It tends to work better for smaller datasets and be more computationally efficient than its counterpart.

On the other hand, skip-grams work in the opposite direction, trying to infer the missing terms that are most likely to appear around a given word; in its architecture there is no bias term and no activation function between input layer and the single hidden layer, only a Softmax activation function (to generate the probability distribution using the number 1 to set the activation) between the latter and its output layer (Campesato, 2021). The skip-gram

method tends to work better than CBOW for words that appear rarely in the text and provides a more nuanced result.

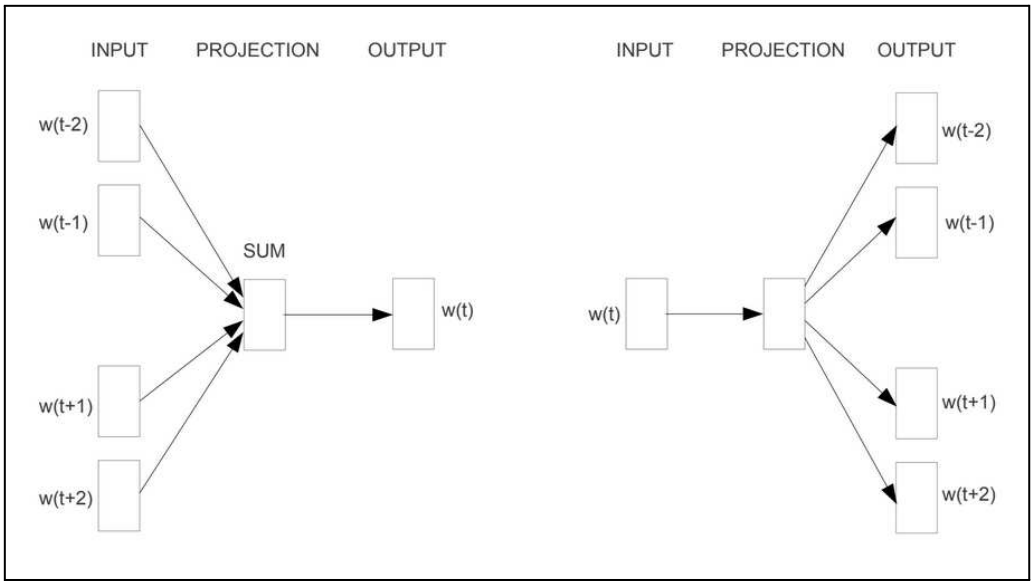


Figure 8. Word2Vec training models by Mikolov et al (2013).

Finally, the main benefit of using this algorithm is its efficiency in terms of space and time requirements, allowing for the development of larger word embeddings from larger bodies of text Brownlee (2019). The technique is not all-encompassing, however, providing only one word embedding per word, and does present some limitations, amongst which is its shallow neural network with one hidden layer and the fact that fine tuning is not possible – these have now been solved through the use of the Attention-based Mechanism, which allows for the representation of a single word as different vectors based on context (Campesato, 2021).

3.3.4.3 GloVe

In the same line as Embedding Layer and Word2Vec, GloVe is “limited to one word embedding for every word, which means that a word that’s used with two or more different contexts will have the same embedding for every occurrence of that word” (Campesato, 2021, p. 186).

Developed by researchers at the University of Stanford, it tries to join “both the global statistics of matrix factorization techniques like LSA with the local context-based learning in Word2Vec” (Brownlee, 2019).

This technique works by calculating a co-occurrence matrix of terms that happen within a given context, and decomposing it into a global matrix – meaning it relies more heavily on terms that happen frequently together than on the local context itself (Campesato, 2021). The author specifies these two steps as:

1. Construction of the co-occurrence matrix of dimensionality equal to words x context.

2. Factoring the matrix into a matrix of dimensionality equal to word x features.

In the first matrix, the number of rows set by context represents all the instances the given term has appeared in the corpus – meaning its frequency. These are then grouped into a single vector per term into the global matrix, and so it has a lower dimensionality (Campesato, 2021).

GloVe is a statistics-based technique, instead of a Machine Learning-based one as Embedding Layer and Word2Vec (Brownlee, 2019), and despite being more efficient with a great capability to capture a deeper understanding of a given term's meaning, it has important limitations: it does not support Out-of-Vocabulary words nor polysemy (words with several meanings based on their context) (Campesato, 2021).

3.3.5 Transformer-based Embedding

As mentioned previously, attention-based mechanisms such as Transformer-based Embedding architectures have surpassed in recent years the techniques mentioned previously, especially Neural Network based ones. Differently from Word2Vec and GloVe, attention-based mechanisms take into account all of the words present in a sentence, meaning that each time a given term is used, it will have a different embedding (Campesato, 2021).

Another novelty is that the word order is registered and kept track of: after word embedding, the technique applies positional encoding as an additional step; the sum of both of these vectors is used as input for the Transformer (Vetsch, 2022).

Traditional Transformer-based architectures use a encoder/decoder model (Vetsch, 2022), also called attention layers, and are generally constructed of an "encoder component that contains six 'sub' encoders, as well as a decoder component that also contains six 'sub' decoders" (Campesato, 2021, p. 462). These, however, can be considered a hyperparameter during training and be tuned as needed.

The attention layers are coordinated through a feed-forward network to process the input, meaning the data does not circle backwards (Brownlee, 2019).

3.4 Named Entity Recognition (NER)

Another part of NLP text preprocessing tasks, now as a subfield of Information Extraction, is called Named Entity Recognition (NER). The idea is to handle a tokenized body of text and identify its compositional elements, to finally classify them based on predefined groups, to indicate people/users, places, organizations, dates and many others (Vetsch, 2022). The categories can be customized by the developer to suit the task at hand.

NER can actually assist with the transformation of unstructured data into structured data, although it can be sensitive to too few or too many tokens during the process. The result is a compilation of named entities, so "a word or a phrase that distinguishes one 'item' from other items in a corpus" (Campesato, 2021, p. 160). This entity, in its own right, can be a single token or a combination thereof: Anandika & Mishra (2019) provide the example of "*The Great Lakes*": it is a single entity, a geographical location, composed of three tokens.

However, the classification is only as good as the dataset is relevant to the task: language can be very field or task specific and, as seen earlier, may change over time; for example, training a NER model with twitter data will not work well for classifying entities in a scientific journal, and so on. This factor may represent a problem, as some languages or fields can be more challenging to gather data on.

Other difficulties a NER model can find relate to Nested Entities and Ambiguity: the former is related to entities hidden inside other entities, which makes their detection particularly tricky, while the latter relates to the instances in which the same term refers to more than one entity – the example given here by the author is the word "*Jordan*", as it can be either a name (person) or a river (location) (Goyal et al, 2018).

There are currently four main NER techniques, discussed in further detail in the following sections, that can be used a myriad of tasks, not only Information Extraction, but also Text Understanding (aiding the comprehension of relationships among entities), Document Summarization (capture information about the most important entities), Search & Retrieval, and even sentiment Analysis (naming entities such as companies, brands and products), among many others.

3.4.1 Rule-based NER

Relying on specific rules typically written by a linguistic expert, these earlier methods do not demand the use of labeled data. The method is also known as the Linguistic Approach, and the rules tend to be hand-crafted based on Grammatical, Syntactic, Orthographic or Dictionary rules (Campesato, 2021; Vetsch, 2022).

The rule-based NER algorithm then looks for patterns that match the rules inside the corpus to identify potential candidates for given categories. Although this method can be more expensive, it works very well for field-specific tasks or limited data – such as fields and languages where it is difficult to acquire a large amount of high-quality data. The issues reside in tasks that need to be scalable or encompass too wide a subject.

3.4.2 Feature-based supervised learning NER

ML methodologies tend to be more advantageous for NER, as it is essentially a classification task, and the field has many algorithms that deal really well with them. The idea of Feature-based supervised learning is to follow the usual pipeline for training a classification ML model: based on a subset of labeled corpus data, it is trained to identify the entities within it; according to the occurrences found, the model calculates the probability of a given category based on its context (Srinidhi, 2021)

During training, the features need to be chosen carefully. Authors Goyal et al (2018, p. 25) classify the feature space into three categories: List Lookup (based on resources like lexicons and dictionaries), Document and Corpus features (based on the document's structure and content) or Word-based features (the orthographical and contextual features of the corpus).

There are many ML models available for NER, "such as hidden Markov models (HMM), decision trees, maximum entropy models, support vector machines (SVM), and conditional random fields (CRF)" (Campesato, 2021, p 161).

3.4.3 Unsupervised learning NER

As this method does not require labeled data, it is quite convenient when high quality annotated data is very difficult or expensive to acquire. The idea is to train a model “that considers the structural and distributional features of data to find more learning about the data” (Goyal et al, 2018, p. 25).

Typically, there are two methods: the first works through clustering, applying distributional statistics to group similar entities based on the context in which they appear, and the second works through an association rules-based approach, where the model looks for patterns within a very large dataset to find patterns and similarities (Goyal et al, 2018).

3.4.4 Deep learning NER

Deep learning based NER applications require a large amount of annotated data, which may be difficult or expensive to acquire, though they tend to be much more accurate and time efficient (Campesato, 2021; GeeksforGeeks, 2022). This is the main reason why this work does not undertake any deep learning approach and as such does not go into further detail.

3.5 Syntax and Parsing

The last essential topic regarding the structural and grammatical analysis of textual sentences depend on two essential concepts: Syntax and Parsing. Mainly because these two fields have an important role in the relationship between words and how sentences are structured.

As discussed in Section 3.1.1, on the challenges faced by the NLP field, syntax deals with the rules set by a given language to structure a sentence as an independent unit. This includes “the word order, the dependency relationships between these words and, in some languages, the relationships of agreement as well as the case marking” (Goyal et al, 2018, p. 127). It is part of how the patterns and regularities that occur within a language are governed and thus can be used as the basis to explain human language to computers.

In sum, the Syntax guides the linguistic form of sentences, without placing too much importance on the meaning behind them – the latter is regulated by the field of semantics.

Parsing (or syntax analysis), on the other hand, “consists of the decomposition of sentences in major syntactic units and of the identification of dependency relationships” (Goyal et al, 2018, p. 129-130). This means identifying the phrases, clauses and grammatical roles of the words, and is usually done in a graphical form, such as a tree – thus parsing can also be a great asset for providing explanations, especially for shorter pieces of text or “serve as a basis for modeling valence shifters such as negation, intensifiers, and diminishers” (Pang & Lee, 2008, p. 22).

So while the Syntax encompasses the actual rules that guide the structure of sentences within a language, Parsing represents the evaluation of the sentences based on these rules.

A very important concept that connects Syntax and its subsequent analysis for NLP tasks is called Part of Speech (POS): they indicate the function, the role a given term develops within a sentence – meaning nouns, verbs, adjectives, conjunctions and so on. By categorizing the

tokens, the analysis of the text becomes easier and serves as a starting point for specific tasks. One such example is the improvement of Search Function; by signaling a term as nouns or adjectives, which tend to be more semantically representative and may better characterize texts than verbs or pronouns, algorithms are more well equipped for information retrieval (Goyal et al, 2018).

3.5.1 Part-of-speech Tagging

This process of signaling each token based on its grammatical role within an entire corpus is called POS tagging. These tags tend to vary through different languages, meaning there is no one true standard for each one: instead, tagset have been developed and some of them are more widely used than others (Goyal et al, 2018). The author provides as an example the Penn Treebank tagset for English, which has more than 30 different tags – including cardinal numbers and foreign words – and the Xerox tagger1, which goes beyond 70 such tags.

Typically, there can be as many tags as needed, as the grammatical roles can be even further specified. A noun, for instance, is said to be Common when referring to general classes of people, places and objects, but Proper when speaking about specific people, places and objects.

Even if it may seem simple, POS tagging can be quite a complex task. The main issue found is ambiguity, because the same word (or more specifically graphic form) can take the role of multiple parts of speech in any given language (Goyal et al, 2018). The word "*heat*" can serve as either a noun or a verb, while "*large*" can serve as either an adjective, a noun or a verb.

Beyond the simple number of tags chosen to classify it, the token itself may be either variable or invariable, meaning it can take on more than one form: depending on gender – if the language itself uses them, like the Latin languages – or plural form. For example, in English adjectives tend to be invariable, so in the phrase "*the eggs are blue*" the adjective "*blue*" would remain the same if there was only one egg, while in French it would be different depending on the gender of the subject and its quantity.

Other issues found in the process of POS tagging relate to spelling errors or grammatical errors typically found in user generated context, like written texts or transcribed spoken language, and new words that were not present in the training corpus. These can easily trip speech taggers and may make it necessary to use strategies such as contextual heuristics, for example (Bengfort et al, 2018; Goyal et al, 2018)

In sum, through the rules set by the Syntax of a given language, tags can be assigned to represent the role taken by each token inside of a corpus to facilitate NLP tasks or subtasks – one of which is parsing.

According to Campesato (2021), POS tagging can be done through four methodologies:

1. Lexical based: sets the tag by the most frequently occurring use-case.
2. Rule based: coordinated through the pattern set by grammatical rules – such as an "s" at the end of the word signals a plural. This is an obvious example for the main

issue of this methodology: the rules are strongly related to the language in question (German and Italian, for example, may use the letter “e” for signaling plural) and tend to not handle exceptions very well.

3. Probabilistic method: assigns the tag based on the likelihood that a token with a particular tag will occur in the phrase.

4. Deep learning: as the name indicates, this method uses deep learning algorithms such as RNN to define the tags based on contextual use.

As for Parsing, Khurana (2022, p. 3717) explains that after using POS tagging “at lexical level, words are grouped to phrases and phrases are grouped to form clauses and then phrases are combined to sentences at syntactic level.” This means that the analysis goes beyond the lexical level and more information can be glimpsed; for example, changing the word order can affect the comprehension of a sentence, and parsing is the next step to reflect that beyond just the context of the word. For this step, there are typically two main subfields, discussed as follows.

3.5.2 Dependency Parsing

The idea behind syntactic analysis through dependency parsers is to first identify the main word inside of a phrase and subsequently look for links that modify it; this method results in an overlapping structure of arcs that represent the meaningful structures and relationships within a sentence (Dong & Liu, 2020). The author provides the visual example seen in Figure 9 below, using the phrase “*How many teaspoons are in a tablespoon?*”.

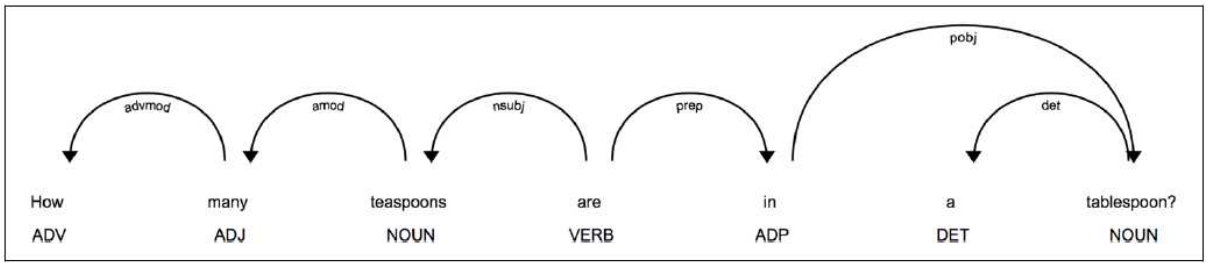


Figure 9. Logic of a dependency parsing by Dong & Liy (2020)

The example uses the module DisplaCy, found in the open source NLP library SpaCy, to define how the words relate to and modify each other: using the Universal POS tagset (written by linguists, rather than by developers), the main word in the sentence is “*are*”, a verb.

As such, it becomes clear that Dependency Parsing focuses on the relationships at the word level in a given sentence, and represents them as a graph in which each word is a node. As the name suggests, this method points out the dependency among the words that form a sentence.

This type of parser is quite popular due to its capacity to produce a quick and accurate grammatical analysis, however it may lack depth of information (Dong & Liu, 2020).

3.5.3 Constituency Parsing

On the other hand, the idea behind the Constituency Parsing method is to discover and define the hierarchical structure of a sentence, usually using a tree to represent the syntactic structure found – as seen in Figure 10.

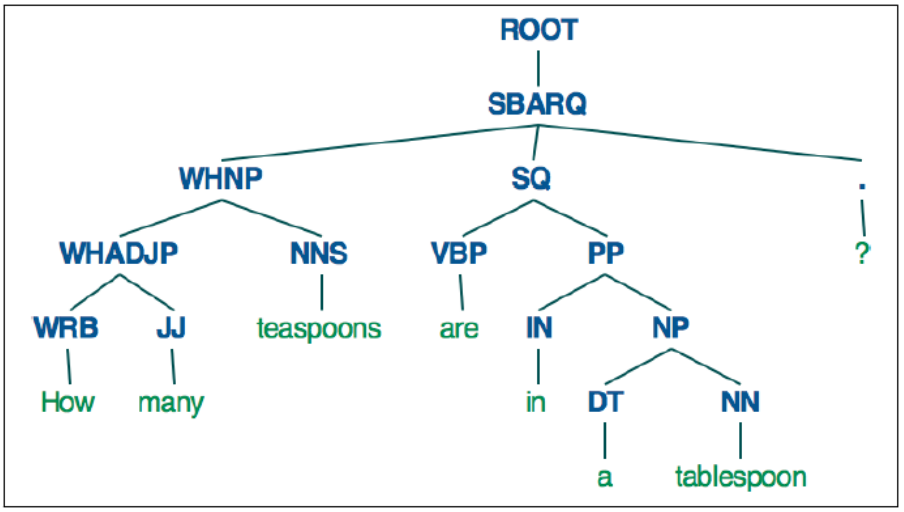


Figure 10. Logic of constituency parsing by Dong & Liu (2020)

Using the same example, the authors Dong & Liu (2020) use the Stanford CoreNLP package – Java-based NLP tools – to show how constituency parsers can find more complex interrelationships and nested structures: through tree traversal algorithms and without explicit relationship between nodes. There may be more than one way of building such a tree, however, due to the already discussed ambiguity.

The constituency parser trees are built with “terminal leaf nodes, the part-of-speech tag, and the word itself. The nonterminal nodes represent phrases that join the POS tags into related groupings” (Dong & Liu, 2020, p. 225). The focus here is the relationships that exist at a sentence level, so the root phrase found by the tool is signaled with the tag SBARQ (which represents as a direct question, because it begins with a “wh”-word). Other pertinent notations in this example are WHNP (a noun phrase using a “wh”-word) and SQ, the main clause of the root phrase.

It’s clear that this method of parsing provides more information than Dependency Parsing, and some care must be taken to evaluate its relevance to the taks. Although Constituency Parsing also suffers from ambiguity, the level of syntactic detail found can be very effective for identifying questions and extracting queryable data (Dong & Liu, 2020).

4

Sentiment Analysis

As briefly introduced in Section 1, Sentiment Analysis is now a widespread sub-task within the NLP field for teaching computer algorithms how to systematically evaluate, classify and extract subjective information from texts, and their related affective states (such as opinions and attitudes regarding a particular subject) (Lei & Liu, 2021). Authors Dong & Liu (2020, p. 13) point out that this popularity is due to its great capacity to pick up on the tone of text, which “can convey a lot of information about the subject’s perspective and lead to aggregate analyses of reviews, message polarity, or reactions”.

It belongs to the wider field of Information Extraction, which are typically solved as classification problems – either as end-goal or as a sub-task of defining an effective summary (Pang & Lee, 2008). This field of study has received many names, which this work uses interchangeably, such as Opinion Mining and Subjectivity Analysis. To a smaller extent, terms such as Review Mining or Appraisal Extraction can also be found in the literature. All of these have as their goal the use of computation treatment to discover “opinion, sentiment and subjectivity in text” (Pang & Liu, 2008, p. 5).

Authors Mäntylä et al (2018, p. 1) mention that the value of understanding other people’s thoughts and opinions is likely “as old as verbal communication itself”, citing the fact that historically, leaders have been interested in the opinion of subjects – either to control opposition and dissent or to boost their own popularity, even books have been written in either ancient Eastern and Western societies on the subject, such as the widely known classics “The Art of War" and “Iliad”. In ancient times, the voting method was defined as the tool to tally up people’s opinions on a myriad of public matters and communal life, while modern times began using questionnaires to do the same around the 20th century – to quantify and measure opinions.

However valuable and sought after, it is not always easy: “in general, sentiment and subjectivity are quite context-sensitive, and, at a coarser granularity, quite domain dependent” (Pang & Lee, 2008, p. 13). This means that people’s opinions, how they write

and what they say, how it will be interpreted and what it represents, as previously established, is entirely subjective and can vary greatly from one domain to the other.

This inherent subjectivity of this task is often classified in terms of binary polarity, meaning good and bad, or happy and unhappy, positive and negative (Lei & Liu, 2021). The main idea is that, given a piece of text that contains an opinion "about one single issue or item, classify the opinion as falling under one of two opposing sentiment polarities, or locate its position on the continuum between these two polarities" (Pang & Lee, 2008, p. 16). The author goes on to mention that since many of these tasks work on determining two opposing sides, they are particularly suited for binary classification Machine Learning techniques.

But the field can extract more details, if necessary, through multiclass classification. One way of doing so is through the addition of a neutral state when the text does not lean particularly strongly towards either polar side, while another is to substitute this scale altogether and use the emotions themselves as insight: happiness, anger, sadness, joy and so on.

The main tasks tackled by the SA nowadays are as follows.

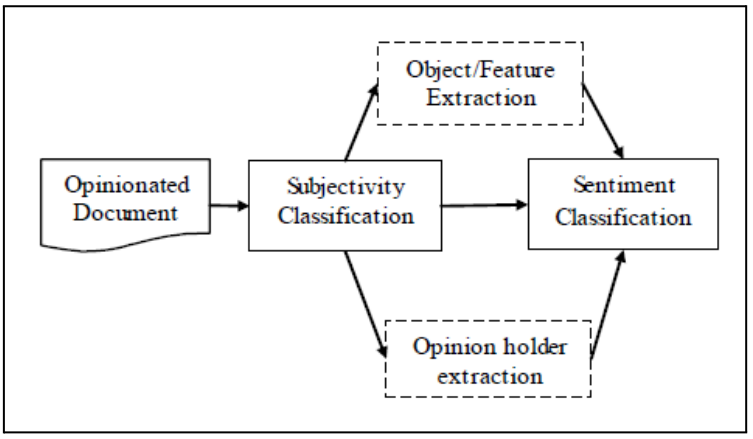


Figure 11. Sentiment Analysis tasks by Kumar & Sebastian (2012).

Following the acquisition of a corpus of opinionated data, SA can be used to perform:

1. Subjectivity Classification, which tries to identify the content of a document as opinionated (subjective, containing a point of view or emotion) or as the objective statement of a fact.

2. Sentiment Classification, when the document is opinionated, to discover its polarity (either binary or multiclass). Can encompass both classification and ranking of sentiment, and solves a huge plethora of questions such as "how positive is this text?", "rank these texts from most positive to most negative" and assigning labels "where is this text on a scale from between liberal and conservative political thinking"(Pang & Lee, 2008).

3. Subtask of Opinion Holder Extraction, to establish the source of the opinion, be it direct or indirect.

4. Subtask of Object/Feature Extraction, to discover the target entity, what or who the text is talking about.

These possibilities developed over time, as Sentiment Analysis as a field of study and its history are quite recent. Even if, at a push, the basic idea can be traced back many centuries ago, the scientific papers published on the subject were very few until two decades ago (Ahlgren, 2016). This late development can be attributed to factors such as the recent evolution and broadening of Machine Learning methods, along with the refinement of NLP techniques and the more ready availability of user generated data since the 1990s due to the development of the World Wide Web (mainly its evolution from 1.0 to 2.0) (Pang & Lee, 2008; Kumar & Sebastian, 2012).

Initially, the sparse corpus of published research on sentiment analysis of public opinion, according to Mäntylä et al (2018), dates back to the time of the Word War II, and was very political in nature; the modern understanding of the field started to gather attention and branch out its focus around mid-2000's, using user generated content online, to reach many other areas like finances and the prediction of the stock market and public sentiment of terrorist attacks.

Indeed, the Internet nowadays makes it possible to "find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics — that is, people we have never heard of" (Pang & Lee, 2008, p. 1). And the availability of data today is only getting bigger.

Not only that, but the data itself is also incredibly relevant: "the interest that individual users show in online opinions about products and services, and the potential influence such opinions wield" (Pang & Lee, 2008, p. 2) cannot be taken lightly.

For this reason, sentiment analysis research shifted its focus in the last couple of decades and was dominated by business applications on ecommerce and social media, deepening the need for methods that deal with nuance in user generated content: Mäntylä et al (2018) mention that this development led to the deepening of NLP techniques in general, as issues such as irony and sarcasm now became more prevalent in the datasets used, and furthered the fields' advancement towards more complex and nuanced emotion detection – instead of only its general polarity.

One solution found is mentioned by Ahlgren (2016, p. 2), saying that more advanced methods now "do not treat all words equally but assign more weight to important words depending on their position in the sentence." They go on to mention that even more sophisticated techniques will be developed for nuanced understanding as computational power becomes less expensive and NLP methods evolve. However, one of the main issues Sentiment Analysis has faced along its history as a field is how domain-specific its application tends to be, or how "one collection of words that is efficient for one domain most likely will not perform as well in another domain" (Ahlgren, 2016, p. 1).

4.1 Development of SA on Customer Reviews

After understanding what is the field of Sentiment Analysis, how it has developed over time and why businesses apply it, the workflow used to arrive at the analysis itself should be discussed and compared to other SA tasks.

Many practical applications of this were seen in Section 1.2, especially in the context of intelligence and how corporations use SA for assessing brand reputation, public opinion on products and services, recommendation systems and tracking trends to make more informed strategic decisions on how to stay relevant and ahead of competitors. More specifically, companies can gather user generated content – which has an incredible power to influence other consumers – and adapt their strategy, product development process, marketing message, branding positioning and much more to suit their needs. Doing so can be quite expensive and time consuming to undertake, because the qualified data needed is not always easy to gather and, if needed, label. And this is the first step into the process.

The typical sources used by businesses and organizations around the word for gathering data – typically user generated content – are:

1. Social media platforms, such as Twitter, Facebook and Instagram. Some of them may offer APIs (Application Programming Interface) specific for this task.

2. Product reviews or ratings, typically through e-commerce websites, a field in which Amazon.com is the most widely used due to its huge user-base and diversity of products. These tend to come with an indication if it is positive or negative, such a 5-star rating system.

3. Online forums and blogs, which contain vast amounts of information regarding to user's thoughts, especially non-product related, such as politics and people. These can be harder to process the subjective value, because it's typically written as free-form text.

4. News articles, to a lesser extent, however they may contain public comments on matters such as politics, health, science, economics and future trends.

5. Lastly, surveys and questionnaires can be created by businesses and shared with their clients for acquiring very targeted information.

One point of attention, however, is that some traditional sources of data – such as Amazon.com and Twitter – have started to introduce barriers against web crawlers and APIs that work as one, such as use-based payments and data quantity limitations, further complicating the collection of qualified data used by businesses.

In the context of academic research, on the other hand, nowadays there are many publicly available sentiment analysis datasets – gathered and maintained by communities such as Kaggle, for example. These are typically composed of labeled data regarding movie and product reviews or tweets, and on a smaller scale other social media posts and even news articles.

While this work is done within the context of research, the goal is to undertake the task of performing sentiment analysis on product reviews to analyze customer satisfaction and improvement points as it would happen in a business context. This is the reason why it was chosen to forego more easily accessible public datasets, but rather investing the time and resources to create a real-world data set under today's constraints, dealing with issues of labeling, quality, data quantity and sparsity and so on.

Thinking specifically about customer reviews about products, services or even events, the information they provide can range on many assorted aspects, from the most general overview to opinions on very specific features based on what each person considers important and valuable. This extraction and analysis of each single feature is a more standard information extraction task, but allows for a deeper understanding of public sentiment, and can be simpler than other SA tasks such as news article summarization, for example, which may contain even opinions that do not belong to the writer themselves – and as such require a further step to associate the opinions to their respective holders (Pang & Lee, 2008).

So for customer review SA there can be two main subtasks: identifying product features and subsequently extracting the opinions associated with them. Typically, the first one is done through the assumption that features will be represented by nouns or noun phrases (like the "quality" or the "performance" of a product); however, as not all nouns will be features, author Bakrey (2023) proposes an heuristic to solve the problem, saying that "adjectives appearing in the same sentence as frequent features are assumed to be opinion words, and nouns and noun phrases co-occurring with these opinion words in other sentences are taken to be infrequent features". So for the second subtask, extracting the opinions related to each feature, one can simply extract the adjectives connected to frequently used nouns found.

If the task is summarizing the reviews, on the other hand, the authors Pang & Lee (2008, p. 17) mention that doing so, beyond indicating the opinion, may indicate the reason why the customer had those feelings and cement its value – its trustworthiness: "identifying pro and con reasons can potentially be used to help decide the helpfulness of individual reviews: evaluative judgments that are supported by reasons are likely to be more trustworthy".

The time of the reviews can also be considered important. Indeed, many websites will allow for viewing reviews in chronological order or its reverse: this allows not only for the evaluation of sentiment, but also how this sentiment has changed and evolved over time. By widening the scope from reviews to news articles, for example, one can see how public opinion on politicians and campaigns or companies and their reputation can be tracked over weeks, months or even years to create a timeline. In this way, one can define new strategies and keep track of its development and effectiveness.

Still, as discussed extensively in previous sections, languages are quite complex and simply looking at the list of words associated with a given user, such as "great" and "amazing", does not always indicate that the subjective meaning of the text is positive, for example (Bengfort et al, 2018).

One very straightforward example on the importance of wordsense is given by Bengfort et al (2018) using the word "*sick*": a given interlocutor may use it in a sentences such as "*that kick-flip was sick*" and "*the chowder made me sick*", and each one of these has a different

meaning and directly opposite polarities. Another example, more complex, shows the issue of different meanings even within the same context: if the interlocutor uses the word "*bland*" to describe hot peppers, it is quite a negative comment; if the word is used to describe a cough syrup, on the other hand, it may very well be positive.

It is clear that the field is much more elaborate than tallying up the sums of positive and negative terms and doing so without a deeper context evaluation can yield vastly inaccurate results. And the basis for this is the next steps: preparing the data and defining how to represent it.

The task to be performed needs to be clear, because it will shape how the data is cleaned and preprocessed. As discussed in previous sections, data cleaning begins with determining data structure and finding and removing incomplete inputs. This is quite standard, and the textual documents in a corpus tend to be contained into matrix representations, such as dataframes. So after making sure that the columns in the dataset and their denominations are pertinent and the empty inputs are cleared, the text itself is considered.

For simpler tasks, one can start by removing punctuations and special characters that create noise and do not serve a specific purpose within the text. URLs are removed next, followed by stop words like articles such as "*the*" and "*a*", as both of these do not add information that impacts the understanding of the text and its meaning. Then comes the process of lowercasing all of the remaining words, so as to simplify the dataset and make sure that the same word is seen as such – instead of being interpreted differently due to case-sensitivity. Lastly, the documents are tokenized as needed, typically into single words or sentences, and normalized through lemmatization or stemming.

It is clear that the initial steps already influence the subsequent ones. For example, say one removes all punctuation in the beginning; this complicates the tokenization process if the goal is to do so at the sentence level, as this is typically done through using the full stop (".") as the boundary from one sentence to the next. Even further, removing all uppercase letters and stop words impacts named entity recognition tasks, for example "Apple" as a company will be understood as the same entity as "apple" the fruit, and "the people" becomes a more generic counterpart of "people", respectively.

As for stemming and lemmatization, again removing uppercase letters may confuse names, for example "John" "Running" has its last name stemmed into "run" and the entity would be misunderstood. The same can happen with keeping punctuation, as "running!" may not be stemmed correctly into its stem "run".

Further regarding the choices on text representation, one very common and widespread method is TF-IDF, regarding term presence versus term frequency. Authors Pang & Lee (2008) mention better results were seen when considering only term presence for polarity classification – the binary valued vector showed 1 for a term that occurred within the document, and 0 for non-occurrences; in their words, thinking of information retrieval tasks in general, "while a topic is more likely to be emphasized by frequent occurrences of certain keywords, overall sentiment may not usually be highlighted through repeated use of the same terms" (Pang & Lee, 2008, p. 21).

If one chooses to use more complex representations of text, such as sentences or paragraphs, for Sentiment Analysis they can have opposing labels within the same

document – meaning that the overall sentiment is the amalgamation of the sequence of the opinion contained in each subunit (Pang & Lee, 2008). So instead of representing text as a bag of features, one can use different subunits and try to model the relationship among them: the authors propose a procedure for classifying movie reviews by initially detecting and removing the objective parts, the objective sentences of the review containing not an opinion but facts or perhaps a description of the plot, and applying the polarity classification only on the text that's left, the subjective part.

Another common technique used for Sentiment Analysis tasks is POS tagging, mostly as an aide for text disambiguation: a strong presence of adjectives has typically been found to indicate subjective and opinionated texts (Pang & Lee, 2008). This means that by signaling them as such, these adjectives and their polarity can be used to guide sentiment classification as the main – but not only – feature.

These are NLP-based steps and need to be carefully taken into consideration so as to make sure the data that will compose the basis of any model to be built is solid. After the text preprocessing, there are typically two major methods to perform sentiment analysis tasks: lexicon-based and supervised machine learning – they can also be combined to work together in a hybrid fashion, such as using ML to identify subjective text or expand a lexicon, and use a lexicon to classify the text (Mudinas et al, 2012; Lei & Liu, 2022)

Both of these methods contain a wide variety of techniques to deal with Sentiment Analysis tasks, as exemplified in Figure 12. This work will go into more details in further sections dedicated to each of these fields.

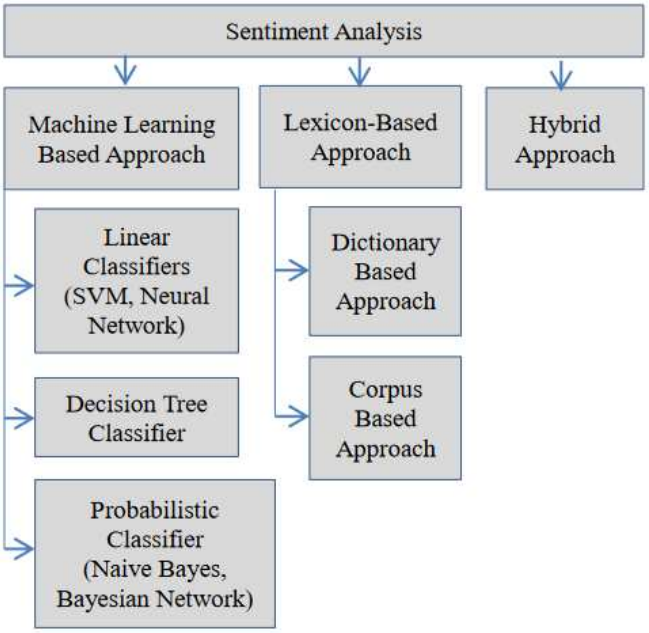


Figure 12. Sentiment Analysis techniques by Sadia et al (2018).

Regardless of the method chosen, in the end one may choose to represent the resulting sentiment information as a summary, and it is typically done through Pang & Lee (2008):

1. Aggregation of voting metric, such as star system or letter grades;
2. Highlight opinions that occur frequently;

3. Highlight points of consensus and of disagreement;
4. Identification of the various communities to which users belong;
5. Identification of levels of authority among users.

Number 1 is the most common for customer reviews, be it for products or services, for books, movies, courses and many others. This type of dataset is typically already labeled with the voting metric itself, and the model can be trained and used for a faster and deeper understanding of customer feedback, applied on product improvement, market research and competitor analysis, reputation management or simply checking the impact of a marketing campaign.

They can also be used along with statistical representation, seen for example in ecommerce platforms to indicate the average grade of a product or service and the number of reviews per number of stars; the latter can be an important indicator, as a product with an average of 50% reviews on the negative side is more worrying if it has a very high number of reviews overall than if it only has a few. People intuitively see reviews through this logic (Pang & Lee, 2008).

One possible issue, however, is that critics do not always agree with each other. Authors Pang & Lee (2008) explains this through the example of a star-rating system, saying that a person can see 3 stars or lower out of 5 as a negative opinion, while a more relaxed grader could see 3 stars as something positive; even if two critics agree on the number of stars itself, for example, they may do so for different reasons – one may see very strong positive points, but also very negative ones, while another sees only average characteristics.

For this reason, a researcher may choose to represent the data by summarizing it, like numbers 2 and 3 above, and verify the most common opinions found and cover both positive and negative ideas, instead of only their representative score (Pang & Lee, 2008). This type of task is very connected to extraction of topic-based information, with the added consideration of subjective and opinionated text.

4.2 Lexicon Based SA Approaches

Widely used for their simplicity and interpretability, lexicon-based methods are an important component of NLP tasks. They are among the most basic tools within the field to enable computers to understand and manipulate human language, being defined by author Bakrey (2023) as a “collection of words or phrases that are used in a speech or a specific field as they play a role [...] providing information about the meanings, uses, and grammatical properties of the words it contains”.

The author complements the definition by saying that lexicons are typically task-specific, so the words and phrases contained therein are associated with certain features and styles; this means that, for example, for POS segmentation the words are associated to their grammatical uses, while for sentiment analysis they can be related to their polarity.

The idea can be broken down even further when thinking about their application. For the field of this research, for instance, if the goal is to perform sentiment analysis on a given corpus of text inside of a more general context, like blog posts or news articles, one can use more general purpose lexicons – typically dictionary or corpus based. This means that they

contain either the polarity or the score of words as intended for their most commonly found uses. One point of attention is that using continuous sentiment scores add a level of complexity that is oftentimes undesired, and so less common than their counterpart (Deng et al, 2017).

Two issues are typically found with more general-context lexicons: insufficiency and inaccuracy, in which the first clearly relates to the lack of words and sentiments specific to the text domain, and the latter refers to the fact that the most common meaning of a given word may not be the one used in the analyzed text domain (Deng et al, 2017). As such, textual analysis tasks can often be improved by lexicon extension for each specific task and the context of its application.

If the task is the sentiment analysis of customer reviews, for example, one may use a domain-specific lexicon. So it should be able to take into consideration not only the purpose of the text, reviewing a product or a service, but also *what* is being reviewed. For example, the word "*long*" within the context of restaurant reviews may be highly negative, associated with long lines or long wait times, while for cellphones it may be quite positive, indicating long battery life. This is typical for SA applied along with aspect extraction, so research interested in both the sentiment and the aspect to which it refers.

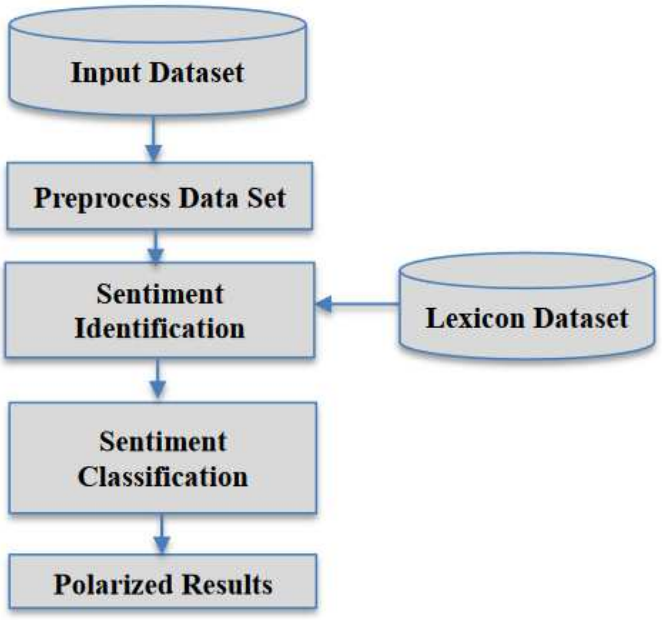


Figure 13. General Sentiment Analysis framework by Sadia et al (2018).

Lexicon-based approaches can be deeply task-specific and bring a high level of interpretability to the analysis, which does not exist with supervised machine learning techniques – these, though they may achieve better performance, tend to work in a black-box manner that does not allow for easy explanations (Mudinas et al, 2012). But they are not all created equal.

4.2.1 Sentiment Lexicon Generation Approaches

Lexicons can be created manually, which is very time consuming – sometimes taking years to reach a level adequate for use. Typically, this method is used to answer the need for very high quality lexicons and for very niche applications, and the labeling is done through field specialists and experts, such as scientists and doctors which tend to be regarded as particularly reliable, but costly and time consuming. Another option is crowdsourcing, in which a large number of people label the dataset a little bit at a time, so it is faster but may not be as reliable.

One example of this is the Linguistic Inquiry and Word Count (LIWC) lexicon (Nichols, n.d.), based on a general context collection of words and used to determine the degree of positive and negative emotions in a given document. It started as a psychological study in the 1980s (Tausczik & Pennebaker, 2009), trying to relate the personal texts and stories written by patients and mental health improvement: the documents were initially subjected to the classification of specialists in a wide variety of dimensions, like emotional, vivid and optimistic. The issue was that many classifications were contradictory, as the specialists did not agree among themselves. So the researchers turned to computer based resources, creating a lexicon to classify texts based on psychological meaning at word-level – taking into consideration grammatical use and personal or impersonal referencing.

Over the last 40 years, the LIWC lexicon has been maintained and expanded into many other languages beyond English, categorizing words in over 100 textual and psychological categories by field specialists, and is now widely used to analyze the sentiments of texts not only for mental health diagnostics, but also marketing research (Nichols, n.d.). One must point out, however, that the maintenance required by manually created lexicons is perpetually ongoing, which may impact the cost and efficiency of the process.

Another, less expensive, approach to create lexicons is called dictionary-based, due to its nature. A small list of seed-words is populated and labeled, usually manually, and then propagated through an online dictionary looking for relationships, such as synonyms and antonyms, and classifying them (Sadia et al, 2018; Darwich, 2019). This method typically results in a task-specific lexicon, but not a context-specific one, because while the seed-words are related to the task at hand, the main idea behind a dictionary is to define the relationship between the words, not their domain-specific information.

One very common online general-domain dictionary used for this purpose is called Wordnet (Sharma, 2022), which groups synonym words through synsets. It can be quite accurate, as this dictionary stores the synonym pairs based on their context and usage.

Derived from it specific for SA tasks, the SentiWordNet lexicon relates the synonym pairs, the synsets, to their respective sentiment polarity scores – positive, negative or objective – as represented in values from between 0 and 1; the final sum of all three is 1 (Baccianella, 2010; Sharma, 2022). It is a publicly available resource and widely used due to its capacity to encompass word meaning: Baccianella (2010, p. 2200) 2200 explains the lexicon through the example of "the synset *'estimable'*, corresponding to the sense 'may be computed or estimated' of the adjective estimable, has an Objective score of 1.0, while the synset *'estimable'* corresponding to the sense 'deserving of respect or high regard' has a Positive score of 0.75, a Neg score of 0.0, and an Obj score of 0.25". It is also important to use POS

tagging along with SentiWordNet, so as to allow the algorithm to correctly identify the use case of each word.

Another dictionary-based sentiment analysis lexicon is VADER (Valence Aware Lexicon and sEntiment Reasoner), built specifically for use in the context of social media and made publicly available for use in Python code; is was based on three preset lexicons, among them LIWC, along with human validation, and detects both the polarity of the emotions present in a document (positive, negative and neutral) and their intensity (Al-Shabi, 2020; Malde, 2022). The latter is quantified on a scale of -4 to 4, most negative word to most positive, respectively, which VADER normalizes and returns as a value between -1 and 1. This interval is also known as the valence of the sentiment.

It is a great resource for up to date analysis of user generated content, as it contains commonly found slang terms, abbreviations and emojis. As such, the lexicon finds applications ranging from social media monitoring, to customer review analysis and market research.

VADER can be used through the NLTK (Natural Language Toolkit) Python library and requires very little or no text preprocessing at all, because it was built to automatically deal with negation, capitalization (so how "*AMAZING work*" differs from "*amazing work*") expressive punctuation (such as "??" or "!!" at the end of a sentence) and the removal of stop words. [95, site] All of these factors make VADER a widely popular tool, but it may provide a lower accuracy for particularly context-dependent tasks in which the language used is different from the general social media environment.

Author Al-Shabi (2020) compared the performance of several publicly available lexicons, as it can be common when performing NLP tasks to employ more than one single lexicon. Among them, both SentiWordNet and VADER, on the polarity classification of two different datasets composed of user generated tweets. Using Accuracy as performance metric, the VADER lexicon outperformed all others on both runs, especially for positive and negative classes. They particularly recommend its use for the classification of short texts.

A more recent development is the lexicon named Sentimentr, mentioned by authors Lei & Lie (2022): it can be quite accurate for general purpose uses as the lexicon has incorporated rules to deal with valence shifting. This means that it uses weights to adjust the overall sentiment value of a sentence based on how it is composed. If there are elements such as negators (i.e: not, doesn't), intensifiers (i.e: highly, really), downtoners (i.e: a little, slightly) or adversative conjunctions (i.e: but, however), the output is adjusted accordingly. Sentimentr was developed for Sentiment Analysis tasks with the R language.

The third, and final, method to create a lexicon is called corpus-based: this approach specifying selected labeled seed-words and "exploiting co-occurrence statistics or syntactic patterns in a text corpus" (Darwich, 2019, p. 297). So instead of using pairs of synonym/antonym words, corpus-based methods try to capture the semantic distance of the words that compose the corpus to the seed-words within their context of use to derive their polarity, creating a lexicon that is not only task-specific, but also domain-specific. This is a distinctive advantage, for example, using a corpora of user Tweets to infer polarity and create a social-media specific lexicon (Deng et al, 2017).

This is a clever way of picking up on internet slang or more informal terms, which are beyond the capabilities of dictionary-based lexicons unless the term has been specifically included in it, and deriving a sentiment classification tool specific for the domain in which it will be used (Darwich, 2019).

In general, due to the constraints and advantages presented, SA tasks are typically done in a two step process: lexicon creation or extension based on the vocabulary found in the given dictionary/corpus and measurement of sentiment strength (Mudinas et al, 2012). In other words, in the absence of manually labeled datasets, many tasks can be solved by "first creating a sentiment lexicon in an unsupervised manner, and then determining the degree of positivity of a text unit via some function based on the positive and negative indicators" (Pang & Lee, 2008, p. 27). This process results in a task – and field, if necessary – specific lexicon, in which the output can be the association of words to a polarity (positive, negative, neutral) or to its weight on a scale (for example, +5 if very positive, and -5 if very negative).

One must pay attention, however, to how the dictionary is maintained for these rule-based methods: they are typically not updated often and as such may lack the presence of new word-pair relationships in a timely manner (Deng et al, 2017). Lastly, even though rule-based techniques can be pretty direct and adapted to fit the context to a certain extent, the fact remains that it is a technique that focuses mostly on individual words (Campesato, 2021).

4.2.2 Use of Lexicons on NLP Pipelines

In general, lexicon-based methods are classified as a rule-based approach, because they are built through a fully or partly human-written set of rules – manually or through dictionaries and seed-word lists – and applied through pre-defined logic, such as the sum or the average of the assigned scores.

For the task of Sentiment Analysis itself, the idea is that "each word in the document is checked against the sentiment lexicon and the sentiment scores are recorded as matched words are found" (Deng et al, 2017, p. 66). With the matches found, many calculations can be performed: the simplest method is to sum the difference between positive and negative scores of each word within the text, and use the result as overall sentiment score on a given scale. Another possibility is to find the average value among all the tokens present in the text – or perhaps the weighted average, depending on words that may carry a higher impact, such as adjectives.

The output of this process can be used by itself, as a number in a reference scale – a score, or be further processed to classify the document into its overall sentiment: for example, three class classification into Positive, Negative or Neutral.

The manner in which lexicons can be integrated into the pipeline of NLP tasks, such as sentiment analysis, is done through several steps. As discussed in previous sections, the first one is Text Preprocessing the available data. This step is gonna vary widely based on the lexicon (os lexicons) chosen to use and how it deals with the numerous grammatical components of the text, if more or less handling is required.

Next comes the use of the lexicon itself. It will compare the words found on each document to its word/valence pair and calculate the valence of each one.This value will then be used to

aggregate them into a single one – following a predefined heuristic – and compare it to the threshold set for each class.

The final result can at last be used to gather a deeper understanding of the users and their needs, be it the class or the score. It can be used for analysis and reporting, visualizing trends and making decisions.

But needless to say, the lexicon-technique is not failproof. As discussed previously, it needs to be maintained over time regardless of the method chosen to build the lexicon itself and the method does not solve characteristic NLP issues such as sarcasm and irony, for example, or non-sentiment words used to convey opinions.

4.2.3 Evaluation of Lexicon-based approaches

The manner in which the output it evaluated is through measures seen previously in the section on Machine Learning techniques. The first, and by far the most prevalent, is Accuracy. It measures the exactness of a classifier, how well one can trust that a review evaluated as positive truly is positive (Jenhani et al, 2016). It is a great performance indicator and widely used to compare the quality among models, but may not be enough to evaluate the results from very imbalanced datasets, so other metrics can be used.

Recall (Sensitivity) represents the model's capacity to identify the positive instances identified, among all the actual positive instances. It can be used along with the Precision metric, which represents the proportion of true positives that should be identified in comparison to all that were classified as such.

Lastly, and quite common for comparisons, is the F1 score. It represents the so-called harmonic mean between precision and sensitivity, providing an overview of both, and should be as close to the value of 1 as possible.

As seen so far, the use of lexicon-based methods to solve Sentiment Analysis tasks can be very straightforward. There are numerous publicly available lexicons, which – unless created manually – makes the process accessible and less expensive than other approaches. But issues inherent to NLP are not fully solved by it, like irony and figurative sayings, as the full context in which the words exist cannot be grasped in its totality.

This deeper contextual understanding of the textual data is better reached through another line of methodologies, known as supervised machine learning. Authors Yusof et al (2015) set out to compare both approaches based on published literature in the field, and concluded that while lexicon-based methods can be less expensive and work particularly well for cross-domain applications, if a large body of textual data is available machine-learning methods may achieve a higher accuracy – especially for context related meaning – because of the way in which it learns.

4.3 Supervised ML SA Approaches

As already established in this work, the use of machine learning techniques as a business practice has become indispensable. Sentiment analysis is just one more facet of it, one more application in which it stands out, this time by allowing any given organization to more effectively apply the concept of "customer in the center".

When “compared with conventional computational linguistics methods, machine-learning is more into achieving a higher level of automatic language processing, understanding, prediction, and production, and its algorithms may thus be more sophisticated” (Lei & Liu, 2022, p. 5) This difference from more conventional techniques is given by the fact that not only it involves different features, but also has the potential of modeling the numerous relationships found in the text at varying levels, between tokens and sub-document units – some may be specific to the task and others not even known by the developer (Pang & Lee, 2008). In other words, machine learning based sentiment analysis tends to achieve a deeper understanding of the textual meaning contained in a given document due to its capacity to extrapolate relationships within the subunits that compose it.

And while there is a wide variety of models that can be applied, discussed at length in section 2.6, the basis is typically the same: supervised ML techniques “classify the texts in the test dataset into one of the predefined sentiment categories based on the results of machine-learning from the training dataset” (Lei & Liu, 2022, p. 17) – meaning it goes through the labeled data to learn how to generalize the underlying sentiment expressed in the text and predict its class. [Sen et al, 2019; Campesato, 2021).

Authors Haberzettl & Bernd (2018) conducted a survey on machine learning-based SA research to discover which were the most widely used models and feature extraction techniques, ranging from customer reviews, tweets, social media posts, news, forums and many other mediums. Based on the papers with the highest number of citations, they found 12 dominant models and 11 techniques – that were later classified based on how many times they appeared in the research and how many times they were part of the best model trained.

Regarding the models, the authors found that out of the 12 present in literature, only 5 were part of the best solution at least once: “Support Vector Machine (SVM), Artificial Neural Network (ANN), Naive Bayes (NB), Logistic Regression (LR) or Maximum Entropy (ME) and k-nN nearest neighbor (k-nN)” (Haberzettl & Bernd, 2018, p. 9). SVM, for example, was part of the best solution found in over 50% of the papers reviewed, showing good promise for its use.

As for the feature extraction techniques, because of the heterogeneity of the papers, their data, objectives and metrics, comparison among the research becomes tricky. The authors point out that almost half of the papers reviewed used only the model’s Accuracy as the metric to decide between one technique or another, which is not expressive enough, and the datasets used vary greatly amongst themselves, from quantity to quality. So while one may say that the most widely used techniques were n-grams, term presence and POS tagging, with the first being the most frequent method among the best solutions found, one should consider it within a larger scale of the task and the data.

Now, creating the machine learning models themselves can be quite a long process, and the first step is arguably the most important: to collect the labeled dataset. Like other methods, it is often contaminated with noise (outliers) and missing values, perhaps even features that need to go through a transformation to become usable, the so-called dummy variable (Sen et al, 2019). Here, also, there are many ways of dealing with faulty data, typically done during data preprocessing, such as removing irrelevant variables and deleting duplicate or

empty entries – for textual data, this can also go for punctuation, unnecessary spaces and stop words (Naeem et al, 2020).

It is common after data preprocessing to employ visualization techniques and better understand how it is distributed. So ”this includes word clouds for positive and negative words in the dataset, bar-plot for the class distribution” (Naeem et al, 2020, p. 484) and allows for a quick general outline, indicating class imbalances or more common words.

The remainder process outline is better summarized by authors Sen et al (2019), Naeem et al (2020) and Lei & Liu (2022), when mentioning that clean labeled data is subsequently decomposed into distinct features of the text or its sentiments. These can be high-frequency words or phrases, their grammatical POS tags, NER tags, BOW, n-grams and so on. Much like the lexicon-based methods, this feature selection relies a lot on the task itself, what it means to accomplish, and the expertise of the developer.

It also depends on the granularity of the analysis: at a document level, in which the polarity is assigned to the entirety of the document, it is assumed it relates to a single entity and so needs no further development, while a sentence level the idea is to differentiate between objective and subjective phrases to grade and sum their individual polarities; the deepest type of analysis, however, requires further feature extraction and selection and as such is called aspect level (or feature level) (Chauhan, 2021). It is applied when the text refers to more than one entity and it is important to extract them, as well as the opinion expressed along with it, and authors Avinash & Sivasankar (2018, p. 2) complement this by saying that essentially “feature extraction is one of the dimensionality reduction techniques used in machine learning to map higher dimensional data onto a set of low dimensional potential features”

If one has enough computing power, however, they may use the method known as *brute-force*, in which most, if not all, possibilities are tested, including raw data, transformations and any other conceivable approach. The idea is to isolate the best features through testing them all.

Quite a few researches have been done on this topic, with varying degrees of success. Authors Chauhan et al (2021) for example compared the results achieved with the most common supervised machine learning models through the BOW and TF-IDF feature selection approaches. Using Accuracy as a metric, they concluded that the former performs slightly better than the latter, especially when paired with stemming as the normalization technique instead of lemmatization. Authors Avinash & Sivasankar (2018) on the other hand chose to compare the performance of the same TF-IDF approach with the word embedding technique doc2vec, an extension of Word2Vec seen in section 3.3.4. On the data set of movie and product reviews, although both approaches achieved acceptable results, again the TF-IDF models’ overall performance was worse than the alternative.

Authors Trupthi et al (2016) proposed a comparison between the traditional BOW methodology, and then pairs it with bigram collocations, so pairs of words frequently used that provide opinion such as “not good”, and high information feature extraction. The traditional method by itself, as expected, had the worse performance overall, while the simple addition of commonly used bigrams improved the model’s accuracy by 9%. However,

by removing low information features and keeping the high information features, the accuracy of the model improved by 20% in relation to the initial, traditional BOW method.

So feature extraction and selection has a very big impact on the final model and needs to be tested and investigated for each task due to the wide range of possibilities.

Next, this corpus is randomly divided into a training and a testing dataset: the former typically contains more than 70% of the labeled text and will be the learning basis for the chosen classification machine learning models like Decision Trees, Random Forest and SVM, while the latter will use the remaining data to measure their classification performance in the last step of the process.

One point to keep in mind is that regression models can be quite useful for predicting the positivity contained in a given text that expresses opinions, like reviews, because it typically tries to discover the similarity between one class and another. Classification models, on the other hand, typically try to search for the characteristics, the features found within each class without focusing too much on the scale and how close one class is to another (Pang & Lee, 2008).

The testing stage typically uses the metrics of Accuracy, F1 and Precision to compare and evaluate the models trained, discussed at length in section 2.7. The threshold set for the minimum acceptable level of performance may vary, as there is no one size fits all: different tasks will demand different approaches. First of all, data characteristics and availability of resources will have a large impact on the end result, which needs to be within expectations. Now, in the sentiment analysis field for example, the task of brand monitoring on social media may require a bigger focus on negative opinions as they indicate issues the firm needs to pay attention to, so metrics used should reflect that to proactively identify and take action to solve them. In the same vein, if the goal of social media monitoring is to identify and promote positive content, it may be more appropriate to focus on metrics that reflect the model's capacity to correctly identify text in the positive sentiment class.

Another point is the business impact of the resulting model: if it is meant to have a big impact on decisions, a higher, more strict threshold may be necessary. Using the same example of social media monitoring, if the goal of the machine learning model is to filter out the negative or offensive content, instead of promoting positive ones, maybe even to comply with national and international regulations, using a higher performance threshold may be more indicated. However, by prioritizing flagging every single negative or offensive piece of content so accurately, the model may flag a few positive ones on the way – which could result in user frustration and lessen engagement. Conversely, too much offensive content going through may cause damage to the company reputation.

Class imbalances, mentioned previously, add another dimension to the metrics employed: if the researcher knows that within the dataset one or more classes are vastly dominant in regards to others, the F1 and AUC-ROC curve metrics can be employed.

In the end, the model best suited to the task, if there is such a one, can finally be used on new and unseen data for its final purpose.

Although machine learning models for sentiment analysis tend to be more reliable overall than lexicon based ones, their average accuracy is usually around 75% and not all of them

provide a good level of explainability. Even further, beyond the large requirements for good quality, balanced labeled data, they typically are trained with specific domain datasets, meaning that these models to generalize well only within those contexts (Chauhan et al, 2021).

In the business field, the dataset used may contain biases that will result in a model with skewed or unfair predictions, especially if they are user generated. Even further, concerns regarding privacy and regulations are becoming more and more important because the last decade signified the convergence of factors that allowed for the widespread training and use of machine learning models, and the trend is to keep on growing. This means that while the use of machine learning in general is an amazing tool for advancement and acquiring further knowledge and advantages, it needs to be done carefully and respectfully – the ethical concerns are discussed further, in section 4.5.

It should be remarked that this work will not discuss deep learning approaches, as they are outside of the scope and will not be used.

4.4 Ethical implications and guidelines

Sentiment Analysis is a field of study currently undergoing a strong wave of development. Where today the main data source behind it is written text, authors Pang & Lee (2008) see it as a soon-to-be standardized part of many products dealing also with speech. They pick back up the idea introduced in the beginning of this section, that leaders – in a wide sense, not just political – have always been interested in public opinion and the possibilities that this knowledge provides. They illustrate it with Edward Bulwer-Lytton's widely known phrase, "the pen is mightier than the sword".

Public opinion has power. It has sparked and ended wars, revolutions, mass hysteria and outrage, it is the reason why fake news is constantly being spread and perpetuated by companies and organizations to defend their own self interest (Pang & Lee, 2008).

This is to illustrate the fact that all of the SA applications mentioned so far in this work had as its goal making a positive impact, but it does not encompass all it can be used for. Author Mohammad (2022, p. 1) puts it succinctly by saying that "sentiment analysis can be a facilitator of enormous progress (e.g., in improving public health and commerce) but also an enabler of great harm (e.g., for suppressing dissidents and manipulating voters)." Author Patti et al (2017) present the ethical dilemma by saying that the same technology used to discover and counteract terrorists, homophobic and racist hate-speech on social media (which is becoming an indispensable task due to its incredible capability of proliferating), is also used to discover and influence people's political orientation.

This is important because journalist enquiries have discovered that data analysis companies played a role in elections and referendums by associating the information gathered on user's personalities to sentiment analysis of their content to better customize campaigns for still undecided voters towards one side or the other (Patti et al, 2017).

So analyzing social media for the consensus on news with a high level of impact, like vaccines acceptance or disease awareness for public health monitoring in the public sector is meant to bring a positive impact, as well as, on a smaller scale, the consensus on a given

brand or product in the private sector to perform product or reputation improvement. Both of these applications look to make a positive impact, but the script can be easily flipped.

The technology can be used to manipulate public opinion towards a specific end goal. In the public sector, for example, to monitor and censor dissent and in the private sector as customer manipulation – they cannot control UGC, but they can monitor and influence it (Pang & Lee, 2008). The authors point out still that the latter is already part of normal public relations efforts, citing the research of Zabin & Jefferies (2008), that discovered that more than half of the companies engaging in social media monitoring actively and frequently contribute to consumer conversations, a third of them looking to "sway opinion, correct misinformation, solicit feedback, reward loyalty, test new ideas, or for any number of other reasons".

Not to mention the issues related to the data itself. Because typical sentiment analysis technologies use UGC as its basis, consulting a large number of people who are not known, creates the issue of trustworthiness: it means that their reliability can be put into question (Pang & Lee). Even further, one must consider if what is expressed in the data contain human bias, like prejudices and stereotypical ideas, and if it has an impact on the result: it is quite common today to find chatbots as the first line of troubleshooting for products and services in general, they are mean to interact with people of all backgrounds and should reflect this in their speech; more personally on the user themselves, the researcher should consider what is ethical to infer about a person's opinion expressed online, as the data that can actually be acquired typically goes into more personal details such as age and location, space-time movements – issues of privacy (Patti et al, 2017).

In this line, there have been several nations that implemented in the last decade national policies for protecting user's privacy while online, trying to guarantee specifically the correct confidentiality and anonymization of data. Over 70% of the countries worldwide have data privacy laws regarding how corporations may use the data of their customers and users – personal or otherwise – such as the General Data Protection Regulation (GDPR) implemented in 2018 by the European Union block and the General Data Privacy Law (GDPL) in Brazil, implemented in 2020 (Horizon Framework Programme of the European Union, 2020; UNCTAD, 2023).

Looking specifically at the data represented by customer generated reviews, one should first evaluate its impact to know if it is a worthwhile pursuit, if there can be motivation to engage with the task at all. Authors Pang & Lee (2008) point out that due to the rising online ecommerce, there is a lot of research on its impact on customer purchase, and the overarching consensus seems to be that reviews of some items have more impact than others – for example, customers tend to not pay attention to the reviews of cheaper items, while their impact when talking about more expensive items is considerable. They also point out that the studies in this field can be contradictory, with varying positive and negative results depending on factors such as timing of the feedback and characteristics of the seller. One thing is widely accepted, however: the overall polarity of the totality of reviews has a significant economic effect for the company.

However, these issues in particular are not related only to sentiment analysis tasks, but to data-mining technologies in general, and imply that their development are not only an

opportunity for research, but also for new reflections and responsibilities (Pang & Lee, 2008; Patti et al, 2017).

These technologies, especially virtual agents such as chatbots and applications based on speech (like Apple's Siri and Amazon's Alexa), can go so far as to have the ability to "promote beliefs and behaviors in human users is obtained through the creation of an empathic relationship with the user, assuming their goals and emotions" (Patti et al, 2017, p. 154) and it can be quite challenging setting up safeguards and teaching these agents the moral aspects of human behavior.

Author Mohammad (2022) proposes 50 principles to guide research and implementation of automatic emotion recognition technologies, regarding the data, the task design, its method, impact and evaluation and finally the project's implications for privacy.

5

Development

This section presents the case study, from its development, data collection and analysis to the development of the machine learning models chosen based on the literature review introduced so far.

5.1 Project context and objectives

As “nowadays, individuals, organizations and government agencies are increasingly using the content in social media for decision making” (Ziora, 2021, p. 78) and the amount of user generated content is only growing more each year (Maharani, 2013), it becomes increasingly important to develop a strong body of research and evidence on the methods available to do it.

Even more so, NLP has numerous sources of influence with considerable impact on the manner in which the task is solved. Characteristics such as the amount and quality of the data, along with its format – text or audio – and the language it is represented in, were discussed at length in previous sections. These can branch out into considerations of the availability of resources, like lexicons and dictionaries, of which English is by far the language with the largest amount of NLP resources. In this sense, the task itself shapes the process, but only so far as the data and resources allow for it.

As stated in Section 1.3, even if the NLP field of Sentiment Analysis is not a new technology, it has been rising sharply in both public and private interest in the last decade due to the huge availability of data allowed for by the advent of the Internet and Social Media, along with the availability of the processing power required to analyze it. In this context, this work intends on contributing to the growing scientific corpus by evaluating the impact that various

NLP and text representation techniques have on the results of Sentiment Analysis of user generated content.

While not exhaustive, the literature review built thus far serves as support for the case study developed, following the objectives:

1. Analyze the applicability of NLP and machine learning techniques for solving complex business problems and generating valuable insights.

This objective entails detailed investigation into the specific use cases and industries in which ML holds most promise, and the advantages they may offer in terms of data driven decision making for strategic planning.

2. Evaluate technical challenges and limitations associated with the implementation of ML in business processes.

Typically, one can expect technical limitations related to data quality and quantity, model accuracy, computational resources. The goal is to assess and develop strategies to overcome them.

3. Compare and evaluate different machine learning models for the chosen application.

Considering previous resources and choices, evaluate the impact of the final result. This could mean factors like data types, scalability and interpretability – decisions over which the researcher has control as well as not.

To reach these three goals set out, this work has chosen the De'Longhi La Specialista Espresso Machine (model EC9335BK), as it fits both requirements: it has extractable user generated data and is of italian manufacture.

5.2 Methodology

In order to achieve the goal of extracting insights out of public reviews of the chosen product, a standardized and repeatable heuristic should be defined. As such, the methodology followed is set in six specific steps:

I. Define the field of research: evaluation of consumer sentiment through the use of Natural Language Processing and Sentiment Analysis on product reviews. More in depth, how the available text manipulation techniques may affect the results.

II. Realize a literature review: so as to better understand how the field has developed in the last decade, a few strings of keywords were used to find the most relevant publications; these include the initial field study: “industrial applications of Natural Language Processing” and “business applications of Natural Language Processing”. To get a more specific outlook, “Sentiment Analysis with Machine Learning models” and “Sentiment Analysis with lexicon approach” were also used. Because of their use as synonyms, terms such as “opinion mining” was also considered.

III. Select relevant research: as the initial queries returned a too varied selection of publications, they were carefully filtered to remove any content not related to the current topic. Further, the studies selected should comply with: (a) a real-world, scalable application of NLP in either industrial or business setting; (b) either perform or allow for the use along with machine learning techniques; (c) research done using the English language.

IV. Data gathering and analysis: the dataset was gathered from online resources using an API, and later analyzed using R language to obtain initial insights into what it contains and how to proceed.

V. Model choice, application and tuning: the text was preprocessed based and fed into the training and testing of models most recommended by the literature review. They were tuned as needed.

VI. Final evaluation: the models were evaluated by themselves and later compared to each other based on standard field metrics. The results were finally discussed and future works were proposed.

The initial steps (I, II and III) are presented in detail in sections 1 through 4 of this work, while the remaining steps represent the practical application and development of the case study and are described in section 5.

5.3 Data gathering and pre-processing

The product selected, in line with the prerequisite of being of italian origin, is the La Specialista Espresso Machine (Model EC9335BK) by manufacturer De'Longhi. The company is based in the italian city of Treviso, largely focused on the design and engineering of home appliances. In the early 2000s, De'Longhi pioneered the first super-automatic coffee machine, named Magnifica, and has been a reference in the sector ever since (De' Longhi Group, n.d.).

Looking for product reviews in English, due to a wider availability of resources, the main online selling platform found is Amazon.com – an almost ubiquitous presence in consumer review research and analysis. The website uses the 5-star rating system, in which 1 star is a negative review, while a 5 stars review means the product exceeds expectations.

To calculate a product's overall rating, Amazon.con forgoes the simple use of the average of all ratings, choosing instead to place weights on the date the review was posted, its trustworthiness and if the consumer did buy the item through the website or not. This should provide a more rounded and all-encompassing overview, meaning it could have greater value and information for prospective buyers.

It is not, however, obligatory for the consumer to write a review attached to the rating. This means that the vast majority of the ratings tend to be only the 5-star scale, without textual explanations or comments – significantly reducing the data availability for NLP analysis.

For the machine in question (ASIN B08DHW3GGG), at the time of this research the website had a total of 2.009 global consumer ratings that made up a product score of 3.8; out of these, 795 had textual reviews attached divided in the following manner:

- 1-star: 249 reviews
- 2-star: 42 reviews
- 3-star: 49 reviews
- 4-star: 73 reviews
- 5-star: 382 reviews

Due to recent restrictions set by the platform, the website shows only 100 reviews per star rating at any given time, meaning that the largest theoretical dataset that could be acquired directly is 500 reviews. As seen above, some product ratings do not even reach this limit, further reducing the available dataset to be created.

For this research, the data gathering was done through the use of an API named Apify. Oftentimes regarded as the more ethical manner of practicing web-scraping due to its good practices on using the server's resources, the use of APIs allows for a simpler and faster data gathering process – while still maintaining it customizable. Typically, these tools work in a more transparent manner, and provide better data quality and consistency.

Apify, for example, has a scraper designed specifically for the reviews on the Amazon platform: the main information it requires to work are the product (or products) page in which to look for the data, and the specification of which reviews to gather and how to sort them. The output can be customized as necessary.

For this research specifically – based on the platform's restrictions – at least 5 runs were necessary, one for each rating (1 star, 2 stars and so on). The output chosen was a .csv file for each run, containing 23 data columns and one row per review, to be subsequently integrated. Additionally, some 1-star and 5-star reviews were added manually, meaning that the total dataset gathered contained 422 inputs.

As established previously, "having an appropriate dataset is the key element for increasing performance of such systems" (Goularas, D., & Kamis, 2021, p. 5), perhaps presenting more advantages than even experimenting with many different settings and configurations.

Initially, the data was cleaned to remove data columns deemed unnecessary for the project: *countryCode*, *position*, *reviewCategoryUrl*, all of the *reviewImage* columns, *reviewUrl*, *reviewdIn*, *totalCategoryRatings* and lastly *totalCategoryReviews*. Next, the reviews in languages other than English were removed: since the Amazon domain evaluated is North America, there were also inputs in Spanish and French languages – which are out of the current scope.

In the end, the total usable dataset contains 399 entries, with the following 9 data columns:

- *Country.*
- *Date*.
- *isVerified*, which takes into consideration if the user acquired the product through the website.

- The univocal *productAsin* as checkmark that all reviews are pertinent to the selected product.
- *ratingScore*, containing the score from 1 to 5 stars.
- *reviewDescription* containing the textual review.
- *reviewReaction* containing the number of people who found the review useful.
- *reviewTitle,* which contains few words and tends to be straight to the point.
- *variant* to define the model bought.

These entries are divided into 5 categories, from 1 to 5 stars, and appear to be quite unbalanced. People tend to vote more on the extremes, than on the neutral ground, to express their opinions clearly. The initial configuration seen shows the following scores:

- 1-star: 129 reviews
- 2-star: 38 reviews
- 3-star: 39 reviews
- 4-star: 68 reviews
- 5-star: 125 reviews

```
# A tibble: 399 × 9
   country       date                isVerified productAsin ratingScore reviewDescription
   <chr>         <dttm>              <chr>      <chr>            <dbl> <chr>
 1 United States 2023-07-23 00:00:00 true       B08DHW3GGG           1 I received the machine…
 2 United States 2023-07-15 00:00:00 true       B08DHW3GGG           1 After unboxing, cleani…
 3 United States 2023-06-27 00:00:00 true       B08DHW3GGG           1 I lost my money manufa…
 4 United States 2023-06-26 00:00:00 false      B08DHW3GGG           1 Purchased this machine…
 5 United States 2023-06-21 00:00:00 true       B08DHW3GGG           1 First time descaling a…
 6 United States 2023-06-12 00:00:00 true       B08DHW3GGG           1 I ordered a new machin…
 7 United States 2023-06-08 00:00:00 true       B08DHW3GGG           1 Was so wanting this it…
 8 United States 2023-05-21 00:00:00 true       B08DHW3GGG           1 I had a few of these m…
 9 United States 2023-05-14 00:00:00 true       B08DHW3GGG           1 I ordered a De'Longhi …
10 United States 2023-05-13 00:00:00 false      B08DHW3GGG           1 Bought it from a big b…
# ℹ 389 more rows
# ℹ 3 more variables: reviewReaction <chr>, reviewTitle <chr>, variant <chr>
```

Figure 14. Data set tibble summary representing the initial 10 rows.

The columns of real interest for this project are: *ratingScore*, as it contains the label, the sentiment, *reviewDescription*, as it is the main comment made by the user the reasoning behind the rating, and *reviewTitle*, because being shorter, this work makes the assumption that is contains words – tokens – that denote and explain the feeling in a direct, straight to the point manner and can be useful.

5.4 Data analysis

Realistically, the first thing to know is that the dataset is quite small and needs to be dealt with as such, as it will affect later decisions. For example, using Deep Learning models, which have an amazing reputation for Accuracy in NLP tasks, is not a viable option in the long run as they tend to require very large datasets to actually learn and work as intended. On the other hand, Naive Bayes can be a good choice in its place, so it remains a possibility.

As seen previously on Figure 12, the dataset saved into a tibble named *coffeeReview* contains 9 columns with well organized column names, most of which use data type <chr>, meaning strings. The two exceptions to this are *date*, stored as <dttm> to represent calendar dates and times, and *ratingScore*, stored as <dbl>, part of the numeric class in R. This sets the operations that can be performed in each one of them; for example the mean value of the scores collected is around 3.06, with no weight placement, meaning it's lower than the value calculated by the Amazon platform with the full collection of reviews (3.8 rating).

```
> coffeeReview %>% glimpse()
Rows: 399
Columns: 9
$ country           <chr> "United States", "United States", "United States", "United Stat…
$ date              <dttm> 2023-07-23, 2023-07-15, 2023-06-27, 2023-06-26, 2023-06-21, 2023-06-12, 2023-06…
$ isVerified        <chr> "true", "true", "true", "false", "true", "true", "true", "true", "true", "false"…
$ productAsin       <chr> "B08DHW3GGG", "B08DHW3GGG", "B08DHW3GGG", "B08DHW3GGG", "B08DHW3GGG", "B08DHW3GG…
$ ratingScore       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ reviewDescription <chr> "I received the machine and just to found out the machine is meant for brazil. I…
$ reviewReaction    <chr> NA, NA, "One person found this helpful", "One person found this helpful", NA, NA…
$ reviewTitle       <chr> "do not buy this machine. at lease the the one sold by Amazon.com", "Pump doesn'…
$ variant           <chr> "Size: Stainless SteelColor: SilverPattern Name: Espresso Machine", "Size: Stain…
```

Figure 15. Glimpse at the initial dataset, its columns and data types.

Regarding the date, a plot can be made to check any trends based on when the review was posted, as seen in Figure 16. Even if not inside the particular scope of this project, one could argue that the drop in the number or reviews posted between 2020 and 2021 may be related to the beginning of the COVID-19 pandemic, during which consumers changed buying patterns to prioritize essential, perishable goods (Wertz, 2021).

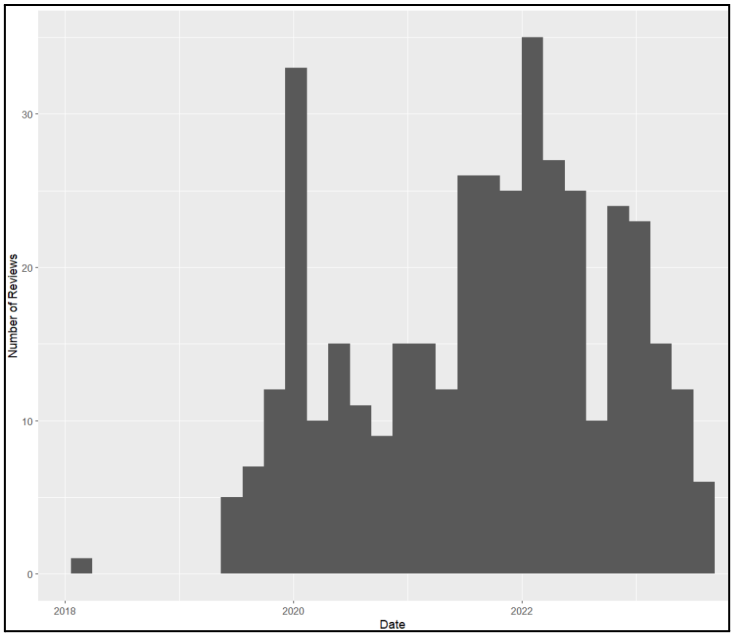


Figure 16. Distribution of reviews posted over the years of 2018 and 2023.

For general data cleaning, no duplicated data was found, however the column *reviewReaction* had a high number of NA values – meaning they are Not Available, blank. To remain inside the data type, these were set to “0 people found this helpful” so it can be used if deemed necessary.

Regarding the distribution of the data, Figure 17 below shows the entries in a visual manner: there is a clear data imbalance between the extreme points, 1 and 5 stars, and the remaining, more temperate ratings of 2, 3 and 4 stars.

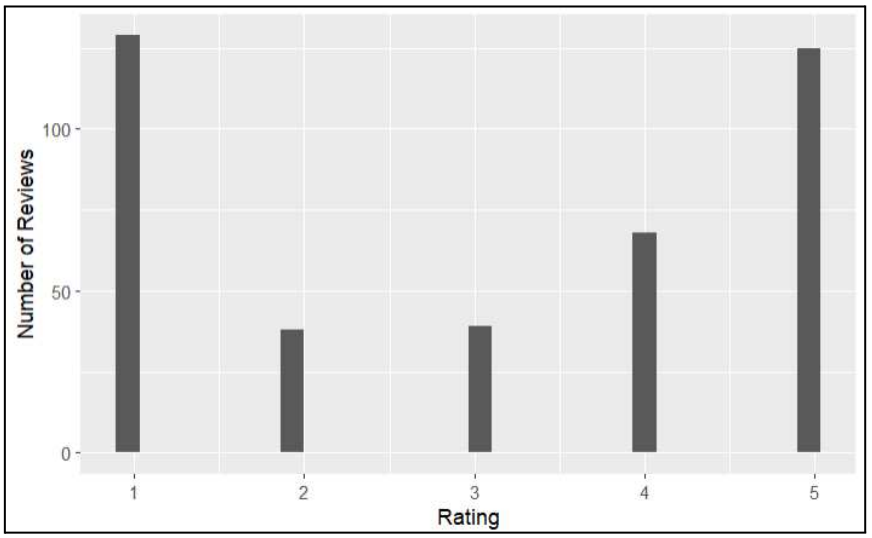


Figure 17. Distribution of numbers of reviews per rating, 1 to 5 stars.

Because the idea is to predict the overall sentiment contained in a given review and distill it into their polarity, the 5-star system was transformed into the labels of [-1, 0 1] representing respectively negative, neutral or positive polarity.

The original ratings were divided according to the findings of Gupta et al (2021), in which mostly negative sentiments are contained in 1 and 2 star ratings (condensed into -1 polarity), while mostly positive ones are contained in 4 and 5 star ratings (condensed into 1 polarity). Lastly, 3 stars represents the neutral class (0). This transformation allows for the simplification of the data and provides a better focus on the sentiment behind the review, condensing the desired information, instead of reflecting the degree of satisfaction shown by the customer.

One needs to keep in mind the possible ramifications of such a transformation, as it is considered an inductive bias. It does depend on the assumption of an equal distance between the ratings: the idea that the distance between 1 and 2 stars is the same as the one between 4 and 5 stars. It can also introduce bias into the data by creating large differences between the number of datapoints in each class, especially in an already imbalanced dataset. For the dataset used here, the neutral category is clearly less predominant, seen in Figure 18, and should be dealt with accordingly.

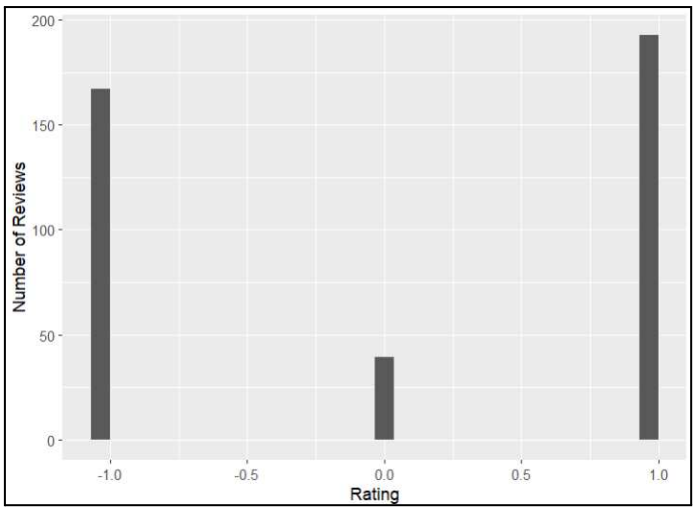


Figure 18. Distribution of reviews in classes after transformation.

The transformation also represents a loss in information granularity, in exchange for better simplicity and clarity in the results, which in this case is acceptable due to the size of the data set and the method used.

Regarding textual preparation on the reviews themselves and the review titles (respectively *reviewDescription* and *reviewTitle*), numbers, punctuations and special characters were removed, uppercase characters were transformed into lowercase. This represents the removal of noise from the datasets. Due to the fact that the process does not contain the use of lexicons to infer meaning, the contractions were kept as they were in the original text.

Next, both columns were tokenized at word-level and ordered by frequency, seen in a summary in Figure 19. The tibble *tokenReview* (a) represents the words contained in the reviews posted by the consumers throughout all the rates, and it is clear that the most common words are stop words – these do not provide valuable information and as such can be removed. As for *tokenTitle* (b), since the input by the customers was considerably shorter (and more to the point), the words are more likely to be pertinent to the task at hand, seen in the use of “great” 45 times and “coffee” 42 times.

(a)

```
> print(tokenReview)
# A tibble: 4,279 × 2
   wordsReview     n
   <chr>       <int>
 1 the          2538
 2 to           1365
 3 a            1322
 4 and          1272
 5 it           1174
 6 i            1172
 7 is            775
 8 machine       697
 9 of            654
10 this          588
# i 4,269 more rows
```

(b)

```
> print(tokenTitle)
# A tibble: 666 × 2
   wordsTitle     n
   <chr>      <int>
 1 machine       89
 2 not           56
 3 the           50
 4 it            46
 5 great         45
 6 a             42
 7 coffee        42
 8 to            36
 9 espresso      35
10 but           31
# i 656 more rows
```

Figure 19. Top 10 words sorted by frequency before removing stop words in (a) tokens from Review column and (b) tokens from Title column.

After removing the stop words from both sets of tokens, the content in the dataset becomes clearer in the case of *tokenReview*, now containing highly indicative words. It also confirms the assumption that this textual data is composed of subjective and opinionated text on the topic chosen. As for the set of *tokenTitle*, the 4th most frequent word is “love”, a high indication of positive sentiment, while the 10th most used word is “*don’t*” – a negation that could denote a given prevalence of negative sentiment.

```
> print(tokenReview)
# A tibble: 3,754 × 2
   wordsReview     n
   <chr>       <int>
 1 machine       697
 2 coffee        378
 3 espresso      264
 4 water         219
 5 delonghi      143
 6 grinder       137
 7 time          137
 8 beans         126
 9 milk          113
10 grind         102
# ℹ 3,744 more rows
```

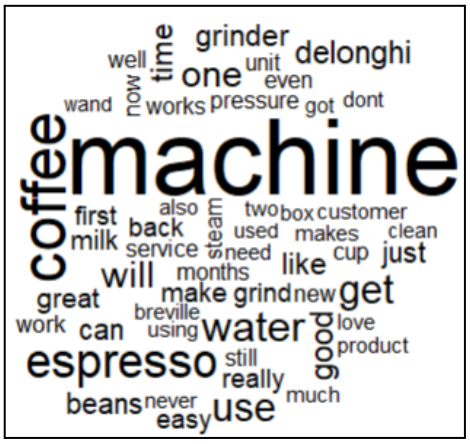
```
> print(tokenTitle)
# A tibble: 462 × 2
   wordsTitle     n
   <chr>      <int>
 1 machine       89
 2 coffee        42
 3 espresso      35
 4 love          17
 5 buy           16
 6 quality       16
 7 months        14
 8 water         14
 9 service       13
10 dont          12
# ℹ 452 more rows
```

Figure 20. Top 10 words sorted by frequency after removing stop words in (a) tokens from Review column and (b) tokens from Title column.

Negative words were not removed, as negation is an important factor to Sentiment Analysis tasks.

One can try and infer several meanings from the words contained in both (a) and (b), however tokens made from single words, n-grams in which n = 1, cannot be used as indicators by themselves as the context is not present. The 6th word in the *tokenTitle* classification, quality, can be associated with both “good quality” and “bad quality” sentences.

To extend a little more, a word cloud can be created to include more words.

(a) 

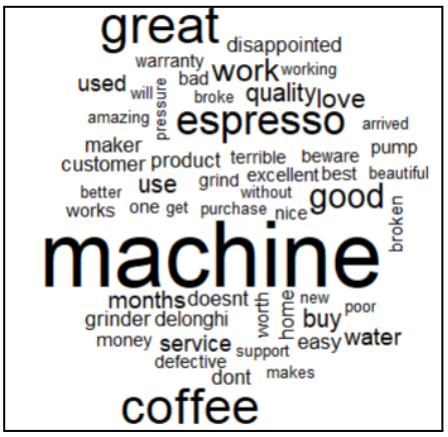
(b) 

Figure 21. Word Clouds for (a*) tokenReview* and (b) *tokenTitle* .

As expected from a dataset regarding a particular product, the most frequent words for both fields, reviews and titles, are coffee, machine and espresso. A few other topics that show up and one can assume users take into consideration when evaluating the product are its pressure, easiness of use and the product's warranty. Interestingly, the word "Breville" among the most commonly used refers to a direct competitor brand, so another assumption is that the product is actively compared to their coffee machines by the market.

Some concerns can be raised by the prevalence of words such as "defective" and "broke". It can be a point to investigate further, especially if one assumes they can be connected to the instance of "warranty" present also.

5.5 Natural language processing

Starting from from the clean and tokenized data (also an NLP technique), four datasets were created to compare the models created based on the textual preprocessing approach chosen, stemming:

- *dataReview*: contains the reviews and their respective labels (ratings), with no additional cleaning steps.
- *dataTitle*: contains the titles and their respective labels (ratings), with no additional cleaning steps.
- *dataReview_ST*: contains the reviews and their respective labels (ratings), but the words contained in it were stemmed.
- *dataTitle_ST*: contains the titles and their respective labels (ratings), but the words contained in it were stemmed.

The idea behind stemming the tokens is to reduce word variation and its impact (by reducing them to their root form), again here with the goal of simplifying the data. For sentiment analysis tasks, specifically, stemming can be advantageous because semantically related terms can then be treated as one, as they should be.

One possible issue stemming may present is that by simply removing the end part of each token, the algorithm assumes its root without basis on a dictionary, such as occurs in the lemmatization technique. Incorrect meaning and spelling may then occur.

The chosen data representation for all four datasets is the TF-IDF, in an attempt to highlight the most important words within the corpora. The idea is to teach the machine learning models what words are more relevant to the label of the review or title, negative (-1), neutral (0) or positive (1) – emphasizing important terms while downplaying the common ones. It can also serve as a manner of feature weighting.

The downside of using this type of text representation is that the matrices can be very sparse, wasting memory and computational power to run.

Both of these techniques, stemming as text preprocessing and defining the TF-IDF as text representation, belong to the NLP field and impact all subsequent steps.

At this point, the researcher can perform several NLP tasks, not only sentiment analysis. Examples of this are document clustering, hoping to cluster reviews of a given label by their common words and themes to better understand possible influences on customer satisfaction, or topic modeling, looking to uncover latent topics on the reviews – categorizing them to gain better insights.

This work then moved on to training the machine learning models.

5.6 ML Model training

As discussed in Section 1.1.3, there are a few ubiquitous models when performing Sentiment Analysis, chief among them Support Vector Machines, Random Forest and Naive Bayes. The last one is a particular choice, due to the small dataset available for the case study.

Before training each model, each dataset was partitioned into a training dataset containing 80% of the reviews and a testing one, containing the remaining 20%. The function *smote()* was used in all of the training datasets to oversample the neutral class (0) and improve the imbalance among the classes.

All models were trained using both 5-fold and 10-fold cross validation for the four datasets: *dataReview*, *dataReview_ST*, *dataTitle* and *dataTitle_ST*. Using the Accuracy metric to evaluate which path is better, it becomes clear that even though the difference is small, with an improvement of around 2%, in general using a 10-fold cross validation strategy yields better models – especially for Support Vector Machines.

The comparison for the models created for each dataset is seen in Figure 22, ordered by Accuracy. Also, some insight on the performance of each one can already be glimpsed: interestingly the performance of the Naive Bayes is the worst through all four datasets, even when the cross-validation process used *tuneGrid()* method to look for the best model combining different La Place smoothing corrections (interval [1:3]) and the use (*true)* or not (*false*) of kernel density estimation – respectively either considering the data as a normal distribution or not.

The Random Forest models were trained using *tuneLength* of 5, meaning the algorithm tested 5 different combinations of hyperparameters and selected the best possible one among them. These are typically the number of trees and the number of random randomly selected predictors per split.

As for the Support Vector Machine models trained, the *tuneLength* value used was 10. Using the Linear SVM (*method = "svmLinear"*), the main hyperparameter to test and compare is C, the cost brought by the trade-off between maximizing the margin and minimizing the errors. At a first glance, considering only the Accuracy metric, SVM seems to be the most promising model.

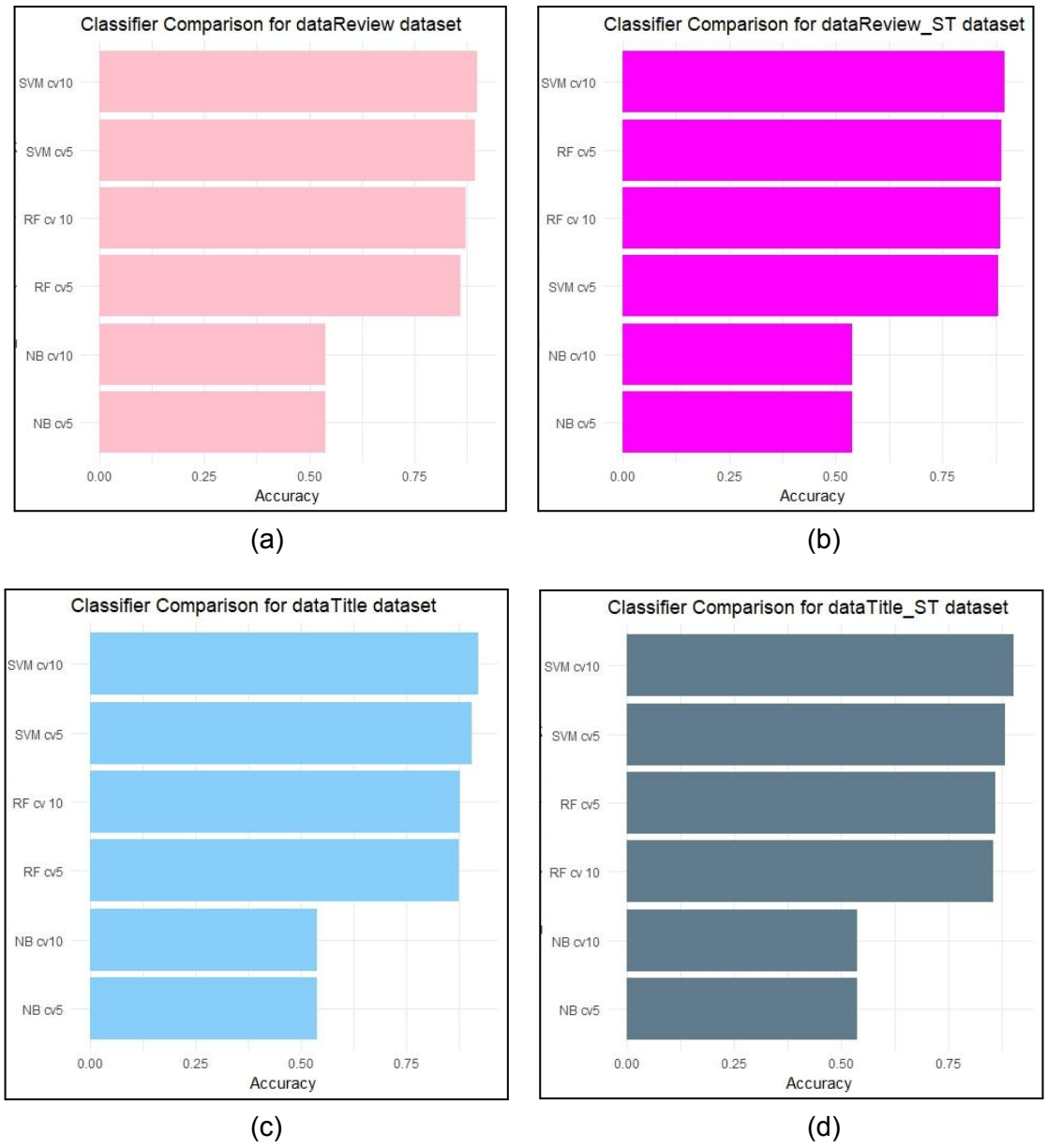


Figure 22. Comparison of Accuracy achieved by each model trained on both 5-fold and 10-fold cross validation, seen for (a) dataReview dataset, (b) dataReview_ST dataset, (c) dataTitle dataset and (d) dataTitle_ST dataset.

Because it is clear that in general the 10-fold cross validation models performed slightly better, the 5-fold cross validation models are no longer evaluated. From this point on, all models shown were created through 10-fold cross validation.

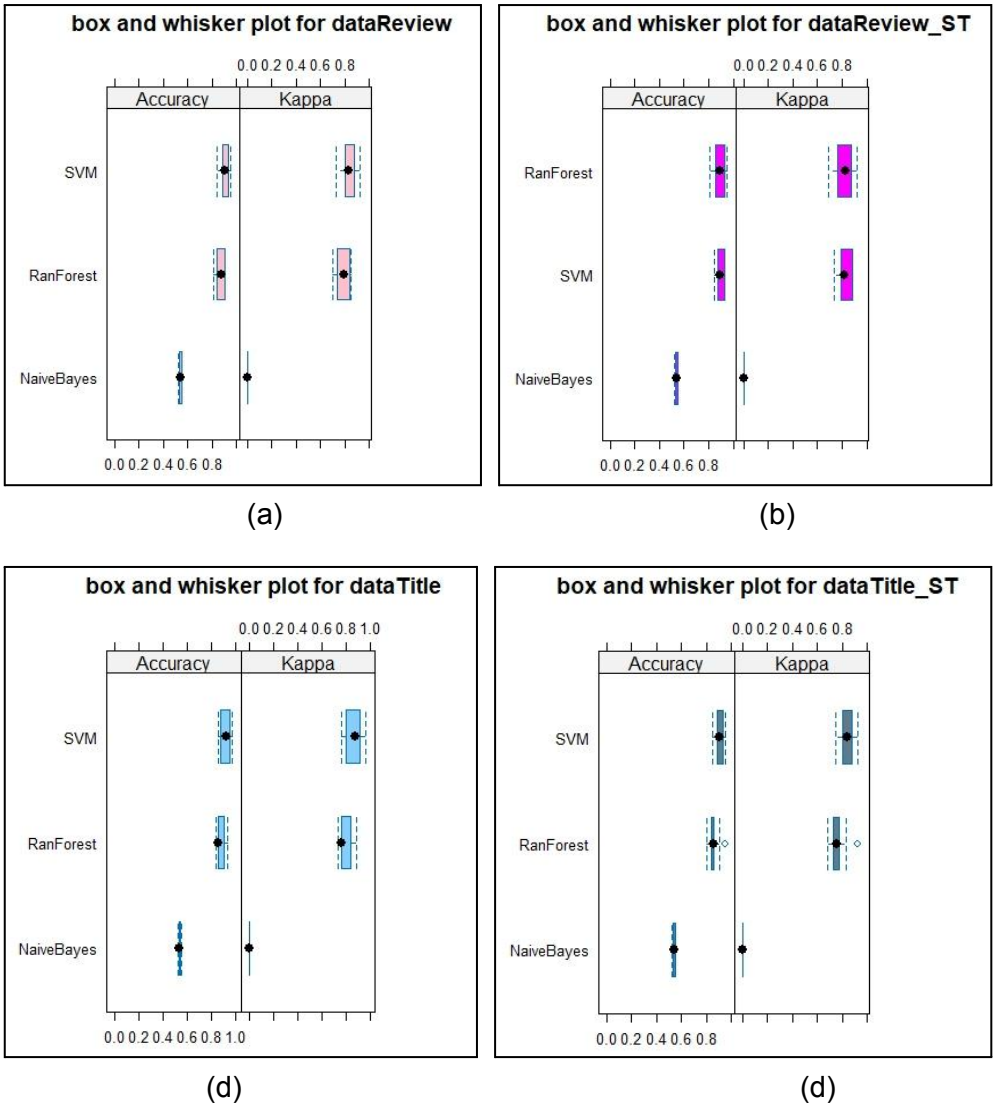


Figure 23. Comparison of Accuracy and Kappa achieved by each model when tested on the testing dataset, seen for (a) dataReview dataset, (b) dataReview_ST dataset, (c) dataTitle dataset and (d) dataTitle_ST dataset.

Another way of visualizing the performance and comparing the models between one another is through the *resamples()* function, testing each one on the test datasets. It is remarked that these have not undergone oversampling, as the training dataset have, remaining imbalanced to better reflect real-world data.

It is also reflected on Figure 23 the performance of the SVM, when compared to both other models: above 80% for all datasets. Also reflected here was the poor performance of the Naive Bayes. This time the models are accompanied by the Kappa value, which reflects the level of agreement or disagreement between the predictions, if they are most likely correct or simply happened by chance such as in a random chance classifier. This value should be maximized, as having a high inter-rate reliability means all models trained are in high agreement with one another.

The Kappa value does not reflect, however, the validity of the previews: it only states that the models agree with one another, not that they are agreeing to the correct answer. That is why it is paired to the Accuracy value.

In general, beyond the poor performance of the Naive Bayes models, they have a very low level of agreement between one another.

In general, the best model for all datasets seems to be SVM, possibly due to its robustness when dealing with large scale features and high dimensional data, except for the stemmed reviews (*dataReview_ST*): this one was better modeled by the Random Forest model. Perhaps it can be attributed to the randomness factor, due to bootstrap sampling and feature sampling the model may have led to a better fit.

The metrics ROC and AUC work for binary classifiers, not multiclass, so they were not used in this work. However, one can perform a deeper analysis using the Confusion Matrix of each model.

Seen on Figure 24 for the dataset *dataReview*, the x-axis Reference represents the true value of each input, while the y-axis Prediction represents the model output. The goal is to have the heatmap busier (darker color) in the central diagonal line, meaning the prediction and the true value match for all three classes.

As expected, the testing data is highly unbalanced as it should reflect real world data and so wasn't corrected: only 7 inputs belonging to the neutral (0) class. It was also the one with the largest number of misclassifications for both the SVM and Random Forest models.

The Naive Bayes model is working essentially as a purely random classifier: all of the inputs are predicted to be neutral. Class imbalance could be the reason for this, the model learning to maximize the predictions by setting all inputs into the majority class. However, the neutral class is the imbalance one and has been corrected for, meaning that other causes are more likely.

Among these, it is possible that the model was not able to effectively discriminate between the features belonging to one class over the other, or it may have violated the assumption of independence between the classes – a core principle of the Naive Bayes model. Either way, this performance means it is a bad candidate for the task and the data.

Both SVM and Random Forest are better at predicting positive reviews, but the latter performs slightly better for both positive (1) and negative ones (-1). This is a point in which one should consider the weight of each class: if the negative reviews have a larger weight, as those are the ones to be considered in further detail by the business to solve the issues contained therein looking to improve customer satisfaction, the Random Forest model could be more interesting. Even for further hyperparameter tuning, looking for better Specificity values (a low 53% in the current configuration).

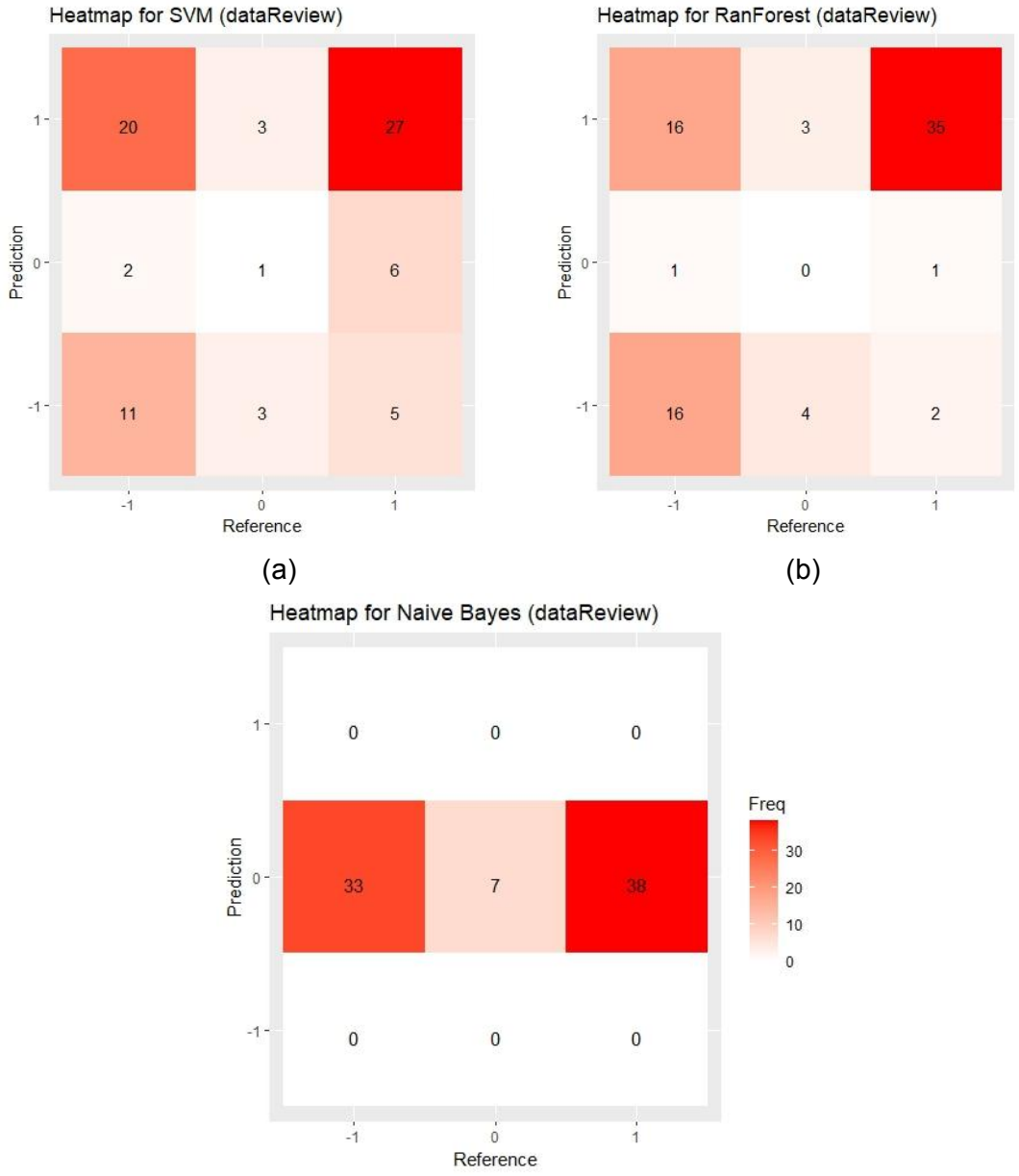


Figure 24. Comparison of the Confusion Matrix of each model learned from the *dataReview* dataset, being (a) SVM, (b) Random Forest and (c) Naive Bayes.

As for the second dataset *dataTitle*, containing non-stemmed titles of each review and the corresponding label, the Naive Bayes model shows the same behavior as seen for the initial dataset and as such can be discarded.

Seen in Figure 25, the most balanced model seems to be SVM and its heat map reflects the desired highlighted central diagonal. It shows, however, a tendency to classify neutral (0) inputs as positive (1), classifying over half of the available inputs as such.

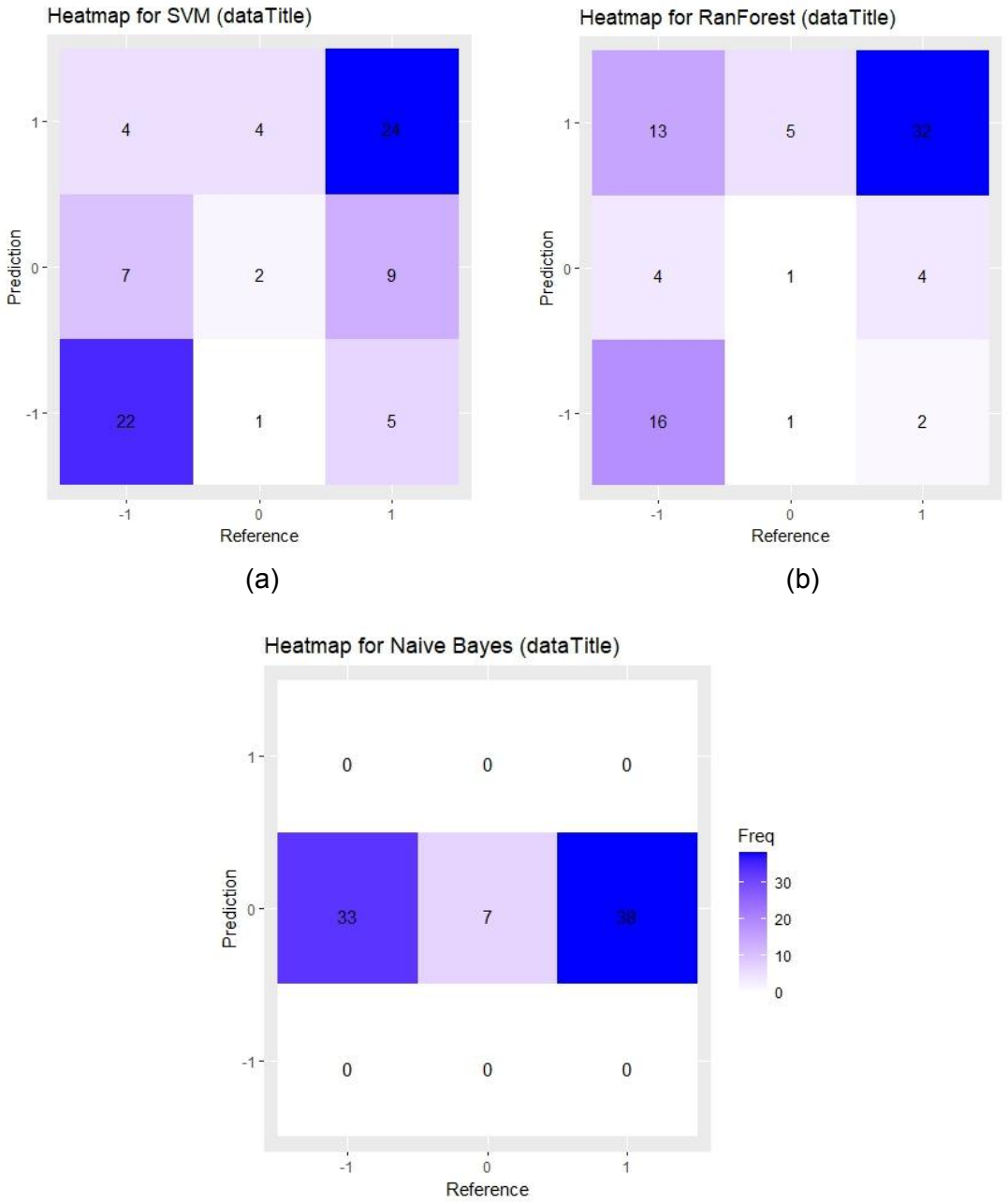


Figure 25. Comparison of the Confusion Matrix of each model learned from the *dataTitle* dataset, being (a) SVM, (b) Random Forest and (c) Naive Bayes.

In this case, the SVM model shows a better aptitude to correctly classifying negative reviews. However, its capacity regarding neutral classes is less than ideal – misclassifying a total of 4 inputs as positive and 1 as negative, from the total of 7 reviews within the testing dataset. Many of the positive reviews were also misclassified as neutral.

In general, it is the best model so far, seemingly balanced and good for classifying negative reviews correctly.

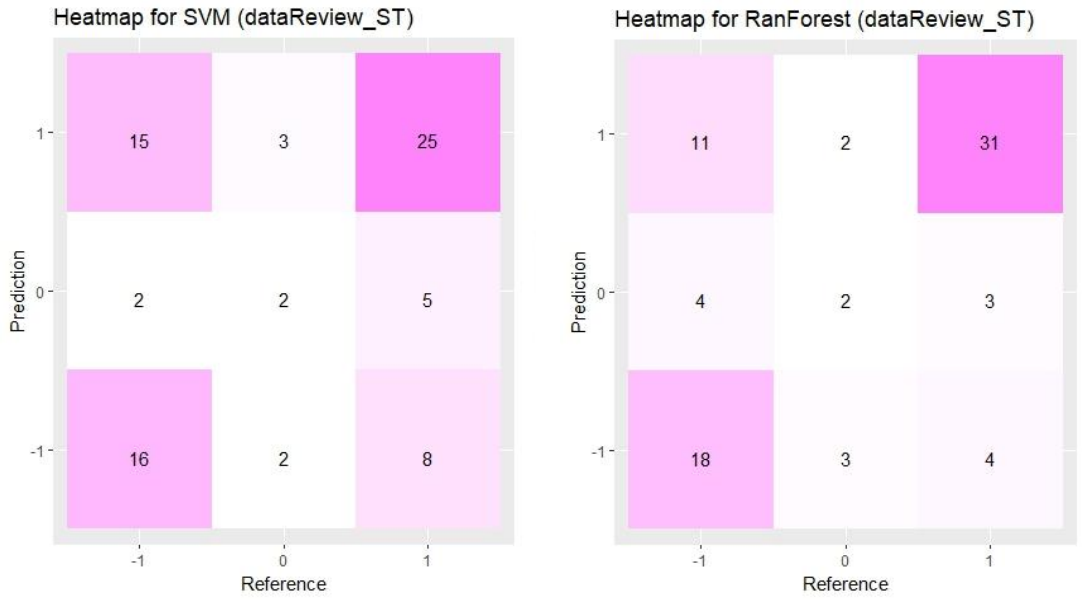
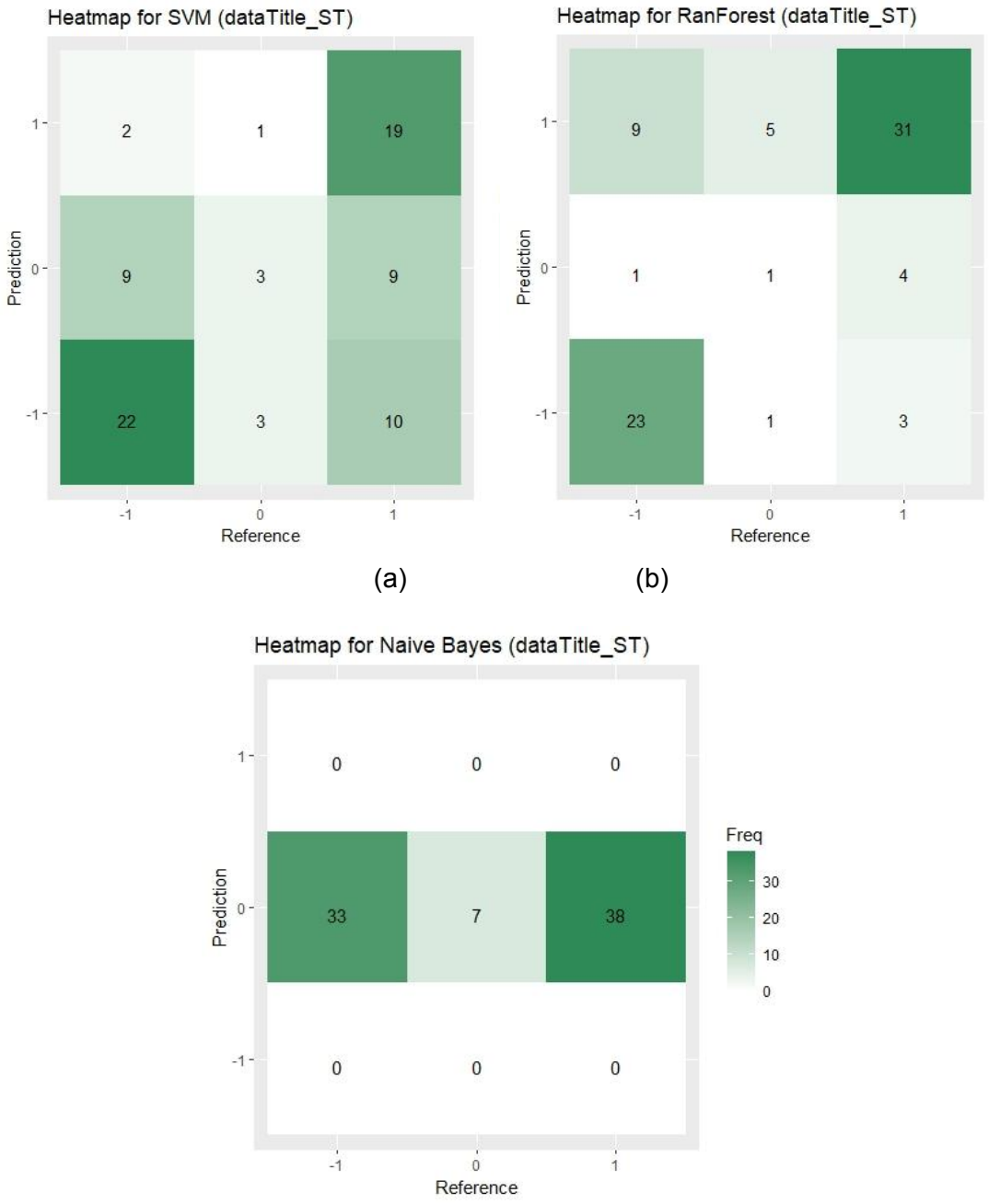


Figure 26. Comparison of the Confusion Matrix of each model learned from the *dataReview_ST* dataset, being (a) SVM, (b) Random Forest and (c) Naive Bayes.

Now, to check whether stemming the datasets improved the models trained, Figure 26 presents the Confusion Matrix heat maps for *dataReview_ST*. The initial impression is that the Naive Bayes model maintains the same behavior already seen and discussed.

Taking one step further to stem the data does not appear to have made an impact on the resulting models' performance. On the contrary, where the Random Forest model for non-stemmed titles correctly classified 35 positive reviews, the same model for the stemmed titles dataset achieved 31 correct classifications for the class. There was a slight increase for the correct classifications of the negative reviews, going from 16 correct predictions to 18.

The same is true for the SVM model, in which the non-stemmed data model correctly classified 27 positive reviews (from the total of 38) while the stemmed data one recognized 25 as such. A slight improvement for negative reviews, again, going from 11 in the former to 16 in the latter.



(a) (b)

(c)

Figure 27. Comparison of the Confusion Matrix of each model learned from the *dataTitle_ST* dataset, being (a) SVM, (b) Random Forest and (c) Naive Bayes.

Lastly, in figure 27 the Confusion Matrix heat maps for the models trained on the fourth dataset, *dataTitle_ST*, stemmed titles. Already expected, the Naive Bayes maintains the

behavior of a random classifier, but this time the SVM shows the poorest performance so far for predicting positive reviews. It kept, however, the same level of Sensitivity as the non-stemmed title dataset, correctly classifying 22 out of 33 negative reviews.

5.7 Model Selection

A few considerations to be taken into account when selecting the model, beyond just its performance, are its interpretability and computational efficiency. In the former case, as this task is not designed for topic modeling or understanding the reasoning behind the classification, interpretability is not a priority and can be foregone. As for the latter, the efficiency is important – but can be manipulated with optimizations such as minimizing the vocabulary size and using n-grams sparingly (both implemented in this project).

The idea of this process has been to solve complex business problems and generate valuable insights, while testing different NLP approaches. If one takes the stand that negative sentiments are a priority, as the assumption that they represent the points to be improved upon for achieving better customer satisfaction, there are two promising models to further refine. Both of these models were trained using the titles, instead of the reviews themselves, confirming the initial assumption that this data may me more straight to the point and contain words that better reflect the class.

The first was a Support Vector Machine model trained on the non-stemmed dataset *dataTitle,* correctly classifying 22 negative reviews out of a total of 33, and 24 positive ones, out of 38. It seems to be generally equilibrated and can be worked on further to minimize false positives and maximize true negatives. This could be done through adjusting the decision threshold to reflect the priority set, and the testing of different kernels – beyond just the linear one used.

The second promising model is the Random Forest trained on the stemmed dataset *dataTitle_ST*, which correctly classified 31 positive reviews out of a total of 38, and 23 negative ones out of 33.

6

Conclusion

In this work three main objectives were set: to analyze the applicability of NLP and Machine Learning techniques in solving complex business problems, to evaluate the associated technical challenges and limitations and lastly, to compare and discuss the results obtained through the different machine learning models developed.

The first and second ones, extensively examined in Sections 2 to 4 and later tested in the case study, have shown that the technology and resources currently available can bring immense benefits to data-driven decision making in the business context. By collecting relevant, opinionated user generated content available online (product reviews) and manipulating it using NLP techniques and good practices, such as tokenization, TF-IDF text representation and standard data cleaning, valuable insights can be extracted.

Some of the technical challenges discussed and overcome were the data itself and its cleaning: UGC can contain several issue such as slangs, typos and non-recognized characters like emojis, and the size of textual data can bring a large impact on computational efficiency – this was dealt with through minimizing the vocabulary to only necessary terms and limiting the number of n-grams using only n = 1.

The two suppositions made initially garnered interesting results: the first, comparing machine learning models trained on the reviews themselves and models trained on their titles, assuming that the latter would contain more direct, valuable words. The second, comparing models learned from stemmed and non-stemmed data, to evaluate the impact of the technique – specially because the dataset is considered small, and so could benefit from condensing the tokens into their essential stems.

In total, four datasets were created, two using non-stemmed data (from the reviews themselves, *dataReview*, and from the titles, *dataTitle)* and two using stemmed data (from the reviews, *dataReview_ST*, and from the titles, *dataTitle_ST*). To further condense and simplify the data, the original 5-star rating system used as class labels was transformed into polarities: positive (1), negative (-1) and neutral (0).

The four datasets were used to learn three models widely successful in Sentiment Analysis tasks: Support Vector Machine, Random Forest and Naive Bayes. The last one was chosen specifically because of its renowned capability of generalizing well from smaller datasets.

All of the models were tested with both 5-fold and 10-fold cross validation, the latter one showing better performance by a small margin, and comparing various combinations of hyperparameter tuning.

The final resulting models showed interesting results: the initial bet on the Naive Bayes did not provide the expected results, as all of them behaved as general random classifiers and so were discarded.

On the other hand, the assumptions made about using the titles instead of the reviews themselves confirmed their promise: the two best models trained were done so using the titles TF-IDF representation. The first one, the SVM model trained on the *dataTitle* dataset and the second one, the Random Forest model trained on the stemmed *dataTitle_ST* dataset.

All models classified better for the positive class, which should be taken into consideration. If the idea is to use Machine Learning models to identify possible issues to improve upon and achieve better customer satisfaction levels, the focus should be on correctly identifying negative instances. So this is the first suggestion for future work development, to expand on the trade-off between Sensitivity (correctly spotting true positives) and Specificity (correctly identifying negative values) to emphasize the latter.

Further considerations to develop are: introduction of higher level n-grams with the intention of expanding the context within the dataset (as unigrams contain no context whatsoever) and attempting the use of larger, more balanced datasets which would also allow for a deep learning approach. Another suggestion to be tested is using a different, hybrid technique, to pair the models with an appropriate lexicon and expand on context.

Lastly, specifically for acquisition of insights, it is suggested to perform topic modeling tasks to evaluate not only the consumer satisfaction, but also the factors that influence it.

7

R code

```
library(readxl) # data
library(tibble) # data
library(ggplot2) # visualization
library(tidytext) # tokenization
library(dplyr) # data manipulation
library(tm)  # text mining
library(SnowballC) # stemming
library(caret) # machine learning models
library(pROC) # for evaluating classifiers

########################## ORIGINAL DATASET ##############################
# import dataset as tibble
coffeeReview <- read_excel("coffeeReviews_int_clean_transf.xlsx",
                           col_names = TRUE)
coffeeReview %>% print()

# brief overview of variables and data types
coffeeReview %>% glimpse()

# mean of rating scores in the data set
coffeeReview$ratingScore %>% mean()

# data plot by date or review
qplot(date,
      data = coffeeReview,
      xlab = "Date",
      ylab = "Number of Reviews")

qplot(ratingScore,
      data = coffeeReview,
      xlab = "Rating",
      ylab = "Number of Reviews")

# Number of inputs per rating score 1 to 5
coffeeReview %>% filter(ratingScore==-1) %>% count() # 167
```

```
coffeeReview %>% filter(ratingScore==0) %>% count() # 39
coffeeReview %>% filter(ratingScore==1) %>% count() # 193

# Dealing with missing values in the column reviewReaction
library(tidyr)
coffeeReview %>% summarise(count = sum(is.na(reviewReaction)))

coffeeReview <- coffeeReview %>%
  replace_na(list(reviewReaction="0 people found this useful"))

# Check for duplicated data
sum(duplicated(coffeeReview)) # zero

# Remove numbers from reviewDescription and reviewTitle columns
coffeeReview$reviewTitle <- gsub(pattern = "[[:digit:]]",
                                 replacement = "",
                                 coffeeReview$reviewTitle)

coffeeReview$reviewDescription <- gsub(pattern = "[[:digit:]]",
                                       replacement = "",
                                       coffeeReview$reviewDescription)

coffeeReview %>% select(6,8) %>% print(n=50)

# Set all characters to lowercase
coffeeReview$reviewTitle <- tolower(coffeeReview$reviewTitle)

coffeeReview$reviewDescription <- tolower(coffeeReview$reviewDescription)

coffeeReview %>% select(6,8) %>% print(n=50)

# Remove punctuation from reviewDescription and reviewTitle columns
coffeeReview$reviewTitle <- gsub(pattern = "[[:punct:]]",
                                 replacement = "",
                                 coffeeReview$reviewTitle)

coffeeReview$reviewDescription <- gsub(pattern = "[[:punct:]]",
                                       replacement = "",
                                       coffeeReview$reviewDescription)

coffeeReview %>% select(6,8) %>% print(n=80)

########################## TOKENIZATION ############################

tokenReview_full <- coffeeReview %>% unnest_tokens(output = "wordsReview",
                                                   input = reviewDescription,
                                                   token = "words") %>%
                                 count(wordsReview, sort = TRUE)

tokenTitle_full <- coffeeReview %>% unnest_tokens(output = "wordsTitle",
                                                  input = reviewTitle,
                                                  token = "words") %>%
                                 count(wordsTitle, sort = TRUE)

print(tokenReview_full)
print(tokenTitle_full)

# remove stop words from tokenized data
tokenReview <- tokenReview_full %>%
```

```
  anti_join(stop_words, by = c("wordsReview" = "word"))

tokenTitle <- tokenTitle_full %>%
  anti_join(stop_words, by = c("wordsTitle" = "word"))

print(tokenReview)
print(tokenTitle)

############################ WORD CLOUD ##############################

library(wordcloud)

wcReview <- coffeeReview %>%
  select(reviewDescription) %>%
  summarize(text = paste(reviewDescription, collapse = " ")) %>%
  pull(text)

wcReview <- removeWords(wcReview, stopwords("english"))

wordcloud(words = strsplit(wcReview, " ")[[1]], min.freq = 60, random.color=FALSE)

wcTitle <- coffeeReview %>%
  select(reviewTitle) %>%
  summarize(text = paste(reviewTitle, collapse = " ")) %>%
  pull(text)

wcTitle <- removeWords(wcTitle, stopwords("english"))

wordcloud(words = strsplit(wcTitle, " ")[[1]], min.freq = 5, random.color=FALSE)


###################### BoW (tokenized no stop words) ######################

bowReview <- coffeeReview %>%
  unnest_tokens(output = "wordsReview", token = "words", input =
reviewDescription) %>%
  anti_join(stop_words, by = c("wordsReview" = "word")) %>%
  count(id, wordsReview, sort = TRUE)
bowReview

bowTitle <- coffeeReview %>%
  unnest_tokens(output = "wordsTitle", token = "words", input = reviewTitle) %>%
  anti_join(stop_words, by = c("wordsTitle" = "word")) %>%
  count(id, wordsTitle, sort = TRUE)

bowReview
bowTitle

########################## TF-IDF REPRESENTATION ##########################

coffeeReview <- coffeeReview %>%
  mutate(ratingScore = factor(ratingScore))

tfidfReview <- coffeeReview %>% unnest_tokens(output = "wordsReview",
                                              input = reviewDescription,
                                              token = "words") %>%
                            anti_join(stop_words, by = c("wordsReview" = "word"))
%>%
```

```
                                count(wordsReview, sort = TRUE) #%>% loses all
other columns
                                bind_tf_idf(wordsReview, id, n)


tfidfTitle <- coffeeReview %>% unnest_tokens(output = "wordsTitle",
                                             input = reviewTitle,
                                             token = "words") %>%
                                anti_join(stop_words, by = c("wordsTitle" =
"word")) %>%
                                count(wordsTitle, sort = TRUE) #%>% loses all
other columns
                                #bind_tf_idf(wordsTitle, id, n)


# create a corpus from the original tibble dataset

corpusReview <- VCorpus(VectorSource(coffeeReview$reviewDescription))
head(meta(corpusReview))
corpusReview

corpusTitle <- VCorpus(VectorSource(coffeeReview$reviewTitle))
head(meta(corpusTitle))
corpusTitle

inspect(corpusReview[1:5])
inspect(corpusTitle[[58]])

# further text cleaning
corpusReview <- tm_map(corpusReview, stripWhitespace) #remove extra whitespace
corpusTitle <- tm_map(corpusTitle, stripWhitespace)

corpusReview <- tm_map(corpusReview, removeWords, stopwords("english")) #remove
stop words
corpusTitle <- tm_map(corpusTitle, removeWords, stopwords("english"))

STcorpusReview <- tm_map(corpusReview, stemDocument) #stemming
STcorpusTitle <- tm_map(corpusTitle, stemDocument)

# By this point, we have 4 datasets: non stemmed (corpusReview / corpusTitle)
# and stemmed: (STcorpusReview / STcorpusTitle)

#Create the Document Term Matrix based on TF-IDF value
#1
dtmReview <- DocumentTermMatrix(corpusReview,
                                control = list(weighting = function(x)
                                                    weightTfIdf(x, normalize =
                                                                FALSE)))
inspect(dtmReview[1:10, 300:320])

#2
dtmTitle <- DocumentTermMatrix(corpusTitle,
                                control = list(weighting = function(x)
                                weightTfIdf(x, normalize =
                                            FALSE)))
inspect(dtmTitle[1:10, 400:420])

#3
dtmReview_ST <- DocumentTermMatrix(STcorpusReview,
```

```
                                control = list(weighting = function(x)
                                      weightTfIdf(x, normalize =
                                                  FALSE)))
inspect(dtmReview_ST[1:10, 300:320])

#4
dtmTitle_ST <- DocumentTermMatrix(STcorpusTitle,
                                control = list(weighting = function(x)
                                weightTfIdf(x, normalize =
                                                  FALSE)))
inspect(dtmTitle_ST[1:10, 300:320])

# Convert matrices into dataframes
#1
matrixReview <- as.matrix(dtmReview)
View(matrixReview[1:30,1:30])

dfReview <- as.data.frame(as.matrix(dtmReview))
View(dfReview)

#2
matrixTitle <- as.matrix(dtmTitle)
View(matrixTitle[1:30,1:30])

dfTitle <- as.data.frame(as.matrix(dtmTitle))
View(dfTitle)

#3
matrixReview_ST <- as.matrix(dtmReview_ST)
View(matrixReview_ST[1:30,1:30])

dfReview_ST <- as.data.frame(as.matrix(dtmReview_ST))
head(dfReview_ST)

#4
matrixTitle_ST <- as.matrix(dtmTitle_ST)
View(matrixTitle_ST[1:30,1:30])

dfTitle_ST <- as.data.frame(as.matrix(dtmTitle_ST))
head(dfTitle_ST)

# By this point, the four datasets are in the form of dataframes, accounting for
TF-IDF of each term in each document
# they are (dfReview / dfTitle) not stemmed and (dfReview_ST / dfTitle_ST) stemmed

# Extract rating score from original dataset and add to the clean data as labels
for training

scores <- coffeeReview$ratingScore

#1 DATASET
dataReview <- data.frame(
  Sentiment = factor(scores),
  dfReview
)
View(dataReview)

#2 DATASET
dataTitle <- data.frame(
```

```
  Sentiment = factor(scores),
  dfTitle
)
View(dataTitle)

#3 DATASET
dataReview_ST <- data.frame(
  Sentiment = factor(scores),
  dfReview_ST
)
View(dataReview_ST)

#4 DATASET
dataTitle_ST <- data.frame(
  Sentiment = factor(scores),
  dfTitle_ST
)
View(dataTitle_ST)

###################### DATA FOR MACHINE LEARNING MODELS ######################
library(performanceEstimation)
varUnder = 6
varOver = 1

###### dataReview DATASET (not stemmed TF-IDF representation of reviews) ######

# split train and test datasets, and adjust unbalanced classes (oversampling with
smote function)
set.seed(1234)
trainIndex1 <- createDataPartition(dataReview$Sentiment, p = 0.8, list = FALSE)
train_dataReview <- dataReview[trainIndex1, ]
test_dataReview <- dataReview[-trainIndex1, ]

smote_dataReview <- smote(Sentiment ~ ., train_dataReview, perc.over =
varUnder,perc.under=varOver)


####### dataTitle DATASET (not stemmed TF-IDF representation of titles) #######

# split train and test datasets, and adjust unbalanced classes (oversampling with
smote function)
set.seed(1234)
trainIndex2 <- createDataPartition(dataTitle$Sentiment, p = 0.8, list = FALSE)
train_dataTitle <- dataTitle[trainIndex2, ]
test_dataTitle <- dataTitle[-trainIndex2, ]

smote_dataTitle <- smote(Sentiment ~ ., train_dataTitle, perc.over =
varUnder,perc.under=varOver)


####### dataReview_ST DATASET (stemmed TF-IDF representation of reviews) ######

# split train and test datasets, and adjust unbalanced classes (oversampling with
smote function)
set.seed(1234)
trainIndex3 <- createDataPartition(dataReview_ST$Sentiment, p = 0.8, list = FALSE)
train_dataReview_ST <- dataReview_ST[trainIndex3, ]
test_dataReview_ST <- dataReview_ST[-trainIndex3, ]
```

```
smote_dataReview_ST <- smote(Sentiment ~ ., train_dataReview_ST, perc.over =
varUnder,perc.under=varOver)


######## dataTitle_ST DATASET (stemmed TF-IDF representation of titles) ########

# split train and test datasets, and adjust unbalanced classes (oversampling with
smote function)
set.seed(1234)
trainIndex4 <- createDataPartition(dataTitle_ST$Sentiment, p = 0.8, list = FALSE)
train_dataTitle_ST <- dataTitle_ST[trainIndex4, ]
test_dataTitle_ST <- dataTitle_ST[-trainIndex4, ]

smote_dataTitle_ST <- smote(Sentiment ~ ., train_dataTitle_ST, perc.over =
varUnder,perc.under=varOver)

######################### MACHINE LEARNING MODELS ########################

#1 dataReview DATASET (not stemmed TF-IDF representation of reviews)
# SUPPORT VECTOR MACHINE
set.seed(1234) #linear svm 5 fold cv
svm5_dataReview <- train(form=Sentiment~.,
                         data=smote_dataReview,
                         method = "svmLinear",
                         trControl = ctrl5,
                         preProcess = c("center","scale"),
                         tuneLength=10,
                         metric="Accuracy"
)
svm5_dataReview

set.seed(1234) #linear svm 10 fold cv
svm10_dataReview <- train(form=Sentiment~.,
                         data=smote_dataReview,
                         method = "svmLinear",
                         trControl = ctrl10,
                         preProcess = c("center","scale"),
                         tuneLength=10,
                         metric="Accuracy"
)
svm10_dataReview # difference in accuracy from 5 fold to 10 fold svm is less than
1%

# RANDOM FOREST
set.seed(1234)
rf5_dataReview<-train(form=Sentiment~.,
                    data=smote_dataReview,
                    method="rf",
                    tuneLength = 10,
                    trControl=ctrl5,
                    metric="Accuracy"
)
rf5_dataReview

set.seed(1234)
rf10_dataReview<-train(form=Sentiment~.,
                    data=smote_dataReview,
                    method="rf",
                    tuneLength = 10,
```

```
                    trControl=ctrl10,
                    metric="Accuracy"
)
rf10_dataReview

# NAIVE BAYES
set.seed(1234)
nb_dataReview <- train(form=Sentiment~.,
                           data=smote_dataReview,
                           method = "naive_bayes",
                           trControl = ctrl5,
                           tuneGrid = expand.grid(    # try different laplace
corrections
                           usekernel = c(TRUE, FALSE),
                           laplace = 0:3,
                           adjust = 1:3
  ),
  preProcess = c("zv", "center", "scale")
  # center and scale the data
)
nb_dataReview

set.seed(1234)
nb10_dataReview <- train(form=Sentiment~.,
                    data=smote_dataReview,
                    method = "naive_bayes",
                    trControl = ctrl10,
                    tuneGrid = expand.grid(    # try different laplace corrections
                           usekernel = c(TRUE, FALSE),
                           laplace = 0:3,
                           adjust = 1:3
                    ),
                    preProcess = c("zv", "center", "scale")
                    # center and scale the data
)
nb10_dataReview

#2 dataTitle DATASET (not stemmed TF-IDF representation of titles)
# SUPPORT VECTOR MACHINE
set.seed(1234) #linear svm 5 fold cv
svm5_dataTitle <- train(form=Sentiment~.,
                           data=smote_dataTitle,
                           method = "svmLinear",
                           trControl = ctrl5,
                           preProcess = c("center","scale"),
                           tuneLength=10,
                           metric="Accuracy"
)
svm5_dataTitle

set.seed(1234) #linear svm 10 fold cv
svm10_dataTitle <- train(form=Sentiment~.,
                           data=smote_dataTitle,
                           method = "svmLinear",
                           trControl = ctrl10,
                           preProcess = c("center","scale"),
                           tuneLength=10,
                           metric="Accuracy"
)
```

```
svm10_dataTitle # difference of 2% in accuracy

# RANDOM FOREST
set.seed(1234)
rf5_dataTitle<-train(form=Sentiment~.,
                     data=smote_dataTitle,
                     method="rf",
                     tuneLength = 10,
                     trControl=ctrl5,
                     metric="Accuracy"
)
rf5_dataTitle

set.seed(1234)
rf10_dataTitle<-train(form=Sentiment~.,
                     data=smote_dataTitle,
                     method="rf",
                     tuneLength = 10,
                     trControl=ctrl10,
                     metric="Accuracy"
)
rf10_dataTitle

# NAIVE BAYES
set.seed(1234)
nb_dataTitle <- train(form=Sentiment~.,
                     data=smote_dataTitle,
                     method = "naive_bayes",
                     trControl = ctrl5,
                     tuneGrid = expand.grid(    # try different laplace corrections
                            usekernel = c(TRUE, FALSE),
                            laplace = 0:3,
                            adjust = 1:3
                     ),
                     preProcess = c("zv", "center", "scale")
                     # center and scale the data
)
nb_dataTitle

set.seed(1234)
nb10_dataTitle <- train(form=Sentiment~.,
                     data=smote_dataTitle,
                     method = "naive_bayes",
                     trControl = ctrl10,
                     tuneGrid = expand.grid(    # try different laplace corrections
                            usekernel = c(TRUE, FALSE),
                            laplace = 0:3,
                            adjust = 1:3
                     ),
                     preProcess = c("zv", "center", "scale")
                     # center and scale the data
)
nb10_dataTitle

#3 dataReview_ST DATASET (stemmed TF-IDF representation of reviews)
# SUPPORT VECTOR MACHINE
set.seed(1234) #linear svm 5 fold cv
svm5_dataReview_ST <- train(form=Sentiment~.,
                            data=smote_dataReview_ST,
```

```
                          method = "svmLinear",
                          trControl = ctrl5,
                          preProcess = c("center","scale"),
                          tuneLength=10,
                          metric="Accuracy"
)
svm5_dataReview_ST

set.seed(1234) #linear svm 10 fold cv
svm10_dataReview_ST <- train(form=Sentiment~.,
                          data=smote_dataReview_ST,
                          method = "svmLinear",
                          trControl = ctrl10,
                          preProcess = c("center","scale"),
                          tuneLength=10,
                          metric="Accuracy"
)
svm10_dataReview_ST # essentially no difference between 5 and 10 fold cv

# RANDOM FOREST
set.seed(1234)
rf5_dataReview_ST<-train(form=Sentiment~.,
                     data=smote_dataReview_ST,
                     method="rf",
                     tuneLength = 10,
                     trControl=ctrl5,
                     metric="Accuracy"
)
rf5_dataReview_ST

set.seed(1234)
rf10_dataReview_ST<-train(form=Sentiment~.,
                     data=smote_dataReview_ST,
                     method="rf",
                     tuneLength = 10,
                     trControl=ctrl10,
                     metric="Accuracy"
)
rf10_dataReview_ST

# NAIVE BAYES
set.seed(1234)
nb_dataReview_ST <- train(form=Sentiment~.,
                     data=smote_dataReview_ST,
                     method = "naive_bayes",
                     trControl = ctrl5,
                     tuneGrid = expand.grid(  # try different laplace corrections
                            usekernel = c(TRUE, FALSE),
                            laplace = 0:3,
                            adjust = 1:3
                     ),
                     preProcess = c("zv", "center", "scale")
                     # center and scale the data
)
nb_dataReview_ST

set.seed(1234)
nb10_dataReview_ST <- train(form=Sentiment~.,
                          data=smote_dataReview_ST,
```

```
                          method = "naive_bayes",
                          trControl = ctrl10,
                          tuneGrid = expand.grid(  # try different laplace
corrections
                          usekernel = c(TRUE, FALSE),
                          laplace = 0:3,
                          adjust = 1:3
                          ),
                          preProcess = c("zv", "center", "scale")
                          # center and scale the data
)
nb10_dataReview_ST

#4 dataTitle_ST DATASET (stemmed TF-IDF representation of titles)
# SUPPORT VECTOR MACHINE
set.seed(1234) #linear svm 5 fold cv
svm5_dataTitle_ST <- train(form=Sentiment~.,
                          data=smote_dataTitle_ST,
                          method = "svmLinear",
                          trControl = ctrl5,
                          preProcess = c("center","scale"),
                          tuneLength=10,
                          metric="Accuracy"
)
svm5_dataTitle_ST

set.seed(1234) #linear svm 10 fold cv
svm10_dataTitle_ST <- train(form=Sentiment~.,
                          data=smote_dataTitle_ST,
                          method = "svmLinear",
                          trControl = ctrl10,
                          preProcess = c("center","scale"),
                          tuneLength=10,
                          metric="Accuracy"
)
svm10_dataTitle_ST

# RANDOM FOREST
set.seed(1234)
rf5_dataTitle_ST<-train(form=Sentiment~.,
                          data=smote_dataTitle_ST,
                          method="rf",
                          tuneLength = 10,
                          trControl=ctrl5,
                          metric="Accuracy"
)
rf5_dataTitle_ST

set.seed(1234)
rf10_dataTitle_ST<-train(form=Sentiment~.,
                          data=smote_dataTitle_ST,
                          method="rf",
                          tuneLength = 10,
                          trControl=ctrl10,
                          metric="Accuracy"
)
rf10_dataTitle_ST

# NAIVE BAYES
```

```
set.seed(1234)
nb_dataTitle_ST <- train(form=Sentiment~.,
                         data=smote_dataTitle_ST,
                         method = "naive_bayes",
                         trControl = ctrl5,
                         tuneGrid = expand.grid(  # try different laplace
corrections
                         usekernel = c(TRUE, FALSE),
                         laplace = 0:3,
                         adjust = 1:3
                         ),
                         preProcess = c("zv", "center", "scale")
                         # center and scale the data
)
nb_dataTitle_ST

set.seed(1234)
nb10_dataTitle_ST <- train(form=Sentiment~.,
                         data=smote_dataTitle_ST,
                         method = "naive_bayes",
                         trControl = ctrl10,
                         tuneGrid = expand.grid(  # try different laplace
corrections
                         usekernel = c(TRUE, FALSE),
                         laplace = 0:3,
                         adjust = 1:3
                         ),
                         preProcess = c("zv", "center", "scale")
                         # center and scale the data
)
nb10_dataTitle_ST

###################### MODEL VISUALIZATION FOR CV ##########################

library(forcats)
classifiers_dataReview <- c("SVM cv5", "SVM cv10", "RF cv5", "RF cv 10", "NB cv5",
"NB cv10" )
accuracy_dataReview <- c(0.8942628, 0.8991565, 0.8582042, 0.8704380, 0.5384703,
0.5385157)
dataframe_dataReview <- data.frame(Classifier = classifiers_dataReview, Accuracy =
accuracy_dataReview)

ggplot(dataframe_dataReview, aes(x = Accuracy, y = fct_reorder(Classifier,
Accuracy))) +
  geom_bar(stat = "identity", fill = "pink") +
  labs(title = "Classifier Comparison for dataReview dataset", x = "Accuracy") +
  theme_minimal()

classifiers_dataReview_ST <- c("SVM cv5", "SVM cv10", "RF cv5", "RF cv 10", "NB
cv5", "NB cv10" )
accuracy_dataReview_ST <- c(0.8797476, 0.8961309, 0.8893287, 0.8871304, 0.5384682,
0.5385157)
dataframe_dataReview_ST <- data.frame(Classifier = classifiers_dataReview_ST,
Accuracy = accuracy_dataReview_ST)

ggplot(dataframe_dataReview_ST, aes(x = Accuracy, y = fct_reorder(Classifier,
Accuracy))) +
  geom_bar(stat = "identity", fill = "magenta") +
```

```
  labs(title = "Classifier Comparison for dataReview_ST dataset", x = "Accuracy")
+
  theme_minimal()

classifiers_dataTitle <- c("SVM cv5", "SVM cv10", "RF cv5", "RF cv 10", "NB cv5",
"NB cv10" )
accuracy_dataTitle <- c(0.9061675, 0.9207537, 0.8750143, 0.8775456, 0.5384682,
0.5385157)
dataframe_dataTitle <- data.frame(Classifier = classifiers_dataTitle, Accuracy =
accuracy_dataTitle)

ggplot(dataframe_dataTitle, aes(x = Accuracy, y = fct_reorder(Classifier,
Accuracy))) +
  geom_bar(stat = "identity", fill = "lightskyblue") +
  labs(title = "Classifier Comparison for dataTitle dataset", x = "Accuracy") +
  theme_minimal()

classifiers_dataTitle_ST <- c("SVM cv5", "SVM cv10", "RF cv5", "RF cv 10", "NB
cv5", "NB cv10" )
accuracy_dataTitle_ST <- c(0.8845955, 0.9036776, 0.8605278, 0.8560535, 0.5384682,
0.5385157)
dataframe_dataTitle_ST <- data.frame(Classifier = classifiers_dataTitle_ST,
Accuracy = accuracy_dataTitle_ST)

ggplot(dataframe_dataTitle_ST, aes(x = Accuracy, y = fct_reorder(Classifier,
Accuracy))) +
  geom_bar(stat = "identity", fill = "lightskyblue4") +
  labs(title = "Classifier Comparison for dataTitle_ST dataset", x = "Accuracy") +
  theme_minimal()

############### PREDICTION CAPABILITIES AND CONFUSION MATRIX ##############

#1 dataReview DATASET (non stemmed TF-IDF representation of reviews)

svm5_dataReview_predict <- predict(svm5_dataReview, newdata = test_dataReview)
svm5_dataReview_cm <- confusionMatrix(svm5_dataReview_predict,
test_dataReview$Sentiment)
svm5_dataReview_cm$overall['Accuracy']

svm10_dataReview_predict <- predict(svm10_dataReview, newdata = test_dataReview)
svm10_dataReview_cm <- confusionMatrix(svm10_dataReview_predict,
test_dataReview$Sentiment)
svm10_dataReview_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_svm10_dataReview <- as.matrix(svm10_dataReview_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_svm10_dataReview), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle("Heatmap for SVM (dataReview)")

rf5_dataReview_predict <- predict(rf5_dataReview, newdata = test_dataReview)
rf5_dataReview_cm <- confusionMatrix(rf5_dataReview_predict,
test_dataReview$Sentiment)
rf5_dataReview_cm$overall['Accuracy']
```

```
rf10_dataReview_predict <- predict(rf10_dataReview, newdata = test_dataReview)
rf10_dataReview_cm <- confusionMatrix(rf10_dataReview_predict,
test_dataReview$Sentiment)
rf10_dataReview_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_rf10_dataReview <- as.matrix(rf10_dataReview_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_rf10_dataReview), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle("Heatmap for RanForest (dataReview)")

nb_dataReview_predict <- predict(nb_dataReview, newdata = test_dataReview)
nb_dataReview_cm <- confusionMatrix(nb_dataReview_predict,
test_dataReview$Sentiment)
nb_dataReview_cm$overall['Accuracy']

nb10_dataReview_predict <- predict(nb10_dataReview, newdata = test_dataReview)
nb10_dataReview_cm <- confusionMatrix(nb10_dataReview_predict,
test_dataReview$Sentiment)
nb10_dataReview_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_nb10_dataReview <- as.matrix(nb10_dataReview_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_nb10_dataReview), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "red") +
  ggtitle("Heatmap for Naive Bayes (dataReview)")

#2 dataTitle DATASET (non stemmed TF-IDF representation of title)

svm5_dataTitle_predict <- predict(svm5_dataTitle, newdata = test_dataTitle)
svm5_dataTitle_cm <- confusionMatrix(svm5_dataTitle_predict,
test_dataTitle$Sentiment)
svm5_dataTitle_cm$overall['Accuracy']

svm10_dataTitle_predict <- predict(svm10_dataTitle, newdata = test_dataTitle)
svm10_dataTitle_cm <- confusionMatrix(svm10_dataTitle_predict,
test_dataTitle$Sentiment)
svm10_dataTitle_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_svm10_dataTitle <- as.matrix(svm10_dataTitle_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_svm10_dataTitle), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "blue") +
```

```
  ggtitle("Heatmap for SVM (dataTitle)")

rf5_dataTitle_predict <- predict(rf5_dataTitle, newdata = test_dataTitle)
rf5_dataTitle_cm <- confusionMatrix(rf5_dataTitle_predict,
test_dataTitle$Sentiment)
rf5_dataTitle_cm$overall['Accuracy']

rf10_dataTitle_predict <- predict(rf10_dataTitle, newdata = test_dataTitle)
rf10_dataTitle_cm <- confusionMatrix(rf10_dataTitle_predict,
test_dataTitle$Sentiment)
rf10_dataTitle_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_rf10_dataTitle <- as.matrix(rf10_dataTitle_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_rf10_dataTitle), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "blue") +
  ggtitle("Heatmap for RanForest (dataTitle)")

nb_dataTitle_predict <- predict(nb_dataTitle, newdata = test_dataTitle)
nb_dataTitle_cm <- confusionMatrix(nb_dataTitle_predict, test_dataTitle$Sentiment)
nb_dataTitle_cm$overall['Accuracy']

nb10_dataTitle_predict <- predict(nb10_dataTitle, newdata = test_dataTitle)
nb10_dataTitle_cm <- confusionMatrix(nb10_dataTitle_predict,
test_dataTitle$Sentiment)
nb10_dataTitle_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_nb10_dataTitle <- as.matrix(nb10_dataTitle_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_nb10_dataTitle), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "blue") +
  ggtitle("Heatmap for Naive Bayes (dataTitle)")

#3 dataReview_ST DATASET (stemmed TF-IDF representation of title)

svm5_dataReview_ST_predict <- predict(svm5_dataReview_ST, newdata =
test_dataReview_ST)
svm5_dataReview_ST_cm <- confusionMatrix(svm5_dataReview_ST_predict,
test_dataReview_ST$Sentiment)
svm5_dataReview_ST_cm$overall['Accuracy']

svm10_dataReview_ST_predict <- predict(svm10_dataReview_ST, newdata =
test_dataReview_ST)
svm10_dataReview_ST_cm <- confusionMatrix(svm10_dataReview_ST_predict,
test_dataReview_ST$Sentiment)
svm10_dataReview_ST_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_svm10_dataReview_ST <- as.matrix(svm10_dataReview_ST_cm$table)
```

```
# Plot the heatmap
ggplot(data = as.data.frame(matrix_svm10_dataReview_ST), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "orchid1") +
  ggtitle("Heatmap for SVM (dataReview_ST)")

rf5_dataReview_ST_predict <- predict(rf5_dataReview_ST, newdata =
test_dataReview_ST)
rf5_dataReview_ST_cm <- confusionMatrix(rf5_dataReview_ST_predict,
test_dataReview_ST$Sentiment)
rf5_dataReview_ST_cm$overall['Accuracy']

rf10_dataReview_ST_predict <- predict(rf10_dataReview_ST, newdata =
test_dataReview_ST)
rf10_dataReview_ST_cm <- confusionMatrix(rf10_dataReview_ST_predict,
test_dataReview_ST$Sentiment)
rf10_dataReview_ST_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_rf10_dataReview_ST <- as.matrix(rf10_dataReview_ST_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_rf10_dataReview_ST), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "orchid1") +
  ggtitle("Heatmap for RanForest (dataReview_ST)")

nb_dataReview_ST_predict <- predict(nb_dataReview_ST, newdata =
test_dataReview_ST)
nb_dataReview_ST_cm <- confusionMatrix(nb_dataReview_ST_predict,
test_dataReview_ST$Sentiment)
nb_dataReview_ST_cm$overall['Accuracy']

nb10_dataReview_ST_predict <- predict(nb10_dataReview_ST, newdata =
test_dataReview_ST)
nb10_dataReview_ST_cm <- confusionMatrix(nb10_dataReview_ST_predict,
test_dataReview_ST$Sentiment)
nb10_dataReview_ST_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_nb10_dataReview_ST <- as.matrix(nb10_dataReview_ST_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_nb10_dataReview_ST), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "orchid1") +
  ggtitle("Heatmap for Naive Bayes (dataReview_ST)")

#4 dataReview_ST DATASET (stemmed TF-IDF representation of reviews)

svm5_dataTitle_ST_predict <- predict(svm5_dataTitle_ST, newdata =
test_dataTitle_ST)
```

```
svm5_dataTitle_ST_cm <- confusionMatrix(svm5_dataTitle_ST_predict,
test_dataTitle_ST$Sentiment)
svm5_dataTitle_ST_cm$overall['Accuracy']

svm10_dataTitle_ST_predict <- predict(svm10_dataTitle_ST, newdata =
test_dataTitle_ST)
svm10_dataTitle_ST_cm <- confusionMatrix(svm10_dataTitle_ST_predict,
test_dataTitle_ST$Sentiment)
svm10_dataTitle_ST_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_svm10_dataTitle_ST <- as.matrix(svm10_dataTitle_ST_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_svm10_dataTitle_ST), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "seagreen4") +
  ggtitle("Heatmap for SVM (dataTitle_ST)")

rf5_dataTitle_ST_predict <- predict(rf5_dataTitle_ST, newdata = test_dataTitle_ST)
rf5_dataTitle_ST_cm <- confusionMatrix(rf5_dataTitle_ST_predict,
test_dataTitle_ST$Sentiment)
rf5_dataTitle_ST_cm$overall['Accuracy']

rf10_dataTitle_ST_predict <- predict(rf10_dataTitle_ST, newdata =
test_dataTitle_ST)
rf10_dataTitle_ST_cm <- confusionMatrix(rf10_dataTitle_ST_predict,
test_dataTitle_ST$Sentiment)
rf10_dataTitle_ST_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_rf10_dataTitle_ST <- as.matrix(rf10_dataTitle_ST_cm$table)

# Plot the heatmap
ggplot(data = as.data.frame(matrix_rf10_dataTitle_ST), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "seagreen4") +
  ggtitle("Heatmap for RanForest (dataTitle_ST)")

nb_dataTitle_ST_predict <- predict(nb_dataTitle_ST, newdata = test_dataTitle_ST)
nb_dataTitle_ST_cm <- confusionMatrix(nb_dataTitle_ST_predict,
test_dataTitle_ST$Sentiment)
nb_dataTitle_ST_cm$overall['Accuracy']

nb10_dataTitle_ST_predict <- predict(nb10_dataTitle_ST, newdata =
test_dataTitle_ST)
nb10_dataTitle_ST_cm <- confusionMatrix(nb10_dataTitle_ST_predict,
test_dataTitle_ST$Sentiment)
nb10_dataTitle_ST_cm$overall['Accuracy']

# Heatmap of confusion matrix
matrix_nb10_dataTitle_ST <- as.matrix(nb10_dataTitle_ST_cm$table)

# Plot the heatmap
```

```
ggplot(data = as.data.frame(matrix_nb10_dataTitle_ST), aes(x = Reference, y =
Prediction, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "seagreen4") +
  ggtitle("Heatmap for Naive Bayes (dataTitle_ST)")

######################## COMPARING DIFFERENT MODELS ############################
# using the resampling function, which essentially compares models using the same
resampling profiles (data)

#1
rsmp_dataReview <-
resamples(list(SVM=svm10_dataReview,RanForest=rf10_dataReview,NaiveBayes=nb10_data
Review))
bwplot(rsmp_dataReview, layout = c(3, 1), main="box and whisker plot for
dataReview", fill = "pink")

post_dataReview <-
postResample(pred=svm5_dataReview_predict,obs=test_dataReview$Sentiment)

#2
rsmp_dataTitle <-
resamples(list(SVM=svm10_dataTitle,RanForest=rf10_dataTitle,NaiveBayes=nb10_dataTi
tle))
bwplot(rsmp_dataTitle, layout = c(3, 1), main="box and whisker plot for
dataTitle", fill = "lightskyblue")

#3
rsmp_dataReview_ST <-
resamples(list(SVM=svm10_dataReview_ST,RanForest=rf10_dataReview_ST,NaiveBayes=nb1
0_dataReview_ST))
bwplot(rsmp_dataReview_ST, layout = c(3, 1), main="box and whisker plot for
dataReview_ST", fill = "magenta")

#4
rsmp_dataTitle_ST <-
resamples(list(SVM=svm10_dataTitle_ST,RanForest=rf10_dataTitle_ST,NaiveBayes=nb10_
dataTitle_ST))
bwplot(rsmp_dataTitle_ST, layout = c(3, 1), main="box and whisker plot for
dataTitle_ST", fill = "lightskyblue4")
```

8

References

[1] Maharani, W. (2013). Microblogging sentiment analysis with lexical based and machine learning approaches. *2013 International Conference of Information and Communication Technology*. https://doi.org/10.1109/icoict.2013.6574616

[2] Algefes, A., Aldossari, N., Masmoudi, F., & Kariri, E. (2022). A Text-mining approach for crime tweets in Saudi Arabia: From analysis to prediction. *7th International Conference on Data Science and Machine Learning Applications*. https://doi.org/10.1109/cdma54072.2022.00023

[3] Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. M. (2013). Efficient estimation of word representations in vector space. *arXiv (Cornell University)*. http://export.arxiv.org/pdf/1301.3781

[4] Rathi, M., Malik, A., Varshney, D., Sharma, R., & Mendiratta, S. (2018). Sentiment Analysis of Tweets Using Machine Learning Approach. *2018 Eleventh International Conference on Contemporary Computing*. https://doi.org/10.1109/ic3.2018.8530517

[5] Goularas, D., & Kamis, S. (2019). Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data. *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications*, 978-1-7281-2914–3. https://doi.org/10.1109/deep-ml.2019.00011

[6] Cambria, E. (2016). Affective Computing and Sentiment Analysis. *IEEE Intelligent Systems*, *31*(2), 102–107. https://doi.org/10.1109/mis.2016.31

[7] Vargas, M. P., Parra, O. J., & Rico, M. J. (2017). Business perception based on sentiment analysis through deep neuronal networks for Natural Language Processing. *Lecture Notes in Computer Science*, *10531*, 365–374. https://doi.org/10.1007/978-3-319-67380-6_33

[8] Riekert, M., Leukel, J., & Klein, A. (2016). Online media sentiment: Understanding machine learning-based classifiers. *ResearchGate*.

https://www.researchgate.net/publication/303862940_Online_media_sentiment_Understanding_machine_learning-based_classifiers/citations

[9] Mudinas, A., Zhang, D., & Levene, M. (2012). Combining lexicon and learning based approaches for concept-level sentiment analysis. *WISDOM '12: Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*. https://doi.org/10.1145/2346676.2346681

[10] Tsytsarau, M., & Palpanas, T. (2011). Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, *24*(3), 478–514. https://doi.org/10.1007/s10618-011-0238-6

[11] Quarteroni, S. (2018). Natural language processing for industry. *Informatik Spektrum*, *41*(2), 105–112. https://doi.org/10.1007/s00287-018-1094-1

[12] May, M. C., Neidhöfer, J., Körner, T., Schäfer, L., & Lanza, G. (2022). Applying natural language processing in manufacturing. *Procedia CIRP*, *115*, 184–189. https://doi.org/10.1016/j.procir.2022.10.071

[13] Single, J. I., Schmidt, J., & Denecke, J. (2020). Knowledge acquisition from chemical accident databases using an ontology-based method and natural language processing. *Safety Science*, *129*, 104747. https://doi.org/10.1016/j.ssci.2020.104747

[14] Gui, Z., & Harth, A. (2021). Towards a Data Driven Natural Language Interface for Industrial IoT Use Cases. *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*. https://doi.org/10.1109/ichms53169.2021.9582450

[15] Sergeeva, M. B., Voskobovich, V. V., & Kukharenko, A. M. (2022). Data processing in Industrial Internet of Things (IIoT) applications : Industrial Agility. *2022 Wave Electronics and Its Application in Information and Telecommunication Systems (WECONF)*. https://doi.org/10.1109/weconf55058.2022.9803390

[16] van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L. (2018). Challenges and opportunities of applying natural language processing in business process management. International Conference on Computational Linguistics. "COLING 2018: The 27th International Conference on Computational Linguistics: Proceedings of the Conference: August 20-26, 2018 Santa Fe, New Mexico, USA". Stroudsburg, PA: Association for Computational Linguistics, 2018, p. 2791-2801.

[17] Obradovic, D. (2016, January 15). *THE ROLE INNOVATION ON STRATEGIC ORIENTATIONS AND COMPETITIVENESS OF ENTERPRISES*. OBRADOVIC | Ecoforum Journal. http://ecoforumjournal.ro/index.php/eco/article/view/305. Date accessed: 24 Apr. 2023.

[18] Bahja, M. (2021). Natural Language Processing Applications in business. In *IntechOpen eBooks*. https://doi.org/10.5772/intechopen.92203

[19] Ziora, L. (2021). Natural language processing in the support of business organization management. *IntelliSys 2021: Intelligent Systems and Applications, 76–83.* https://doi.org/10.1007/978-3-030-82199-9_6

[20] Global Risk Institute. (2022, November 9). *Teaching Computers to Understand Human Language: How Natural Language Processing is Reshaping the World of Finance - Global Risk Institute*. https://globalriskinstitute.org/publication/teaching-computers-to-understand-human-language-how-natural-language-processing-is-reshaping-the-world-of-finance/

[21] Zhecheva, D., Nenkov, N. (2022). Business demands for processing unstructured textual data – text mining techniques for companies to implement. *Access to science, business, innovation in digital economy, ACCESS Press, 3(2): 107-120*. https://doi.org/10.46656/access.2022.3.2(2)

[22] Xing, F.Z., Cambria, E. & Welsch, R.E. (2018). Natural language based financial forecasting: a survey. *Artif Intell Rev* 50, 49–73. https://doi.org/10.1007/s10462-017-9588-9

[23] Trappey, A. J., Trappey, C. V., Wu, J., & Wang, J. W. C. (2020). Intelligent compilation of patent summaries using machine learning and natural language processing techniques. *Advanced Engineering Informatics*, *43*, 101027. https://doi.org/10.1016/j.aei.2019.101027

[24] Vashisht, V., & Dharia, P. (2020). Integrating Chatbot Application with Qlik Sense Business Intelligence (BI) Tool Using Natural Language Processing (NLP). In *Lecture notes in networks and systems* (pp. 683–692). https://doi.org/10.1007/978-981-15-2329-8_69

[25] García-Méndez, S., De Arriba-Pérez, F., Barba-Seara, O., Fernández-Gavilanes, M., & González-Castaño, F. J. (2021). Demographic Market Segmentation on Short Banking Movement Descriptions Applying Natural Language Processing. *2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, 141–146. https://doi.org/10.1109/iscsic54682.2021.00035

[26] Hartmann, J., & Netzer, O. (2023). Natural language processing in marketing. In *Review of marketing research* (pp. 191–215). https://doi.org/10.1108/s1548-643520230000020011

[27] Tirunillai, S., & Tellis, G. J. (2014). Mining Marketing Meaning from Online Chatter: Strategic Brand Analysis of Big Data Using Latent Dirichlet Allocation. *Journal of Marketing Research*, *51*(4), 463–479. https://doi.org/10.1509/jmr.12.0106

[28] Timoshenko, A., & Hauser, J. R. (2019). Identifying Customer Needs from User-Generated Content. *Marketing Science*, *38*(1), 1–20. https://doi.org/10.1287/mksc.2018.1123

[29] Chakraborty, I., Kim, M., & Sudhir, K. (2022). Attribute Sentiment Scoring with Online Text Reviews: Accounting for Language Structure and Missing Attributes. *Journal of Marketing Research*, *59*(3), 600–622. https://doi.org/10.1177/00222437211052500

[30] Alpaydin. (2016). *Introduction to Machine Learning.* (3rd ed.). The MIT Press.

[31] Kour, H., & Gondhi, N. K. (2020). Machine Learning Techniques: a survey. In *Lecture notes on data engineering and communications technologies* (pp. 266–275). https://doi.org/10.1007/978-3-030-38040-3_31

[32] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning.* (2nd ed.). The MIT Press.

[33] Ziora, L. (2020). Machine learning solutions in the management of a contemporary business organization. *Journal of Decision Systems*, *29*(sup1), 344–351. https://doi.org/10.1080/12460125.2020.1848378

[34] Zabin, J., & Jefferies, A. (2008). Social media monitoring and analysis: Generating consumer insights from online conversation. *Aberdeen Group Benchmark Report*. https://robertoigarza.files.wordpress.com/2008/10/rep-social-media-monitoring-and-analysis-aberdeen-group-2008.pdf

[35] Muhammad, I. Z., & Zhu, Y. (2015). SUPERVISED MACHINE LEARNING APPROACHES: a SURVEY. *ICTACT Journal on Soft Computing*, *05*(03), 946–952. https://doi.org/10.21917/ijsc.2015.0133

[36] Jiang, T., Gradus, J. L., & Rosellini, A. J. (2020). Supervised Machine Learning: A brief primer. *Behavior Therapy*, *51*(5), 675–687. https://doi.org/10.1016/j.beth.2020.05.002

[37] Mrabet, M. a. E., Makkaoui, K. E., & Faize, A. (2021). Supervised Machine Learning: A Survey. *2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet)*, 1–10. https://doi.org/10.1109/CommNet52204.2021.9641998

[38] Choudhary, R., & Gianey, H. K. (2017). Comprehensive Review On Supervised Machine Learning Algorithms. *2017 International Conference on Machine Learning and Data Science (MLDS)*, 37–43. https://doi.org/10.1109/mlds.2017.11

[39] Mahesh, B. (2019). Machine Learning Algorithms – A Review. *ResearchGate*. https://doi.org/10.21275/ART20203995

[40] Sen, P. C., Hajra, M., & Ghosh, M. (2019). Supervised Classification Algorithms in Machine Learning: A Survey and review. In *Advances in intelligent systems and computing* (pp. 99–111). https://doi.org/10.1007/978-981-13-7403-6_11

[41] N, T. R., & Gupta, R. (2020). A survey on machine learning approaches and its techniques: *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. https://doi.org/10.1109/sceecs48394.2020.190

[42] Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review*, *52*(1), 273–292. https://doi.org/10.1007/s10462-018-09677-1

[43] Salmony, M. Y. A., & Faridi, A. R. (2021). Supervised Sentiment Analysis on Amazon Product Reviews: A survey. *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*. https://doi.org/10.1109/iciem51511.2021.9445303

[44] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744. https://doi.org/10.1007/s11042-022-13428-4

[45] Khan, W., Daud, A., Nasir J. A., Amjad, T. (2016) A survey on the state-of-the-art machine learning models in the context of NLP. *Kuwait Journal of Science*. Vol. 43 No. 4.

[46] Naeem, S., Logofătu, D., & Muharemi, F. (2020). Sentiment analysis by using supervised machine learning and deep learning approaches. In *ICCCI 2020 Advances in*

Computational Collective Intelligence (pp. 481–491). https://doi.org/10.1007/978-3-030-63119-2_39

[47] Usha, G. R., & Dharmanna, L. (2021). Sentiment Analysis on Business Data using Machine Learning. *2021 Second International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, 1–6. https://doi.org/10.1109/ICSTCEE54422.2021.9708593

[48] Rain, C. (2012). *Analysis in Amazon reviews using probabilistic machine learning*. https://www.semanticscholar.org/paper/Analysis-in-Amazon-Reviews-Using-Probabilistic-Rain/f0afe9ea9d286248336ee9dc4e954aecde3475bb

[49] Singla, Z., Randhawa, S., & Jain, S. (2017). Sentiment analysis of customer product reviews using machine learning. *2017 International Conference on Intelligent Computing and Control (I2C2)*, 1–5. https://doi.org/10.1109/i2c2.2017.8321910

[50] Sultana, N., Kumar, P., Patra, M. R., & Alam, S. S. (2019). Sentiment Analysis for Product Review. *2019 International Journal of Soft Computing*. https://doi.org/10.21917/ijsc.2019.0266

[51] Thada, V., & Shrivastava, U. (2020). Sentiment mining of product opinion data. *International Journal of Innovative Technology and Exploring Engineering*, *9*(3), 1218–1222. https://doi.org/10.35940/ijitee.c8641.019320

[52] Hájek, P., & Barushka, A. (2019). A Comparative Study of Machine Learning Methods for Detection of Fake Online Consumer Reviews. *3rd International Conference on E-Business and Internet*, 18–22. https://doi.org/10.1145/3383902.3383909

[53] Faul, A. C. (2019). A concise introduction to machine learning. (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781351204750

[54] Alpaydın, E. (2021). *Machine learning*. The MIT Press. https://doi.org/10.7551/mitpress/13811.001.0001

[55] Brownlee, J. (2020, September 3). *What is a Hypothesis in Machine Learning?* MachineLearningMastery.com. Retrieved June 25, 2023, from https://machinelearningmastery.com/what-is-a-hypothesis-in-machine-learning/

[56] Dy, J. G., & Brodley, C. E. (2004). Feature Selection for Unsupervised Learning. *Journal of Machine Learning Research*, *5*, 845–889.

[57] Yale University Department of Statistics and Data Science. (n.d.). *Linear Regression*. Retrieved July 1, 2023, from http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm

[58] Kutner, M. H. (2005). *Applied Linear Statistical Models* (5th ed.). McGraw-Hill/Irwin.

[59] Kleinbaum, D. G., & Klein, M. (2010). *Logistic regression: A Self-Learning Text* (3rd ed.). Springer.

[60] Waila, P., Marisha, Singh, V. K., & Singh, M. K. (2012). Evaluating Machine Learning and Unsupervised Semantic Orientation approaches for sentiment analysis of textual reviews. *2012 IEEE International Conference on Computational Intelligence and Computing Research*, 1–6. https://doi.org/10.1109/iccic.2012.6510235

[61] Campesato, O. (2021). *Natural language processing and machine learning for developers*. Mercury Learning and Information.

[62] Roldós, I. (2020b, December 22). *Major Challenges of Natural Language Processing (NLP)*. MonkeyLearn Blog. Retrieved August 8, 2023, from https://monkeylearn.com/blog/natural-language-processing-challenges/

[63] Yse, D. L. (2021, December 31). Text normalization for Natural Language Processing (NLP). *Medium*. Retrieved August 8, 2023, from https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646

[64] Martin, J. H., & Jurafsky, D. (n.d.). *Speech and Language Processing* (Chapter 02, 3rd Draft). Stanford University Press. Retrieved August 8, 2023, from https://web.stanford.edu/~jurafsky/slp3/2.pdf

[65] Sharou, K. A., Li, Z., & Specia, L. (2021). Towards a Better Understanding of Noise in Natural Language Processing. *Proceedings of Recent Advances in Natural Language Processing*, 53–62. https://doi.org/10.26615/978-954-452-072-4_007

[66] Dong, G., & Liu, H. (2020). *Feature engineering for machine learning and data analytics* (1st ed.). CRC Press.

[67] Bengfort, B., Ojeda, T., & Bilbro, R. (2018). *Applied Text Analysis with Python: Enabling Language Aware Data Products with Machine Learning*. O'Reilly Media.

[68] Khanna, C. (2022, January 6). Word, Subword and Character-based tokenization: Know the difference | Towards Data Science. *Medium*. Retrieved August 8, 2023, from https://towardsdatascience.com/word-subword-and-character-based-tokenization-know-the-difference-ea0976b64e17

[69] Khanna, C. (2022b, November 28). Text preprocessing: Stop words removal | Chetna | Towards Data Science. *Medium*. Retrieved August 8, 2023, from https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

[70] Shmueli, G., Bruce, P. C., Gedeck, P., & Patel, N. R. (2019). *Data mining for business analytics: Concepts, Techniques and Applications in Python*. John Wiley & Sons.

[71] Srinidhi, S. (2021, December 12). Understanding word n-grams and n-gram probability in natural language processing. *Medium*. Retrieved August 9, 2023, from https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058

[72] Li, H., Cai, D., Xu, J., & Watanabe, T. (2022). Residual Learning of Neural Text Generation with n-gram Language Model. *Findings of the Association for Computational Linguistics: EMNLP 2022*. https://doi.org/10.18653/v1/2022.findings-emnlp.109

[73] Brownlee, J. (2019, August 7). What are word embeddings for text? *MachineLearningMastery.com*. Retrieved August 9, 2023, from https://machinelearningmastery.com/what-are-word-embeddings/

[74] Vetsch, R. (2022, January 22). NLP — from word embedding to transformers | by Robin | medium. *Medium*. Retrieved August 10, 2023, from https://medium.com/@RobinVetsch/nlp-from-word-embedding-to-transformers-76ae124e6281

[75] Anandika, A., & Mishra, S. (2019). A Study on Machine Learning Approaches for Named Entity Recognition. *2019 International Conference on Applied Machine Learning (ICAML)*. https://doi.org/10.1109/icaml48257.2019.00037

[76] Marshall, C. (2021, December 13). What is named entity recognition (NER) and how can I use it? *Medium*. Retrieved August 11, 2023, from https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d

[77] GeeksforGeeks. (2022, October 18). *Named Entity recognition*. Retrieved August 11, 2023, from https://www.geeksforgeeks.org/named-entity-recognition

[78] Goyal, A., Gupta, V., & Kumar, M. (2018). Recent Named Entity Recognition and Classification techniques: A systematic review. *Computer Science Review*, *29*, 21–43. https://doi.org/10.1016/j.cosrev.2018.06.001

[79] Kurdi, M. Z. (2016). *Natural Language Processing and Computational Linguistics 1* (1st ed.). Wiley. https://doi.org/10.1002/9781119145554

[80] Lei, L., & Liu, D. (2021). *Conducting Sentiment Analysis* (Elements in Corpus Linguistics). Cambridge: Cambridge University Press. doi:10.1017/9781108909679

[81] Ahlgren, O. (2016). Research on Sentiment Analysis: The First Decade. *2016 IEEE International Conference on Data Mining Workshops*, 890–899. https://doi.org/10.1109/icdmw.2016.0131

[82] Kumar, A., & Sebastian, T. M. (2012). Sentiment Analysis: A Perspective on its Past, Present and Future. *International Journal of Intelligent Systems and Applications*, *4*(10), 1–14. https://doi.org/10.5815/ijisa.2012.10.01

[83] Mäntylä, M., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis — A review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16–32. https://doi.org/10.1016/j.cosrev.2017.10.002

[84] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1–2), 1–135. https://www.cs.cornell.edu/home/llee/omsa/omsa.pdf

[85] Bakrey, M. (2023, March 25). All about Lexicons In NLP - Mohamed Bakrey - Medium. *Medium*. Retrieved August 20, 2023, from https://mohamedbakrey094.medium.com/all-about-lexicons-in-nlp-12ada00c2821

[86] Mudinas, A., Zhang, D., & Levene, M. (2012b). Combining lexicon and learning based approaches for concept-level sentiment analysis. *WISDOM'12*. https://doi.org/10.1145/2346676.2346681

[87] Nichols, R. *Linguistic Inquiry and Word Count | Centre for Human Evolution, Cognition, and Culture*. (n.d.). Retrieved August 25, 2023, from https://hecc.ubc.ca/quantitative-textual-analysis/qta-practice/linguistic-inquiry-and-word-count

[88] Tausczik, Y. R., & Pennebaker, J. W. (2009). The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. *Journal of Language and Social Psychology*, *29*(1), 24–54. https://doi.org/10.1177/0261927x09351676

[89] Baccianella, S. (2010). SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf

[90] Sadia, A., Bashir, F., & Khan, F. (2018). An Overview of Lexicon-Based Approach For Sentiment Analysis. *2018 3rd International Electrical Engineering Conference*.

[91] Darwich, M., Noah, S. a. M., Omar, N., & Osman, N. A. (2019). Corpus-Based Techniques for Sentiment Lexicon Generation: A Review. *Journal of Digital Information Management*, *17*(5), 296. https://doi.org/10.6025/jdim/2019/17/5/296-305

[92] Sharma, S. (2022, August 20). Sentiment analysis using the SentiWordNet Lexicon - Srishti Sharma - Medium. *Medium*. Retrieved August 25, 2023, from https://srish6.medium.com/sentiment-analysis-using-the-sentiwordnet-lexicon-1a3d8d856a10

[93] Deng, S., Sinha, A. P., & Zhao, H. (2017). Adapting sentiment lexicons to domain-specific social media texts. *Decision Support Systems*, *94*, 65–76. https://doi.org/10.1016/j.dss.2016.11.001

[94] Al-Shabi, M. A. (2020). Evaluating the performance of the most important Lexicons used to Sentiment analysis and opinions Mining. *ResearchGate*. https://www.researchgate.net/publication/343473213_Evaluating_the_performance_of_the_most_important_Lexicons_used_to_Sentiment_analysis_and_opinions_Mining

[95] Malde, R. (2022, March 30). A short introduction to VADER - towards data science. *Medium*. Retrieved August 25, 2023, from https://towardsdatascience.com/an-short-introduction-to-vader-3f3860208d53

[96] Jenhani, F., Gouider, M. S., & Saïd, L. B. (2016). Lexicon-Based System for Drug Abuse Entity Extraction from Twitter. In *Communications in computer and information science* (pp. 692–703). https://doi.org/10.1007/978-3-319-34099-9_54

[97] Yusof, N. N., Mohamed, A., & Abdul-Rahman, S. (2015). Reviewing classification approaches in sentiment analysis. In *Communications in computer and information science* (pp. 43–53). https://doi.org/10.1007/978-981-287-936-3_5

[98] Chauhan, A., Agarwal, A., & Sulthana, R. (2021). Performance analysis of machine learning algorithms and feature extraction methods for sentiment analysis. *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*. https://doi.org/10.1109/icses52305.2021.9633882

[99] Avinash, M., & Sivasankar, E. (2018). A study of feature extraction techniques for sentiment analysis. In *Advances in intelligent systems and computing* (pp. 475–486). https://doi.org/10.1007/978-981-13-1501-5_41

[100] Trupthi, M., Pabboju, S., & Narasimha, G. (2016). Improved feature extraction and classification — Sentiment analysis. *International Conference on Advances in Human Machine Interaction (HMI - 2016),*. https://doi.org/10.1109/hmi.2016.7449189

[101] Haberzettl, M., Bernd, M. (2018). A Literature Analysis for the Identification of Machine Learning and Feature Extraction Methods for Sentiment Analysis. *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*. https://doi.org/10.1109/icdim.2018.8846980

[102] Saif M. Mohammad; Ethics Sheet for Automatic Emotion Recognition and Sentiment Analysis. *Computational Linguistics* 2022; 48 (2): 239–278. doi: https://doi.org/10.1162/coli_a_00433

[103] Patti, V., Damiano, R., & Bosco, C. (2017). Ethical implications of analyzing opinions, emotions and interactions in social media. *2017 Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. https://doi.org/10.1109/aciiw.2017.8272606

[104] *Data protection and privacy legislation worldwide*. (n.d.). UNCTAD. Retrieved September 3, 2023, from https://unctad.org/page/data-protection-and-privacy-legislation-worldwide

[105] Horizon 2020 Framework Programme of the European Union. (2018, November 29). *FAQ - GDPR.eu*. GDPR.eu. Retrieved September 3, 2023, from https://gdpr.eu/faq/

[106] *Cyberlaw Tracker: Country detail*. (n.d.). UNCTAD. Retrieved September 3, 2023, from https://unctad.org/page/cyberlaw-tracker-country-detail?country=br

[107] Bhaskaran. (2023, February 24). Deterministic vs Stochastic Machine Learning: Which Approach Reigns Supreme in the World of AI? - AITechTrend. *AITechTrend - Further into the Future*. Retrieved September 9, 16 C.E., from https://aitechtrend.com/deterministic-vs-stochastic-machine-learning-which-approach-reigns-supreme-in-the-world-of-ai/

[108] Kothari, V., [vivekkothari]. (2023, April 19). *Difference between Deterministic and Non deterministic Algorithms*. GeeksforGeeks. Retrieved September 16, 2023, from https://www.geeksforgeeks.org/difference-between-deterministic-and-non-deterministic-algorithms/

[109] Mehta, S. (2022, May 10). *Deterministic vs Stochastic Machine Learning*. Analytics India Magazine. Retrieved September 16, 2023, from https://analyticsindiamag.com/deterministic-vs-stochastic-machine-learning/

[110] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018b). *Foundations of Machine Learning, second edition*. MIT Press.

[111] Prasad, A. (2022, January 5). Regression Trees | Decision tree for Regression | Machine learning. *Medium*. Retrieved September 17, 2023, from https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047

[112] Rathi, M., Malik, A., Varshney, D., Sharma, R., & Mendiratta, S. (2018b). Sentiment Analysis of Tweets Using Machine Learning Approach. *2018 Eleventh International Conference on Contemporary Computing (IC3)*. https://doi.org/10.1109/ic3.2018.8530517

[113] IBM. (n.d.). *What is Random Forest? | IBM*. Retrieved September 17, 2023, from https://www.ibm.com/topics/random-forest

[114] Sutton, H. (2020). *Quantifying structure in random forests* [M.Sc Thesis]. Simon Fraser University.

[115] Lohith, O., Jha, A., & Tamboli, S. C. (2023). Comparative Analysis of Random Forest Regression for House Price Prediction. *International Journal of Creative Research Thoughts*, *11*.

[116] Syam, N., & Kaul, R. (2021). Random forest, bagging, and boosting of decision trees. In *Emerald Publishing Limited eBooks* (pp. 139–182). https://doi.org/10.1108/978-1-80043-880-420211006

[117] Zhu, J. (2018b). Probabilistic Machine Learning: Models, Algorithms and a Programming Library. *Probabilistic Machine Learning: Models, Algorithms and a Programming Library Jun Zhu Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 5754–5759. https://doi.org/10.24963/ijcai.2018/823

[118] Murphy, K. P. (2012). *Machine learning: A Probabilistic Perspective*. MIT Press.

[119] Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452–459. https://doi.org/10.1038/nature14541

[120] Martin, J. H., & Jurafsky, D. (n.d.). *Speech and Language Processing* (Chapter 04, 3rd Draft). Stanford University Press. Retrieved August 8, 2023, from https://web.stanford.edu/~jurafsky/slp3/2.pdf

[121] Kaur, P. (2022). Sentiment analysis using web scraping for live news data with machine learning algorithms. *Materials Today: Proceedings*, *65*, 3333–3341. https://doi.org/10.1016/j.matpr.2022.05.409

[122] Ng, Q. X., Yau, C. E., Lim, Y. L., Wong, L., & Liew, T. M. (2022). Public sentiment on the global outbreak of monkeypox: an unsupervised machine learning analysis of 352,182 twitter posts. *Public Health*, *213*, 1–4. https://doi.org/10.1016/j.puhe.2022.09.008

[123] Gupta, S., Bisht, S., & Gupta, S. (2021). Sentiment Analysis of an Online Sentiment with Text and Slang Using Lexicon Approach. In *Springer eBooks* (pp. 95–105). https://doi.org/10.1007/978-981-16-1502-3_11

[124] W. Suwanpipob, N. Arch-int and M. Wattana, "A Sentiment Classification from Review Corpus using Linked Open Data and Sentiment Lexicon," *2021 13th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Chiang Mai, Thailand, 2021, pp. 19-23, doi: 10.1109/ICITEE53064.2021.9611898.

[125] Bigné, E., Ruiz, C., Pérez‐Cabañero, C., & Cuenca, A. C. (2023). Are customer star ratings and sentiments aligned? A deep learning study of the customer service experience in tourism destinations. *Service Business*, *17*(1), 281–314. https://doi.org/10.1007/s11628-023-00524-0

[126] *History | De' Longhi Group - Corporate website*. (n.d.). Retrieved October 20, 2023, from https://www.delonghigroup.com/en/group/history

[127] Wertz, J. (2021, January 31). Changes In Consumer Behavior Brought On By The Pandemic. *Forbes*. Retrieved October 20, 2023, from https://www.forbes.com/sites/jiawertz/2021/01/31/changes-in-consumer-behavior-brought-on-by-the-pandemic/

Acknowledgements

In the vast world of academia, where every paragraph feels like a triumph and every bibliography entry a small victory, I can now finally enter this work. It has been the result of over two incredible years of adventures in a new field, a new university and, above all, a new country.

First and foremost, I extend my deepest gratitude to Associate Professor Marta Disegna, who has supported and guided me from the beginning to the end of this thesis project. Your knowledge and patience have transformed it into a coherent, well thought out work.

To Nicolò Biasetton, who saved the applied project from utter ruin and myself from hours upon hours of staring into the screen, going blind debugging one part of the code or another. Trust that your previous experiences and projects in the field were put to good use.

My sincerest gratitude for the support crew of this entire journey, from beginning to end – and beyond. To Salva for all of the coffee (brioche) and the pep talks, to Aliya for always being the brighter side of everything, and to Nima for taking me in and being the voice of reason (usually). To Anastasiia, for finding the courage to speak up and becoming a dear friend. Every single one of you is nothing short of exceptional. You have weathered the storms (exams) and celebrated every small victory (anything above 27) alongside me, and I wouldn't change any part of it all.

To the ones I have carried with me from 10.000 kilometers away, for never being far. To Carla, for all of the calls, the grown up discussions, steering me away from procrastination and giving me all of the support when you were also going through it yourself. To Alexya for venturing all the way out here (twice) just to teach me how to order a coca cola con ghiaccio. To Lara, for over two decades of friendship, simple words are not enough. Every single one of you is (stuck) with me for life.

Above all else, to my family.

To my family, for the unwavering belief in me, even before I had it myself, for keeping me sane, for laughing and crying together from so far a distance, and the occasional motivational cat photo. I could not have achieved this without you (and the cats).

Last, but certainly not least, I want to thank myself. It was challenging and it was hard work, it was overwhelming and joyous. This thesis represents the culmination of over two years of growth, of adventures, deadlines and (some) study. And I did it.