NOVA IMS
Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

A Sentiment Analysis model to evaluate people's opinion about artificial intelligence

Maria Nápoles Sarmento Buzaglo

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

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

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

A SENTIMENT ANALYSIS MODEL TO EVALUATE PEOPLE'S OPINION ABOUT ARTIFICIAL INTELLIGENCE

by

Maria Nápoles Sarmento Buzaglo

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

Advisor: Mauro Castelli

Co Advisor*:* Leonardo Vanneschi

October 2019

ACKNOWLEDGEMENTS

To NOVA Information Management School (IMS) for the excellence in teaching, for every available opportunity provided to its students and for its facilities and excellent organization.

To Leonardo Vanneschi for creating such an interesting master program and for his help, availability and support in every single decision regarding to my master’s thesis.

To my advisor, Mauro Castelli, who have helped and guided me through this challenge and gave me every condition to my success in the development of this dissertation.

To my projects’ team during the 1st year of my master’s in Advanced Analytics, João Manso and Gonçalo Pio, who made me progress as a person and were essential to achieve my goals as a student during the masters.

To my parents, Sofia and Rui, who are always supporting me in everything and encouraged me to go ahead with my master decisions.

ABSTRACT

With the use of internet, people are much more able to express and share what they think about a certain topic, their ideas and so on. Facebook and Twitter social networks, YouTube, online review sites like Zomato, online news sites or personal blogs are platforms that are usually used for this purpose. Every business wants to know what people think about their products; many people and politicians want to know the prediction for political elections; sometimes it can be useful to understand how opinions are distributed in some controversial themes. Thus, the analysis of textual data is also a need to stay competitive.

In this work, through Sentiment Analysis techniques, different opinions from different online sources regarding to artificial intelligence are analyzed - a controversial field that have been a target of some debate in recent years.

First, it is done a careful revision of the concept of Sentiment Analysis and all the involved techniques and processes such as data preprocessing, feature extraction and selection, sentiment classification approaches and machine learning algorithms – Naïve Bayes, Neural Networks, Random Forest, Support Vector Machine, Logistic Regression, Stochastic Gradient Descent. Based on previous works, the main conclusions, regarding to which techniques work better in which situations, are highlighted. Then, it is described the followed methodology in the application of Sentiment Analysis to artificial intelligence as a controversial field. The auxiliary tool used for this work is Python. In the end, results are presented and discussed.

KEYWORDS

Sentiment Analysis, Artificial Intelligence, AI, Natural Language Processing, NLP, Machine Learning, Binary Classification, Opinion Mining

RESUMO

Com o uso da internet as pessoas estão muito mais aptas a expressar e partilhar o que pensam sobre um determinado assunto, a partilhar as suas ideias e muito mais. As redes sociais como o Facebook e o Twitter, o YouTube, sites de avaliação online como o Zomato, sites de notícias online ou blogs pessoais são plataformas que geralmente são utilizadas para este fim. Todas as empresas querem saber o que as pessoas acham dos seus produtos; muitas pessoas e políticos querem saber a previsão para as eleições políticas; muitas vezes é útil saber qual é a distribuição das opiniões relativamente a um determinado tema controverso. Posto isto, a análise de dados textuais é também uma necessidade para ganhar vantagem competitiva.

Neste trabalho são analisadas, através de técnicas de Análise de Sentimentos, diferentes opiniões vindas de diferentes fontes online sobre inteligência artificial – um tema controverso que tem sido alvo de algum debate nos anos mais recentes.

Primeiro, é feita uma revisão cuidadosa do conceito de Análise de Sentimentos e de todas as técnicas e processos associados tais como o pré-processamento dos dados, a seleção e extração de atributos, abordagens de classificação de sentimentos e algoritmos de aprendizagem automática – Naïve Bayes, Neural Networks, Random Forest, Support Vector Machine, Logistic Regression, Stochastic Gradient Descent. Com base em trabalhos realizados, são destacadas as principais conclusões sobre que técnicas funcionam melhor e em que situações. Em seguida, é descrita a metodologia que foi seguida na aplicação de Análise de Sentimentos à inteligência artificial como um tema controverso. A ferramenta auxiliar utilizada neste trabalho é o Python. No final os resultados são apresentados e discutidos.

PALAVRAS-CHAVE

Análise de Sentimentos, Inteligencia Artificial, IA, Processamento de Linguagem Natural, PLN, Aprendizagem Automática, Classificação Binária, Mineração de Opiniões

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

NLP	Natural Language Processing
SA	Sentiment Analysis
AI	Artificial Intelligence
API	Application Programming Interface
POS	Part-of-Speech Tagging
PMI	Point-wise Mutual Information
LSI	Latent Semantic Index
PCA	Principal Component Analysis
TF-IDF	Term Frequency – Inverse Document Frequency
LSA	Latent Semantic Analysis
NB	Naïve Bayes
BN	Bayesian Network
LR	Logistic Regression
SVM	Support Vector Machine
ANN	Artificial Neuron Networks
SGD	Stochastic Gradient Descent
RF	Random Forest
TPR	True Positive Rate
FPR	False Positive Rate
ROC	Reception Operating Characteristic
AUC	Area Under the Curve
NLTK	Natural Language Toolkit

1. INTRODUCTION

The increased use of Internet and online activities (like chatting, conferencing, surveillances, ticket booking, online transactions, e-commerce, social media communications, blogging and micro-blogging, clicks streams, etc.) lead us to extract, transform, load and analyze a huge amount of structured and unstructured data, at a fast pace, referred to as Big Data. Such data can be analyzed using a combination of Data Mining, Web Mining and Text Mining techniques in various real-life applications **(Ravi K., Ravi V., 2015)**.

The analysis of textual data is a need to stay competitive. Every business wants to know what people think about their products. This can be done by analyzing the customer comments such as reviews that people left in blogs, Twitter, Facebook, forums, e-commerce web sites, etc. Text mining is then a semi-automated process of extracting knowledge from unstructured textual data sources **(Aggarwal and Zhai, 2012)**. Most advanced text mining or text analytics software use sophisticated Natural Language Processing (NLP) algorithms.

NLP is a sub-field of Artificial Intelligence focused on enabling computers to understand and process human languages **(Indurkhya and Damerau, 2010), (Manning and Schutze, 1999)**. There are many fields of application of NLP. Text classification and categorization, Named Entity Recognition, Speech Recognition, Machine (language/idiom) Translation, Spam Detection, Sentiment Analysis (SA), etc. This last one, Sentiment Analysis (also known as sentiment detection or opinion mining), is the focus of this project.

SA focus is to extract the sentiment expressed in a piece of text (an entire document, a sentence, a review). Through the inherent reasoning that human being has, it can be an easy task for humans to understand if a piece of text is objective or subjective and, in case of subjective text, to classify it as positive, negative or even neutral in terms of sentiment expressed. However, it is a time-consuming task when the goal is to perform that task over millions of texts. That is why the need to use computational programs comes up. Sentiment Analysis helps in achieving various goals like observing public mood regarding political movement, the measurement of customer satisfaction, movie sales prediction and many more.

Computers are awesome working with structured and standardized data, and they can process data much faster than humans can. But humans do not communicate through “structured data” nor do speak binary. Humans use words to communicate and this is a form of unstructured data. Computers do not have the same intuitive understanding of natural language that humans do. Therefore, they cannot really understand what the language is really trying to say. That is why general NLP is not an easy subject and has some challenges.

The main goal of this project is to apply SA to documents/reviews containing people’s opinion about artificial intelligence, as a binary classification problem (positive opinion or negative opinion). Following the main goal, there are 3 sub-goals:

1) To understand the best techniques of feature extraction, feature selection and data preprocessing that better work for this problem;

2) To explore various machine learning algorithms and to understand the one(s) that better work(s);

3) To understand the best combination (measured by specific performance evaluation measures) between what and how many features to use and with which machine learning algorithm to use.

2. LITERATURE REVIEW – SENTIMENT ANALYSIS

Sentiment Analysis, one of the applications of NLP, is a classification task and the goal is to analyze people's opinion about an entity. Basically, it tries to answer the question "What people think about a certain topic?". It can be applied to financial markets, politics, voices of customers, sentiment analysis in spam email detection, etc. This last one is a classification problem that classifies a new email as a spam email or a normal email **(A. Heydari et al., 2015).**

Given two simple examples:

- A brand may want to know what people's opinion/sentiment is regarding the mobile that came up a few days ago. The objective is to know what improvements could make sense to include in the next generation of that mobile;

- During the campaign for the elections, through Sentiment Analysis is possible to understand whose campaign is more negative – the election results can be predicted from political posts.

Giving a more formal definition, Sentiment Analysis is the computational study of opinions, sentiments, emotions and attitude expressed in texts towards an entity **(W. Medhat et al., 2014)**.

SA can be considered as a classification problem. Regarding the type of classification in SA, it is possible to attribute one of two classes (for instance: positive or negative) which is called binary classification, or it can be classified in one of more than 2 classes (for instance: very positive, positive, negative and very negative) which is called multi-class classification.

There are 3 main levels of classification in SA: document-level classification, sentence-level classification and aspect-level classification.

Regarding the document-level classification, the whole document is considered as an information unit. It aims to classify the whole document in a positive or negative sentiment.

About sentence-level classification, it aims to classify the sentiment in each sentence. Sentences can be seen as short documents.

Aspect-level classification classifies the sentiment with respect to the specific aspects of entities **(W. Medhat et al., 2014)**.

Document-level and sentence-level classification do not provide the necessary detail needed about all the aspects of the entity. Let us suppose that reviews from a restaurant are being analyzed. The review, in general, can be classified, for instance, as positive; however, a particular part of the review can be about the high volume of the environment music that is unpleasant. So, at a general level, the restaurant has a positive review but, if the concern is related to the classification of the environment music, it should be classified as negative. This is true for both document and sentence level. This means that when it is supposed to get the sentiment associated to a specific aspect (the quality of the music in that restaurant) inside the general target (restaurant's opinion) it should be analyzed at an aspect level **(W. Medhat et al., 2014)**.

There are many resources that contains reviews, comments, posts, etc, expressing opinions about a specific target. Social networks as Facebook and Twitter, as well as micro-blogging sites, are very good sources of information because people just share their opinions and discuss them. Some micro-blogging sites like Twitter and Sina-Wiebo made available their Application Programming Interface (API) to collect public data from their sites **(Ravi K., Ravi V., 2015)**.

The points below describe the general idea of how SA flows:

1) First of all, is necessary to retrieval and prepare the text documents – there are lots of preprocessing work;
2) After data preprocessing, comes the Subjectivity Classification (also known as Subjectivity Detection) - it is important to detect if the statement is a fact or is about opinion; it just matters to keep going if it is about opinion;
3) Sentiment Classification: once it is ensured that it is an opinions piece of text, the goal is to classify it in one of two opposing sentiments: positive or negative (binary classification case).
4) Target Identification: it is necessary to identify accurately the target of the expressed sentiment.
5) In the end, once the sentiments of all text data points in the document are identified and calculated, they can be aggregated and it is possible to understand the general sentiment about that target.

Thus, the goal is to perform the above description using a variety of automated tools.

2.1. Data Preprocessing in Sentiment Analysis

Much of the effectiveness of NLP and particularly of Sentiment Analysis is dependent on the preprocessing of textual data. A big part of the accuracy achieved reflects the preprocess phase that data pass through. To build better algorithms is necessary to play with clean data. There are many useful procedures to help in the text cleaning and normalization.

2.1.1. Tokenization

Tokenization is the process of splitting paragraphs into sentences, or sentences into individual words **(Indurkhya and Damerau, 2010)**, **(Manning and Schutze, 1999)**. Each unit (each sentence, each word or something else like a number or a punctuation mark) is called token. This is an early step of processing because enables a much easier analysis. There are pre-trained algorithms for this. To split sentences into individual words, the most common approach is to split across white spaces. To split a paragraph into sentences, it is common to split it across punctuation followed by a capital letter. Although the majority of the European languages has whitespaces as delimiters of tokens, there are many languages such as Japanese and Chinese that do not use word spaces. So, in such languages, word segmentation is a much more major and challenging task. Then, tokenization can be split into 2 approaches: languages where the whitespace is the delimiter and languages where the whitespace is not the delimiter **(Jackson & Moulinier, 2002)**.

Some pre-trained algorithms (like sent_tokenize from nltk library from Python) already consider possible situations like, for instance, "Hello Mr. Smith, welcome home."; it will consider "Mr. Smith" as part of the same sentence even with the punctuation followed by a capital letter. In the other hand, word tokenization can have some problems when a word is abbreviated or is possessive. An example: "It's my first year in Lisbon", ["It", "'s", "my", "first", "year", "in", "Lisbon"]).

2.1.2. Capitalization

It is quite common that text has a variety of capitalization reflecting the beginning of sentences, proper nouns, etc. Therefore, without considering this, the algorithm would consider "Cool", "cool" and "COOL" as 3 distinct words. The most common approach to solve this issue is to reduce everything to the lower case for simplicity.

The problem is that at the same time it would be normally to keep the two types of Brown in Richard Brown and brown paint distinct or, for instance, "US" from United States can be reduced to "us" interpreted as a personal pronoun **(Manning et al., 2008)**. A simple heuristic is to change to lowercase letters the capital letters that appear at the beginning of a sentence and in things like headings and titles, while other words with capital letters that appear in the middle of the sentences are assumed names and their uppercase letters are preserved.

2.1.3. Stop Words

One of the problems regarding the NLP task is the high dimensionality. To reduce the processing time, it is important to reduce the dimensionality. Most of the words in a given text are connecting parts of a sentence rather than showing subjects, objects or intent. So, this kind of words like "the", "and" or in Portuguese "o", "a", "uma" are words that do not contribute a lot for the decision between positive or negative.

Stop words, also called empty words as they usually do not bear with much meaning, represent noise in the retrieval process. These words damage retrieval performance, since they do not discriminate between relevant and non-relevant documents **(Indurkhya and Damerau, 2010)**.

However, there are often words like "not" that are considered stop words and are dropped. This can be a big problem because, for instance, "I am not happy with the last Iphone that Apple introduced on the market". Considering "not" as a stop word, it is easy to lose the idea of denial. Just the words "happy", "last", "Iphone", "Apple", "introduced" and "market" are going to be kept on and the remain words are going to be dropped. Then, for sure the algorithm would classify the above example as a positive sentence. Usually, there are pre-built libraries of stop words. One might also create their own stop words dictionary.

2.1.4. Steaming

This is a process where the words are reduced to a root by dropping affixes (prefixes, suffixes, etc.) from the words. Stemming procedures aim to identify the stem of a word and use it in lieu of the word itself. Grouping words having the same root under the same stem (or indexing term) may increase the success rate when matching documents to a class **(Indurkhya and Damerau, 2010), (Manning et al., 2008)**.

Stemming procedures ignore word meanings and thus tend to make errors. For instance, the stem of “studies” is “studi”. Reductions may produce a root word that is not an actual word in fact. This does not necessarily adversely affect its efficiency. The danger is when for instance words like “universe” and “university” are reduced to the same root “univers”. Another situation is when there are 2 words that could be reduced to the same stem, for instance “studying” and “studies”, but in fact are reduced to different stems “study” and “studi” respectively.

2.1.5. Lemmatization

The goal of lemmatization technique is the same of the stemming technique. However, lemmatization is the process of reducing the inflected words to the root of those words bearing in mind the morphology of the word **(Manning et al., 2008)**. Therefore, based on the morphological analysis of the word, a word is reduced to its root word that is an actual word. The root word here is called Lemma. To do that, it is necessary to have detailed dictionaries that the algorithm can look through to link the form back to its lemma. An example: “studies” -> lemma: “study”, “better” -> lemma: “good”.

When there are words that are equal in terms of spelling but with different meanings, they are going to have the same stem but different lemmas. An example: in Portuguese “canto” that is the 1st form of the verb to sing and “canto” that means “corner”; the stem will be “canto” for both but, the lemma for the 1st one will be “cantar” (verb) and for the 2nd one “canto”.

When there are words that have different spelling but with similar meaning, these words are going to have different stems but the same lemma. For instance, “am” and “are”. The lemma of both words is “be”.

Both Stemming and Lemmatization, helps to achieve the root forms of inflected words. These are text normalization techniques. There are algorithms that have been developed to perform these both tasks. Stemmer is easier to build than lemmatizer because this last one requires the creation of dictionaries that allow the algorithm to look to the proper form of the word. Therefore, it is more difficult to build a lemmatizer but can conduct to better results.

2.1.6. Part-of-Speech Tagging (POS)

Part-of-Speech tagging is the process of making up a word in a corpus as corresponding to a particular part of speech based on both its definition and its context **(Indurkhya & Damerau, 2010)**. Therefore, each word of the document is classified, for instance, as a noun, or a verb, or an adjective, etc. There are words that can be, without any context, a noun and a verb at the same time. That is why it depends on the context. POS tagging is nothing more than a classification problem where each token has a probability of belong to a category (noun, adverb, interjection, etc) to be the class of that specific word. POS tagging is executed on tokens, after tokenization process.

2.1.7. Spelling Correction

Spelling correction is a useful pre-processing step because this helps to reduce multiple copies of words. There are of course typographic errors and misspelling. For example, “analytics” and “analytcs” will be treated as different words even they are used in the same sentence. There are also variants produced by punctuation and tokenization mainly due to the lack of naming convention. For instance, “Nurr77,” “Nurr-77,” or “Nurr 77”. Regional variations also introduce variants: the difference between British and American English for “colour” and “color”.

The standard strategy to reduce the negative impact caused by spelling errors or orthographic variation is to relate similar (however not identical) forms in one way or another **(Indurkhya & Damerau, 2010)**. However, it often takes some time to make these corrections and they may not to be accurate. Before the spelling correction step, some cases should be treated as following: “your” many times is written abbreviated like “ur”. If there is no previous treatment for this situation, the word “ur” might be transformed into “or”.

2.1.8. Rare Words Removal

The features that rarely occur in data are usually unreliable and they do not have much predictive power. Words that rarely occurs in a text can be removed. Because they are so rare, these words are going to add noise to the model. The model can overfit when rare words make part of the features of the model. For instance, if there are just 2 comments in the training test with that rare word and both are negative documents, then that feature will be considered a very discriminatory feature and the unseen comment will be classified almost sure as negative.

There are algorithms using a simple smoothing technique and eliminates those features that appear less than a threshold (e.g., less than 10 times). There are some other studies that use more sophisticated smoothing methods, such as using a Gaussian prior on the model parameters, which improve the performance when compared with the frequency cutoff technique **(Curran and Clark, 2003).**

2.1.9. Negation

Handling negation can be an important concern in opinion analysis. Consider the difference between *I really like this movie* (positive) and *I don’t like this movie* (negative). The negation expressed by *don’t* completely alters the inferences drawn from the predicate like. Similarly, negation can modify a negative word to produce a positive review (*don’t dismiss this film, doesn’t let us get bored*).

A commonly baseline used in sentiment to deal with negation is attaching “NOT” to words occurring close to negation, that use terms such as “no”, “don’t”, “never”, until the first punctuation mark. Thus, the phrase “*didn’t like this movie, but I*” becomes “*didn’t NOT_like NOT_this NOT_movie, but I*” **(Das and Chen, 2001)**. Newly formed words like NOT_like, NOT_recommend will thus occur more often in negative documents and act as cues for negative sentiment, while words like NOT_bored, NOT_dismiss will acquire positive associations.

2.2. Feature Extraction and Feature Selection

In Sentiment Analysis, the classifier does the attribution of negative or positive. The classifier will base the decision in a set of features. Thus, discriminatory features will help to get a better classification (based on specific performance evaluation measures). As a start point, let us assume that every word in a document is considered as a feature. Some of these words/features can be dropped applying the preprocessing steps previously presented. For the remaining features, some feature extraction techniques followed by feature selection techniques can be applied.

Feature extraction is the process of combining one or more existing features to create a smaller number of possible more insightful features. In this process, features are typically not chosen or disregarded, but simply combined.

- N-Gram Words - an N-Gram is a set of N successive words. For instance, “very good” in a 2-gram; “not good at all” is a 4-gram. The use of N-Grams as features provide more context. For instance, when the word “good” appears in a text, the natural trend is to say that this text is positive even if the actual expression that occurs is “not good”. These kinds of mistakes can be easily avoided with the introduction of n-grams as features. For each N-Gram as features, the goal is to count how many times it appears in a text/review/comment. The use of bi-grams usually improves the performance of the model. A higher order of n-grams has less obviously effects **(Jurafsky D. and Martin J. H., pp 38-43, 2018)**.

Feature selection is the process of choosing the features that are useful to classify all the texts/reviews/comments that are to be classified. While feature selection is also desirable in other classification tasks, it is particularly important in text classification due to the high dimensionality of text features and the existence of irrelevant (noisy) features. Techniques such as rare words removal, stemming and lemmatization are text normalization techniques that reduce the dimensionality by themselves. However, there are specific feature selection methods **(Aggarwal and Zhai, 2012)**. Feature selection methods can be divided into lexicon-based methods and statistical methods.

Lexicon-based methods need human annotation; usually, these approaches begin with a small set of seed words and then this set is expanded through synonym detection or online-resources to obtain a larger lexicon **(W. Medhat et al., 2014)**. This proved to have many difficulties as reported by Whitelaw et al, **(Whitelaw et al, 2005)**. Statistical approaches are fully automatic.

Below, some of the most frequently used statistical methods in feature selection:

- Point-wise Mutual Information (PMI) – this is a measure of the level of co-occurrence between a word and a class **(W. Medhat et al., 2014)**. When PMI value is greater than 0, it’s because the word is positively correlated to the class. The word is negatively correlated to the class when PMI value is less than 0. Let us assume there are two possible classes, positive and negative (binary classification). When the word w and positive class are positively correlated it means that the word w tends to occur in positive class; if, at the same time, the word w does not occur with negative class, this word is probably a discriminatory feature that should be kept (high PMI value). If the word tends to occur in both positive and negative classes, this word is not a discriminatory feature (small PMI value).

- Chi-Square – is a different way to compute the lack of independence between the word w and a particular class i. Both Chi-Square and PMI are different ways of measuring the correlation between terms and categories. One major advantage of the Chi-Square measure over the PMI is that it is a normalized value and, therefore, Chi-Square values are more comparable across terms in the same category **(Aggarwal and Zhai, 2012)**.

- Latent Semantic Index (LSI) - While feature selection attempts to reduce the dimensionality of the data by picking from the original set of attributes, feature transformation methods create a new (and smaller) set of features as a function of the original set of features. LSI is a typical example of such a feature transformation method. LSI transforms the text space to a new axis system, which is a linear combination of the original word features. To achieve this goal, Principal Component Analysis (PCA) techniques **(Jolliffee IT., 2002)** are used to determine the axis-system that retains the greatest level of information about the variations in the underlying attribute values. The main disadvantage of LSI is that it is an unsupervised technique blind to the underlying class distribution. Therefore, the features found by LSI are not necessarily the directions along which the class-distribution of the underlying documents can be best separated **(Aggarwal and Zhai, 2012)**.

- TF-IDF - if a word occurs multiple times in a document, we should boost its relevance, as it should be more meaningful than other words that appear fewer times (Term Frequency - TF). At the same time, if a word occurs many times in a document but also in many other documents, maybe it is because the word is just a frequent word and not, necessarily, because it is a relevant/meaningful word (Inverse Document Frequency – IDF). So, TF (word, text) = (nr of times the word occurs in the text) / (nr of words in the text); IDF (word) = log ((nr of texts)/(nr of texts where the word occurs)); TD-IDF(word, text) = TF(word, text) * IDF(word). **(Jurafsky D. and Martin J. H., pp 112-115, 2018).** There are words in a document like “the”, “is”, “of”, that occurs many times but are not important. We could resolve this situation by adding these words to a list of stop words and removing them before analysis. However, some of these words might be more important in some documents than others. This is why a list of stop words is not a sophisticated approach to adjusting term frequency for commonly used words.

2.3. SENTIMENT CLASSIFICATION APPROACHES

There are two main approaches to the problem of extracting sentiment automatically: a lexicon-based approach **(Scharl and Weichselbraun, 2008)** and a machine learning approach **(Boiy and Moens, 2009)**. It is possible to consider a hybrid approach as well - a mix of both lexicon-based and machine learning approaches **(Diana Maynard and Adam Funk, 2011)**.

2.3.1. Lexicon-based Approach

The lexicon-based approach calculates the semantic orientation of words in a review by obtaining words polarity from a lexicon such as the SentiWordNet **(Esuli and Sebastiani, 2006)**, **(S. Baccianella et al., 2010)**. A lexicon is a list of positive and negative words. SentiWordNet is an example of a domain-independent lexicon, which means that contains general positive and negative words not in a specific domain (for example, not specific about restaurants). It can be used a domain-specific lexicon whenever it is available since domain-specific lexicons better indicate the word polarities in that domain. For instance, the word "small" has a positive connotation in a cell phone domain and a negative connotation in a hotel domain.

Dictionaries for lexicon-based approach can be created manually or automatically using seed words to expand the list of words. There are 3 main approaches in order to collect an opinion word list: Manual approach, Dictionary-Based approach and Corpus-Based approach.

- Manual approach - very time consuming and it is not used alone. Usually, it is combined with the other two automated approaches as a final check to avoid mistakes that result from automated methods **(W. Medhat et al., 2014)**.

- Dictionary-Based approach – first, a small set (set W) of opinion words are collected manually with known orientations. Then, their synonyms and antonyms are searched, in an automatic way, in a well-known corpus (such as WordNet dictionary **(George A. Miller, 1995)**, **(Tufi and Stefenescu, 2012)**) and added in the seed set of words (set W). Then, a new iteration starts, and the synonyms and antonyms of the synonyms and antonyms added previously are also added. The iterative process stops when no new words are found. After the process is completed, manual inspection can be carried out to remove or correct errors **(Hu Minging and Liu Bing, 2004)**, **(Kim S, Hovy E., 2004)**. The big disadvantage of the dictionary-based approach is the inability to find opinion words with specific domain and context specific orientations.

- Corpus-Based approach - this approach helps to solve the problem of finding opinion words with specific context orientation (from specific domains). Therefore, a manual seed list is also created, with opinion words, and other opinion words are found in a large corpus based on syntactic patterns (words that occur together, etc) or patterns that occur together. For instance, the conjunction "and" can represent conjoined adjectives that usually have the same orientation. The word "but" can usually represent opinions changes. In order to determine if two conjoined adjectives are of the same orientation or not, learning is applied to a large corpus **(Hatzivassiloglou V and McKeown K, 1997)**. Corpus-based approach alone is not so effective than dictionary-based approach because it is hard to prepare huge corpus covering all English words. However, the advantage of this approach is the ability to find domain and context specific opinion words and their orientations using a domain corpus. The corpus-based approach is performed using a statistical approach or a semantic approach.

Statistical approaches can be used to find co-occurrence patterns or seed opinion words. The polarity of a word can be identified by studying the occurrence frequency of the word in a large annotated corpus of texts. If a word occurs more frequently in positive texts than in negative texts, then its polarity is positive; if the word occurs more frequently in negative texts, its polarity is negative. If it has equal frequencies, then it is a neutral word. The similar opinion words frequently appear together in a corpus. This is the main observation that these state-of-the-art methods are based on. If two words appear together frequently in the same context, they are likely to have the same polarity. Thus, the polarity of an unknown word can be determined by calculating the relative frequency of co-occurrence with another word **(W. Medhat et al., 2014)**. PMI **(P. Turney, 2002)** and LSA **(S. Deerwester et al.,1990)** are two statistical approaches.

2.3.2. Machine Learning approach

The text classification methods using a machine learning approach can be roughly divided into supervised and unsupervised (semi-supervised) learning methods. The supervised methods make use of a large number of labeled training documents. The unsupervised methods are used when it is difficult to find these labeled training documents. The supervised learning approach uses machine learning techniques to establish a model from a large corpus of reviews. It is necessary to get data (texts/documents/reviews) already labeled and divide this set of data into training data and test data. The classifier will learn, through the training data, to classify as positive or negative based on the considered features and then it is used the test set to understand if it was well learned based of specific error measures. Therefore, machine learning methods depend on the existence of labeled training documents. As already referred, sometimes it is difficult to create labeled training documents but it is easy to collect the unlabeled documents. Unsupervised learning methods overcomes this difficulty, therefore supervised approaches are typically more successful **(W. Medhat et al., 2014)**.

2.4. Machine Learning algorithms

2.4.1. Supervised Learning

Naïve Bayes (NB)

This is a probabilistic algorithm that computes the probability of, a document D, belonging to each one of the possible classes, based on the distribution of the words in the document. The class with higher probability is the one chosen as a label for the document D **(W. Medhat et al., 2014)**.

3 possible methods/models inside Naïve Bayes method can be considered: Multinomial NB, Bernoulli NB and Gaussian NB. In the first one, the considered features used by the classifier are the frequency

of the words present in the document. In the second one, the considered features are Boolean variables (if a word occurs in a text or not). The last one is when predictors take up continuous values (and this last one is not for the purpose of this work).

The line of reasoning of this algorithm: for each word of the unseen document, see the frequency (or the presence or not) of that word in positive training documents and the frequency in negative training documents. If the word appears more often in negative documents then the word is considered as negative word, otherwise is considered as positive word. Then, it is necessary to count the number of negative words and positive words that appear in this unseen document. If it has more negative words, it is labeled as negative. Otherwise, it is labeled as positive.

$$p(c|x) = \frac{P(X|C)p(c)}{p(x)}$$

Naïve Bayes **(1)**

Where:

- P(c|x) – prob. of class c, given the attributes x (the values of the features)
- P(c) – prior probability of class c;
- P(x|c) – prior prob. that a given feature set is being classified as a label.
- P(x) – prior probability that a given feature set is occurs.

(W. Medhat et al., 2014)

NB Assumption - the conditional independence assumption: the probabilities p(feature_i|C), for every i, are independents, given class C **(W. Medhat et al., 2014)**.

Despite its simplicity and the fact that its conditional independence assumption clearly does not hold in real-world situations, Naive Bayes-based text categorization still tends to perform surprisingly well **(Pang et al., 2002)**;

Kang and Yoo proposed an improved NB classifier to solve the problem of the tendency for the positive classification accuracy to appear up to approximately 10% higher than the negative classification. They showed that using this improved algorithm with restaurant reviews narrowed the gap between the positive accuracy and the negative accuracy compared to NB and Support Vector Machines (SVM) **(kang et al., 2011)**.

Bayesian Network (BN)

Bayesian Network is a probabilistic algorithm as well. The main assumption of the NB classifier is the independence of the features. The other extreme assumption is to assume that all the features are fully dependent. This leads to the Bayesian Network model which is a directed acyclic graph whose nodes represent random variables, and edges represent conditional dependencies. BN is considered a complete model for the variables and their relationships. Therefore, a complete joint probability distribution over all the variables is specified for a model **(W. Medhat et al., 2014)**.

In Text Mining, the computation complexity of BN is very expensive; that's why it is not frequently used. On the other hand, it has been shown that allowing limited levels of dependence can provide good tradeoffs between accuracy and computational costs **(Aggarwal and Zhai, 2012)**.

Logistic Regression (LR)

This is another probabilistic algorithm that computes the probabilities of a document D belong to each one of two classes or one of many classes and the class with higher probability is the one chosen as label for document D.

The line of reasoning of this algorithm: there are weights, wi, that are real numbers associated to each feature xi. Each weight represents how important a feature is for the classification decision. Positive weights mean that features are associated with the class and negative weights means that it isn't associated with the class. There is also a bias term, the intercept term, which is a real number associated to the weight inputs **(Jurafsky D. and Martin J.H., p.84, 2018)**. In this kind of problems, the dependent variable should take values between 0 and 1 and a threshold can be defined (usually 0,5) to decide if it takes the value 0 or 1 (that can represent positive or negative, spam or not spam email, etc).

$$z = \left(\sum_{i=1}^{n} w_{\mathrm{i}} x_{\mathrm{i}}\right) + b$$

Linear Probability Model **(2)**

Equation 2 represents the line of reasoning presented above. This is called linear probability model. This model has some limitations and the main one is that z takes values between (-) infinity and (+) infinity. That's why another model is adopted in which the dependent variable is, for sure, always between 0 and 1. Logit and Probit models are models that satisfy this condition **(Jurafsky D. and Martin J.H., pp 84-85, 2018)**.

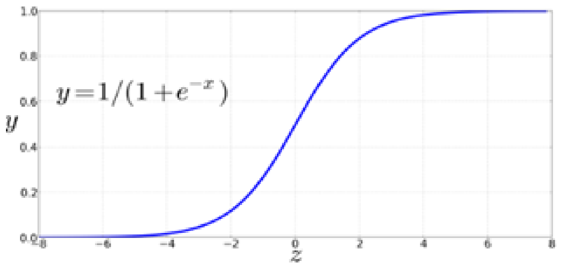


Figure 1 - Graphical representation of Sigmoid Function
Taken from **Jurafsky D. and Martin J.H., p.84, 2018**

Summing up, given values for features (x) considered in the model, the P (ŷ=1|x) is computed. If the result is bigger than the threshold/decision boundary, then ŷ = 1 (that represents one of the classes); otherwise, ŷ = 0.

Comparing both Naïve Bayes and Logistic Regression, NB strongly assumes the conditional independence of the features. Imagining that the same feature f1 was added twice: in this case, there are two features, f1 and f2, which are perfectly correlated once they are a copy of each other. Naïve Bayes will treat f1 and f2 as if they were separate, multiplying them both, overestimating the evidence. Logistic Regression is much more robust to correlated features; LR will assign part of the weight to f1 and part to f2. Then, when there are many correlated features, Logistic Regression will assign a more accurate probability than Naïve Bayes. Naïve Bayes doesn't work well with correlated features while Logistic Regression deal with correlated features once it can have better accuracy in probabilities; Naïve Bayes works extremely well (even better than Logistic Regression) on very small datasets **(Ng and Jordan, 2002)** or short documents **(Wang and Manning, 2012)** and it's easy to implement as well as very fast to train; Logistic Regression works better on larger documents or datasets **(Jurafsky D. and Martin J.H., p.87, 2018)**.

Support Vector Machine (SVM)

Support Vector Machines **(Cortes and Vapnik, 1995)** are a set of supervised learning methods used for both classification and regression. SVM is considered a linear classifier.

The line of reasoning of this algorithm: it is a kind of large-margin classifier, rather than probabilistic, in contrast to Naïve Bayes and Logistic Regression, **(Pang et al., 2002)**. It is a vector space-based machine learning method where the goal is to find the decision boundary (geometrically talking, a hyperplane) between two classes which is maximally far from any point in the training data (possible discounting some points as outliers or noise).

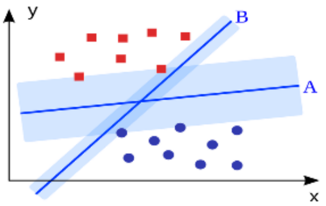


Figure 2 - Linear separable binary classification problem
representing 2 different possible boundaries

In Figure 2, the line A is the chosen one as decision boundary for test data point classification once the line A has a larger margin than the line B. The points divided by the line A have to travel much more to cross the division than if the data was divided by B.

When there is a two-class separate training set, there are lots of possible linear separators. For instance, Perceptron algorithm (Neural Networks) find just any linear separator, Naïve Bayes search for the best linear separator according to some criteria; the SVM criterion is a decision surface that is maximally far away from any data point.

The problem of finding the optimal hyperplane is an optimization problem and can be solved by optimization techniques (like Lagrange Multipliers).

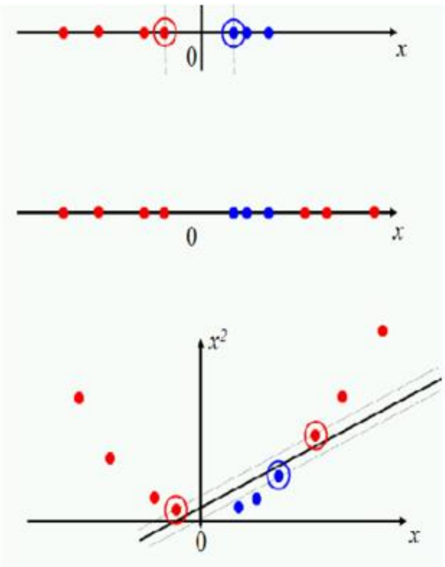


Figure 3 - Projecting data that is not linearly separable into a higher dimensional space can make it linearly separable

Non-Linear Separable problems – this happens when the data set does not allow the classification by a linear classifier. One way to solve this problem is to map the data on a higher dimensional space and then to use a linear classifier in that dimensional space. The idea is to map the original feature space to some higher dimensional feature space where the training set is separable; this is done through Kernel functions **(Hofmann et al., 2008)**.

SVM have been shown to be highly effective at traditional text categorization, generally outperforming Naive Bayes**, (Joachims, 1998)**.

Artificial Neural Networks (ANN)

Artificial Neural Networks are computational techniques that aim to realize a very simplified model of the human brain trying to mimic the behavior of the brain to solve problems. Neural networks are considered linear classifiers and consist of many neurons where the neuron is its basic unit. Each neuron is able to only perform very simple tasks, and Neural Networks are able to perform complex calculations because they are typically composed of many neurons, strongly interconnected between each other.

While a large variety of networks have been proposed in the literature **(Haykin and Network, 2004)**, the main structure shared by the different models is the one in which the network is composed of units (also called neurons) and connections between them, which together determine the behavior of the network **(M. Castelli et al., 2019)**.

The inputs to the neurons are the word frequencies in each document. There is a set of weights which are associated with each neuron used in order to compute the function of its inputs f(.) **(W. Medhat et al., 2014)**.

The choice of the network type depends on the problem to be solved. Perceptron is a simple Neural Network that can be composed of one single neuron or several neurons arranged in a single layer. By its definition, it can only have one layer of neurons, but that layer can have m neurons. Perceptron is able to solve linear separable problems (binary and multi-class classification problems). It can separate data into 2^m distinct classes.

Multi-layer neural networks can use the Back-Propagation algorithm to learn **(Hecht-Nielsen et al., 1988)**. The input of the neurons in a given layer are the outputs of the neurons of the previous layer. The training process is more complex because the errors need to be back-propagated over different layers. To solve non-linear separable problems, it must be used a neural network with more than 1 layer.

There are implementations of Neural Networks for text data which are found in **(Ruiz and Srinivasan, 1999)** and **(Ng Hwee Tou et al., 1997)**.

Comparison between SVM and ANN were presented regarding document-level sentiment analysis **(Moraes and Valiati, 2012)**. Their experiments indicated that ANN produced superior results to SVM except for some unbalanced data contexts. They confirmed some potential limitations of both models that have been rarely discussed in the SA literature, like the computational cost of SVM at the running time and ANN at the training time.

Decision Trees

A decision tree is a kind of machine learning algorithm that can be used for classification and regression. A tree is a set of nodes connected through edges and where loops are not present. Decision tree learning is used to approximate discrete-valued target functions, in which the learned function is represented by a decision tree. Learned trees can also be re-represented as sets of if-then rules to improve human readability **(Michalski et al., 2013)**. Decision tree classifier provides a hierarchical decomposition of the training data space in which a condition on the attribute value is used to divide the data **(J.R. Quinlan, 1986)**. When applied to text classification problems, the condition or predicate is the presence or absence of one or more words. The division of the data space is done recursively until the leaf nodes contain certain minimum number of records which are used for the purpose of classification **(W. Medhat et al., 2014)**. In a decision tree the internal nodes are tests and leaf nodes are categories. Each internal node tests one attribute and each branch from a node selects one value for the attribute. The attribute used to make the decision is not defined. So, the attribute which gives maximum information can be used.

The decision tree implementations in text classification tend to be small variations on standard packages such as ID3, and C4.5 both proposed by Quinlan **(Quinlan, 1993)**. Li and Jain have used the C5 algorithm, which is a successor to the C4.5 algorithm. According to the experimental results, Naïve Bayes classifier outperforms Decision Trees classifier **(W. Medhat et al., 2014)**.

2.4.2. Weakly Supervised Learning and Unsupervised Learning

It can be difficult to create labeled training documents in text classification, but it is easy to collect unlabeled documents. While it’s easy to collect the unlabeled documents, it is not easy to manually categorize them for creating training documents. The unsupervised learning methods overcome this difficult.

In many scenarios, while users cannot afford to label many documents for training a classifier, they can provide a small amount of seed information for the classification task. Such seed information may arrive in various forms: a set of representative keywords for each class, a few labeled documents, or even only the surface names of the classes. Such a problem is called weakly-supervised or semi-supervised text classification **(Yu Meng et al., 2018)**.

A research work was presented by Ko and Seo **(ko and Seo, 2000)**. The proposed method divides the unlabeled documents into sentences and categorizes each sentence using keyword lists of each category and sentence similarity; the keyword lists are defined for each category by hand, which contain special features of each category sufficiently. Then it uses the categorized sentences for training. This proposed method shows a similar degree of performance, compared with the traditional supervised learning methods.

Another strategy was proposed by Yulan and Zhou **(Yulan and Zhou, 2011)**. They have proposed a strategy that provides weak supervision at the level of features rather than instances. They obtained an initial classifier by incorporating prior information, extracted from an existing sentiment lexicon, into sentiment classifier model learning. They refer to prior information as labeled features and use them directly to constrain model’s predictions on unlabeled instances using generalized expectation criteria. In their work, they were able to identify domain-specific polarity words, clarifying the idea that the polarity of a word may be different from a domain to the other. They showed that their approach attained better performance than other weakly supervised sentiment classification methods and it is applicable to any text classification task where some relevant prior knowledge is available.

2.5. Performance evaluation

To understand how good a technique (a classifier) is, it is fundamental to understand how it performs when applied to unseen data. There are performance measures used to evaluate binary classifiers and performance measures used to evaluate multi-class classifiers. For the purpose of this work, only matter the measures for binary classifiers.

In binary classification, each observation belongs to one of two classes. Let's suppose the set {p,n} contains the two possible classes where p denotes the positive class and n denotes the negative class.

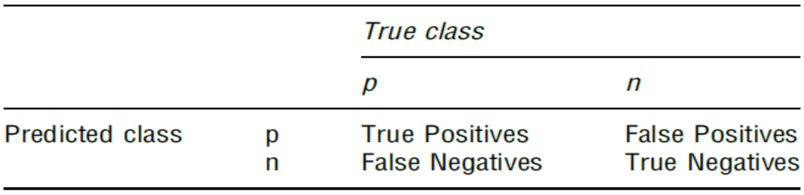
TP – "True Positive" – if the instance is positive in fact, and is classified as positive;

FN – "False Negative" – if the instate is positive in fact, but is classified as negative;

TN – "True Negative" – if the instance is negative in fact, and is classified as negative;

FP – "False Positive" – if the instance is negative in fact, but is classified as positive;

Given a classifier and a set of observations, it can be built a two-by-two confusion matrix:



		True class	
		p	*n*
Predicted class	p	True Positives	False Positives
	n	False Negatives	True Negatives

Table 1 - Confusion matrix. Across the top are the true class labels and down the side are the predicted class labels. Each cell contains the number of predictions made by the classifier that fall into that cell.
(M. Castelli et al., 2019)

The numbers along the major diagonal of the confusion matrix represent the correct decisions made. Based on the information reported in the confusion matrix it is possible to get the most often measures to evaluate the performance of binary classifiers:

1) Accuracy: ratio of the number of correctly predicted observations and the total number of observations **(M. Castelli et al., 2019)**.

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

Accuracy represents the overall effectiveness of a classifier. One may think that the higher the accuracy, the better the model. However, considering just the accuracy as a performance measure can lead to some misleading interpretation about the classifier. Accuracy is a great measure but only when the datasets are balanced – the number of observations belonging to one class is similar to the number of observations belonging to the other class.

For instance, the problem of trying to identify who is terrorist in a flight. One model for this problem with more than 99% of accuracy is to classify every single person as not terrorist. Given 800 million average passengers and 19 confirmed terrorists, this model achieves an accuracy of 99,9999%. This is an example in which accuracy can be clearly not an adequate metric. The terrorist detection task is an imbalanced classification problem. To avoid this issue regarding the accuracy measure, other measures are also considered.

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

(4) (5)

2) Precision: the number of true positives divided by the total number of observations classified as positive. This measure can be thought of as a measure of classifiers exactness **(M. Castelli et al., 2019)**.

3) Recall: the number of true positives divided by the total number of observations that are positive. Recall is used to express the effectiveness of a classifier to identify positive labels **(M. Castelli et al., 2019)**.

   Let us assume that positive class is to be terrorist. If we label all observations (individuals) as terrorists, recall goes to be 1.0. However, although recall=1, it is not a perfect classifier. When we increase the recall, we decrease the precision. In this case, precision will be 0.

4) F-measure: is a measure that allows the evaluation of the performance of a classifier considering both precision and recall **(M. Castelli et al., 2019)**.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

(2)

5) AUC – Area under the curve: to introduce AUC measure it is necessary to introduce the concept of ROC (Reception Operating Characteristic) curve **(Fawcett,2006)**. ROC curve is a graphical plot that illustrates the performance of a binary classifier system. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various values of the threshold. Let us consider the (quite popular) case where the output of the model is a number in [0,1]. In this case, to combine the FPR and the TPR into one single

metric, first it is computed the two former metrics with many different threshold values (for example 0,00; 0,01; 0,02; … ;1,00), then the corresponding points are plotted on a single graph, with the FPR values on the abscissa and the TPR values on the ordinate. The resulting curve is called ROC curve, and the metric we consider is the area under this curve (AUC), also called AUROC. The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification **(Fawcett,2006)**.

3. FIELD OF APPLICATION: ARTIFICIAL INTELLIGENCE

3.1. ARTIFICIAL INTELLIGENCE AND ITS IMPORTANCE

Nearly everywhere we look today, we see intelligent systems talking to us (such as Siri), offering recommendations (such as Netflix, Spotify and Amazon), providing financial advice (Schwab's Intelligent Portfolio) and winning game shows (IBM's Watson). We see systems emerging to improve voice recognition, image interpretation, face recognition and even self-driving cars, based on techniques such as deep learning efforts. Other work aims to advance natural language understanding and generation so that the machines can communicate with us on our own terms.

Artificial intelligence (AI) is a sub-field of computer science. Computer Science is the study of computers and computational systems. Computer scientists deal mostly with software and software systems; these include their theory, design, development and application. Artificial Intelligence aims at the development of computers capable of doing things that are normally done by people - in particular, things associated with people acting intelligently. It is the theory and development of computer systems able to perform tasks normally requiring human intelligence. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans such as the ability to reason, discover meaning, generalize or learn from experience. The term AI was coined in 1956, but AI has become more popular today thanks to increased data volumes, advanced algorithms and improvements in computing power and storage.

Artificial Intelligence has two key components: Automation and Intelligence. AI performs automated tasks using intelligence. Some of the sub-fields of AI are: Reasoning, Automated Learning and Scheduling, Machine Learning, Natural Language Processing, Robotics, General Intelligence. It works by combining large amounts of data with fast, iterative processing and intelligent algorithms, allowing the software to learn automatically from patterns or features in the data.

The AI systems are efficient enough to reduce human efforts in various areas. That makes artificial intelligence such an essential and special technology right now. In order to perform various activities in the industry, many of them are using artificial intelligence to create machine slaves that perform various activities on a regular basis. The artificial intelligence applications help to get the work done faster and with accurate results. Error free and efficient worlds are the main motivations behind artificial intelligence. In the recent years, many sectors have started using AI technology to reduce human efforts and to get efficient and faster results. Lots of data are necessary to train deep learning models because they learn directly from the data. The more data we can feed them, the more accurate they become.

3.2. ARTIFICIAL INTELLIGENCE AS A CONTORVERSIAL FIELD

AI is changing the way the world operates, in both our personal and our business lives. Many benefits are possible to get from this kind of technology. However, there are some reasons why many people have doubts regarding to this subject.

People are worried about the impact of AI on jobs. However, according to a recent study from **PricewaterhouseCoopers (PwC)**, AI will create 7.2 million jobs over the next 20 years, slightly more than the seven million predicted to be automated. However, the impact of new technologies will not be felt evenly - sectors like healthcare and education are set to benefit, whereas manufacturing and transport are estimated to see the largest jobs decrease. That's why the new technologies will have an uneven effect on industries. Robots have already taken many jobs on the assembly line – but now this could extend to new levels. Take, for example, the concept of driverless cars, which could displace the need to have millions of human drivers, from taxi drivers to chauffeurs, very quickly.

Regarding to the distribution of power, in the minds of some people, AI carries the risk of taking control away from humans. Nations that are in possession of artificial intelligence could theoretically kill humans. An example of an uncontrol case was when Facebook designed chatbots to negotiate with one another, and the bots made up their own way of communicating. The model that allowed two bots to have a conversation—and use machine learning to constantly iterate strategies for that conversation along the way—led to those bots communicating in their own non-human language and no one could understand that conversation.

There are accidents that came up from artificial intelligence errors. Who is criminally liable when, for instance, a self-driving car fatally strikes a pedestrian? This is another debate point about this theme.

Some people think about if it is ethically and morally correct to have androids, human-like robots or recreate intelligence, a gift of nature that shouldn't be recreated.

AI has become a topic of debates and discussions where for some people it is consider a blessing and for others it is a disaster. Having said that, is artificial intelligence a threat or a blessing? What is the general opinion of people/society regarding Artificial Intelligence?

3.3. SA TECHNIQUES FOR ADDRESSING SOCIETY'S OPINION ON ARTIFICIAL INTELLIGENCE

Sentiment Analysis is a subfield of Natural Language Processing that is a subfield of Artificial Intelligence. Artificial intelligence is a controversial topic once people are divided between the efficiency of those technologies and the possible problems that come up from them. There are lots of people that agree with their use and development and others that believe it is not good for society in general for a lot of reasons, some of them presented above.

An artificial intelligence technique is going to be used - Sentiment Analysis - to help in the analysis of opinions about artificial intelligence topic, in order to understand the general people's opinion about it.

4. METHODOLOGY

The goal of this project, as already referred, is to develop a model that evaluates the polarity of documents containing opinions regarding to artificial intelligence. Figure 4 contains an overview of all the phases of the work.

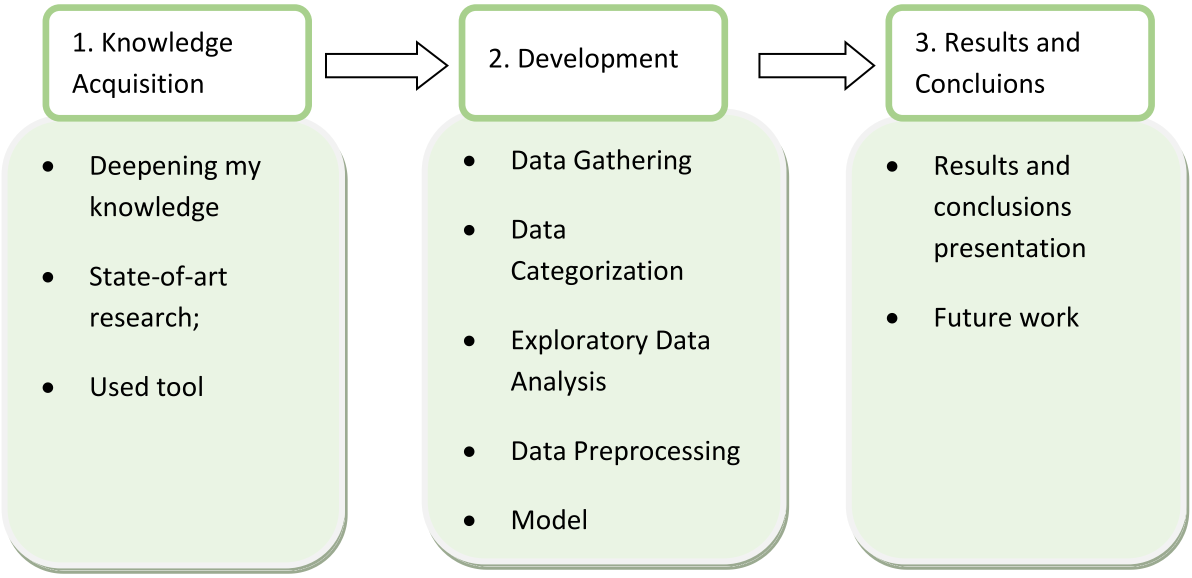


Figure 4 - General overview of the project steps

The first phase consisted in the acquisition of the first notions about sentiment analysis techniques as well as in the continuous development of the knowledge in these fields. An oriented learning process of the main tool used (Python – Scikit-Learn library) was also done. Once familiarized with the main concepts, a deep literature review was performed in order to get a further knowledge detail and to present the state-of-art.

The second phase is about all the process regarding to the achievement of the final model. It's going to be described with detail in the next subchapters.

In the last phase, the results that come up from the development phase are presented and discussed. Conclusions and suggestions for future work are the last topics presented.

The developed model was trained and tested, however it was not applied to new data (reviews). This means that there was not a final conclusion about the general people's opinion related to artificial intelligence. Therefore, this conclusion could be achieved by applying the model to a new dataset gathered from identical conditions (is the same environment), to ensure it would perform as it did when applied to the test set.

4.1. DEVELOPMENT PHASE

4.1.1. Data Gathering

The collected data consists in a list of opinions about what people think about artificial intelligence, bearing in mind different topics inside this theme such as: loss of jobs or not, self-driving cars, medical issues, AI as a substitute of humans. As it was already referred in literature review, there are 3 main levels of classification in Sentiment Analysis and, in this case, the classification was done at document-level. This means that the opinions are about different topics inside artificial intelligence, but the goal is to classify the sentiment about AI at a general level and not based on an aspect-level (like, for instance, driverless cars).

Opinions were collected, by hand, from different sources like Twitter and Facebook posts and comments of that posts, Quora, comments left in online news, comments from Youtube videos related with this subject. The key words used to search for such data were "Artificial Intelligence", "Automation", "Self-Driving cars", "Intelligent Machines", "Robotic". The data is not restricted to a specific geo-localization neither to a specific population target. Once the data was collected manually, it was possible to previously guarantee the subjectivity of each instance.

Thus, 230 opinions were collected. The small size of the dataset was the biggest constrain in the achievement of a better accuracy in the final model.

4.1.2. Data Categorization

In order to apply a supervised machine learning technique, the data must be labeled in order to train it and then to test it. Thus, it was necessary to attribute one class to each opinion. Considering the two referred approaches of sentiment classification, lexicon-based approach was the followed one.

Lexicon-based approach was applied using Liu and Hu opinion lexicon **(Liu & Hu, 2004)**. Based on word polarity from this dictionary, it was simply counted the number of positive, negative and neutral words in the document, and that document was classified depending on which polarity was more represented. Words that do not appear in the lexicon are considered as neutral. A neutral list was also created for the opinions that have as many positive words as negative words. At the end of this task, the opinions were classified in one of three classes: positive, negative or neutral.

Regarding to the neutral list of opinions, it was performed a manual classification to set the opinions labeled as neutral to the positive or negative class because any neutral class was considered (binary classification problem).

One important thing to take into account is that the associated error in this technique is propagated to the final model error, once the final model is going to be trained above this previous labeled data. Having said that, a quality control was performed to ensure the smallest possible error, by making a manual check. Thus, a percentage of opinions (randomly selected) were read and moved from one class to another in case of wrong initial classification. Because there are just 230 opinions, it was possible to manual check 100% of the opinions.

	Initial (after lexicon-based algorithm)	After negative opinions treatment	After positive opinions treatment
Positive	139	153 -32 →	121
Negative	91 -14 →	77	109

Table 2 - Quality control: resulting movements

After the application of the lexicon-based algorithm, table 2 presents how many opinions were in each one of the positive and negative classes (after the treatment of the neutral list of opinions), after each phase of the quality control. During negative opinions treatment, 14 opinions were moved from negative class to positive class. During positive opinions treatment, 32 opinions were moved from positive class to negative class. Thus, a total of 46 opinions were wrongly classified. This means that the inaccuracy associated to this classification algorithm for this dataset was 20.0%.

Unbalanced data is a common issue that comes up a lot of times. However, in this case, the proportion between positive and negative opinions is quite well balanced having a distribution of 52.6% of positives and 47.4% of negatives.

4.1.3. Data Preprocessing

The data preprocessing is an extremely important phase because the collected text data is unstructured. To introduce some structure to data and to normalize it, some data preprocessing techniques already referred in literature review, were applied.

Some components were removed from the text such as:

- Punctuation marks;
- Stop words: it was created a customized stop words list:

  ['the' , 'a' , 'an' , 'in' , 'on' , 'if' , 'of' , 'or' , 'at' , 'as' , 'what' , 'which' , 'this' , 'that' , 'these' , 'those' , 'just' , 'so' , 'both' , 'through' , 'about' , 'for' , 'is' , 'while' , 'during' , 'to' , 'it']

The removal of punctuation marks was the first preprocessing step performed. This process also removed apostrophe mark which means that words like “didn’t” were transformed in “didnt”.

This customized stop words list, when compared to the pre-defined stop words list from NLTK from Python, doesn’t contain many words that were considered as having a little possibility of being discriminative words such as all words representing negation. In this customized stop words list, also some new words were included.

The text was also all transformed in lowercase, and some words were replaced. Examples of some used replacements: "isnt" by "is not", "arent" by "are not", "didnt" by "did not", "wont" by "will not”.

Lemmatization and Stemming were both considered and tested and, according with literature review, Lemmatization tends to work better and was the applied one. These steps were crucial to get normalized and cleaned text.

Every unseen document to be classified must go through this data preprocessing steps.

4.1.4. Exploratory Data Analysis

In order to go into further detail in the knowledge of the data set, after data cleansing and normalization, a brief exploratory data analysis was performed.

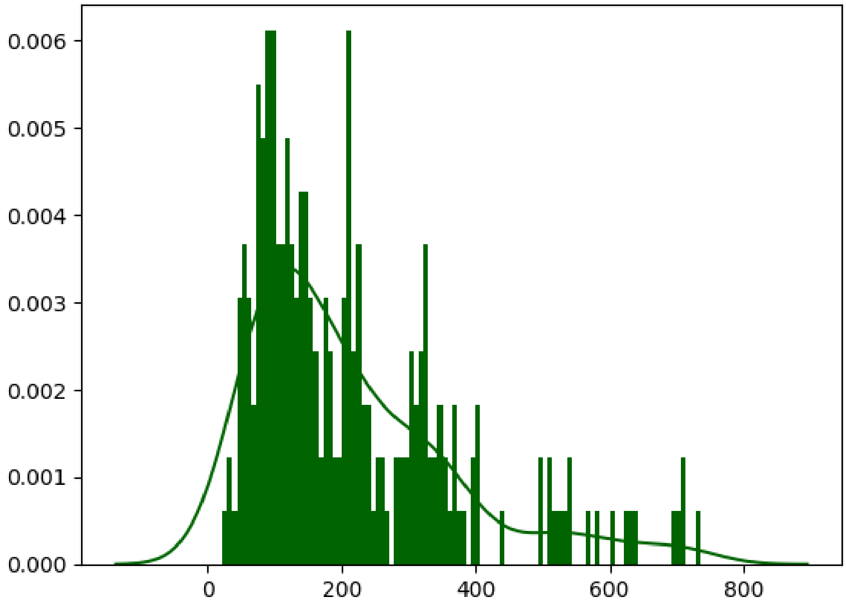


Figure 5 - Opinions length distribution

Figure 5 represents the distribution of the number of characters that each opinion has. The distribution is positively asymmetric. The comments have an average of around 150 characters and those ones that escape of this pattern usually tend to have some more characters. Once there are not any data point (review) much bigger or smaller than the other ones in terms of number of characters, it doesn’t seem to exist any outliers, so no outlier removal was necessary.

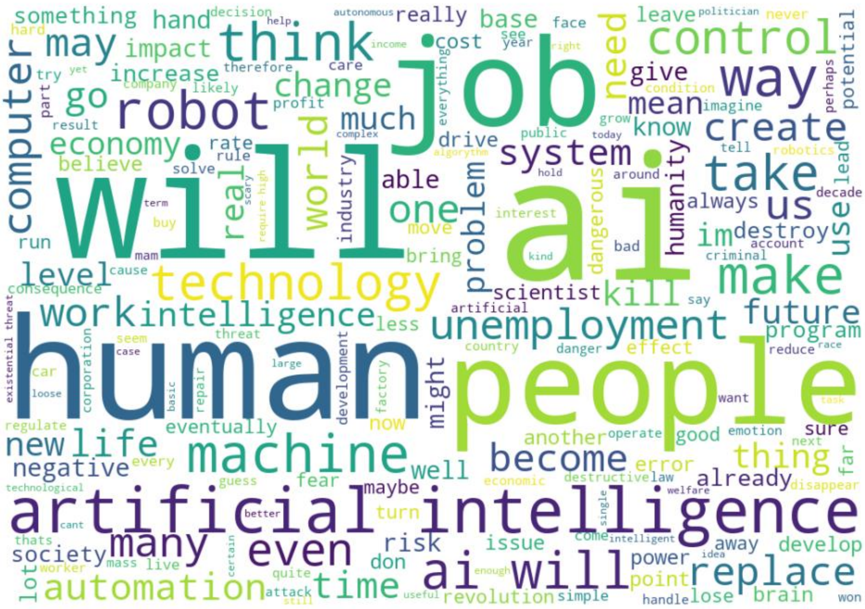


Figure 6 - Words Cloud

A cloud of the words in study was created in order to obtain, in a visual way, the words with bigger frequency in all the dataset after removing stopwords, applying lemmatization, applying lowercase and removing punctuation marks. In the words cloud, the bigger and more visible the word, the higher the frequency of that word. Words or bi-grams such as “will”, “job”, “ai”, “human”, “artificial intelligence”, “ai will”, “robot” are an example of some of the words most referenced in the collected dataset.

4.1.5. Feature Selection

In order to apply a machine learning algorithm to a textual dataset, it is necessary to convert the dataset to a numeric representation. For that purpose, it was used a term-by-document matrix that is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. Usually rows correspond to documents in the collection, columns correspond to terms and each cell represents the frequency. In terms of the value presented in each cell, it also can be considered TF-IDF scores instead of frequency. As already referred in literature review, this measure doesn’t consider a word as an important/meaningful word just by its high frequency in a document, but also takes into account its frequency in the corpus. Thus, words with higher TF-IDF scores are words with high frequency in a specific document and low frequency in all the remaining ones.

Based on works already done and published and by logical reasoning, in general TF-IDF scores can work much better by achieving better performance results. Nevertheless, both of Term Frequency and TF-IDF scores were tested and the final model uses TF-IDF scores. In fact, globally, results were better using TF-IDF scores. In the next chapter part of the results are presented.

For both Term Frequency and TF-IDF, bi-grams were considered, rare words were removed and words that appear a lot of times were also removed. After some experiences, every term that did not appear in 2 documents, minimum, was removed. Terms that appeared in half of the documents or more were also removed. Typically, these terms are the ones that can be compared to stop words but from a specific domain - artificial intelligence domain. An example: "Artificial intelligence" is a bi-gram that probably appears in many documents (positives and negatives). Thus, it won't help in the decision of the polarity of a document.

With the previous techniques, the number of features was already reduced. In order to select only the best features and, thus, reduce the number of features even more, statistical methods were applied: Mutual Information[1], Chi2, F-test[2] and LSI. The values of these statistics are the criteria to decide which words are the best ones to improve the learning process. The best ones are the words most related with the target value. Thus, after executing a lot of runs with different parameters, for each one of these statistical methods, the best combinations between the top percentile of features and the best statistical method were found. The one that worked better was Mutual Information in general. The tables containing the values can be consulted in the annexes.

4.1.6. Model

Once the data preprocessing phase was completed, the data was partitioned in training set and validation set according to the following proportions: 70% for training, 30% for validation. This choice (70%-30%) is a traditional division in Machine Learning. The proportions of the different values of the target variable (positive or negative) were approximately kept in each data set. Having the data partitioned, different models were created (by applying different ML algorithms with different parameters) in order to get the one that gives the best accuracy percentage (always considering precision and recall measures as well and AUC value).

To find a model that is consistent and to ensure the behavior of the classifier won't be much different when applied to a new/unseen data, the K-Fold Cross Validation technique was used. This method helped avoiding models that just performed well in some instances and do not have good generalization ability.

In K-Fold Cross Validation, the original sample is randomly partitioned into k equal size subsamples. Of all the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds) with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation.

Thus, it was decided to use K=3 once it is the number that better approximate the division in 70% - 30%.

7 different machine learning algorithms were tested: Stochastic Gradient Descent (SGD), SVM, LR, Random Forest (RF), ANN, Multinomial NB and Bernoulli NB. According to the literature review, and summing it up, experiences indicates that ANN tends to produce superior results than SVM, NB works extremely well on very small datasets or very short documents, Logistic Regression works

[1] Mutual Information is just the average of PMI (referred in literature review) for all possible outcomes.

[2] F-test is a statistical test to determine whether group means are equal.

better on larger documents or datasets, and SVM have shown to be highly effective at traditional text categorization, generally outperforming NB. Although the results and conclusion described, experiences for this specific application work were done with each one of the 7 algorithms referred.

The methodology followed until achieve the best models was the following:

- Every type of runs was performed for both Term Frequency and for TF-IDF scores.
- For each one of the 7 algorithms, runs were executed with different parameters, always using the python pre-defined feature selection technique – F-test – and percentile 9 to select just the best 9% of the features.
- Then, keeping constant the parameters that achieved the best result for each algorithm (measured by average accuracy, average precision, average recall, average f1 measure and standard deviation of de average accuracy), a set of new runs was performed variating the percentile, which means variating the number of features used. This was done for each one of the feature selection techniques: Mutual Information, Chi2 and F-test.
- Just for Neural Networks, after achieving the best percentile and using the best feature selection technique, more runs were done just by changing the number of hidden neurons in each one of the 2 hidden layers.
- Finally, just for Neural Networks (once the best results were achieved by them), runs were done using LSA, by variating the number of principal components used. It was done just for TF-IDF scores. LSA was not experimented for Naïve Bayes and others, considering that worse results were achieved for Neural Networks using LSA.

SGD	**SVM**	**LR**	**RF**	**NN**	**Multinomial NB**	**Bernoulli NB**
152	98	56	70	172	98	140

Table 3 – Resume of the performed runs

Table 3 represents the total number of experiences executed. Thus, a total of 786 runs were performed using different algorithms, different parameters and different features selection methods (and thus different features).

The best results were achieved using Naïve Bayes (Multinomial and Bernoulli) and Neural Networks algorithms, based on TF-IDF scores and using Mutual Information but considering different number of features used (different percentiles for select the top best features). Table 4 presents the 3 best models achieved in terms of accuracy and AUC value. Results are presented in the next chapter.

id	Algorithm	Smoothing parameter	Percentile and respective average of the number of features used
1	Bernoulli NB	0.01	58% 528 features in average
2	Multinomial NB	0.05	59% 537 features in average

id	Algorithm	Number Hidden Neurons	Activation Function	Solver	Regularization Term	Tolerance	Percentile and respective number of features used
3	Neural Network	70, 67	Relu	lbfgs	1.05	0.005	55% 501 features in average

Table 4 – The three best models

5. RESULTS AND DISCUSSION

For all parameters tested for each one of the 7 algorithms, runs were executed based on Term Frequency and TF-IDF scores. For the majority of them, best results were achieved through TF-IDF scores as already referred and those considered as the 3 best models are not exception. This is possible to verify by analyzing the values in Tables 5 and 6. The results for some of the remaining models can be seen in annexes.

Model ID	Accuracy	Precision	Recall	f1	St_Dv	AUC	St_Dv_AUC	mean features
1	0,671	0,619	0,617	0,616	0,033	0,67	0,02	528
2	0,634	0,634	0,634	0,632	0,052	0,68	0,02	537
3	0,596	0,601	0,596	0,594	0,055	0,64	0,05	501

Table 5 – Average measures of each model based on Term Frequency

Model ID	Accuracy	Precision	Recall	f1	St_Dv	AUC	St_Dv_AUC	mean features
1	0,691	0,691	0,691	0,687	0,075	0,720	0,050	528
2	0,696	0,699	0,696	0,691	0,034	0,730	0,040	537
3	0,705	0,706	0,705	0,704	0,055	0,710	0,020	501

Table 6 - Average measures of each model based on TF-IDF scores

Table 6 represents the average measures on the 3-fold cross validation for each one of the 3 best models. According to the accuracy values, model 3 is the best one – the one that uses Neural Network algorithm. The accuracy of this model is 70.5%. Once the data is quite balanced between both classes and the k-folds partition is stratified, which means that the folds are made by preserving the percentage of samples for each class, accuracy is a very good measure. However, complete balanced is not the same as "almost complete" balanced and the data contains more 5.2% of positive reviews than negative reviews.

Tables 7, 8 and 9 presented below represent the confusion matrices for each model that compares the real values with the predicted values, giving the classification error of each class. The values of each confusion matrix presented are the mean values of each one of the confusion matrices that came up from 3-fold runs. As it is presented, for each one, the number of false positives and false negatives is not exactly the same. The 3 models tend to better perform in the prediction of positive opinions - the associated error in the positive class is lower than in the negative class; in the 3 models there are always more false positives than false negatives. The difference is not that big, but the quantity of test data is not that much as well. Thus, a small variation in the number causes a considerable difference in the error percentage. This little difference in the error can be a consequence of the small imbalance in the data.

		Predicted		
		Positive	Negative	Error
Actual	Positive	30	10	0,250
	Negative	14	23	0,378

Table 7 - Confusion Matrix Model 1

		Predicted		
		positive	negative	Error
Actual	Positive	32	8	0,200
	Negative	15	21	0,416

Table 8 - Confusion Matrix Model 2

		Predicted		
		positive	negative	error
Actual	positive	30	10	0,250
	negative	12	24	0,330

Table 9 - Confusion Matrix Model 3

Thus, even with just a little imbalance in data, precision, recall and f1 values were also taken into account. All these measures can be analyzed in Table 6. However, f1 score (that combines both precision and recall scores as referred in literature review) is almost the same of the accuracy value for the 3 models. Model 3 has the highest f1 score 0.704 (and the highest accuracy precision and recall values), which are pretty good values.

Model 3 (from table 6) seems to be the best one considering that it has the best accuracy and f1 values and the average error value, between both classes, is the lowest one. However, an insightful measure is AUC that comes up from ROC Curve.

Figure 7, 8 and 9 show the ROC curve graphic for each one of the folds and the mean ROC curve for each model presented in Table 6.

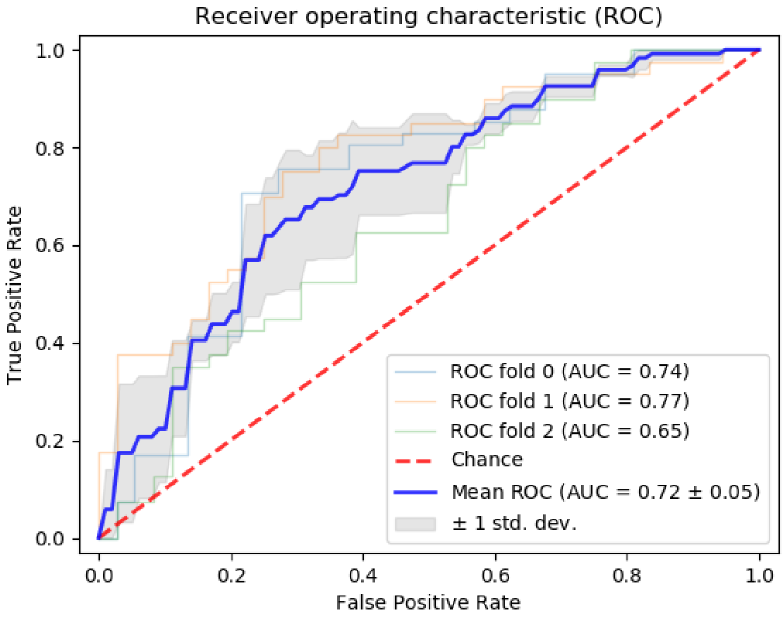


Figure 7 - Average ROC Curve Model 1

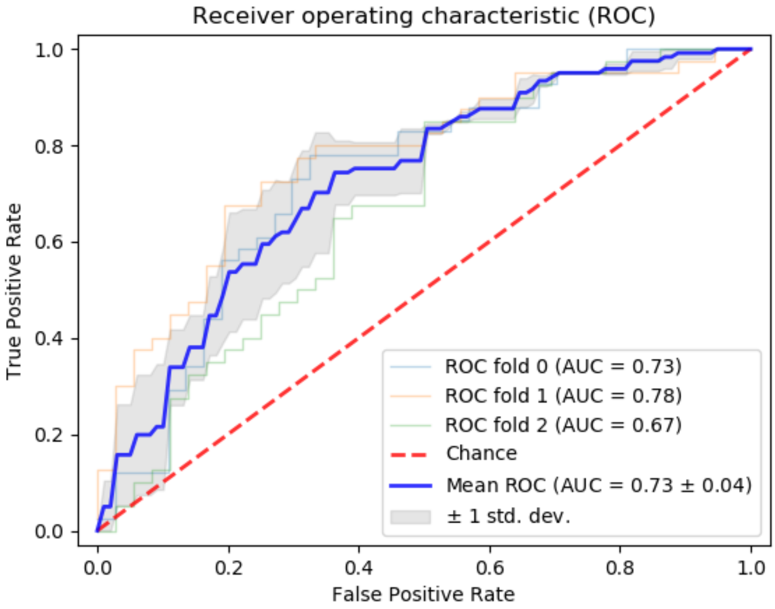


Figure 8 - Average ROC Curve Model 2

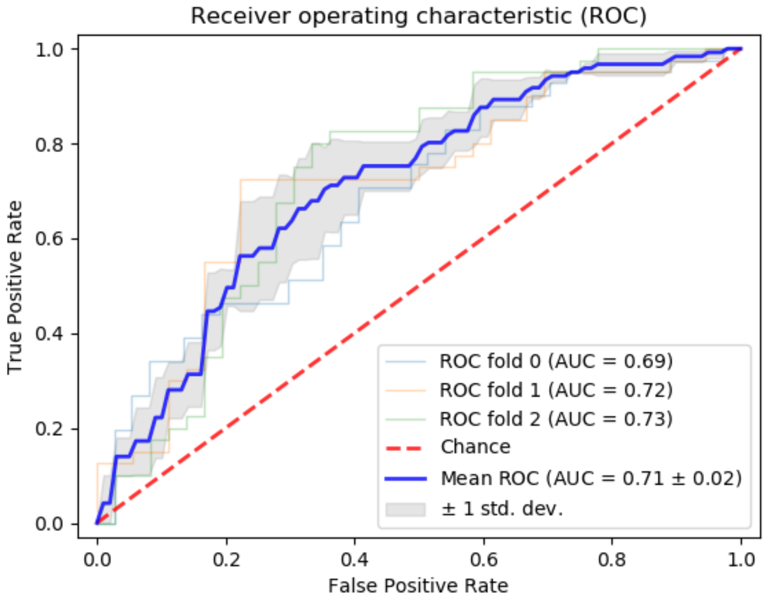


Figure 9 - Average ROC Curve Model 3

ROC curve is created by plotting the True Positive Rate against the False Positive Rate at various values of the threshold. f1 scores presented above are related to a threshold of 0.5. ROC curve can be considered as the average value of the sensitivity (recall) for a test over all possible values of specificity (precision) or vice versa. It summarizes the confusion matrices produced for each threshold without having to, actually, calculate them. Sometimes, for some problems, it can be more important to better classify one class than the other. In that case, it could make sense to optimize the true positive rate without exceeding a specific false positive rate. However, in this case study, the importance of correctly classify a true positive opinion in the right class is exactly the same of correctly classify a true negative opinion in the right class. Therefore, AUC measure was considered.

Regarding to the analysis of the 3 models:

- Although the Model 3 has higher accuracy and f1 score, this is the one of the 3 models that presents the worse average AUC score (0.71), but also the more consistent one in terms of AUC standard deviation (0.02).

- Model 2 is the one that presents better average AUC value (0.73) with average AUC standard deviation of 0.04.

- Model 1 has worse accuracy and f1 score than the other 2 models but contains better AUC value than the model 3.

The 3 models are similar, the 3 of them have balanced values for precision, recall and f1 measure once the data is not so unbalanced like that. Despite of the model 3 having an accuracy of 70.5% in a specific threshold, this model is the worse when considering all the thresholds. Thus, Model 2 was considered the best one with the highest AUC value (0.73) and an accuracy of 69.6%.

This model is working with 537 features in average, selected according to the mutual information method and the smoothing parameter is 0.05 (Table 4).

Without smoothing parameter, when the unseen document contains a word that does not appear in positive training documents for instance, then the probability P(w1|positive), where w1 is the word that doesn't appear, is 0 and, consequently, P(positive|w) = 0. This means that if a word that appears in an unseen text does not appear in any positive document, then the whole probability of the unseen document being labeled as positive is zero. Smoothing parameter (or Laplace smoothing or Lidstone smoothing) consists in the addition of alpha in the numerator and in denominator: p(x1|positive) = (count ("w1", positive) + alpha) / sum (count (w, positive) + alpha). Therefore, if the count of the word w1 in positive training documents is 0, the numerator will be alpha **(Daniel Jurafsky and James H. Martin, pp 67-68, 2018)**. A lot of runs were performed for different smoothing parameters (different alpha values). The best one was achieved with a low alpha defined as 0.05.

According to literature review, Naïve Bayes tends to work extremely well on very small datasets or short documents. Actually, the dataset used is quite small (230 instances) and Naïve Bayes is the champion model. SVM have shown to be highly effective on traditional text categorization, generally outperforming Naïve Bayes. However, the best model using SVM algorithm was achieved based of TF-IDF scores and Mutual Information technique using the best 50% of the features, and the average accuracy was 68.3%, with f1 measure equals to 68.3% as well. Previous works also indicate that ANN produces superior results than SVM except for some unbalanced data and, actually, one of the 3 top models uses Neural Network algorithm and any of them uses SVM.

6. CONCLUSIONS

In order to apply SA techniques to artificial intelligence opinions, and to understand what are the best techniques for this specific binary problem based on a small dataset containing short documents (opinions), experiences were performed with different feature selection techniques – Mutual Information, Chi2, F-test, LSI – and with different machine learning algorithms - SGD, SVM, LR, RF, ANN, Multinomial NB and Bernoulli NB – all of them based on both Term Frequency and on TF-IDF scores. The best combination that came up was a model based on TF-IDF scores, using Mutual Information and Multinomial Naïve Bayes, achieving an accuracy of approximately 70% and an AUC of approximately 73%.

Considering a small dataset with short reviews in general, Multinomial Naïve Bayes algorithm outperforms all the other ones that were tested. Neural networks algorithms tend to perform very well in these conditions as well. In this case study, SVM doesn't outperform Naïve Bayes (in previous works, SVM proved to generally outperform NB in traditional text categorization problems).

Regarding feature selection techniques, Mutual Information outperforms the other techniques (comparisons based on the average values over all the runs). Chi2 and LSA are the ones that perform poorly when combined and used with the aforementioned conditions.

Data preprocessing techniques were also explored in order to get normalized and cleaned text. Removal of rare words, the use of bi-grams, the use of lemmatization - instead of stemming - considering the part-of-speech of the words and a customized stop words list proved to be crucial aspects for the achievement of such good results.

Regarding the importance of the words in a document, TF-IDF worked much better than Term-Frequency of the words.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

Throughout the development of this work some difficulties were encountered. Most of them are related to data gathering. In fact, artificial intelligence has had a lot of discussion in turn of it, but it was not easy to collect a considerable amount of text opinions about it. The small amount of data was a restriction so that the model could learn better to classify, thus achieving better results. However, this limitation did not prevent the proposed ML method to achieve satisfactory results.

Bearing in mind that the collected data already includes various aspects inside artificial intelligence (such as self-drive cars, the possible lake of jobs, and much more) and just includes 230 opinions, sentiment analysis based on aspect-level was not possible. Restrict the data just to a specific aspect inside AI would reduce the dataset a lot.

For future works, I would recommend gathering the data through the Twitter and Facebook APIs in order to get a bigger amount of opinions and, thus, to let the model learn better the features for each class.

I would also recommend to consider a third class - a neutral class – or, in case of opting to keep a binary classification problem, just ignore the opinions that seem to be dubious in terms of positive or negative class. This was not done just to keep the maximum amount of data instances and don’t reduce even more the number of instances.

According to the classifier proposed by Kang and Yoo **(kang et al., 2011)** (an improved NB classifier to solve the problem of the tendency for the positive classification accuracy to appear up to approximately 10% higher than the negative classification), I would suggest to try this improved algorithm for classification performance possible improvement.

Finally, I would recommend working on previously identified limitations, either to erase them or to reduce its impact.

8. BIBLIOGRAPHY

Ravi, K., Ravi, V. (2015). A survey on opinion mining and sentiment analysis: Tasks, approaches and applications. *Knowledge-Based Systems, 89,* 14-46.

Medhat, W., Hassan, A., Korashy, H. (2014). Sentiment analysis algorithms and applications: a survey. *Ain Shams Engineering Journal.* Retrieved from http://dx.doi.org/10.1016/j.asej.2014.04.011.

Hanhoon, K., Joon, Y.S., Dongil, H. (2012*).* Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*.

Pang, B., Lee, L., Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. *Conference on Empirical Methods in Natural Language Processing: Association for Computational Linguistics*, *10*, 79–86.

Ng, A. Y., Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems, 14th International Conference*, 841–848.

Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. *Association for Computational Linguistics, 50th Annual Meeting of the Association for Computational Linguistics vol.2 short papers*, 90–94.

Boiy, E., Moens, M.F. (2009). A machine learning approach to sentiment analysis in multilingual web texts. *Information Retrieval, 12* (5)*,* 526–558.

Scharl, A., Weichselbraun, A. (2008). An automated approach to investigating the online media coverage of US presidential elections. *Journal of Information Technology and Politics, 5* (1), 21–132.

Maynard, D., Funk, A. (2011). Automatic detection of political opinions in tweets. *Proceedings of the 8th international conference on the semantic web, ESWC'11,* 88–99.

Yulan, H., Deyu, Z. (2011). Self-training from labeled features for sentiment analysis*. Information Processing and Management, 47* (4), 606–616. Retrieved from https://research.aston.ac.uk/portal/files/3337561/Self_training_from_labeled_features_for_sentiment_analysis_2011.pdf.

Miller, G.A. (1995). WordNet: a lexical database for English. *Communications of the ACM, 38* (11) ,39–41.

Tufi, D., Stefenescu, D. (2012). Experiments with a differential semantics annotation for WordNet 3.0, *Decis. Support Syst.*, *53*, 695–703.

Esuli, A., Sebastiani, F. (2006). SENTIWORDNET: a publicly available lexical resource for opinion mining. *Proceedings of the 5th Conference on Language Resources and Evaluation LREC-06 Genoa - Italy,* 417–422.

Baccianella, S., Esuli, A., Sebastiani, F. (2010). SENTIWORDNET 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. *Proceedings of LREC- 10 Malta*, 2200–2204.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Proceedings of annual meeting of the Association for Computational Linguistics (ACL'02)*.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*, 391-407.

Indurkhya, N., Damerau, F. J. (2010). Handbook of Natural Language Processing: Second Edition, *Chapman & Hall*.

Manning, C. D., Schutze, H. (1999). Foundations of statistical natural language processing. *MIT Press, 999.*

Heydari, A., Tavakoli, M. A., Salim, N., Heydari, Z. (2015). Detection of review spam: a survey. *Expert Syst. Appl., 42* (7), 3634–3642.

Manning, C. D., Raghavan, P., Schutze, H. (2008). Introduction to Information Retrieval. *Cambridge University Press*. Retrieved from http://books.google.pt/books?id=t1PoSh4uwVcC

Jackson, P., Moulinier, I. (2002). Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization. *John Benjamins Pub*. Retrieved from http://books.google.pt/books?id=jkkoj7U5g4kC

Aggarwal, C. C., Zhai, C. (2012). Mining Text Data. *Springer New York Dordrecht Heidelberg London: Springer Science + Business Media, LLC'12*.

Jolliffee, I. T. (2002). Principal component analysis. *Springer*.

Jurafsky, D., Martin, J. H. (2018). Speech and Language Processing. *Stanford University and University of Colorado at Boulder.*

Castelli, M., Vanneschi, L., Largo, Á. R. (2019). Supervised Learning: Classification. *Guenther, R. and Steel, D. (eds.), Encyclopedia of Bioinformatics and Computational Biology, 1,* 342–349.

Meng, Y., Shen, J., Zhang, C., Han, J. (2018). Weakly-Supervised Neural Text Classification. *Department of Computer Science, University of at Illinois Urbana-Champaign IL- USA*.

HU, M. & LIU, B. (2004) Mining and Summarizing Customer Reviews. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - USA.*

Curran, J. R., Clark, S. (2003). Language independent NER using a maximum entropy tagger. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL, 4,* 164-167.

Das, S., Chen, M. (2001). Yahoo! for Amazon: Extracting market sentiment from stock message boards. *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*.

Whitelaw, C., Garg, N., Argamon, S. (2005). Using appraisal groups for sentiment analysis. *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM)*, 625–631.

Kim S, Hovy E. (2004). Determining the sentiment of opinions. *Proceedings of international conference on Computational Linguistics (COLING'04)*.

Hatzivassiloglou, V., McKeown, K. (1997). Predicting the semantic orientation of adjectives. *Proceedings of annual meeting of the Association for Computational Linguistics (ACL'97)*.

Cortes, C., Vapnik, V. N. (1995). Support vector networks. *Machine Learning, 20*, 273–297.

Hofmann, T., Schölkopf, B., Smola, A. J. (2008). Kernel methods in machine learning. *Ann. Stat., 36*, 1171-1220.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proc. of the European Conference on Machine Learning (ECML)*,137–142.

Haykin, S., Network, N. (2004). A comprehensive foundation. *Neural Networks*, *2*, 41.

Hecht-Nielsen, R., et al. (1988). Theory of the backpropagation neural network. *Neural Networks, 1 (Suppl 1)*, 445–448.

Ruiz, M., Srinivasan, P. (1999). Hierarchical neural networks for text categorization. *Presented at the ACM SIGIR conference.*

Tou, N. H., Wei, G., Kok, L. (1997). Feature selection, perceptron learning, and a usability case study for text categorization. *Presented at the ACM SIGIR conference*.

Moraes, R., Valiati, J. F., Gavião N. W. P. (2013). Document-level sentiment classification: an empirical comparison between SVM and ANN. *Expert Syst Appl, 40*, 621–633.

Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (2013). Machine learning: An Artificial Intelligence Approach. *Springer Science & Business Media*.

Quinlan, J.R. (1993). C4.5: Programs for Machine Learning. *San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.*

Ko, Y., Seo, J. (2000). Automatic text categorization by unsupervised learning. *Proceedings of COLING-00, the 18th international conference on computational linguistics*.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27* (8), 861–874.

9. ANNEXES

Table 10 – Results using Chi2 feature selection technique with different percentiles after to find the best combination of parameters, for some of the algorithms.

	Count Vectorize						**TF_IDF Vectorize**				
	Bernoulli_NB, Alpha = 0.01, Binarize = None, Fit_Prior =True, Class_Prior= None										
	Avg_Accuracy	**Avg_Precision**	**Avg_Recall**	**Avg_F1**	**St_Dv Accuracy**		**Avg_Accuracy**	**Avg_Precision**	**Avg_Recall**	**Avg_F1**	**St_Dv Accuracy**
percentile: 9	0,635	0,635	0,635	0,633	0,027		0,604	0,609	0,604	0,591	0,013
percentile: 15	0,618	0,618	0,618	0,616	0,015		0,626	0,629	0,626	0,620	0,009
percentile: 30	0,604	0,604	0,604	0,602	0,024		0,622	0,623	0,622	0,618	0,026
percentile: 35	0,591	0,591	0,591	0,588	0,026		0,635	0,636	0,635	0,632	0,031
percentile: 40	0,626	0,629	0,626	0,624	0,044		0,617	0,618	0,617	0,614	0,037
percentile: 45	0,630	0,633	0,630	0,628	0,032		0,622	0,622	0,622	0,620	0,026
percentile: 50	0,608	0,609	0,608	0,607	0,049		0,595	0,595	0,595	0,593	0,028
percentile: 55	0,621	0,623	0,621	0,620	0,048		0,604	0,604	0,604	0,602	0,033
percentile: 60	0,604	0,605	0,604	0,602	0,043		0,617	0,617	0,617	0,614	0,011
percentile: 65	0,626	0,627	0,626	0,625	0,038		0,630	0,632	0,630	0,628	0,022
percentile: 70	0,608	0,610	0,608	0,607	0,044		0,626	0,626	0,626	0,624	0,007
percentile: 75	0,621	0,623	0,621	0,620	0,064		0,639	0,640	0,639	0,636	0,018
percentile: 80	0,621	0,623	0,621	0,621	0,057		0,635	0,636	0,635	0,632	0,026
percentile: 85	0,617	0,619	0,617	0,616	0,063		0,635	0,636	0,635	0,631	0,026
percentile: 90	0,621	0,623	0,621	0,621	0,057		0,635	0,636	0,635	0,631	0,026
Multinomial_NB, Alpha = 0.05, Fit_Prior =True, Class_Prior= None											
percentile: 10	0,643	0,644	0,643	0,640	0,043		0,630	0,632	0,630	0,623	0,044
percentile: 15	0,626	0,627	0,626	0,622	0,007		0,635	0,638	0,635	0,630	0,005
percentile: 25	0,609	0,609	0,609	0,608	0,040		0,608	0,609	0,608	0,603	0,044
percentile: 35	0,613	0,613	0,613	0,611	0,019		0,622	0,624	0,622	0,617	0,014
percentile: 40	0,613	0,613	0,613	0,610	0,025		0,613	0,613	0,613	0,609	0,025
percentile: 45	0,635	0,636	0,635	0,630	0,031		0,621	0,623	0,621	0,617	0,028
percentile: 50	0,626	0,627	0,626	0,621	0,047		0,626	0,629	0,626	0,620	0,038
percentile: 55	0,634	0,636	0,634	0,630	0,062		0,630	0,630	0,630	0,627	0,045
percentile: 60	0,630	0,631	0,630	0,627	0,043		0,639	0,641	0,639	0,633	0,035
percentile: 65	0,630	0,629	0,630	0,627	0,070		0,643	0,645	0,643	0,640	0,046
percentile: 70	0,626	0,626	0,626	0,623	0,059		0,639	0,642	0,639	0,634	0,041
percentile: 75	0,626	0,625	0,626	0,624	0,058		0,639	0,642	0,639	0,634	0,053
percentile: 80	0,634	0,634	0,634	0,632	0,062		0,656	0,659	0,656	0,653	0,033
percentile: 85	0,634	0,634	0,634	0,632	0,050		0,652	0,655	0,652	0,647	0,044
SGD, loss: modified_huber, Penalty = 'l1', tol=0.001, alpha = 0.0025, max_iter = 7500											
percentile: 10	0,596	0,602	0,596	0,595	0,073		0,596	0,652	0,596	0,559	0,065
percentile: 20	0,566	0,567	0,566	0,565	0,046		0,583	0,598	0,583	0,567	0,079
percentile: 30	0,600	0,601	0,600	0,600	0,034		0,635	0,639	0,635	0,632	0,009
percentile: 45	0,595	0,594	0,595	0,593	0,060		0,635	0,639	0,635	0,631	0,051
percentile: 50	0,583	0,584	0,583	0,582	0,040		0,657	0,661	0,657	0,655	0,047
percentile: 55	0,579	0,588	0,579	0,574	0,040		0,653	0,654	0,653	0,651	0,055
percentile: 60	0,591	0,594	0,591	0,590	0,014		0,631	0,633	0,631	0,630	0,074
percentile: 65	0,570	0,570	0,570	0,567	0,027		0,653	0,660	0,653	0,651	0,080
percentile: 70	0,565	0,570	0,565	0,563	0,056		0,631	0,641	0,631	0,627	0,093
percentile: 75	0,600	0,601	0,600	0,599	0,050			0,674	0,670	0,669	0,058
percentile: 80	0,600	0,601	0,600	0,599	0,076		0,666	0,671	0,666	0,663	0,051
percentile: 85	0,626	0,625	0,626	0,625	0,019		0,648	0,654	0,648	0,646	0,066
SVM, C=3, Kernel= linear, tol= 0.001, max_iter=95											
percentile: 85	0,605	0,607	0,605	0,604	0,080		0,636	0,638	0,636	0,635	0,097
percentile: 75	0,605	0,609	0,605	0,603	0,069		0,644	0,647	0,644	0,641	0,025
percentile: 65	0,609	0,612	0,609	0,608	0,059		0,635	0,634	0,635	0,634	0,017
percentile: 60	0,592	0,595	0,592	0,590	0,058		0,635	0,634	0,635	0,634	0,014
percentile: 55	0,579	0,584	0,579	0,576	0,082		0,626	0,626	0,626	0,626	0,038
percentile: 50	0,578	0,583	0,578	0,577	0,013		0,635	0,635	0,635	0,634	0,021
percentile: 40	0,583	0,587	0,583	0,582	0,041		0,626	0,625	0,626	0,624	0,029
percentile: 30	0,601	0,602	0,601	0,600	0,093		0,604	0,604	0,604	0,603	0,048
NN, percentile: 55, 'relu', lbfgs, alpha = 1.05, max_iter = 150, early_stop = False, tol = 0.005											
(59,59	0,583	0,589	0,583	0,582	0,075		0,635	0,637	0,635	0,631	0,021
(59,50	0,592	0,597	0,592	0,591	0,070		0,621	0,621	0,621	0,619	0,028
(70,67	0,592	0,598	0,592	0,589	0,116		0,609	0,609	0,609	0,606	0,018

Table 11 - Results using F-test feature selection technique with different percentiles after to find the best combination of parameters, for some of the algorithms.

	Count Vectorize						TF_IDF Vectorize				
	Bernoulli_NB, Alpha = 0.01, Binarize = None, Fit_Prior =True, Class_Prior= None										
	Avg_Accuracy	Avg_Precision	Avg_Recall	Avg_F1	St_Dv Accuracy		Avg_Accuracy	Avg_Precision	Avg_Recall	Avg_F1	St_Dv Accuracy
percentile: 9	0,596	0,602	0,596	0,594	0,015		0,665	0,670	0,665	0,659	0,012
percentile: 15	0,596	0,597	0,596	0,592	0,021		0,617	0,617	0,617	0,612	0,050
percentile: 30	0,639	0,639	0,639	0,638	0,028		0,604	0,602	0,604	0,600	0,067
percentile: 35	0,635	0,638	0,635	0,633	0,018		0,600	0,599	0,600	0,594	0,044
percentile: 40	0,639	0,643	0,639	0,637	0,013		0,617	0,617	0,617	0,613	0,050
percentile: 45	0,621	0,623	0,621	0,620	0,040		0,617	0,618	0,617	0,616	0,024
percentile: 50	0,630	0,632	0,630	0,629	0,044		0,604	0,604	0,604	0,602	0,033
percentile: 55	0,626	0,627	0,626	0,625	0,052		0,604	0,604	0,604	0,602	0,025
percentile: 60	0,608	0,609	0,608	0,607	0,052		0,626	0,626	0,626	0,623	0,007
percentile: 65	0,608	0,610	0,608	0,608	0,040		0,626	0,627	0,626	0,623	0,013
percentile: 70	0,617	0,619	0,617	0,616	0,050		0,630	0,631	0,630	0,628	0,020
percentile: 75	0,621	0,623	0,621	0,620	0,064		0,639	0,640	0,639	0,636	0,018
percentile: 80	0,621	0,623	0,621	0,621	0,057		0,635	0,636	0,635	0,631	0,026
percentile: 85	0,617	0,619	0,617	0,616	0,063		0,635	0,636	0,635	0,631	0,026
percentile: 90	0,621	0,623	0,621	0,621	0,057		0,635	0,636	0,635	0,631	0,026
percentile: 8	0,635	0,644	0,635	0,631	0,060		0,661	0,665	0,661	0,655	0,026
Multinomial_NB, Alpha = 0.05, Fit_Prior =True, Class_Prior= None											
percentile: 10	0,630	0,631	0,630	0,625	0,033		0,652	0,655	0,652	0,647	0,007
percentile: 15	0,609	0,609	0,609	0,604	0,008		0,613	0,612	0,613	0,607	0,044
percentile: 20	0,635	0,634	0,635	0,633	0,048		0,626	0,627	0,626	0,622	0,031
percentile: 35	0,643	0,644	0,643	0,642	0,024		0,595	0,594	0,595	0,589	0,044
percentile: 40	0,643	0,644	0,643	0,640	0,024		0,617	0,618	0,617	0,612	0,033
percentile: 45	0,643	0,644	0,643	0,640	0,024		0,617	0,618	0,617	0,612	0,033
percentile: 55	0,643	0,644	0,643	0,639	0,054		0,634	0,636	0,634	0,630	0,040
percentile: 65	0,625	0,625	0,625	0,623	0,069		0,648	0,649	0,648	0,645	0,031
percentile: 70	0,621	0,621	0,621	0,619	0,062		0,643	0,646	0,643	0,639	0,050
percentile:75	0,626	0,625	0,626	0,624	0,058		0,648	0,651	0,648	0,643	0,038
percentile: 80	0,634	0,634	0,634	0,632	0,062		0,656	0,659	0,656	0,653	0,033
percentile: 85	0,630	0,630	0,630	0,628	0,054		0,639	0,640	0,639	0,635	0,038
SGD, loss: modified_huber, Penalty = 'l1', tol=0.001, alpha = 0.0025, max_iter = 7500											
percentile: 10	0,570	0,580	0,570	0,567	0,062		0,579	0,612	0,579	0,552	0,080
percentile: 20	0,596	0,597	0,596	0,596	0,061		0,618	0,623	0,618	0,610	0,048
percentile: 30	0,574	0,577	0,574	0,574	0,066		0,613	0,613	0,613	0,612	0,034
percentile: 40	0,596	0,596	0,596	0,593	0,031		0,665	0,670	0,665	0,663	0,029
percentile: 42	0,591	0,594	0,591	0,588	0,025		0,670	0,675	0,670	0,667	0,037
percentile: 43	0,600	0,601	0,600	0,599	0,053		0,644	0,649	0,644	0,640	0,036
percentile: 45	0,548	0,551	0,548	0,548	0,032		0,644	0,651	0,644	0,640	0,041
percentile: 50	0,587	0,588	0,587	0,587	0,028		0,631	0,635	0,631	0,628	0,046
percentile: 55	0,604	0,606	0,604	0,604	0,026		0,666	0,670	0,666	0,664	0,051
percentile: 57	0,583	0,583	0,583	0,582	0,027		0,666	0,675	0,666	0,663	0,080
percentile: 60	0,579	0,584	0,579	0,575	0,048		0,631	0,633	0,631	0,630	0,080
percentile: 65	0,574	0,581	0,574	0,569	0,007		0,640	0,648	0,640	0,637	0,107
percentile: 70	0,565	0,567	0,565	0,564	0,069		0,640	0,643	0,640	0,637	0,054
percentile: 75	0,578	0,578	0,578	0,578	0,048		0,640	0,645	0,640	0,638	0,096
percentile: 80	0,613	0,614	0,613	0,612	0,067		0,666	0,668	0,666	0,664	0,067
percentile: 85	0,635	0,634	0,635	0,634	0,005		0,648	0,657	0,648	0,645	0,078
percentile: 90	0,605	0,605	0,605	0,604	0,027		0,657	0,666	0,657	0,653	0,069
SVM, C=3, Kernel= linear, tol= 0.001, max_iter=95											
percentile: 10	0,579	0,594	0,579	0,573	0,068		0,635	0,635	0,635	0,634	0,031
percentile: 20	0,609	0,612	0,609	0,604	0,067		0,605	0,606	0,605	0,601	0,046
percentile: 30	0,592	0,595	0,592	0,591	0,080		0,635	0,634	0,635	0,631	0,088
percentile: 40	0,583	0,591	0,583	0,580	0,054		0,635	0,637	0,635	0,632	0,033
percentile: 50	0,587	0,591	0,587	0,586	0,036		0,609	0,609	0,609	0,608	0,018
percentile: 60	0,605	0,609	0,605	0,603	0,058		0,609	0,608	0,609	0,608	0,008
percentile: 70	0,609	0,615	0,609	0,608	0,020		0,622	0,623	0,622	0,620	0,029
percentile: 80	0,600	0,606	0,600	0,598	0,073		0,635	0,637	0,635	0,634	0,031
percentile: 90	0,583	0,587	0,583	0,581	0,068		0,639	0,642	0,639	0,638	0,051
NN, percentile: 55, 'relu', lbfgs, alpha = 1.05, max_iter = 150, early_stop = False, tol = 0.005											
(59,59	0,592	0,597	0,592	0,590	0,046		0,635	0,635	0,635	0,632	0,005
(59,50	0,583	0,589	0,583	0,581	0,041		0,592	0,509	0,592	0,536	0,058
(70,67	0,583	0,590	0,583	0,581	0,028		0,630	0,632	0,630	0,628	0,002

Table 12 - Results using Mutual Information feature selection technique with different percentiles after to find the best combination of parameters, for some of the algorithms (yellow values represent good models).

	Count Vectorize						TF_IDF Vectorize				
	Bernoulli_NB, Alpha = 0.01, Binarize = None, Fit_Prior =True, Class_Prior= None										
	Avg_Accuracy	Avg_Precision	Avg_Recall	Avg_F1	St_Dv Accuracy		Avg_Accuracy	Avg_Precision	Avg_Recall	Avg_F1	St_Dv Accuracy
percentile: 9	0,613	0,615	0,613	0,611	0,053		0,591	0,604	0,591	0,573	0,040
percentile: 15	0,609	0,613	0,609	0,606	0,051		0,605	0,607	0,605	0,601	0,108
percentile: 20	0,635	0,636	0,635	0,634	0,009		0,579	0,581	0,579	0,572	0,035
percentile: 25	0,644	0,643	0,644	0,643	0,036		0,634	0,635	0,634	0,631	0,044
percentile: 30	0,635	0,635	0,635	0,634	0,031		0,656	0,657	0,656	0,654	0,024
percentile: 35	0,622	0,621	0,622	0,621	0,021		0,660	0,662	0,660	0,655	0,056
percentile: 40	0,648	0,648	0,648	0,647	0,021		0,634	0,634	0,634	0,629	0,062
percentile: 45	0,648	0,651	0,648	0,647	0,005		0,639	0,639	0,639	0,633	0,063
percentile: 50	0,639	0,643	0,639	0,637	0,025		0,643	0,644	0,643	0,636	0,063
percentile: 55	0,617	0,619	0,617	0,616	0,037		0,660	0,661	0,660	0,657	0,073
percentile: 60	0,613	0,614	0,613	0,612	0,044		0,674	0,674	0,674	0,672	0,027
percentile: 65	0,613	0,615	0,613	0,612	0,053		0,669	0,670	0,669	0,668	0,022
percentile: 56	0,630	0,633	0,630	0,629	0,011		0,665	0,665	0,665	0,661	0,075
percentile: 57	0,609	0,610	0,609	0,607	0,037		0,665	0,665	0,665	0,661	0,065
percentile: 58	0,617	0,619	0,617	0,616	0,033		0,691	0,691	0,691	0,687	0,075
percentile: 59	0,621	0,623	0,621	0,620	0,048		0,687	0,687	0,687	0,684	0,048
percentile: 63	0,617	0,619	0,617	0,616	0,037		0,669	0,670	0,669	0,668	0,022
percentile: 66	0,630	0,633	0,630	0,628	0,046		0,669	0,670	0,669	0,668	0,022
percentile: 69	0,621	0,623	0,621	0,621	0,044		0,669	0,670	0,669	0,668	0,022
percentile: 72	0,621	0,623	0,621	0,621	0,057		0,674	0,675	0,674	0,672	0,027
percentile: 75	0,621	0,623	0,621	0,621	0,057		0,670	0,671	0,670	0,668	0,026
percentile: 78	0,617	0,619	0,617	0,616	0,063		0,670	0,670	0,670	0,668	0,026
percentile: 81	0,621	0,623	0,621	0,621	0,057		0,674	0,674	0,674	0,672	0,027
	Multinomial_NB, Alpha = 0.05, Fit_Prior =True, Class_Prior= None										
percentile: 58	0,630	0,630	0,630	0,627	0,054		0,691	0,695	0,691	0,686	0,052
percentile: 56	0,634	0,636	0,634	0,631	0,040		0,674	0,675	0,674	0,668	0,071
percentile: 60	0,639	0,639	0,639	0,636	0,047		0,665	0,667	0,665	0,661	0,016
percentile: 59	0,634	0,634	0,634	0,632	0,052		0,696	0,699	0,696	0,691	0,034
	SGD, loss: modified_huber, Penalty = 'l1', tol=0.001, alpha = 0.0025, max_iter = 7500										
percentile: 58	0,596	0,597	0,596	0,596	0,043		0,644	0,652	0,644	0,640	0,105
percentile: 30	0,639	0,643	0,639	0,638	0,066		0,631	0,635	0,631	0,627	0,067
percentile: 35	0,640	0,644	0,640	0,639	0,088		0,600	0,619	0,600	0,590	0,063
percentile: 40	0,639	0,644	0,639	0,638	0,063		0,605	0,609	0,605	0,603	0,099
percentile: 45	0,639	0,643	0,639	0,638	0,080		0,622	0,623	0,622	0,621	0,060
percentile: 50	0,631	0,633	0,631	0,629	0,070		0,644	0,651	0,644	0,642	0,103
percentile: 55	0,630	0,634	0,630	0,629	0,013		0,640	0,648	0,640	0,637	0,096
percentile: 56	0,605	0,607	0,605	0,604	0,066		0,626	0,643	0,626	0,618	0,053
percentile: 57	0,583	0,582	0,583	0,583	0,021		0,635	0,641	0,635	0,632	0,100
percentile: 59	0,618	0,618	0,618	0,618	0,041		0,640	0,645	0,640	0,638	0,084
percentile: 60	0,587	0,591	0,587	0,586	0,040		0,653	0,657	0,653	0,652	0,108
percentile: 65	0,595	0,595	0,595	0,595	0,037		0,653	0,656	0,653	0,652	0,073
percentile: 67	0,583	0,586	0,583	0,582	0,047		0,666	0,668	0,666	0,665	0,055
percentile: 69	0,561	0,563	0,561	0,560	0,050		0,692	0,697	0,692	0,690	0,051
percentile: 68	0,574	0,577	0,574	0,573	0,062		0,687	0,691	0,687	0,686	0,044
percentile: 70	0,583	0,586	0,583	0,582	0,047		0,679	0,689	0,679	0,675	0,055
percentile: 74	0,591	0,594	0,591	0,590	0,034		0,687	0,696	0,687	0,685	0,066
percentile: 72	0,587	0,590	0,587	0,587	0,054		0,683	0,694	0,683	0,679	0,049
percentile: 75	0,578	0,580	0,578	0,578	0,060		0,683	0,693	0,683	0,680	0,062
percentile: 77	0,591	0,593	0,591	0,590	0,069		0,688	0,692	0,688	0,686	0,076
percentile: 79	0,596	0,596	0,596	0,595	0,033		0,670	0,673	0,670	0,668	0,059
	SVM, C=3, Kernel= linear, tol= 0.001, max_iter=95										
percentile: 40	0,548	0,551	0,548	0,548	0,046		0,666	0,667	0,666	0,665	0,073
percentile: 45	0,565	0,571	0,565	0,564	0,035		0,653	0,655	0,653	0,651	0,085
percentile: 50	0,613	0,619	0,613	0,612	0,053		0,683	0,684	0,683	0,683	0,036
percentile: 52	0,570	0,577	0,570	0,567	0,047		0,674	0,675	0,674	0,674	0,048
percentile: 57	0,591	0,596	0,591	0,590	0,001		0,661	0,662	0,661	0,660	0,029
percentile: 60	0,596	0,599	0,596	0,595	0,088		0,678	0,680	0,678	0,677	0,017
percentile: 62	0,605	0,612	0,605	0,601	0,080		0,670	0,671	0,670	0,669	0,037
percentile: 65	0,605	0,610	0,605	0,603	0,058		0,662	0,664	0,662	0,660	0,076
percentile: 70	0,609	0,614	0,609	0,608	0,055		0,640	0,640	0,640	0,638	0,054
percentile: 75	0,600	0,606	0,600	0,598	0,073		0,636	0,638	0,636	0,633	0,085
percentile: 80	0,605	0,609	0,605	0,604	0,074		0,648	0,649	0,648	0,647	0,063
percentile: 85	0,596	0,600	0,596	0,593	0,081		0,657	0,660	0,657	0,655	0,081
	NN, percentile: 55, 'relu', lbfgs, alpha = 1.05, max_iter = 150, early_stop = False, tol = 0.005										
(59,59), tol = 0.0001, alpha=1	0,622	0,630	0,622	0,620	0,041		0,688	0,690	0,688	0,686	0,063
(59,59), tol = 0.0001, alpha=1	0,618	0,626	0,618	0,616	0,048		0,688	0,690	0,688	0,686	0,063
(59,59), tol = 0.001, alpha=1	0,609	0,617	0,609	0,607	0,047		0,683	0,685	0,683	0,682	0,070
(59,59), tol = 0.003, alpha=1	0,609	0,614	0,609	0,608	0,047		0,692	0,694	0,692	0,691	0,077
(59,59), tol = 0.004, alpha=1	0,600	0,606	0,600	0,599	0,047		0,688	0,690	0,688	0,687	0,074
(59,59), tol = 0.005, alpha=1	0,613	0,619	0,613	0,611	0,028		0,688	0,690	0,688	0,687	0,074
(59,59), tol = 0.0001, alpha=1.05	0,622	0,630	0,622	0,618	0,021		0,696	0,699	0,696	0,695	0,059
(59,59), tol = 0.0001, alpha=1	0,622	0,630	0,622	0,618	0,021		0,696	0,699	0,696	0,695	0,059

(60,60	0,604	0,614	0,604	0,601	0,026		0,696	0,697	0,696	0,695	0,059
(65,65	0,617	0,623	0,617	0,616	0,022		0,622	0,538	0,622	0,567	0,092
(300,300	0,588	0,595	0,588	0,585	0,077		0,622	0,539	0,622	0,567	0,099
(59,57	0,618	0,621	0,618	0,617	0,015		0,679	0,680	0,679	0,678	0,061
(59,55	0,613	0,620	0,613	0,611	0,028		0,692	0,694	0,692	0,691	0,055
(59,53	0,600	0,606	0,600	0,599	0,028		0,692	0,694	0,692	0,691	0,055
(59,51	0,605	0,606	0,605	0,604	0,046		0,674	0,675	0,674	0,674	0,055
(59,50	0,631	0,642	0,631	0,628	0,070		0,701	0,702	0,701	0,700	0,063
(60,55	0,618	0,621	0,618	0,617	0,024		0,688	0,690	0,688	0,686	0,064
(60,57	0,592	0,595	0,592	0,589	0,058		0,696	0,698	0,696	0,695	0,049
(60,62	0,618	0,620	0,618	0,617	0,024		0,696	0,697	0,696	0,695	0,059
(70,70	0,614	0,614	0,614	0,613	0,054		0,662	0,664	0,662	0,660	0,062
(70,75	0,626	0,629	0,626	0,626	0,041		0,688	0,690	0,688	0,686	0,063
(70,73	0,644	0,650	0,644	0,641	0,028		0,670	0,672	0,670	0,668	0,049
(70,69	0,627	0,639	0,627	0,622	0,047		0,653	0,572	0,653	0,598	0,111
(70,67	0,596	0,601	0,596	0,594	0,055		0,705	0,706	0,705	0,704	0,055
(70,68	0,587	0,601	0,587	0,582	0,040		0,683	0,684	0,683	0,682	0,048
(70,72	0,600	0,604	0,600	0,599	0,031		0,675	0,676	0,675	0,674	0,085
(70,71	0,613	0,617	0,613	0,613	0,043		0,679	0,681	0,679	0,678	0,069
(90,85	0,631	0,635	0,631	0,630	0,049		0,688	0,690	0,688	0,687	0,074
(90,80	0,635	0,637	0,635	0,634	0,034		0,683	0,686	0,683	0,682	0,070
(90,83	0,605	0,610	0,605	0,603	0,053		0,670	0,672	0,670	0,669	0,071
(90,86	0,596	0,604	0,596	0,594	0,039		0,683	0,684	0,683	0,682	0,049
(90,87	0,635	0,644	0,635	0,633	0,039		0,666	0,667	0,666	0,665	0,069
(90,88	0,600	0,602	0,600	0,599	0,040		0,692	0,693	0,692	0,691	0,055
(90,89	0,635	0,638	0,635	0,634	0,034		0,613	0,530	0,613	0,559	0,086
(90,90	0,639	0,647	0,639	0,636	0,051		0,692	0,693	0,692	0,691	0,055
(90,91	0,604	0,610	0,604	0,603	0,046		0,688	0,689	0,688	0,687	0,074
(90,92	0,609	0,612	0,609	0,608	0,060		0,688	0,690	0,688	0,686	0,064
(90,93	0,613	0,621	0,613	0,611	0,043		0,687	0,690	0,687	0,686	0,052
(100,95	0,570	0,574	0,570	0,569	0,050		0,688	0,688	0,688	0,687	0,063
(100,98	0,578	0,581	0,578	0,578	0,012		0,670	0,671	0,670	0,669	0,049
(100,99	0,613	0,617	0,613	0,613	0,041		0,626	0,545	0,626	0,570	0,093
(100,100	0,617	0,625	0,617	0,615	0,024		0,683	0,686	0,683	0,682	0,070
(100,105	0,600	0,606	0,600	0,599	0,016		0,609	0,525	0,609	0,554	0,079
RandForest, Gini, max_depth=20, n_estimators=200, max_feat: auto, min_sample_split= 4, min_sample_leaf = 2, random_state =42											
percentile: 30	-	-	-	-	-		0,609	0,610	0,609	0,608	0,017
percentile: 35	-	-	-	-	-		0,570	0,569	0,570	0,568	0,027
percentile: 40	-	-	-	-	-		0,617	0,617	0,617	0,616	0,033
percentile: 45	-	-	-	-	-		0,600	0,600	0,600	0,597	0,042
percentile: 50	-	-	-	-	-		0,622	0,624	0,622	0,618	0,034
percentile: 55	-	-	-	-	-		0,574	0,573	0,574	0,573	0,021
percentile: 60	-	-	-	-	-		0,574	0,573	0,574	0,573	0,021
percentile: 65	-	-	-	-	-		0,587	0,586	0,587	0,585	0,019
Logistic Regression, l2, C = 1, max_iter = 90, tol = 0.2											
percentile: 30	-	-	-	-	-		0,622	0,634	0,622	0,605	0,006
percentile: 35	-	-	-	-	-		0,652	0,667	0,652	0,640	0,046
percentile: 40	-	-	-	-	-		0,635	0,652	0,635	0,618	0,027
percentile: 45	-	-	-	-	-		0,648	0,663	0,648	0,634	0,049
percentile: 50	-	-	-	-	-		0,644	0,657	0,644	0,630	0,013
percentile: 55	-	-	-	-	-		0,635	0,650	0,635	0,617	0,018
percentile: 60	-	-	-	-	-		0,652	0,670	0,652	0,637	0,016
percentile: 65	-	-	-	-	-		0,643	0,662	0,643	0,627	0,022

Table 13 – Results using LSI feature selection technique with different percentiles just for the model considered the best one in terms of accuracy and f1 measure.

LSA												**TF_IDF Vectorize**				
Model	**Parameters**											**Avg_Accuracy**	**Avg_Precision**	**Avg_Recall**	**Avg_F1**	**St_Dv Accuracy**
NN	**(70,67**	**relu**	**lbfgs**	**alpha = 1.05**	**max_iter = 150**	**early_stop = False**	**tol = 0.005**	**percentile: 55**	**N_Components: 2**		0,526	0,538	0,526	0,461	0,013
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 6**		0,609	0,614	0,609	0,605	0,100
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 10**		0,596	0,597	0,596	0,586	0,068
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 14**		0,604	0,613	0,604	0,596	0,045
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 17**		0,574	0,575	0,574	0,567	0,034
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 20**		0,613	0,639	0,613	0,596	0,050
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 23**		0,604	0,620	0,604	0,598	0,092
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 26**		0,591	0,608	0,591	0,581	0,079
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 30**		0,605	0,612	0,605	0,600	0,105
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 50**		0,604	0,608	0,604	0,600	0,057
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 70**		0,639	0,641	0,639	0,637	0,047
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 90**		0,613	0,613	0,613	0,613	0,016
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 100**		0,613	0,613	0,613	0,612	0,016
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 130**		0,635	0,638	0,635	0,633	0,054
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 150**		0,648	0,654	0,648	0,646	0,056
NN	(70,67	relu	lbfgs	alpha = 1.05	max_iter = 150	early_stop = False	tol = 0.005	percentile: 55	**N_Components: 151**		0,657	0,661	0,657	0,655	0,036