University of Padova

Sentiment Analysis on Online Automotive Forums

Supervisor
PROF. NICOLA FERRO

Master Candidate
GIUSEPPE RAVAGNANI

Abstract

Contents

viii

1

Introduction

2

State of the Art

In this chapter it will be presented the state of the art of the sentiment analysis. Since most common approaches are based on machine learning techniques, we will also introduce the basics of supervised learning and the most important algorithms used in this work. Finally, some datasets of sentiment analysis will be presented along with the state of the art of sentiment analysis algorithm.

2.1 Sentiment Analysis

From the introduction of online social media, people generate continuously vast quantities of contents such as texts, reviews, comments, videos, pictures. This enormous quantity of data offers the possibility to access and understand users' perspective about topics of interest, and this can be exploited, for instance, to make market predictions, business choices, predict outcomes of political elections. *Sentiment Analysis* is an active area of research motivated to improve the automated recognition of sentiment expressed in text [1].

From the whole scenic of online social media, Twitter* is the most used to achieve contents, due to a very active community and the simplicity to get data from

---

*Twitter (https://twitter.com/) is an online micro-blogging social network, where people interact with short messages, known as "tweets". It was created in March 2006 by Jack Dorsey, Noah Glass, Biz Stone, and Evan Williams and launched in July of that year.

Twitter's Application Programming Interface (API).

2.1.1 Twitter Sentiment Analysis

From recent statistics dated 2018, Twitter counts 326 million monthly active users, 100 million daily active users, which send 500 million tweets per day (80% from mobile). That large amount of users and messages motivates the fact that most of the researches are made using Twitter's data, so usually this area is referenced as Twitter Sentiment Analysis (TSA).

In addition to the plenty of data, Twitter becomes a case study because tweets often express users' opinions about a topic of interest. TSA has been applied to mine information from users, in order to explain and make predictions on different social phenomena, for instance in [2] researches studied the outcomes of political elections, in [3] stock market movements and in [4] product sales. Sentiment analysis has also been applied without involving Twitter's data, for instance in [5] the goal was to monitor the software deployment.

We noticed that sentiment analysis has a wide variety of applications, however even if the attention of researches and the growing of the literature, the performances of state of the art approaches are very poor, due to the difficulties on interpreting the natural language.

Sentiment analysis can be summarized as the automatic classification task of sentiment polarity expressed in text (for instance with classes positive, negative and neutral), identifying the stance about a topic, opinion holder identification, and sometimes considering various aspects of a topic. The sentiment polarity classification problem can be modeled in different ways, depending on the task. It can be considered a binary classification (positive/negative) or three-way classification (positive/negative/neutral), but sometimes a more fine-grained classification is needed, considering also slightly positive and slightly negative classes. Sentiment polarity classification may be also domain specific: for instance texts classified as positive express in favor of a certain brand, or a candidate of the elections, or optimism about social issues.

Approaches for sentiment analysis can be categorized in two main classes [1]:

- The first class contains methods that involves the use of a lexicon of opinion-related terms, with a scoring method to evaluate the sentiment. These

methods are used in unsupervised applications, but they achieve low performances, since they are unable to consider some important part of the sentence, for instance context, novel vocabulary, indicators for sentiment expressions;

- The second class is represented by methods based on feature representation of texts, that involves the use of machine learning approaches to derive relationships between feature values and sentiment using supervised learning. These methods require a large set of labeled training samples to train a model, but they have the limit that if they are thought for domain specific, they does not work on broader applications. In the next sections we will focus on these latter methods.

In general, the effectiveness of the information derived from TSA campaigns, are critically dependent on the approach to evaluate the sentiment expressed by users. Some approaches are directly taken from literature and they are thought for more established social media, like product reviews and web forums. However, several characteristics make hard to apply the same techniques on tweets. These characteristics include brevity of texts (less than 140 characters length), adoption of novel language, twitter specific communication elements, strong sentiment class imbalance (usually most of tweets are labeled as neutral), and stream based tweets generation. Considering these characteristics is crucial to achieve adequate performances from TSA models. For these reasons, some researches focus more on sentiment analysis on other social media, for instance on Facebook* status updates that can contain up to 5000 characters. Other social media have the characteristic of long texts, but strong imbalance among classes: for example online reviews are predominantly positive or negative, while web forums are mostly neutral. In this review we will focus more on TSA, but the same techniques can be applied (eventually with some differences) on other types of social media. In the next paragraph I'm going to discuss some techniques to address issues on tweets due to their brevity.

---

*Facebook (https://www.facebook.com) is a social network founded by Mark Zuckerberg aling with fellow Harvard College students and roommates Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. Facebook was launched on February 4 2004.

2.1.2 TECHNIQUES TO ADDRESS TWEETS' ISSUES

As previously said, tweets are text messages limited to 140 characters, and their brevity impact the performance of sentiment analysis techniques, since there are just few terms that capture information about user's sentiment. This fact implies also the very sparsity of feature vectors that represent tweets. The length limitation, motivates users to adopt novel form of expression, slang, acronyms and emoticons. Moreover, novel terms are continuously evolving, so some of them may not be considered in all TSA approaches.

One class of techniques addresses this issue propagating known sentiment information, coming from vocabulary of emoticons and lexicon of sentiment terms, through tweets to identify new expressions related to sentiment, looking for instance the co-occurrence or proximity of tuples of terms. Once propagated, the vocabulary is enlarged, so a bigger set of lexicon sentiment information may improve effectiveness of TSA methods [6].

Another approach is based on feature expansion, adding more related content or forming combination of the current one. In this task, some lexical resources like WordNet are useful to include words' synonyms, or add semantic concepts related to named entities [7].

Twitter has communication elements that are not present in other social media, and can contain information about sentiment polarity. These elements also increase the vocabulary and so they can complicate sentiment analysis. Some of these components are user mentions (@username) used for direct the tweet to the mentioned user, hashtags (#hashtag) used to connect the tweet to other that threat the same arguments. Especially the second ones sometimes contains information about sentiment, for instance hashtags like #sad or #bad may express negative feelings, or contrary #good or #wonderful may express positive ones. For these facts it is important to include these elements in the sentiment decision, paying attention that not always they are not entirely included in the body of the tweet.

One class of techniques involves the just discussed Twitter-specific communication elements. It consists on removing, grouping or correcting them in a preprocessing task. This lead to complexity reducing and consequently reducing the feature sparsity. Some approaches removes any occurrences of hashtags, while some oth-

6

ers replace them with common tokens, for instance *"USER_MENTION"*. In some other cases, syntactic exaggerations (like multiple exclamation marks) are simplified and misspellings are corrected. Also emoticons are handled for instance grouping positive emoticons (like *:)* or *:-)*) and the same for negative ones. Some other approaches include these elements in the feature vectors.

In the next paragraph we will discuss how sentiment class imbalance is handles.

2.1.3 TECHNIQUES TO ADDRESS SENTIMENT CLASS IMBALANCE

As previously said, tweets are predominantly neutral (while for instance product reviews are generally either positive or negative), so usually TSA faces class imbalance issues that can affect classification tasks. Large class imbalances are problematic especially for machine learning because they induce bias, so learned models have preferences on predicting neutral class (in the case of TSA).
One strategy to overcome these issues consists on expanding the training set utilized to train the machine-learning classifiers. In this way the set of comments will contain various form of expressions of sentiment expressed in tweets, so it may improve the classification performance.
Another approach consists on applying multiple classifiers in an ensemble. Since classifiers' performance may vary, the ensemble model the overall classification more stable. A similar approach consists on applying firstly a subjectivity classifier (responsible of select the subject), then a sentiment classifier (responsible of the sentiment classification).

In the next paragraph we will discuss how to handle temporal dependencies.

2.1.4 TECHNIQUES TO ADDRESS TEMPORAL DEPENDENCIES

One issue on performing TSA on streaming is that topic and language may change rapidly, and model may not be calibrated on needed ones. Tweets sometimes are also temporary dependent, since a user can express his opinion using more tweets. This characteristic of communication can be shared with other types of social media, for instance web forums. In this class, the temporal dependency can be

7

shown on quotes: usually happens that a user expressed his opinion on something someone on the conversation said. In these cases the sentiment of the comment is dependent by the referenced text, to it may be useful to take in account both the comment and the reference to decide the sentiment. The majority of sentiment classifiers threat tweets as independent instances, sometimes losing information about ordering or temporal references.

2.1.5 Common approaches

Sentiment analysis is a composition of diverse problems, in fact some steps are needed to perform opinion mining on given texts, since texts are coming from different resources and different formats. Data acquisition and preprocessing are mandatory sub-tasks required for opinion mining.

Data acquisition is an essential task for collecting the dataset, since online resources generally don't satisfy all requirements, for instance the focused domain, or the social media where data are extracted. For this task, big social media such as Twitter and Facebook share API for collecting public data, contrary for other online resources such as forums, *web scraping* [†] is a useful, but less comfortable technique.

Raw data collected with in the previous task must be preprocessed, since they are full of noise (useless parts of sentences that do not give any information). Some common steps are: tokenization, stop word removal, stemming, Parts Of Speech (POS) tagging, and feature extraction and representation [8]. Tokenization is used to break a sentence into words, phrases, symbols, or other meaningful tokens by removing punctuation marks. Stopwords are also dropped since they don't carry information so they don't contribute on analysis. Stemming is the process to transform a word into its root form by dropping the suffix. POS tagging is performed to recognize different parts of speech in the text. Due to the characteristic sparsity and noise of the data in natural language processing problems, feature selection is an important preprocessing step, critical for the final performance.

Sentiment classification, as said before, can be performed using machine learning

---

[†]Web scraping is a technique for extracting data from websites. Web scraping software access the websites through Hypertext Transfer Protocol, sometimes using automatic crawlers, then the downloaded page is parsed in order to extract the needed information.

approaches as well as lexicon based approaches. Machine learning approaches can be divided in supervised learning and unsupervised learning methods: for what concerns supervised ones, common algorithms are Decision Trees, Support Vector Machines (SVM), Neural Networks (NN), Naïve Bayes. Lexicon based approaches uses either dictionaries or corpus based approaches. Dictionary methods use existing resources collecting information about words along with their positive or negative sentiment strength. Corpus based approaches, instead, are based upon the probability of occurrence of a sentiment word along with positive or negative set of words by performing search on texts taken from the web. The two different approaches are summarized in Figure 2.1.

Since this work is focused on machine learning approaches, the main steps, common to most text classification algorithms, deserve a better explanation.
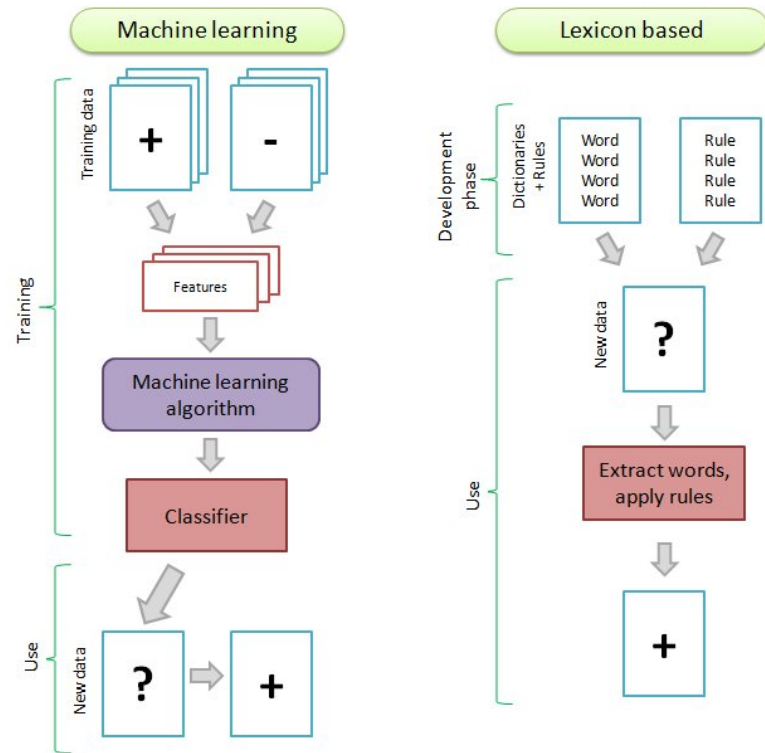
Figure 2.1: Comparison between machine learning and lexicon-based approaches on sentiment analysis. Taken from [9].

9

Tokenization

Given a characters sequence, tokenization is the task of splitting it into pieces, called *tokens*, while also dropping some useless characters like punctuation. Below an example:

Input: Friends, Romans, Countrymen, lend me your ears;

Output: ['Friends', 'Romans', 'Countrymen', 'lend', 'me', 'your', 'ears']

It is important to give some definitions [10]: a token consist in a sequence of characters grouped together in a useful semantic unit, a *type* is the class of all tokens containing the same character sequence, a *term* is a type that is included in a reference dictionary. The major question on tokenization is to select which are the correct tokens to use, for instance, for "*aren't*", which is the more correct tokenization between "*arent*", "*are n't*" or "*aren t*? A simple strategy is to split on all non-alphanumerical characters, but in this case "*aren t*" does not look correct In general tokenization is language-specific, so every language has its own rules to make a useful tokenization.

Stopwords removal

A stopword is a commonly used word, such as "the", "a", "an" or "in" that do not contribute on giving information to a text, so usually are removed. It is possible to achieve language-specific resources to get the list of stopwords for a language. An example of this task is below:

Input: "This is a sample sentence, showing off the stop words filtration."

Tokenized: ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']

Stopwords removal: ['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

Stemming

For grammatical reasons, texts use different forms of a word, such as "organize", "organise", "organizing", so it will be useful to refer all these versions to a common

one. The goal of stemming is to reduce the set of possible words by transforming all words to their common base form, for instance:

am, are, is $\Rightarrow$ be

car, cars, car's, cars' $\Rightarrow$ car

The result applied to a complete sentence will be:

Input: "the boy's cars are different colors"

Output: "the boy car be differ color"

Stemming is usually an heuristic process that respects some linguistic rules that cut off suffixes of the words, and sometimes the affixes. A similar but more complex method is called *lemmatization*. It consists on the same task of stemming, but instead of using heuristics rules, it uses dictionaries and morphological analysis of the words, so it aims to remove inflectional endings only and to return the base or dictionary form, also called *lemma*. An example that shows the difference between stemming and lemmatization can be simply shown on what happens to the word "saw": the stemming returns just "s", whereas lemmatization returns "see".

PARTS OF SPEECH TAGGING

POS tagging is the process of marking up a word in a text as corresponding to a particular part of speech, based on both its definition and its context. Identifying parts of speech is more complicated than simply mapping words to their part of speech tag, because usually words may have different POS tags with respect to the context, for instance:

She saw a *bear*.

Your efforts will *bear* fruit.

The same word "bear" has two complete different senses with respect to the context, moreover, it has two different parts of speech: in the first sentence it is a noun, while in the second one it is a verb.

To define POS tagging there are essentially two types of taggers. Rule-based taggers use contextual information to assign tags to unknown or ambiguous words. Disambiguation is done by analyzing the linguistic features of the word, its preceding and following ones and other aspects. For instance if the preceding word is an article, the current word must be a noun. Stochastic taggers consist on models that incorporates frequency os probability. In [11] is proposed a comparison between rule-based taggers and stochastic taggers.

Features extraction and representation

Especially for machine learning text classification techniques, it is fundamental to translate a document into a numerical vector, in a process called *vectorization*, to be processed by machine learning algorithms. Firstly it is mandatory to define the feature to be vectorized, that represent the document in a multidimensional space. The most implemented feature extraction technique is the *N-Gram*. It consists on extracting a set of n-words which occurs "in that order" in the document. Commonly in a feature representation are used 1-gram (which is just the list of words), 2-gram, 3-gram also together, in order to extract more information in the text.

An example of 2-gram:

Input: "After sleeping for four hours, he decided to sleep for another four."

Output: ['After sleeping', 'sleeping for', 'for four', 'four hours', 'four he' 'he decided', 'decided to', 'to sleep', 'sleep for', 'for another', 'another four']

An example of 3-gram:

Input: "After sleeping for four hours, he decided to sleep for another four."

Output: ['After sleeping for', 'sleeping for four', 'four hours he', 'hours he decided', 'he decided to', 'to sleep for', 'sleep for another', 'for another four']

Once defined the set of features it must be represented in a multidimensional space, where each feature is translated in a number. For this task there are several techniques [12]:

Bag Of Words (BoW) is a simplified version of Term Frequency (TF). It is used in several domains such as computer vision, natural language processing, spam filters as well as text classification. In BoW, a body of text is thought of like a bag of words. These words in a matrix are not sentences which structure sentences and grammar, and the semantic relationships between words and their order is ignored. In this model just the presence of a word matters. The list of used words is processed looking at the entire text in a preliminary phase. Below an example:

Input: "As the home to UVA's recognized undergraduate and graduate degree programs in systems engineering. In the UVA Department of Systems and Information Engineering, our students are exposed to a wide range of range"

Bag of Words: ['As', 'the', 'home', 'to', 'UVA's', 'recognized', 'undergraduate', 'and', 'graduate', 'degree', 'program', 'in', 'systems', 'engineering', 'in', 'Department', 'Information', 'students', 'are', 'exposed', 'wide', 'range']

Bag of Features: [1,1,1,3,2,1,2,1,2,3,1,1,1,2,1,1,1,1,1,1]

In BoW with a vocabulary of size $|\Sigma|$, every word is encoded in a vector of size $|\Sigma|$, with 1 at index corresponding to the word, and 0 in every other indexes. This model is the cause of issues due to words position in the sentence: the phrases "this is good" and "is this good" have the same BoW representation. TF is similar to BoW, that consists on counting the number of occurrences of each word instead of just the presence.

Term Frequency-Inverse Document Frequency (TF-IDF) is a method used in conjunction with TF in order to decrease the effect of commonest words (that usually do not carry information). Inverse Document Frequency (IDF) assigns a higher weight to words with either high or low frequencies term in the document. TF-IDF is the combination of both TF and IDF, and it is calculated as

$$TFIDF(w,d) = TF(w,d) * log(\frac{N}{df(t)})$$

Where N is the number of documents and $df(t)$ is the number of documents that contain the term t. Although TF-IDF overcomes to the issue of commonest words, it keeps suffering the fact that similar words are accounted for similarity since all words are independent one another.

Other techniques, called Word Embedding, such as Word2Vec [13] aim to vectorize words instead of the entire document, but these methods are not used in this work, so I'm not going to describe them.

2.1.6 APPLICATIONS

Sentiment analysis can be used for a wide range of applications [14]. The most general application is in e-commerce activities, since websites allow users to submit their experience about shopping and product qualities. They provide also summary of the product by assigning rates or scores that customers can easily see. Sentiment analysis helps these websites by converting dissatisfied customers into promoters by analyzing the huge amount of opinions.

Voice Of Market (VOM) is about determining what customers are feeling about products or services of competitors. Accurate sentiment classification from voice of market involving also temporal information helps in gaining competitive advantage and new product development. Detection of real-time sentiment information may help on planning marketing campaigns and strategies, improving product features in order to satisfy as much customers as possible.

Voice Of the Customer (VOC) consists on knowing what a customer is saying about a product or service, analyzing his reviews and feedback. VOC is a key element of Customer Experience Management. Extracting such information may help on understand what are the features that customers want on a specific product and what are the ones that do not want, for instance costs.

Brand Reputation Management is about managing the reputation in the market. Customers' opinion may enhance or damage brand reputation, so it is important to keep it monitored. Brand information, nowadays, are not just in advertising, public relations, but also in many conversation in online forums or in social media. Sentiment analysis can help brands on knowing how their products, services or the brands themselves are perceived by people and customers.

14

2.2 Sentiment Analysis Datasets

In this section we will going to describe some sentiment analysis dataset of different domains. That can be useful in order to compare the proposed ones with the final dataset utilized in this work.

Pharma [15]

This dataset provides patients reviews about drugs. Reviews are grouped into the three aspects: benefits, side effects and overall comment. Additionally, ratings are available concerning overall satisfaction as well as a five step side effect rating and five steps effectiveness rating. The data was obtained by crawling online pharmaceutical review sites.

The attributes of the dataset are the followings:

- **urlDrugName** (categorical): name of drug

- **condition** (categorical): name of condition

- **benefitsReview** (text): patient on benefits

- **sideEffectsReview** (text): patient on side effects

- **commentsReview** (text): overall patient comment

- **rating** (numerical): 10 star patient rating

- **sideEffects** (categorical): 5 step side effect rating

- **effectiveness** (categorical): 5 step effectiveness rating

Some examples of comments of different sentiments:

- **positive**: *"I was used to having cramps so badly that they would leave me balled up in bed for at least 2 days. The Ponstel doesn't take the pain away completely, but takes the edge off so much that normal activities were possible. Definitely a miracle medication!!"*

- **neutral**: *"I'm taking Gabapentin to treat nerve disorder. My nerves are inflamed and causes to experience pelvic pain and other pelvic disorders. The Gabapentin sometimes works but I do experience relapse. For the migraines I find that the 800mg of Ibprofen works better with no side effect for my migraines, but the maxalt has a better track record."*

15

- **negative**: *"after taking propecia for over a year, starting when i was 20 years of age my hair continued to thin, and i noticed no significant benefits"*

US Airline [16]

Sentiment analysis dataset about problems of major U.S. airline. Tweets acquired from February 2015.

Some meaningful attributes of the dataset are the followings:

- **airline_sentiment**: sentiment

- **airline_sentiment**: confidence

- **airline**: subject airline company

- **name**: user's username

- **text**: text of the comment

- **tweet_created**: timestamp of the tweet

Some examples of comments of different sentiments:

- **positive**: *"@VirginAmerica plus you've added commercials to the experience... tacky."*

- **neutral**: *"@VirginAmerica What @dhepburn said."*

- **negative**: *"@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse"*

IMDB [17]

This dataset is a collection of movie reviews, along with their associated binary sentiment polarity labels. It contains balanced positive and negative instances (no neutral class in this dataset). In the collection, no more than 30 reviews are present for any given movie, in order to avoid correlation between reviews of the same movie.

The attributes of the dataset are the followings:

- **text**: text of the review

- **sentiment**: the sentiment of the review

Some examples of comments:

- **positive**: "*If you like adult comedy cartoons, like South Park, then this is nearly a similar format about the small adventures of three teenage girls at Bromwell High. Keisha, Natella and Latrina have given exploding sweets and behaved like bitches, I think Keisha is a good leader. There are also small stories going on with the teachers of the school. There's the idiotic principal, Mr. Bip, the nervous Maths teacher and many others. The cast is also fantastic, Lenny Henry's Gina Yashere, EastEnders Chrissie Watts, Tracy-Ann Oberman, Smack The Pony's Doon Mackichan, Dead Ringers' Mark Perry and Blunder's Nina Conti. I didn't know this came from Canada, but it is very good. Very good!*"

- **negative**: "*Story of a man who has unnatural feelings for a pig. Starts out with a opening scene that is a terrific example of absurd comedy. A formal orchestra audience is turned into an insane, violent mob by the crazy chantings of it's singers. Unfortunately it stays absurd the WHOLE time with no general narrative eventually making it just too off putting. Even those from the era should be turned off. The cryptic dialogue would make Shakespeare seem easy to a third grader. On a technical level it's better than you might think with some good cinematography by future great Vilmos Zsigmond. Future stars Sally Kirkland and Frederic Forrest can be seen briefly.*"

In table 2.1 are shown some statistics of proposed datasets.

Dataset	Num. instances	Positive	Neutral	Negative
Pharma	5009	15.6%	73.3%	11.1%
U.S. Airline	14641	16.1%	21.2%	62.7%
IMDB	50.000	50%	0%	50%

Table 2.1: Dataset statistics

The proposed datasets present some differences. Since IMDB dataset is thought for sentiment analysis competitions, it is built in order to simplify the classification, setting the data balanced (50% positives - 50% negatives). Another characteristic of IMDB is that it does not contain the neutral class: as explained

before, product reviews social media generally contain just positives and negatives comments, so in this case the neutral class has been omitted. Apart from IMDB, data imbalance issues affect both of the other datasets: Pharma contains lot of neutral comments (73.3%), while U.S. Airline contains mostly negative ones (62.7%). Data imbalance affects real world data, so it is an issue that must be taken in consideration. An important difference between all of the three datasets consists on the language. While tweets of U.S. Airline are very concise (due to the characters limit), movie reviews are long texts. Long texts may contain sentences that supports different sentiments, for instance an user may express positive sentiment to one aspect of a movie (in case of IMDB), while criticize some other aspects. These cases are more difficult to be handled, since datasets should contain one field for each aspect to be considered, and the corresponding sentiment. In general, the presented cases show a very simple and informal language for what concerns tweets of U.S. Airline, while a long, well expressed and full of information texts for Pharma and IMDB. Lastly, Pharma dataset contains also medical-specific terms.

2.3 Supervised Learning

Since the goal of this work is to apply some machine learning techniques for sentiment analysis, it is needed an introduction to machine learning in a technical way and the most commonly used algorithm utilized in supervised learning, and also in this review.

In general, machine learning can be considered a subfield of artificial intelligence, since algorithms in some sense "learn" from data. The core function of machine learning is to automatically find a good predictor based on past experiences. While "learning by memorization" approach is sometimes useful, it lacks an important aspect of learning systems, which is the generalization. To achieve generalization, the learner should scan input data and learn some "rules", so when new data are provided, it can output some meaningful result. While human learners can rely on experience to filter out some meaningless inputs, on machine learning data must be well defined, and based on principles that protect the learner from reaching senseless conclusions.

Before going in detail, it is important to understand why we need machine learn-

ing. There are tasks that humans and animals do everyday, where it is too difficult to extract a well defined program, for instance driving, speech recognition, image understanding, or tasks that humans cannot do, for instance analysis and understanding of large and complex data sets. Other tasks need adaptivity on input data, feature that programmed tools cannot achieve, for instance handwritten text decoding, or the just described sentiment analysis.

Machine learning is, in general, a wide domain, so it is branched in several subdomains with respect to the different approaches and goals. Learning can be divided in supervised and unsupervised, depending on the fact that ground truth examples are provided or not. In unsupervised learning, clustering and pattern recognition are the most famous techniques. Supervised learning, instead, can be viewed as learning from examples, where training samples contain information that test ones do not. From here on we will focus on just supervised learning.

2.3.1 Supervised Learning Model Definition

The fist step is to define a statistical learning model [18].

- **Learner's input**:

  - **Domain set**: an arbitrary set X of samples that we wish to label. Usually samples, also called *instances*, are represented by vectors of *features*.

  - **Label set**: the set $\mathcal{Y}$ of all possible labels, for instance $\mathcal{Y} = \{0, 1\}$

  - **Training data**: the set $S = ((x_1, y_1), \ldots, (x_n, y_n))$ finite sequence of labeled points on $\mathcal{X} \times \mathcal{Y}$. That is the input provided to the learner.

- **Learner's output**: the learner is requested to output a *prediction rule* $h : \mathcal{X} \to \mathcal{Y}$, called also *hypothesis* or *classifier*. It can be used to predict labels of new points.

- **Data generation model**: we assume that instances are generated by some probability distribution D over X, that represents the environment. It is important that the learner should not know anything about the distribution. We assume also that there is a correct labeling function $f : \mathcal{X} \to \mathcal{Y}$, and that $f(x_i) = y_i, \forall i$, almost initially. The labeling function is unknown, since

it is what the learner aims to find. Summarizing, each pais of S is sampled according to D and then labeled by f.

- **Measure of success**: we define the *error of the classifier* the probability that it does not predict the correct label of a point generated according to the generative distribution. Formally, it is defined as:

$$L_{D,f}(h) \stackrel{\text{def}}{=} P_{x \sim D}(h(x) \neq f(x))$$

That is, given a predictor h, the probability that randomly chosen an observation x according to D, we have that $h(x) \neq f(x)$.
$L_{(D,f)}(h)$ is called *generalization error*, or *true error* of h.

2.3.2 EMPIRICAL RISK MINIMIZATION

The learner that receives in input a training set S, outputs an hypothesis h_s : $x \to Y$. The algorithm searches the hypothesis h_s that minimizes the error, but since both D and f are not known, it minimizes the so called *training error L_s* defined as:

$$L_s(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(h_i) \neq y_i\}|}{m}$$

for $[m] = 1, \ldots, m$. Empirical Risk Minimization (ERM) is the task of searching a predictor that minimizes $L_s(h)$ that makes sense, since the only information available to the learner is just the training set S (that should be a snapshot of the world), moreover, since D and f are unknown, minimizing the true error is not possible.

ERM rule, however, may have the side effect that learns "too well" training data, that does not work well on new data from D. This effect is called *overfitting*, and a common solution to prevent it, is searching h in a restricted set of predictors. In this way, ERM rule can be written as:

$$hs \in argmin_{h \in H} L_s(h)$$

By restricting on a finite set of hypotheses H, we *bias* the learner, since we cannot guarantee that the algorithm picks the optimal one. Contrary to overfitting, *underfitting* happens when the learner has poor performance on training data,

Figure 2.2: Visual explanation of overfitting and underfitting. Taken from [19].

for instance it may happen when the hypothesis class is completely wrong with respect to the right (but still unknown) one. A visual example of underfitting and overfitting is shown in Figure 2.2.

ERM rule depends on training set S, so a general assumption is to consider elements in S, independently and identically distributed according to the distribution D. However, picking a restricted number of samples, it is possible that those data are not *representative* of the distribution, and the learner may be affected by this issue. Furthermore, it is more realistic to not assume that labels are fully determined by features on input data. Saying δ the probability that the training set is not representative, and ϵ an *accuracy parameter*, it is possible to interpret $(1 - \delta)$ as a *confidence parameter*, and the event $L_{(D,f)}(h_s) > \epsilon$ a failure of the learner.

After these assumptions, the goal of the learner is to find a predictor that is probably approximately correct, formally an hypothesis h_s that with probability $> (1-\delta)$ (that determines how confident is the solution), upper bounds the generalization error $L_{(D,f)}(h_s) < \epsilon$ (how well the predictor has learned). Since D and f are still not known, the learner tries to find the optimal hypothesis by minimizing the empirical risk.

Machine Learning Tasks

The model discussed so far can be applied in multiple machine learning tasks.

- **Classification**. It is the task of predicting a discrete and finite set of labels.

21

Such algorithms have access to a set of correctly classified samples, and on the base of these instances, learns a predictor that taken a new instance, it predicts its own class. Classification ca be binary or multiclass, depending on the size of the possible set of labels. This work is focused on this task, more precisely on both binary and muticlass classification.

- **Regression**. It is the task of fining some simple patterns on data that a re functional dependencies beyween the sets $\mathcal{X}$ and $\mathcal{Y}$.

A supervised learning task can be seen as an optimization problem that aims to minimize a *loss function*. Introducing some formalism, a loss function ℓ is a function from $\mathcal{X} \times \mathcal{Y}$ to the set of non negative real numbers

$$\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$$

Saying z belonging to $\mathcal{X} \times \mathcal{Y}$, the generalization error becomes:

$$L_D(h) \stackrel{\text{def}}{=} \mathbb{E}_{z \sim D}[\ell(h, z)]$$

While the empirical risk becomes:

$$L_s(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i)$$

The loss function ℓ is characteristic of the learning task: in binary and multiclass classification, the so called 0-1 loss is the correct choice:

$$\ell_{0-1}(h, (x, y)) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if} \quad h(x) = y \\ 1 & \text{if} \quad h(x) \neq y \end{cases}$$

That intuitively counts the number of misclassified samples. While in regression tasks, the squared loss is a better choice:

$$\ell_{sq}(h, (x, y)) \stackrel{\text{def}}{=} (h(x) - y)^2$$

Figure 2.3: Learning curve for training and validation

2.3.3 Model Selection and Validation

Suppose we are facing a classification problem. As explained so far, the common approach is applying the ERM rule on a training set S restricting to a finite set of hypotheses H. However, we want to try different algorithms (or same algorithm with different parameters) on the same training set, and finally select the best predictor. In this way we need new data in order to evaluate the performance of the algorithms, so usually the approach is the following: the training set is splitted in two parts, training and validation sets. The first set is used to train the whole set of algorithms, obtaining the set $H = \{h_1, \ldots, h_r\}$ of best predictors, then these hypotheses are tested on the validation set, and only the hypothesis that reaches the minor loss is picked (actually we apply the ERM rule on H).

In Figure 2.3 it is shown the usual behavior in validation: training error is monotonically decreasing, because on growing the model complexity, the data are more well fitted. The validation error, instead, at some point starts increasing, that is symptom of overfitting. In this way, the predictor that should be chosen on validation is the one with complexity equal to the point on which the validation curve starts increasing, limiting both overfitting and underfitting.

In practical problems, the dataset is splitted into three subsets: the training set is used to train the learning algorithms, the validation set is used to select the best learned model, and the test set is used to estimate the true error. This last

23

set cannot be used in the learning process, otherwise it can bias the final result, so it must be used only once. If the learning fails, new data must be used for test the final model.

K-Fold Cross Validation

Sometimes data are not plenty, so test-validation split may reduce too much the training set size, so a common approach is the so called *k-fold cross validation* [20]. In this technique, data is partitioned in k subsets of equal size called *folds*, usually respecting the labels distribution in a process called *stratification*. Then, for k iterations, the learning algorithm is trained with $k-1$ folds, and validated using the last one, as shown in Figure 2.4. Finally, the score is calculated as the average of the k scores obtained with the k validation tests.
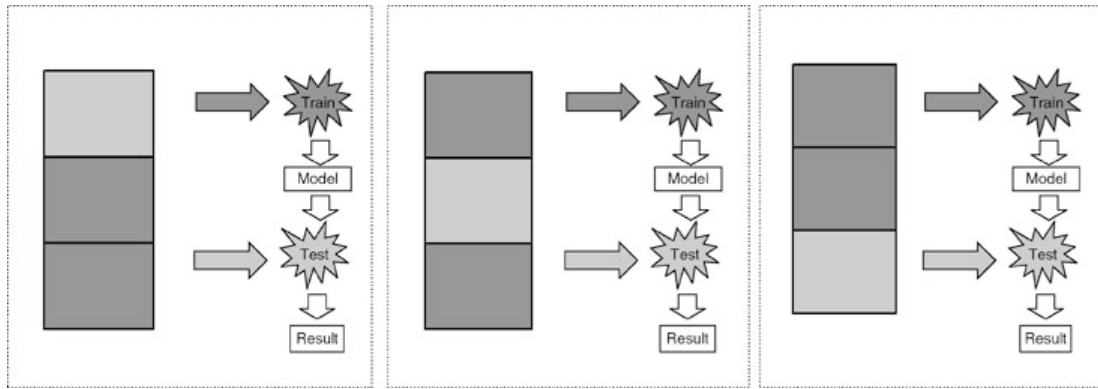
Figure 2.4: Example of k-fold cross validation with k=3. Taken from [20].

2.3.4 Linear Models

One of the most important and commonly used hypotheses class is the one of *linear predictors*. Since this class is used also in this work, it deserves a dedicated focus.
Define the class of affine functions as:

$$L_d = \{h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b$$

L_d is a set of functions, parameterized by $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, that take in input a vector $x \in \mathbb{R}^d$ and outputs $\langle \mathbf{w}, \mathbf{x} \rangle + b$, which is a scalar. The different hypotheses classes are compositions of a function $\Phi : \mathbb{R} \to \mathcal{Y}$ on L_d, for instance in binary classification the function Φ can be chosen as the sign function.

Usually the parameter b is incorporated into $\mathbf{w}$ adding one extra coordinate equal to 1 as follows:

let $w' = (b, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$ and let $x' = (b, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$.

Therefore,

$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle$ so in general the parameter b is omitted because included in $\mathbf{w}$. It follows that each affine function can be thought as a homogeneous linear function in an upper dimension, with a coordinate equal to 1.

The class used for binary classification is the *halfspaces class*. It is made by the composition of the class of linear functions L_d with the sign function, so $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, 1\}$.

$$HS_d = sign \circ L_d = \{sign(h_{\mathbf{w},b}(\mathbf{x})) : \mathbf{x} \in \mathbb{R}^d, h_{\mathbf{w},b}(\mathbf{x}) \in L_d\}$$

Each hypothesis forms an hyperplane that is perpendicular to $\mathbf{w}$, and intersects the vertical axis in $(0, -\frac{b}{w_2})$. The instances that are "above" the hyperplane, which means on the same direction of $\mathbf{w}$, are labeled positively, while the others negatively, as shown in Figure 2.5.

2.3.5 REGULARIZATION

In common machine learning problems, it is important to be on guard against overfitting. In some cases a philosophical solution comes from the *Occam's Razor*, which states that from a pool of solutions, it is better to choose the simplest one, on the basis that they are capturing more fundamental aspects. *Regularization* is an application of this principle to ERM: instead of searching the model that best fits the training data, a penalty based on the model complexity is added. In fact, in general, algorithms tend to overfit the training data, learning a model that is more complex, so adding a penalty on the optimization function, makes
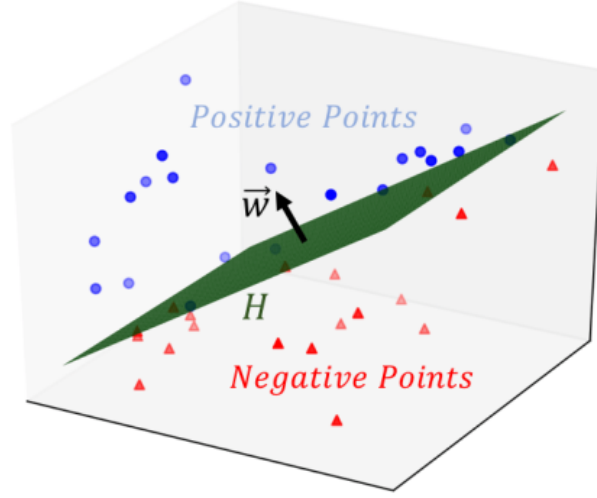
Figure 2.5: Halfspace in $\mathbb{R}^3$. Taken from [21].

the learner more *stable*, that also generalize better [22]. The ERM rule becomes:

$$argmin_{\mathbf{w}}(L_s(\mathbf{w}) + R(\mathbf{w}))$$

Where $R(\mathbf{w}))$ is a regularization function $R : \mathbb{R}^d \to \mathbb{R}$. This approach is called Regularized Loss Minimization (RLM). Intuitively, the model complexity is measured by the value of the regularization function, and the algorithm balances between low empirical risk and simpler hypotheses.

Some common regularization functions are the followings:

- **L1 Regularization**: the regularization function is

$$R(\mathbf{w}) = \lambda \sum_{i=1}^{m} |w_i|$$

- **L2 Regularization**: the regularization function is

$$R(\mathbf{w}) = \lambda \sum_{i=1}^{m} (w_i)^2$$

Writing the expected risk of a learning algorithm as

$$\mathbf{E}_S[L_D(A(S))] = \mathbf{E}_S[L_S(A(S))] + \mathbf{E}_S[L_D(A(S)) - L_S(A(S))]$$

it is shown that it is composed by two terms: the first one describes how well the algorithm fits the training data, and increases with λ, while the second one reflects just the difference between the true risk and the empirical risk. This second term can be seen as the stability of the learner, and decreases with λ. We are facing a tradeoff between fitting and stability that reflects on the choice of the parameter λ.

2.3.6 GRADIENT DESCENT

The goal of a machine learning task is to minimize the risk function $L_d(h)$. It is not possible to minimize it directly, since it depends on the unknown distribution D, so another approach is mandatory. In general there are multiple possible approaches, but the most used derives from the Gradient Descent (GD) [23].
Recall that the gradient $\nabla f(\mathbf{w})$ of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is the vector $\nabla f(\mathbf{w}) = \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)$, that points to the direction of maximum growth of the function. GD is an iterative algorithm that starts from a initial point $\mathbf{w}^{(0)}$, and at each iteration t updates the point by "moving" a step in the opposite direction of the gradient, as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)})$$

where $\eta > 0$ is a parameter that intuitively defines the length of the step. After T iterations of the algorithm, the value of the function should converge to a value $(w)^{(T)}$, which is the supposed minimum (since it is an heuristic, the result is not guaranteed to be the absolute minimum, but just a local minimum). The final result can be either the last vector found $\mathbf{w}^{(T)}$, the best performing $argmin_{t \in T} f(\mathbf{w}^{(t)})$, or the average $\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^{(t)}$ (preferred in the case of non differentiable functions). The pseudocode of GD is shown in Algorithm 2.1.

Algorithm 2.1 Gradient Descent

1: **parameters**: scalar $\eta > 0$, integer $T > 0$
2: **initialize**: $\mathbf{w}^{(1)} = \mathbf{0}$
3: **for** $t = 1, 2, \ldots, T$
4: $\quad \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)})$
5: **output**: $\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^{(t)}$

However, in GD it is required to know the exact gradient of the function, which is unfeasible especially when working on large sets of data, so an approximation may be useful as well. Stochastic Gradient Descent (SGD) requires that the expected value of the update direction is equal to the gradient direction, which is a great advantage in terms of calculations, since it requires less points to be calculated.b The pseudocode of SGD is shown in Algorithm 2.2.
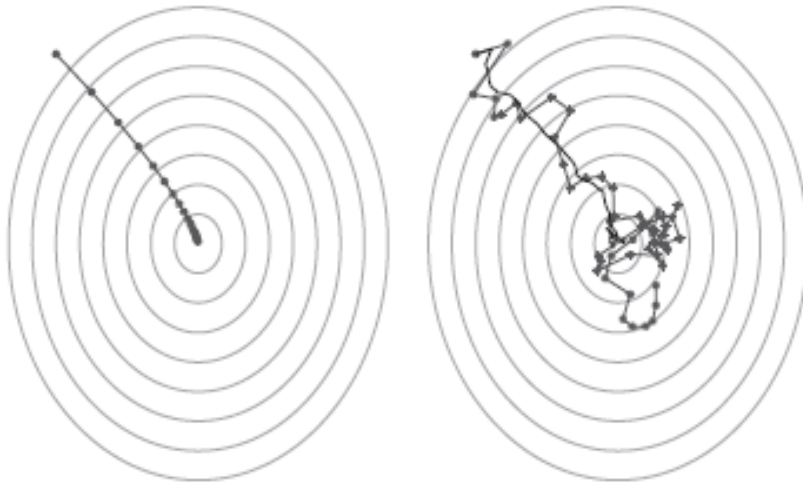

Algorithm 2.2 Stochastic Gradient Descent

1: **parameters**: scalar $\eta > 0$, integer $T > 0$
2: **initialize**: $\mathbf{w}^{(1)} = \mathbf{0}$
3: **for** $t = 1, 2, \ldots, T$
4:     choose $\mathbf{v_t}$ at random from a distribution such that $\mathbb{E}[\mathbf{v_t}|\mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$
5:     $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla \mathbf{v_t}$
6: **output**: $\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^{(t)}$

A visual demonstration of the behavior of both GD and SGD is shown in Figure 2.6. It is shown that SGD to reach the minimum, takes more steps in a direction that is not exactly the gradient direction, but only on the average.

Figure 2.6: Behavior of Gradient Descents (left) and Stochastic Gradient Descents (right). Taken from [24].

2.3.7 Feature Selection

As early explained, supervised learning is based on finding a proper function that joins the input vectors of features $\mathcal{X}$ with their label $\mathcal{Y}$. Sometimes, the output $\mathcal{Y}$ does not depend by all features $(X_1, \ldots, X_d)$, instead, it is decided by a subset of them $(X_{(1)}, \ldots, X_{(r)})$, with $r < d$. With sufficient data and time, it is fine to use all input features for learning a classification algorithm (including also those irrelevant), but in practice there are motivations for that it is preferable to select only relevant ones.

- The irrelevant features will induce great computational cost.

- The irrelevant features may lead to overfitting, for instance if in sentiment analysis the name of the person for which it is going to classify the comment is included in the features set, then the classification algorithm may depend on this (which obviously is a mistake).

- Since the goal of supervised classification is to estimate a function with respect to the input features, it is reasonable to ignore features with low impact on the decision making, in order to keep the model as simple as possible.

The feature selection problem has been studied for many years, in fact many solutions are proposed for machine learning tasks. Below are presented some heuristic feature selection procedures that have been implemented in this work.

Feature Selection for Linear SVM

In linear classifiers, the outcome of a sample vector $x = (x_1, \ldots, x_d)$ is calculated as $y = sign(\mathbf{w}^T \mathbf{x})$. In [25] it is explained a feature selection method that uses the intuitive fact that weights with small absolute value $|w_j|$ does not contribute much on the decision ov the value y. This means that those features are not relevant on the final decision, so they could be discarded. This situation is valid also for SVM with linear kernel, while for other non-linear kernels it is not applicable. The described strategy involves the linear SVM classifier, and it is the following:

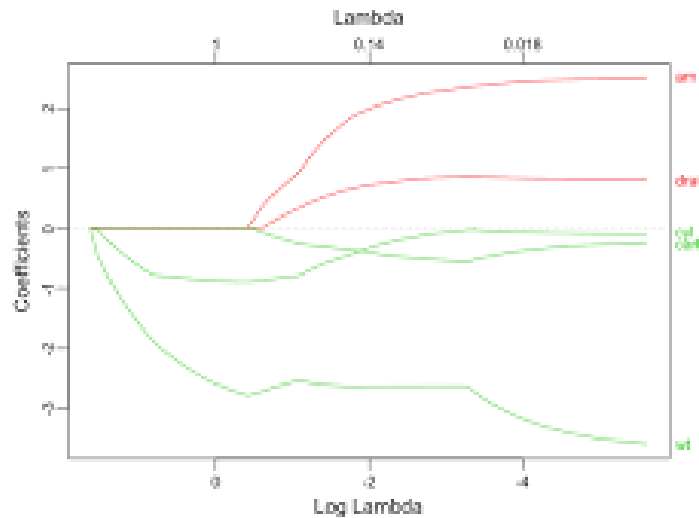- Train the classifier with training data;

- Eliminate the features whose weight is close to zero, formally setting a threshold T, just keep the features $\{w_i : i \in [d], |w_i| > T\}$;

- Finally, re-train the model using just the selected features.

FEATURE SELECTION L_1 NORM REGULARIZATION

In regularization, it is optimize a function composed by both empirical risk, and a regularization function. In case of L_1 norm regularization, the term

$$\lambda \sum_{i=1}^{d} |w_i|$$

especially when λ is large, forces some of the weights w_i to be exactly equal to zero [26]. A typical behavior of the values of the weights with L_1 norm regularization is shown in Figure 2.7.

Figure 2.7: Behavior of weights on L_1 norm regularization on with respect to λ. Taken from [27].

It is possible to see that L_1 norm regularization selects automatically some weights to be nonzero, while sets zero the others. In this situation, the features

to be kept are obviously the one with nonzero values. The strategy is the same as previously mentioned:

- Train the classifier with training data;

- Eliminate the features whose weight is zero (or close to);

- Finally, re-train the model using just the selected features.

INFORMATION GAIN FEATURE SELECTION

Intuitively, Information Gain (IG) measures the amount of information in bits about the class prediction, if the only information available is the presence of a feature and the corresponding class distribution [28].
Given a random variable X, the following definitions comes from Shannon's Information theory [29]:

- The Information associated to an event x from the distribution X is

$$I(x) = \log_2 \frac{1}{\mathbb{P}_X(x)} = -\log_2 \mathbb{P}_X(x)$$

- The Entropy of a random variable X is the average amount of information of the possible events

$$H(X) = -\sum_{x \in X} \mathbb{P}_X(x) \log_2 \mathbb{P}_X(x)$$

Both Information and Entropy are pure numbers, but "measured" in *bits*.
Concretely, IG measures the expected reduction in entropy by choosing a given feature. The IG of a training set S with respect to a feature $i \in [d]$ is calculated as:

$$IG(S, i) = H(S) - \sum_{v \in values(X_i)} \mathbb{P}(X_i = v) H(S_{X_i = v})$$

Where X_i is the set of values of the feature i, and $S_{X_i = v}$ is the training set restricted to the samples where $X_i = v$.
Feature selection using IG works as follows: given the training dataset S, IG score

is calculated for each feature X_i. Next, all scores are sorted in a ranking, then the most important features are selected. The selection can be done either by choosing a threshold on the IG, or by choosing the size of the set of the selected features.

In the followings sections will be presented some of the most used algorithms for binary classification, that are also exploited in this work.

2.3.8 Logistic Regression

Logistic Regression is a model that analyzes the relationships between multiple independent variables, and a categorical dependent variable, namely between the features and the labels in case of a supervised classification task [30].
In the context of linear models, it was described the class of hyperplanes, which is a set of functions from $\mathbb{R}^d$ to $\{0, 1\}$. However, in some cases it may be useful to have a measure of confidence about the prediction. Logistic Regression provides that information by defining the probability that the label of an input instance $\mathbf{x}$ is 1. Calling $\mathbb{P}_{\mathbf{w}}[\mathcal{Y}|\mathcal{X}]$ that conditional probability to get the label $\mathcal{Y}$ given the instance X, under the estimator with parameter $\mathbf{w}$. Statistical theory suggest that the probabilistic classifier should be learned from the training set S using the MLE estimator as:

$$\hat{\mathbf{w}}_{MSE} = argmax_{\mathbf{w}} \quad \mathbb{P}_{\mathbf{w}}[y_1|x_1], \ldots, \mathbb{P}_{\mathbf{w}}[y_n|x_n]$$
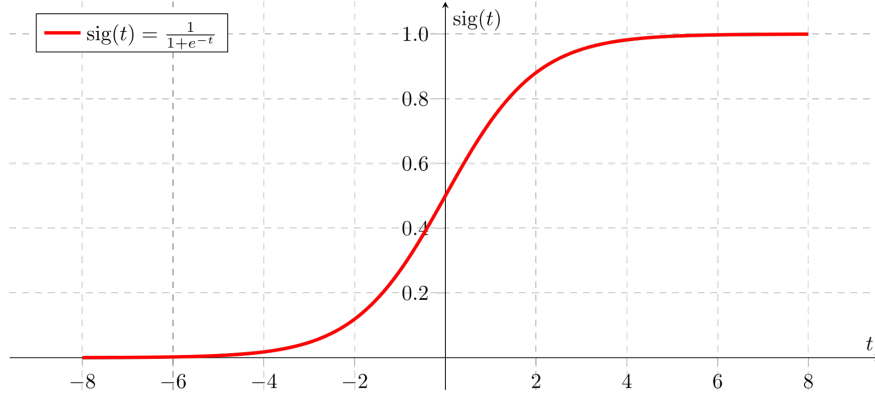
In Logistic Regression the conditional probability $\mathbb{P}_{\mathbf{w}}[\mathcal{Y}|\mathcal{X}]$ is expressed as:

$$\mathbb{P}_{\mathbf{w}}[\mathcal{Y} = y|\mathcal{X} = x] = \frac{1}{1 + e^{y\langle \mathbf{w}, \mathbf{x} \rangle}}$$

where y can be either -1 or 1. This result is obtained by the composition of the prediction of the classifier $y = \langle \mathbf{w}, \mathbf{x} \rangle$ (which has no bounds on the values) with the sigmoid function

$$sig(t) = \frac{1}{1 + e^{-t}}$$

which is shown in Figure 2.8.

Figure 2.8: Sigmoid function.

The hypotheses class is therefore

$$H_{logreg} = sig \circ L_d = \{sig(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{x}, \mathbf{w} \in \mathbb{R}^d\}$$

An interpretation of the formulas may be useful to understand. As long the value $\langle \mathbf{w}, \mathbf{x} \rangle$ is large, the predictor can be considered sure about the prediction (the value is far from the separating hyperplane). Contrary, if the value $\langle \mathbf{w}, \mathbf{x} \rangle$ is low, it means that $\mathbf{x}$ falls nearby the separating hyperplane, so the prediction may be less confident. Applying the prediction value to the sigmoid function, as long the value is large, the result is getting closer to 1, while when the prediction value becomes negative, the final value is getting closer to 0. Therefore, the value obtained by the sigmoid can be viewed as the probability that the input value $\mathbf{x}$ has label 1.

Summarizing all the concepts, the final goal is to have the value $y\langle \mathbf{w}, \mathbf{x} \rangle > 0$ for each $(x, y) \in S$ (or more conveniently $\gg 0$). Recalling the Maximum Likelyhood Estimator solution as

$$\hat{\mathbf{w}}_{MSE} = argmax_{\mathbf{w} \in \mathbb{R}^d} \quad \mathbb{P}_{\mathbf{w}}[\mathcal{Y}|\mathcal{X}]$$

$$= argmax_{\mathbf{w} \in \mathbb{R}^d} \quad \prod_{i=1}^{n} \frac{1}{1 + e^{-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle}}$$

$$= argmin_{\mathbf{w} \in \mathbb{R}^d} \quad \sum_{i=1}^{n} log(1 + e^{-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle})$$

which derives the final optimization problem as:

$$argmin_{\mathbf{w} \in \mathbb{R}^d} \quad \sum_{i=1}^{n} log(1 + e^{-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle})$$

The same result can be reached following the ERM rule by setting the loss function increasing with the probability that the estimator is wrong, formally defining

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = log(1 + e^{-y \langle \mathbf{w}, \mathbf{x} \rangle})$$

The associated ERM problem becomes

$$argmin_{\mathbf{w} \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} log(1 + e^{-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle})$$

2.3.9   SVM

In context of binary classification, Support Vector Machiness became one of the most popular classification methods [31]. SVM is based on the hypotheses class of hyperspaces, already described, and the main concept is to find the hyperplane that best classifies the training data.
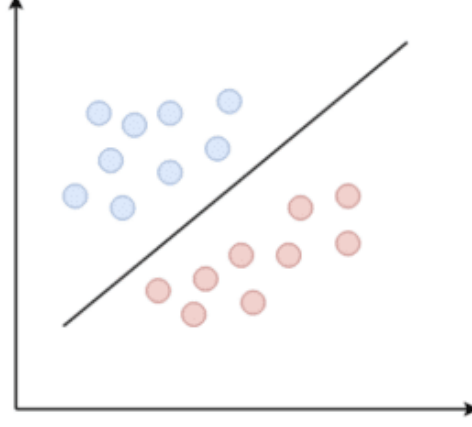
Hard-SVM

Suppose that training data are linearly separable, formally

$$\forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) > 0$$

intuitively, there exists an hyperplane that classifies correctly every instance ot the training set, like the one shown in Figure 2.9

Figure 2.9: Example of linearly separable set.

Given a set of possible separating hyperplanes, it is intuitively better to prefer the one with largest *margin*. The margin of an hyperplane with respect to a training set is defined to be the minimal distance between the hyperplane and a point of the set, formally

$$min_{i \in [m]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$$

Therefore, the Hard-SVM rule, which finds the largest margin in linearly separable case is:

$$argmax_{(\mathbf{w},b):\|\mathbf{w}\|=1} min_{i \in [m]} \quad |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \quad s.t. \quad \forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) > 0$$
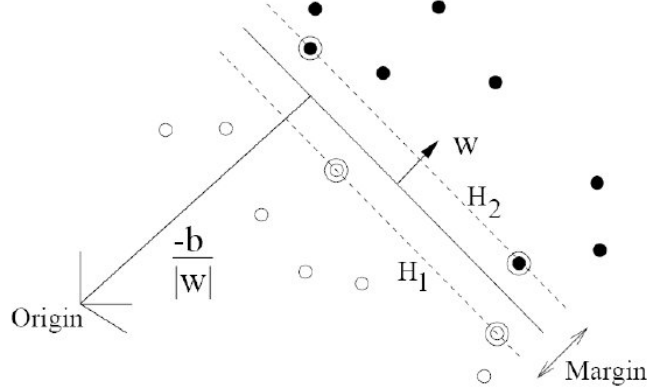
Or another equivalent formulation as a quadratic optimization problem is

$$(\mathbf{w}_0, b_0) = argmin_{(\mathbf{w},b)} \|\mathbf{w}\|^2 \quad s.t. \quad \forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) \geq 1$$

The output should be $\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \hat{b} = \frac{b_0}{\|\mathbf{w}_0\|}$.

The result of the Hard-SVM optimization problem is shown in Figure 2.10.

The points at minimum distance from the hyperplane are called *Support Vectors*.

Figure 2.10: Hard-SVM solution. Taken from [32].

SOFT-SVM

The Hard-SVM formulation supposes that data are linearly separable, however in usual cases, data are not. To handle this situation, Soft-SVM model permits to some points to fall in the wrong side of the hyperplane, but adding a penalty on the objective function represented by *slack* variables ξ_i , that control how far from the wrong side the point $\mathbf{x}_i$ is. The optimization problem becomes

$$argmin_{(\mathbf{w},b)} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \quad s.t. \quad \forall i \in [m], y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) \geq 1 - \xi_i, \xi \geq 0$$

The parameter C is a regularization parameter, that controls the trade-off between margin's size and the training errors.

Solving both the Hard-SVM and Soft-SVM optimization problems is quite hard, due to the complicated constraints [33]. The mathematical tool for simplify the problem is the Lagrangian duality theory [34].

From theory, calling $\mathbf{w}_0$ the solution of Hard-SVM problem, and calling $I = i : |\langle \mathbf{w}_0, \mathbf{x}_i \rangle| = 1$ (the set of indices that defines the support vectors), then there exist coefficients $\alpha_1, \ldots, \alpha_m$ such that

$$\mathbf{w}_0 = \sum_{i \in [I]} \alpha_i \mathbf{x}_i$$

The same result is valid also for Soft-SVM problem. The Lagrangian duality

36
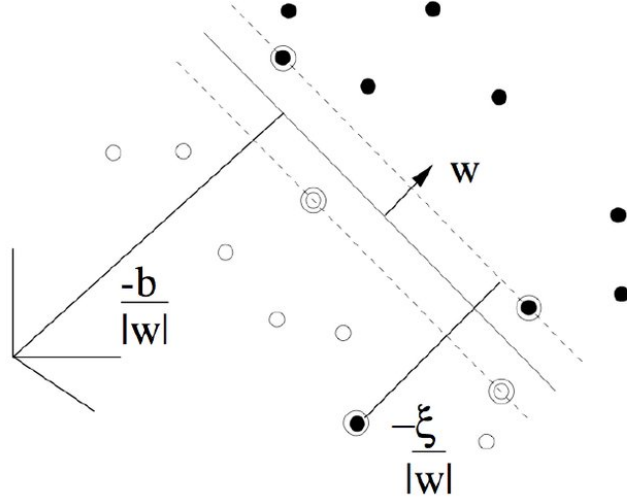
Figure 2.11: Soft-SVM solution. Taken from [32].

theory leads to solving the Hard-SVM dual optimization problem

$$max_{\alpha \in \mathbb{R}^m} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

subject to

$$\forall i, \quad \alpha_i \geq 0, \quad \sum_i y_i \alpha_i = 0$$

and the Soft-SVM dual optimization problem

$$max_{\alpha \in \mathbb{R}^m} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right)$$

subject to

$$\forall i, \quad 0 \leq \alpha_i \leq C, \quad \sum_i y_i \alpha_i = 0$$

The resulted dual problems are much simpler, since the constraints are less complicate.

Sometimes points in the training set are far from being linearly separable, like the one in Figure 2.12. This is because the expressiveness of the halfspaces class is limited. However, it is possible to map the original instance space into another space (possibly of a higher dimension), and then learn a halfspace in that space.
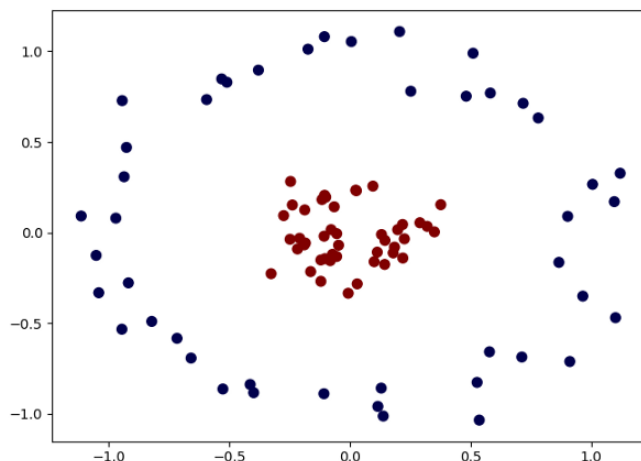
Figure 2.12: Example of non linearly separable data. Taken from [35].

The basic paradigm become:

- Given a domain X, choose a mapping $\phi : X \to F$, for some feature space F;

- Given a training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$, apply the mapping to the instances $\hat{S} = ((\phi(\mathbf{x}_1, y_1)), \ldots, (\phi(\mathbf{x}_m), y_m))$;

- Train a linear predictor h on $\hat{S}$;

- New predictions of some point $\mathbf{x}$ are made on $h(\phi(\mathbf{x}))$.

An example of solution with kernel is shown in Figure 2.13, where data are mapped with the function

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$\mathbf{x} \to (\mathbf{x}, \|\mathbf{x}\|)^2$$
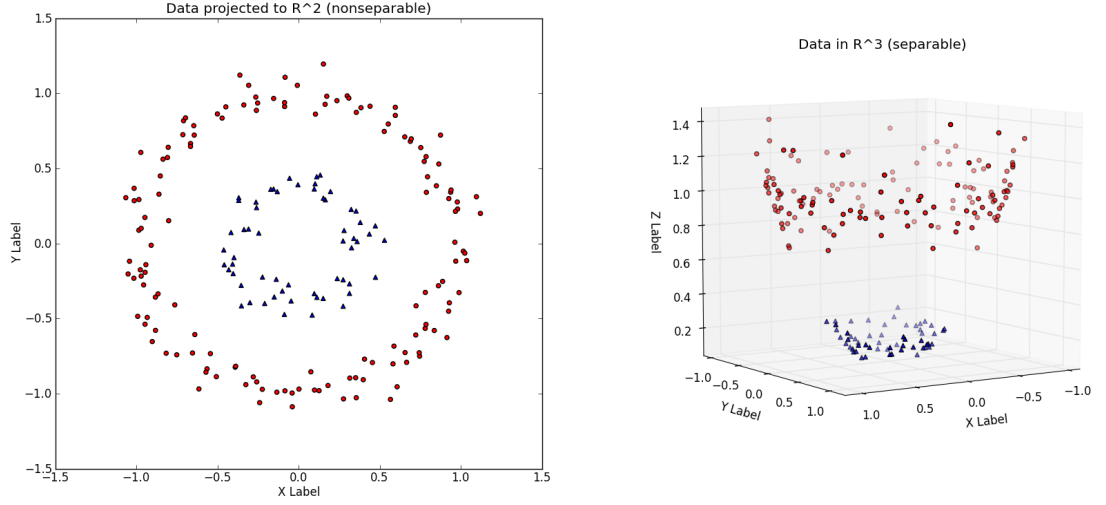
Figure 2.13: Example of solution with kernel. Taken from [36].

Embedding the input space into a higher dimensional space, makes the half-space class more expressive. However, computing the separating hyperplane in a high dimension may be computational expensive. Looking at the SVM dual optimization problem, the increasing computational cost comes from the calculation of $\langle \mathbf{x}, \mathbf{x}' \rangle$, to the calculation of $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ (called also *Kernel Function* $K(\mathbf{x}, \mathbf{x}')$). However, for some kernel functions, it is possible to calculate directly $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ without passing through firstly the calculation of the mapping and then the inner product, that saves some computation. This procedure is called *Kernel Trick*.

Some examples of the most used kernel functions are:

- K-degree Polinomial kernel

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma}}$$

- Radial Basis Function (RBF) kernel

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma(\mathbf{x} - \mathbf{x}')^2}$$

Where σ on gaussian kernel, and γ on RBF kernel are parameter. is shown in

Algorithm **??**

2.3.10 RANDOM FOREST

Recalling that the goal of classification is to minimize a cost function $L_D(h)$ by optimizing the empirical risk $L_s(h)$ (eventually including the regularization term), the so called *Ensemble* models try to obtain the optimal value from a collection of multiple *base learners* $h_1(\mathbf{x}), \ldots, h_J(\mathbf{x})$. These learners are then combined to obtain the final prediction $h(\mathbf{x})$ by "voting" the most common outcome.

$$h(x) = argmax_{y \in \mathcal{Y}} |[j \ for \ j \in [J] \ if \ h_j(\mathbf{x}) = y]|$$

Random Forest is an ensemble learning model, based on *Decision Trees* base learners [37].

DECISION TREES

A decision tree is a flowchart-like tree structure, where each internal node represent a test on an attribute, each branch represent an outcome of the test, class label is represented by each leaf node [38]. For instance, suppose we want to predict whether a person is fit, given some simple information like the age, or what he eat. A decision tree that makes that classification may be the one in Figure 2.14.
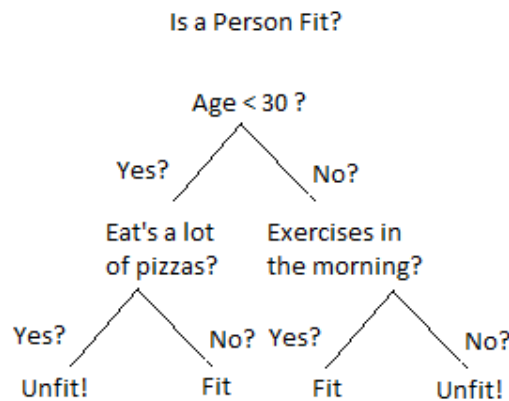
Figure 2.14: Decision Tree for decide whether a person is fit.

The task of learning a decision tree consists on determine the splits that define the tree, such that the loss is minimized. Searching for the optimal decision tree that minimizes the loss is computationally hard, so it is preferable to adopt heuristic rules where the tree is built step-by-step by searching a locally optimal solution.

A general procedure for building a decision tree starts with a single leaf tree (root), and assigns this node the label according to the majority vote among all labels on the training set. Now, for a series of iterations, it is examined the effect of splitting a single leaf. Defining as *gain* a measure of improvement due to the split, the goal for each step is to find the split that reaches the maximum gain.

A possible implementation of decision tree learning algorithm, based on information gain metrics for the splitting criterion, is known as "ID3" (Iterative Dichotomizer 3) [39].

At each iteration, the algorithm selects the attribute with highest information gain as splitting attribute. When information gain approaches to zero, the growing of the tree stops. A recursive solution is shown in Algorithm 2.3.

Algorithm 2.3 ID3(S, A)

1: **input**: training set S, feature subset $A \subseteq [d]$
2: **if** all examples in S are labeled with 1
3:      **return** a leaf 1
4: **if** all examples in S are labeled with 0
5:      **return** a leaf 0
6: **if** $A = \emptyset$
7:      **return** a leaf whose value = majority of labels in S
8: **else**
9:      Let $j = argmax_{i \in A} IG(S, i)$
10:      **if** all examples in S have the same label
11:          **return** a leaf whose value = value of the majority of the labels in S
12:      **else**
13:          Let T_1 be the tree returned by ID3($\{(\mathbf{x}, y) \in S : S : x_j = 1\}, A\ j$)
14:          Let T_2 be the tree returned by ID3($\{(\mathbf{x}, y) \in S : S : x_j = 0\}, A\ j$)
15:          **return** left child of the input node = T2, right child of the input node = T1

A similar approach described in "CART" (Classification and Regression Trees) [40], involves the *Gini* index, instead of information gain. Gini index is calculated

41

as

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Where p_i is the probability of a value i in the distribution of the interested attribute. The Gini index is calculated for each attribute, and the split is made with respect to the attribute with lowest value of Gini index.

RANDOM FOREST

Decision Trees lead easily to overfitting. A solution to prevent the issue, is to construct an ensemble of trees, which is the basic of Random Forest Classifiers. In Random Forest, each decision tree is constructed by applying the mentioned algorithm on the training set S and an additional randome vector Θ, where Θ is sampled from some distribution. The final prediction is obtained by the majority of the votes. There are many ways to define the vector Θ, a possible one is the following:

- Take a random subsample of S, obtaining S';

- Construct a sequence $I_1, I_2, \ldots$ where each I_t is a subset of $[d]$ of size k, generated by sampling k random elements from $[d]$;

Finally, each Decision Tree is based on the set S', where at each split, it is restricted to choosing among the set of features I_t

2.3.11   NAÏVE BAYES

Naïve Bayes is a subset of Bayesian decision theory [41]. It is called naïve because the formulation makes some assumptions. The classifier assumes that each feature only depends on the class of the instance (that implies also that all features are independent each other), as shown in Figure 2.15. However, in real world cases, it is not possible to guarantee this assumption, but the classifier works still reaches good results.
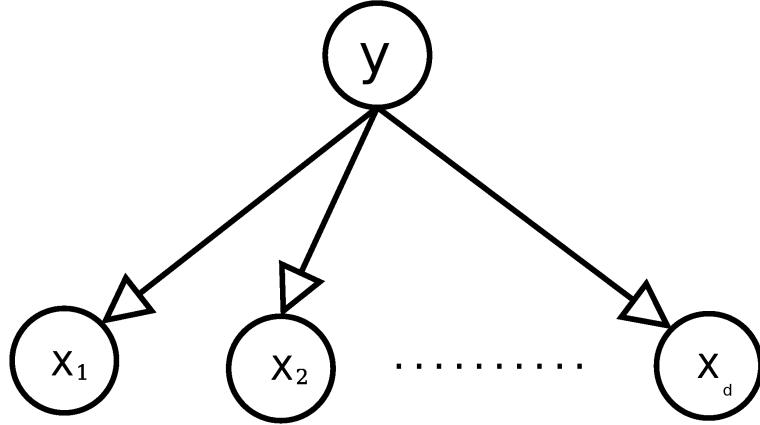
Figure 2.15: Features independency on Naïve Bayes classifier.

Let y denote the class of an observation $\mathbf{x}$. Predict the class of $\mathbf{x}$ by searching the class y that maximize the value of the Bayes Rule (posterior probability)

$$\mathbb{P}(y|\mathbf{x}) = \frac{\mathbb{P}(y)\mathbb{P}(\mathbf{x}|y)}{\mathbb{P}(\mathbf{x})}$$

Exploiting the fact that the features $x_1, \ldots, x_d$ are independent, the following equation is sufficient to predict the most probable class

$$\mathbb{P}(y|\mathbf{x}) = \frac{\mathbb{P}(y)\prod_{i=1}^{d}\mathbb{P}(x_i|y)}{\mathbb{P}(\mathbf{x})}$$

2.3.12 MULTICLASS CLASSIFICATION

Since this point, all classification algorithms are thought for binary classification, namely for a labels set $\mathcal{Y} = -1, +1$. However, many real world classification problems require a larger labels set, leading to a multiclass classification problem. Some algorithms, thought for binary classification, can be extended in the multiclass case, for instance Naïve Bayes and Support Vector Machines, while other approaches convert the multiclass problem into a set of binary ones that are solves using common classification algorithms [42]. Two main approaches belonging to the second category are the so called One-Versus-One (OVO) and the One-Versus-All (OVA). Below are explained some methods for multiclass classification, that have been used in this work.

Multiclass Classification for Naïve Bayes

Naïve Bayes classifier works actually in multiclass case. Given a problem with k classes $(y_1, \ldots, y_k)$, it assigns the label y to an unknown example $\mathbf{x} = (x_1, \ldots, x_d)$ by maximizing the posterior probability, formally

$$y = argmax_{y \in (y_1, \ldots, y_k)} \mathbb{P}(y|x_1, \ldots, x_d)$$

That works independently on 2-labels classes or 3+ labels classes.

One-Versus-All

The simplest approach is to reduce the problem of classifying among k classes into k binary problems, where each problem discriminates a given class from the other $k-1$. For this approach, are needed k different classifiers. For each class, it is trained a binary classifier with the entire training set, where positive samples are the one belonging to that class, while the negative ones are all the others. This approach is summarized in Algorithm 2.4.

Algorithm 2.4 One-Versus-All

1: **input**:
2:     training set $\mathcal{S} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$
3:     algorithm for binary classification A
4: **for each** $i \in Y$
5:     let $S_i = ((\mathbf{w}_1, (-1)^{y_i \neq i}), \ldots, (\mathbf{w}_n, (-1)^{y_i \neq i}))$
6:     let $h_i = A(S_i)$
7: **output**:
8:     the multiclass hypothesis defined by $h(\mathbf{x}) = argmax_{i \in \mathcal{Y}} h_i(\mathbf{x}))$

One-Versus-One

In this approach, each class is compared to each other class. For each pair of classes, a binary classifier is built to discriminate between them, and ignoring all others. Saying k the total number of classes, this method require $\binom{k}{2} = \frac{k(k-1)}{2}$ total binary classifiers. When voting a new instance, it is performed a voting for each classifier, and the class with maximum number of votes is the actual one. This approach is summarized in Algorithm 2.5.

Algorithm 2.5 One-Versus-One

1: **input**:
2:     training set $\mathcal{S} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$
3:     algorithm for binary classification A
4: **for each** $i, j \in Y, i < j$
5:     initialize $S_{i,j}$ to be an empty sequence
6:     **for** $t = 1, \ldots, m$
7:         **if** $y_t = i$ add $(\mathbf{x}_t, 1)$ to $S_{i,j}$
8:         **if** $y_t = j$ add $(\mathbf{x}_t, -1)$ to $S_{i,j}$
9:     let $h_{i,j} = A(S_{i,j})$
10: **output**:
11:     $h(\mathbf{x}) \in argmax_{i \in \mathcal{Y}} \left(\sum_{j \in \mathcal{Y}} sign(j - i) h_{i,j}(\mathbf{x}) \right)$

2.3.13 CLASSIFICATION METRICS

Machine learning classification performance is related to state how well a classifier, that implements a specific machine learning algorithm, makes a correct discrimination between classes. The most basic stated problem consists in binary classification, so most of the metrics are based on this task, and then extended to multiclass problems.

It is important to introduce some terminology about the four types of direct results in binary classification, namely True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN) [43]. These values are commonly displayed in 2 rows by 2 columns of the so called *confusion matrix*, as shown in Figure 2.16.

These four direct outputs of the classifier are also called "base measures", and the meaning is the following: "True" classification results (TP and TN) are the instances that the classifier discriminated correctly, and are located in the diagonal of the confusion matrix, whereas "False" classification results (FP and FN) are the mistakes of the classifier and are displayed in off-diagonal of the confusion matrix. False Positives instances consist on instances that are classified as "positive", but the correct label was "negative", while False Negatives samples consist on in instances that are classified as "negative", but the correct label was "positive". For what seen, it is mandatory to define a "positive" class for measuring a classification metrics on a real world problem, and it depend on the goal of the

Figure 2.16: Confusion matrix for binary classification metrics.

task.

In some problems, False Negatives could be more serious than False Positives, for instance in medical fields it means that a disease is not recognized. In this case it is better to classify positively a disease that is not "malign", instead of ignore it. In the following sections, are explained some performance metrics, derived from the base measures, that have been exploited in this work.

- *Accuracy* is the ratio of total number of correctly classified instances, over the size of the sample set, formally

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

It is the most provided, and the most abused metric in binary classification. Using accuracy as a benchmark measurement has some limitations, in fact it can lead to suboptimal solutions, especially when dealing to imbalance class distributions. Another limitation pf accuracy is that it prodices less distinctive and less discriminable values, consequently it leads to less discriminating power to accuracy in selecting the optimal classifier. In addition, accuracy also lacks in informativeness about minority class instances [44].

- *Precision* is used to measure the positive instances that are correctly classi-

fied from the total predicted ones in positive class, formally

$$Precision = \frac{TP}{TP + FP}$$

- *Recall* is used to measure the fraction of positive samples that are correctly classified, formally

$$Recall = \frac{TP}{TP + FN}$$

- *F-Score* represent the harmonic mean between precision and recall, formally

$$F - score = \frac{2 \times precision \times recall}{precision + recall}$$

To generalize the presented classification metrics from binary to multiclass there are two average methods. Saying $\mathcal{Y} = y_j : j = 1, \ldots, q$ the list of all possible labels, $EM(TP, TN, FP, FN)$ an evaluation metric that is calculated on the number of TP, TN, FP, FN, tp_y, tn_y, fp_y, fn_y the number of classification results with respect to a label y, the average metrics are:

- The *macro-average* is calculated as

$$EM_{macro} = \frac{1}{q} \sum_{y \in \mathcal{Y}} EM(tp_y, tn_y, fp_y, fn_y)$$

- The *micro-average* is calculated as

$$EM_{micro} = EM \left(\sum_{y \in \mathcal{Y}} tp_y, \sum_{y \in \mathcal{Y}} tn_y, \sum_{y \in \mathcal{Y}} fp_y, \sum_{y \in \mathcal{Y}} fn_y, \right)$$

There is no rule about what average score is better. Micro-averaged score may be misleading, because more frequent labels are weighted heavier, and thus in favor of macro-averaged, but in the same way, sometimes it is needed that labels need to be counted proportionally with their influence, so preferring micro-averaged. It depends on the application.

2.4 Sentiment Analysis Algorithms

Sentiment Analysis is a widely studied field of Natural Language Processing, so many algorithms have been proposed in order to achieve the best performance possible. However, the algorithms usually are focused in one type of documents (mostly Twitter's comments), and sometimes in one domain of interest, for instance pharmaceutical.

In this section a sentiment analysis algorithm, that has been taken in some parts for this work, is presented, along with some other works in contest of forum sentiment analysis.

2.4.1 BPEF

Bootstrap Ensemble Framework (BPEF) [45] is a text mining framework for Twitter Sentiment Analysiss. It is thought to overcome some well known issues on TSA, which are high class imbalance and representational richness issues. In [1] it is shown that in experimental results this approach reaches the best performance in its prediction across sentiment classes, as compared to various comparison tools and algorithms. Consequently, BPEF is able to build strong class predictions, from here the choice of implementing some of its parts for this work. In Figure 2.17 the whole procedure of BPEF is summarized.

BPEF is essentially a two-stages process that includes an expansion stages, followed by a contraction stage. By parametrizing the learning procedure, it has been possible to generate a large, diverse set of learned models, that is the expansion stage. After that, the contraction is performed, that is essentially the search of the optimal subset of produced models, which when voted in an ensemble, maximize the overall performance.

As shown in Figure 2.17, expansion is made for three components: datasets, feature and models. At each component, are introduced some parameters which when changed produce newer models. In the following paragraphs are explained the parameters.

48

In addition to the target dataset (the one of interest), some other dataset have been used. It has been produced a category of parameters responsible to mix the additional datasets with the one of interest. By choosing a different combination of datasets it is possible to create a new training dataset which is used to train different models. Datasets have been chosen from similar fields, for instance consumer product reviews. By mixing diverse datasets, it is expanded the number or patterns used to express sentiment polarities, by including instances from different sub-domains. The models learned on instances from mixture of datasets are intended to learn general or subdomain independent sentiment patterns.

Feature Parameters

Different feature types have been adopted as parameter category. Both basic feature types and more generalizing feature variations (used to decrease sparsity on data) have been used as feature parameters.
Feature types consist of unigrams and bigrams of

- Simple words;

- Parts Of Speechs;

- Parts Of Speechs and words combined;

- SentiWordNet features (SWNt), from the need to represent subjective words used scarcely, that may be dropped out during feature selection phase. These words are replaced with their sentiment polarity derived using the SentiWordNet lexical resource [46]. SWNt features are represented as "POS-X", "NEG-X" and "NEU-X", incorporating positive, negative or neutral polarity, where "X" is a positive integer proportional to the strength of the sentiment according to the score derived by aggregating polarity scores across word senses;

- Semantic features (Synset tags derived using WordNet);

On top of these four basic feature groups, it has been applied a feature summarizing technique, like Legomena (replacing once occurring word with a tag) and Named Entity Recognition (NER), in order to reduce sparsity on data.

BPEF includes in its learning stack different classification algorithms commonly used in prior text classification and sentiment analysis algorithms.
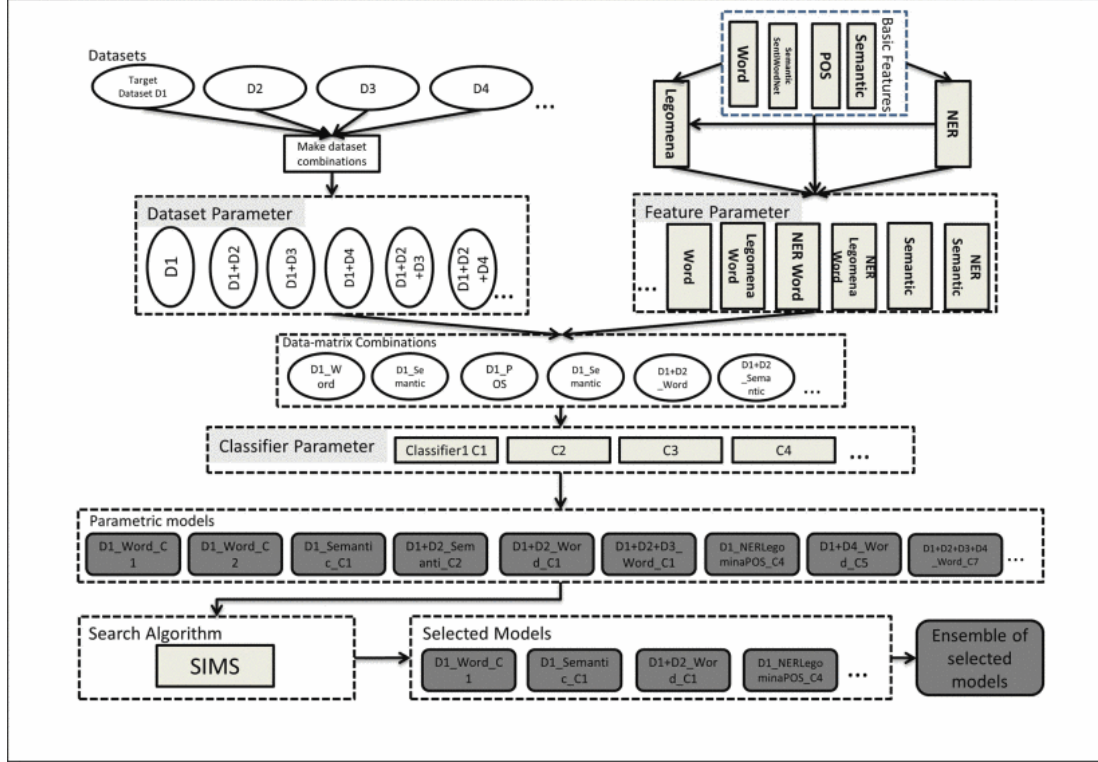
Figure 2.17: Diagram that summarize the BPEF procedure. Taken from [45].

As shown in Figure 2.17, each combination of dataset, feature and classifier, define a new model, so the total learned models are the product of the number of datasets, feature sets and classifiers.

All these models are trained using *bootstrap samples*, which are balanced sub-samples of the training set. This solution may overcome the imbalances issues characteristics of TSA.

Contraction: Step-Wise Model Selection

In expansion stage, a large number of model are generated, while in contraction stage the goal is to improve the performance of the ensemble by keeping only a

subset of the total models, in order to remove redundancy and less useful models. Since there are 2^p possible combination of p parametric models, exhaustive search strategies are infeasible. For this reason, it is proposed a Step-wise Iterative Model Selection (SIMS) approach, which is based on a greedy search, that is more efficient even if it does not guarantee to find the optimal solution.

SIMS is a 2 steps hierarchical application of the Iterative Model Selection (IMS) subroutine. First, IMS is applied on each bootstrap model (which actually is a combination of three binary classifier: Pos-Neu, Pos-Neg, Neu-Neg). IMS (Algorithm 2.6) is used to select a subset of binary classifiers associated to the n bootstrap models comprising $3n$ total binary classifiers. Next, IMS is applied directly on the set of parametric models. The greedy approach searches, at each iteration, the model that gives the best result when combined with the already selected ones. The final SIMS pseudocode is shown in Algorithm 2.7.

Algorithm 2.6 IMS

1: $modelsSelected = []$
2: $parametricModels = [model1, model2, ...]$
3:
4: **input**: $models, modelsSelected$
5: $maxaccuracy = 0$
6: **repeat** until $models$ is empty:
7: $\quad x = argmax_{X \in models}(accuracy(X + modelsSelected))$
8: $\quad accX = accuracy(X + modelsSelected)$
9: **if** $accX > maxaccuracy$
10: $\quad maxaccyracy = accX$
11: $\quad modelsSelected = modelsSelected + x$
12: $\quad models.pop(x)$
$\quad$ **return** $modelsSelected$

Results

In [45], BPEF has been compared with other sentiment analysis tools with respect to recall and accuracy metrics. The final results show that BPEF outperforms the comparison models by a wide margin (16% to 20% higher recall), and also reaches balanced performance across all classes.

It has also been shown the additive contribution of the three component parameters: the study consisted in removing, one at time, all parameters and compared

51

Algorithm 2.7 SIMS

1: $combPerf = []$
2: $binComb = 3$ tuple combination of binary models
3:
4: **for** $B1, B2, B3$ in $BinComb$
5:     $B1Models = \text{IMS}(B1, [])$
6:     $B21Models = \text{IMS}(B1, B1Models)$
7:     $B321Models = \text{IMS}(B1, B21Models)$
8:     $combPerf.\text{append}(B321)$
9: $chosenComb = argmax_{X \in combPerf} \text{accuracy}(X)$
10: $\text{IMS}(chosenComb, [])$

the performance against the SIMS applied on all parametric models. For instance, in order to evaluate the effectiveness of the dataset parameters, those are excluded. The results show that the greatest impact comes from the classifiers parameters, while features parameters have just a shallow impact on the overall performance.

2.4.2 NRC

National Research Council (NRC) developed a classifier for sentiment classification of short messages such as tweets or SMS [47]. In [1] the model was one of the best ones with respect to the accuracy made on several tests. The model is based on a SVM classifier, since its effectiveness on text categorization and robustness on large feature spaces. The algorithm is actually simple, since it is based only on one classifier, but it is the choice of the features that defines the model.
The preprocessing consisted in the normalization of the URLs to "http://someurl", and user mentions to "@someuser". Comments were POS tagged and then tokenized.

2.4.3 ALGORITHMS COMPARISON

2.4.4 RELATED WORKS

Sentiment analysis have been utilized in multiple fields in order to investigate people's opinion. In [48], sentiment analysis has been applied in the medical field, in order to obtain relations between positive and negative opinions and patients' health, more precisely on hearing loss. The study had two stages: the first

consisted on manual annotation of web forums' comment, and the last was the actual classification adopting machine learning techniques. The retrieved dataset consisted in 3515 sentences from 26 threads, that have been extracted from 607 individual posts using regular expression tools. Then, the sentences have been annotated in the three degrees of polarity "positive", "neutral" / "unknown" and "negative" by two manual annotators. All the sentences that did not contain sentiment information have been removed, in fact most of the sentences have been actually discarded, but considering just interested ones, classes resulted balanced, reducing biases to the classifier. It has been also considered the quality of the classification by look for a score that evaluated the agreement of the two annotators, which gives good results, so data may be considered reliable. Classification involves, along with Bag of Words, other few features extracted from each sentence, that are the number of: words with strong subjective (involving a resource like SentiWordNet), words with weak subjective, adjectives, adverbs, pronouns, words having positive, negative and neutral polarities, phrases containing pronouns. Final classification experiments involved machine learning algorithms such as Naïve Bayes, SVM and Logistic Regression. Final results show the best F-score obtained in multiple experiments of 0.685, reached using the Logistic Regression classifier.

In [49] sentiment analysis has been applied in the context of finance, in order to predict stock prices, in fact it has been shown that sentiment expressed in financial forums is correlated to stock prices. Data has been taken from known financial website, that after a filtering on useful information, has been manually annotated in order to apply machine learning classification algorithms. Finally, after tested classification algorithms, it has been study the correlation between sentiment and stock prices. In this case the temporal flow of reviews is relevant for the final outcomes.

In [50] have been tested three different machine learning algorithms for sentiment analysis on Twitter messages in context of electronic products reviews. It follows the typical approach on machine learning text classification: retrieval of tweets exploiting Tweet's API, and classification after preprocessing and feature vector generation. Classification has been made using Naïve Bayes, SVM and Maximum Entropy classifiers, and finally an ensemble of the three methods. All classifiers had almost similar accuracy.

There are lot of works about machine learning techniques on sentiment analysis. In the next chapters is presented a solution for sentiment analysis on Italian automotive forums.

3

Dataset

In this chapter it will be presented the dataset that has been utilized for this work. As previously mentioned, the main focus is asked to be Italian automotive forums, so the first phase consisted in the search of reliable resources from which retrieve the needed information. Information retrieval has been made by crawling the founded web forums. After getting all the information, the final phase for what concerns the dataset consists on annotate all the forums' comments on the base of diverse topics.

In the following sections the whole procedure has been presented in detail.

3.1 Dataset Retrieval

Sentiment analysis is a practical natural language processing problem, most commonly faced involving machine learning techniques. As discussed in detail in Chapter 2, machine learning algorithms need a set of labeled data that represent the environment where the algorithm is supposed to work on. As good sampled data represent the true data distribution, as good the model should work on the real environment, so the dataset retrieval is fundamental for the final outcomes. This work is focused on Italian automotive forums, so the first important phase is to identify some reliable resources where to extrapolate the needed information.

3.1.1 Forums' List

Web forums are websites (or sections of websites) that allow visitors to communicate with each other by posting messages. Most forums allow anonymously visitors to view forum contents, except for registration-required ones. Forums are available for all kinds of topics, from software to health, and like in this case, automotive. Forums are composed by user-generated content, so they are full of users' opinions then, as just mentioned, they are good environments to acquire datasets for sentiment analysis. Forums are commonly subdivided in "areas" or "rooms" which are thematic macro partitions. Rooms are again subdivided into "sections", which are discussions' categories belonging to the same thematic area. Finally, sections contain numerous "threads", which are the actual discussions.

For this work, target resources are identified in Italian automotive web forums or blogs with possibility of discussion, with an active community. The identified resources are: Quattroruote(`https://forum.quattroruote.it/`), Autopareri (`https://www.autopareri.com/forum`), Bmwpassion (`https://www.bmwpassion.com/forum/`), Porschemania (`http://www.porschemania.it/discus/`) and HD-motori (`http://www.hdmotori.it/`). Some statistics are summarized in Table 3.1.

All resources except for Porschemania are brand independent, which means that are treated discussions about every automotive brands independently, while the last one is focused on Porsche * discussions.

The dataset will be composed as a list of comments, so the idea is to download as mush information as possible, that is common to each resource, in order to represent precisely the environment in a structured way like a Comma Separated Values (CSV) file.

3.1.2 Comments' Information

Forums can be seen as a temporal ordered list of comments grouped into discussion. In this way the core part of representing discussions passes through the comment representation. Actually, it is irrelevant the way on which represent the

---

*Porsche is a German sports car manufacturer located in Zuffenhausen in Stuttgart, founded in 1931 by Ferdinand Porsche.

Resource	Type	# Threads	# Messages	# Users
Quattroruote	Forum	121.366	2.413.452	70.707
Autopareri	Forum	n.a	2.146.532	34.963
Bmwpassion	Forum	349.259	7.909.297	78.608
Porschemania	Forum	n.a	n.a	n.a
HDmotori	Blog	n.a.	n.a.	n.a.

Table 3.1: Forums' statistics updated in date 15/08/2019.

information, but it is fundamental decide what information to keep. An idea is to keep all possible information common to comments of all resources, in order to even all entries in a standard way. In Figures 3.1 3.2 3.3 3.4 3.5 are shown one comment for each of the selected resources, with highlighted the information that have been decided to be extracted.

The highlighted information are summarized in the following items. The letter is the reference of the square in the image (not every item is shown because of page dimension):

- **Text (A)**: the text of the comment. It is the most important information about the comment;

- **Quote (B)**: some comments may reference another user's comment in the discussion. This citation is an important information because it gives context about the text of the entry;

- **Topic's Title (C)**: the main title of the discussion;

- **Author (D)**: the username of the user that wrote the comment;

- **Comment's Timestamp (E)**: all comments contain temporal information about when they are written.

Figure 3.1: Sample comment in Quattroruote's forum
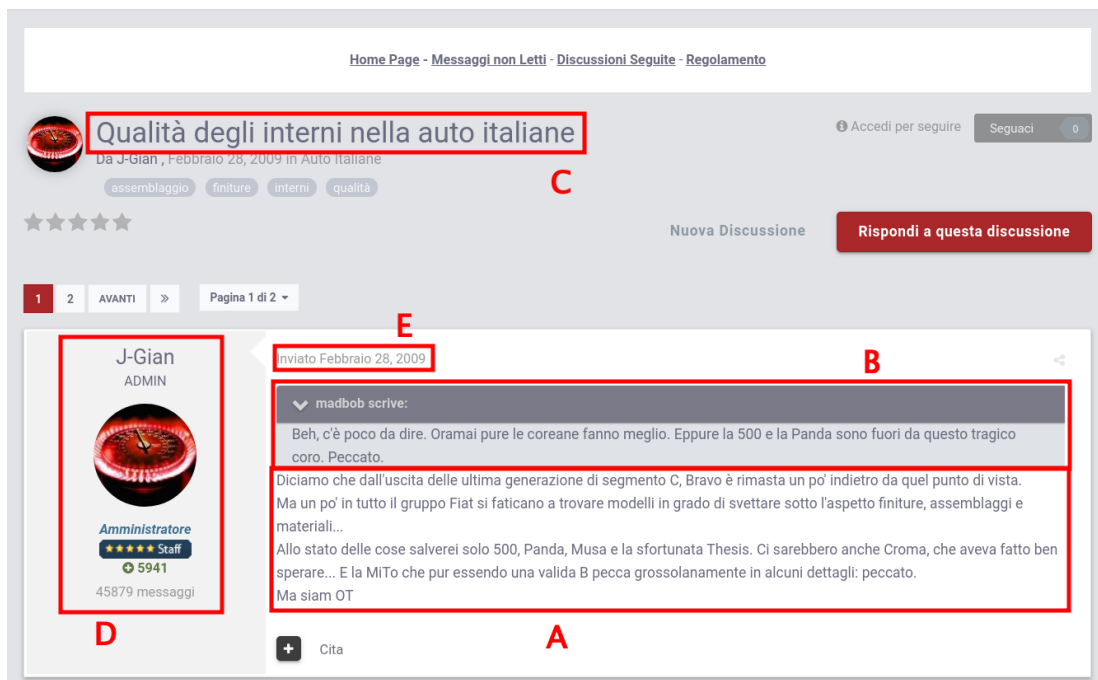
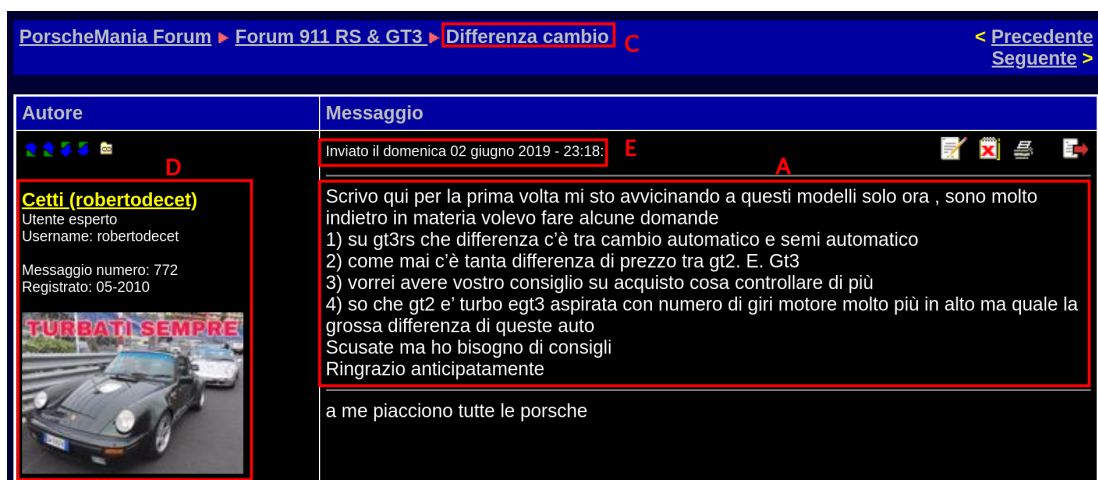**Figure 3.2:** Sample comment in Autopareri's forum

Figure 3.3: Sample comment in Porschemania's forum

3.1.3 Information Gathering

All information needed are viewed in web pages, but they are not accessible from any API or some other data storage interface. For this reason the only solution is to crawl all discussions' web pages and extract the information we need by

Figure 3.4: Sample comment in Bmwpassion's forum

parsing the Hyper-Text Markup Language (HTML) file (HTML scraping). The following paragraphs explain all information and processes used to obtain the desired data explained earlier.

HTML STRUCTURE

Web pages are written in HTML, that is a markup language designed by Tim Berners Lee in 1993. It is the standard markup language designed to be displayed in web browsers. HTML was delivered in several version: HTML 2 comes in 1995, HTML 3 in 1997, HTML 4 in 1997 and the nowadays version HTML 5 in 2014. As a markup language it is composed by textual information tagged by HTML markups. A markup consists of tags which appear inside angled brackets like "< tag >", that give information about how text should be formatted in a browser. Most of the HTML elements are composed by two actual tags: an opening tag <tag> and a closing tag </tag>. Each element may contain other elements and so on, making HTML structured as a tree. Although several HTML markups have been introduced, a correct HTML document must always include certain tags following the standard structure:

Figure 3.5: Sample comment in HDmotori's blog

```html
<!DOCTYPE html>
<html>
    <head>
        <title>This is a title</title>
    </head>
    <body>
        <p>Hello world!</p>
    </body>
</html>
```

The web page content is included into the <body> tag.

Some commonly used HTML tags are the followings (in alphabetical order):

- <!– –>: used to define a comment that is ignored by the browser;

- <!DOCTYPE html>: tells the browser what kind of document it is loaded;

- <a>: used to define an anchor (or hyperlink);

- <article>: used to contain blog entries;

- <b>: used to formatting the text bold;

- <blockquote>: used for quoting an external source;

- <body>: defined the body of the HTML document;

-
: single line break;

- <button>: defines a button that can be clicked;

- <caption>: used to define the table's caption;

- <div>: defines a container;

- <dl>: description list, items into <dd> or <dl> tags;

- <form>: forms for user inputs;

- <h1>, <h2>, ... , <h6>: defines headings. Lower is the number, higher is the level of the heading;

- <head>: head section of the document, used for metadata;

- <header>: contains content to be displayed on top of the body;

- <html>: the root of the document. All tags must be inside this tag;

- <img>: displays an image;

- <input>: used to create forms input elements;

- : ordered list, items in to <li>;

- <style>: used for declaring inline style;

- : unordered list, items into <li>.

Clearly, an HTML document is a composition of all these (and other) tags, that when put together, create into the web browser a readable webpage that displays some content. Moreover, all (or almost all) HTML tags can have attributes that provide additional information about elements, specifying those in the format attribute="value" inside the start tag. This is used, for instance, in <a> tag to specify the hyperlink reference, such as:

This is a link

Important attributes are "id" and "class". The value of the attribute "id" must be unique on the entire document, while more elements can have the same value of the attribute "class". These attributes are used as selectors, for instance for defining the style or the behavior, but since it is not relevant for what concerns the information retrieval strategy, it will be not covered in detail.

HTML PARSER

HTML documents, for their nature of tree structure, can be accessible by Document Object Model (DOM) API. DOM is a platform-independent and language-independent object model for HTML (and other markup languages) documents. It allows programs and scripts to dynamically access and update the content, structure and style of the document [51]. It describes the objects (identified as nodes) that compose the entire document structure and the interfaces to be accessed and manipulated by programs. Using DOM is possible to read, create and modify HTML documents dynamically. An example of DOM of a simple HTML document is shown in Figure 3.6.

DOM implementations create an in-memory representation of the HTML tree, where each element with its attributes is represented with a node. Each node has children nodes that represent the inner elements of the parent, and so on. Starting from the initial <html> tag, referenced as the root of the tree, it is possible to build the entire structure.

In Python language, "Beautiful Soup" (https://www.crummy.com/software/BeautifulSoup/) is a library that implements the DOM interface for HTML documents. It provides a simple interface to parse a HTML document, in order
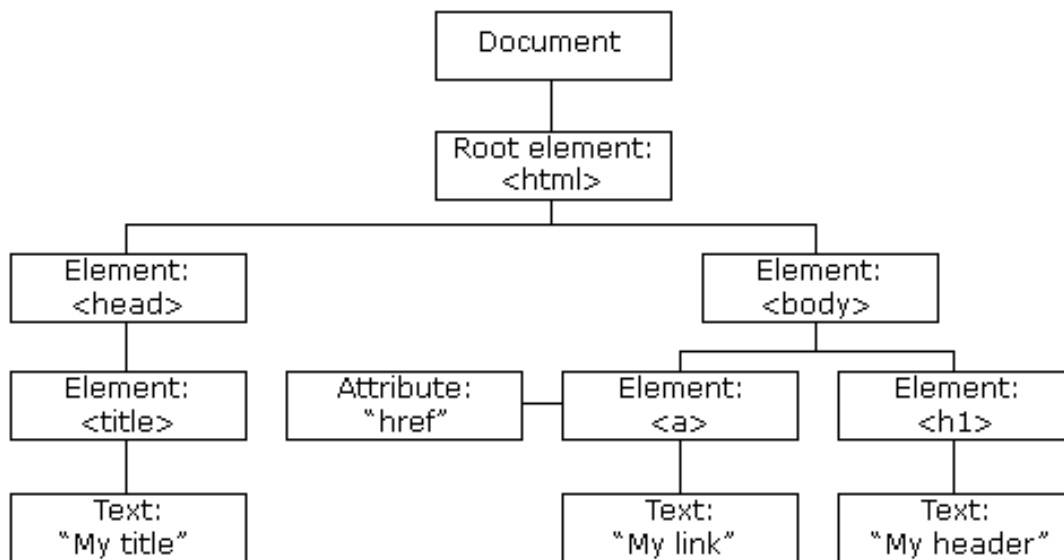
Figure 3.6: Graphical representation of an HTML DOM

to get the needed information. Since for retrieving forums' information there are not available resources, the only solution is to parse HTML web pages, understand where the needed information are located (which actually means find the exact tags where they stand), and finally get them using the given API. Since real web pages have a complicate structure, it is helpful to exploiting browsers' inspector tools. With Beautiful Soup it is possible to query tags by name, by relationship with other tags (for instance, get all children of a given tag), or by attribute. An example of query that retrieves all <p> tags from an HTML document is shown in Listing 3.1.

Listing 3.1: Example of a query in a HTML document with BeautifulSoup

```
from bs4 import BeautifulSoup

# suppose html is a HTML document
soup = BeautifulSoup(html, 'html.parser')
soup.find_all('p')

# it returns a list of <p> tags, like
# [<p>Hi, my name is</p>, <p>John</p>, <p>Live in SF</p>]
```

64

Crawling is the process of automatically browsing the World Wide Web (or an interested part of it) using some programmed scripts called *crawlers*. This process is commonly used by search engines form indexing web pages.

World Wide Web, or in a lower scale a single website, can be viewed as a directed graph where every page contain links to other pages. While viewing web as a directed graph, web pages can be considered as nodes, and hyperlinks can be considered as edges. In this way it is possible to summarize search operations (and obviously also web crawling) as traversing a directed graph [52]. Web crawlers are designed to automatically retrieve information from visited pages.

The working of web crawlers starts with an initial set of URLs, known as *seeds*, then it downloads all these web pages and extracts new links present in the page, eventually filtering out some irrelevant ones. The downloaded pages are then processed and all extracted information are stored. This process, summarized in Figure 3.7, continues recursively until there are no more URLs to be visited.
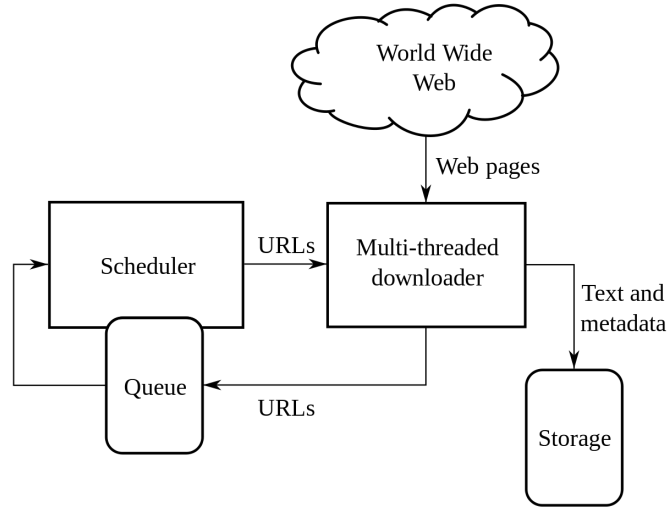
Figure 3.7: Web Crawling process

In this work, for simplicity, have been implemented two different web crawlers for each forum: the first one is supposed to extract all base discussion links starting from a set of manually selected seeds, which are the main pages of thematic areas, while the second is asked to crawl every entire discussion, starting from the previously found URLs. The main structures of the two crawlers are shown in

Algorithms 3.1 3.2, which actually implement a sort of Deep First Search (DFS) traversal algorithm. Page downloading is performed using the "urllib" Python library (https://docs.python.org/3/library/urllib.html).

Algorithm 3.1 Crawler for discussions' base links retrieval

1: **initialize** seeds =[set of thematic areas' links], urls = []
2:
3: **while** seeds is not empty
4:     url = seeds.pop()
5:     next = "next page link"
6:     **for** discussion link d in current page
7:         urls.enqueue(d)
8:     seeds.push(next)
9: **return** urls

Algorithm 3.2 Discussions' Crawler

1: **initialize** comments = []
2:
3: seeds = result from Algorithm 3.1
4: **while** seeds is not empty
5:     next = "next page link"
6:     **for** comment c in current page
7:         comments.enqueue(c)
8:     seeds.push(next)
9: **return** comments

A figurative example, taking in account a sample thematic area from Quattroruote forum is shown in Figure 3.8. In Figure 3.8a it is highlighted a manually selected seed, that points to the list of discussions shown in Figure 3.8b, where the actual discussions' base links are the one highlighted. It is also shown the "next page link" used for reaching all pages of the discussions' list. A discussion's base link points to a page like the one shown in Figure 3.8c, where it is again visible the "next page link" (since also discussions are extended in multiple pages", and obviously the list of comments. Together with comments' crawling it is also done some simple preprocessing using regular expressions, in order to remove some residual HTML tags, and to format dates in a standard way.

(a) Seed link corresponding to a base thematic area.

(b) Base discussions link from previous seed link.

(c) Discussion page.

Figure 3.8: Figurative example of forum crawling.

All information are then stored in a CSV file in order to make simple the final annotation. Some numbers about the crawling process are shown in Table 3.2.

Resource	# Seeds	# Discussions	# Comments
Quattroruote	57	37.721	181.065
Autopareri	51	15.803	552.364
Bmwpassion	5	4.011	75.482
Porschemania	10	14.590	312.222
Forumelettrico	1	139	2.471
HDmotori	1	100	705
TOTAL	125	72.364	1.124.309

Table 3.2: Numbers about crawling process.

Manufacturer-free resources such as Quattroruote and Autopareri have been initialized with a seeds list made of manufacturers' base discussions page, while other resources such as Bmwpassion and HDmotori have been initialized searching for discussions related to Porsche.

As predictable, due to his blog nature, HDmotori is not discussion addicted, in fact it was possible to download just few comments.

3.2 Dataset Annotation and Statistics

The result of the crawling process is a CSV file containing 1.124.309 comments. Some samples are shown in Tables 3.3 3.4 3.5 3.6 3.7 3.8.

Field	Value
Thread	le foto delle nostre Alfa
URL	https://forum.quattroruote.it/threads/le-foto-delle...
Timestamp	2012-05-22 00:00:00
Author	valvonauta_distratto
Quote	La July e il Grigio (meglio tardi che mai...) Saluti
Text	Non si vede un piffero : ?

Table 3.3: Quattroruote's sample comment

Field	Value
Thread	Evo UK test: 599
URL	https://www.autopareri.com/forums/topic/21814-evo...
Timestamp	2007-01-30 20:48:34
Author	poppy84
Quote	
Text	Adesso quanti ne avrebbe la 599... 550 ?

Table 3.4: Autopareri's sample comment

Just seeing few comments is possible to appreciate the heterogeneity of the texts. Some are very tight, like the one in Table 3.4, or some are prolix like the on in Table 3.5. Moreover, in some comments are present citations, while in others not. In general almost all comments are written using a colloquial language, in fact due to the informality of web forums are visible some writing mistakes, bad words and imprecations. Comments are full of technical details about the subject, also brands and models names are omnipresent. All these aspects are relevant, and sometimes fundamental, on sentiment classification.

At this point, the final phase to build the dataset consists on the actual sentiment classification. Sentences may express an unique sentiment polarity, or more generally diverse polarities about different topics. For instance the sentence "My car has awesome suspensions, but the engine is horrible" expresses a positive opinion about the car's mechanics, but a negative stance about the engine. For this

Field	Value
Thread	Dacia Logan MCV 7posti low cost
URL	https://www.bmwpassion.com/forum/threads/dacia...
Timestamp	2004-08-23 00:00:00
Author	mamo
Quote	
Text	Dopo la berlina la station wagon. La gamma della Dacia "Logan" diventa più ampia con una versione più versatile e poliedrica. Merito delle due ante incernierate ai lati che sostituiscono il tradizionale portellone ma anche e soprattutto dell'allestimento con due posti supplementari che rendono la "Logan MCV" (Multi Convivial Vehicle) la più economica "sette posti" sul mercato. D'altra parte i suoi punti di forza sono tutti concentrati nello spazio sia per i passeggeri sia per i bagagli. Grazie al passo allungato di una trentina di centimetri i due posti supplementari sono qualcosa più che due "strapuntini" e il vano posteriore è pensato anche per carichi rilevanti. Si va dai 500 litri della configurazione "cinque passeggeri" ai 2350 di quella "due posti". La meccanica ricalca quella della berlina con la novità di un "benzina" 1600 con otto valvole (90 CV) che affianca il 1400 (75 CV) e il 1600 16V da 105 CV. Unico il turbodiesel che è un "1500 dCi" da 68 CV. I prezzi (Ipt esclusa) partono dagli 8950 euro della "1400" e arrivano ai 12.350 della "1.5 dCi Lauréate". Per le "sette posti" bisogna aggiungere 500 euro con un'unica avvertenza: questo allestimento non è previsto sulle "1400". In arrivo nelle concessionarie italiane a fine gennaio. quattroruote.it

Table 3.5: Bmwpassion's sample comment

reason it is important to catch sentiment position about every topic expressed in comments. From a first look on the forums about the most discussed topics, it comes that usual ones are:

- Brand

- Interiors

- Exteriors

- Mechanics

- Performance

Field	Value
Thread	Consiglio definitivo - premo il bottone rosso?
URL	http://www.porschemania.it/discus/messages/502415/...
Timestamp	2012-12-21 17:36:00
Author	alfi
Quote	
Text	...qualche foto per renderci conto e darti un consiglio piu mirato ?

Table 3.6: Porschemania's sample comment

Field	Value
Thread	Colonnina EVAD Fast Municipio Sant'Ilario d'Enza
URL	https://www.forumelettrico.it/forum/colonnina-enel-fast-a-...
Timestamp	2018-07-16 8:33:00
Author	marco
Quote	
Text	Passato ieri e dopo settimane... la colonnina è ancora spenta e davanti persiste l'installazione del palco. Per cui sposto la discussione in abbandonate e speriamo che la colonnina possa presto venire spostata ed avere maggior fortuna.

Table 3.7: HDmotori's sample comment

- Consumption/Fuel range

- Engine

- Electrical Mobility

- Off-road

- Onboard technology

- Maintenance

- Support/Concessionaires

For each comment it must be specified the sentiment polarity about every identified topic. For annotation it has been chosen a 5-degrees polarity: "very negative", "negative", "neutral", "positive" and "very positive", plus the additional "irrelevant" for topics that have not been mentioned. Moreover, together with the sentiment polarities, it must be also specified, if present and recognized,

Field	Value
Thread	Porsche Taycan: negli USA 3 anni di ricariche rapide incluse fino a 350 kW
URL	http://www.hdmotori.it/2019/01/29/porsche-taycan-3-anni...
Timestamp	2019-01-29 10:46:00
Author	luca bandini
Quote	
Text	arrivano arrivano... piuttosto tesla mi preoccupa perchè tutt'ora non ha piani noti per sostituire la model S che ormai circola da un po' di anni. a livello tecnico è ottima per vendere dovranno modificare l'estetica ?

Table 3.8: HDmotori's sample comment

the manufacturer and the car model, and the overall sentiment of the comment without referring to any topic. Those last annotations have not been used for this work, but maybe in future works they comes useful, and now for free.

The total 1.124.309 crawled comments shuffled and divided into chunks of 250 comments each, in order to make possible the labeling involving crowdsourcing. The main reason of this choice comes from unavoidable deadlines and the need of a adequately large training dataset. Naturally the labeling of all the downloaded comments is unfeasible, so it has been decided a time window of one and an half month, with the purpose of labeling as much comments as possible. In total, have been involved about 20 people (most of them, because of obvious reasons, had labeled just one chunk of comments), to which it was provided a guide where it was specified the way to do the annotation in order to reduce as much as possible misunderstandings. In order to make the labeling simpler, faster and less affected by mispellings, it was provided an Excel sheet where were shown just useful information needed for taking the decision, and the label should be chosen using a dropdown menù. In case of "irrelevant" label, the cell should be left blank, in order to speed up the process. That made the labeling more user friendly than annotating directly on the CSV file. A screenshot of the Excel sheet is shown in Figure 3.9. At the end the total annotations count 7183 entries.

In general, detecting the stance of the user, from a comment, about a certain topic is not easy. Sometimes it is very clear like in this sample:

"Tranquillo il motore va molto bene anche in autostrada. Questa la mia recen-

Figure 3.9: Graphical representation of an HTML DOM

> *sione su S Cross stesso motore: https://forum.quattroruote.it/threads/recensione-s-cross-1-0-cool-2018.119763/#post-2386072"*

where there is a clear evidence on positive polarity about the engine. Contrary, this sample it is not as easy to annotate:

> *"Ciao! presa oggi è uno spettacolo che la metà basta Solo un dubbio non riesco a trovare il modo di condividere la rubrica del mio Galaxy S2 con il coso della macchina dalle istruzioni non è che si capisca un granché bene qualcuno di quelli che l'ha presa ha idea di come si faccia con l'S2?"*

where there is a clear overall positive sentiment, but it is not expressed the topic. For cases like this, it comes the subjectivity of the annotator, for instance the positive sentiment may be referred to the exterior (since the car is "spectacular"). Other cases like this

> *"Per forza la tua Boxster è una giuletta taroccata Boxster!"*

present clearly a negative sentiment, but the topic it is not only unknown, but presumably there is not at all, so the annotation should be "irrelevant" for all categories. Other comments like

"Ciao!!! Fra Punto Evo ed Ibiza la qualità è a favore della seat perchè prati-
camente molti pezzi incluso il motore sono quelli della volkswagen. La
carrozzeria il motore le sospensioni tutta l'elettronica... quindi non posso
che consigliartela ad occhi chiusi. Riguardo ai consumi dubito fortemente
che al primo pieno un 1.3 mjet possa aver fatto 21 km/litro.. come l'ha
rilevato il consumo? Nel caso fosse vero allora sarebbe veramente un'ottima
percorrenza.. Le doti che apprezzo di più della mia ibiza sono la tenuta di
strada il cambio la linea e i consumi."

present comparatives between two entities, in this case cars and engines. For
these situations the rule was to take the decision with respect to the main reach-
able from the text or sometimes from the quote or the topic's title.

Below are shown some statistics about the outcome of the annotation's phase.
Even if the labeling consists in a five-degrees of sentiment polarity, it was reduced
into a three-degrees by grouping "very positive" into "positive" and "very nega-
tive" into "negative". In Table 3.9 are shown the actual number of annotations for
each topic, that are visually represented in Figure 3.10. In Table 3.10 are shown
the percentages of the annotations referred to the total 7183 comments, while
in Table 3.11 are shown the percentages of the annotated sentiments referred to
the total relevant ones (that are all but irrelevant). In Figure 3.11 are shown the
distribution of sentiment polarities with respect to the relevant topics.

As predicable, every class has the majority on "irrelevant" label, which is abso-
lutely right since there are lot of comments that give not information about any
category, and then other comments usually contain just one or a couple of dif-
ferent topics, so in all others they are labeled as "irrelevant". From the statistics
comes some mispredictions, in fact initially the class "offroad" was thought as a
topic that is commonly touched, but actually it is not at all.

From the graphics it is possible to see that the most discussed topics are "brand",
"interiors", "exteriors", "engine" and "maintenance", while the less touched are
"offroad" (as just said) and "electrical mobility". For what concerns "electrical
mobility" topic the point is that it is a new frontier related to innovation, but
forums contain decades of "legacy" mobility, so it is not so strange.

A few interesting observation can be made looking for the bar-charts in Fig-
ure 3.11: recalling what said about unbalances in Twitter Sentiment Analysis'

	Positive	Neutral	Negative	Irrelevant
Brand	651	280	441	5811
Interiors	232	116	142	6693
Exteriors	424	196	246	6317
Mechanics	211	279	221	6472
Performance	258	111	121	6693
Consumption	121	113	87	6862
Engine	317	358	155	6353
Electrical Mobility	31	96	42	7014
Offroad	11	6	4	7162
Technology	117	206	134	6726
Maintenance	110	252	209	6612
Support	108	178	243	6654

Table 3.9: Number of comments for every topic and sentiment polarity.

datasets, here it comes the same issue, in fact most of the topics present visible unbalance, but without an overall preferred polarity. From these statistics it is possible to make supposition about the trends in automotive forums (obviously supposing the dataset is representative): when a brand is mentioned is mostly for praise it, but in general are touched the ends polarities. Moreover, an unsurprising outcomes regards the topics "maintenance" and "support/concessionaires": they show a majority on negatives above positives, and it is explicable since breakages are essentially bad experiences, and so it has repercussions on the comments. And also, if in a car is all fine, presumably there is no motivation to write a comment about maintenance.

What seen is a snapshot of the environment where the sentiment analysis algorithm is supposed to run. The following phase consists on the thinking out a

	% Positive	% Neutral	% Negative	% Irrelevant
Brand	9,06%	3,90%	6,14%	80,90%
Interiors	3,23%	1,61%	1,98%	93,18%
Exteriors	5,90%	2,73%	3,42%	87,95%
Mechanics	2,94%	3,88%	3,08%	90,10%
Performance	3,59%	1,55%	1,68%	93,18%
Consumption	1,68%	1,57%	1,22%	95,53%
Engine	4,41%	4,98%	2,16%	88,45%
Electrical Mobility	0,43%	1,34%	0,58%	97,65%
Offroad	0,15%	0,08%	0,06%	99,71%
Technology	1,63%	2,87%	1,87%	93,63%
Maintenance	1,53%	3,51%	2,91%	92,05%
Support	1,50%	2,48%	3,38%	92,64%

Table 3.10: Percentage distribution of comments for every topic and sentiment polarity.

domain specific sentiment analysis algorithm.

	# Relevant	% Positive	% Neutral	% Negative
Brand	1372	47,45%	20,41%	32,14%
Interiors	490	47,35%	23,67%	28,98%
Exteriors	866	48,96%	22,63%	28,41%
Mechanics	711	29,68%	39,24%	31,08%
Performance	490	52,65%	22,65%	24,70%
Consumption	321	37,70%	35,20%	27,10%
Engine	830	38,20%	43,13%	18,67%
Electrical Mobility	169	18,35%	56,80%	24,85%
Offroad	21	52,38%	28,57%	19,05%
Technology	457	25,60%	45,08%	29,32%
Maintenance	571	19,26%	44,14%	36,60%
Support	529	20,42%	33,65%	45,93%

Table 3.11: Percentage distribution of comments for every topic and sentiment polarity.
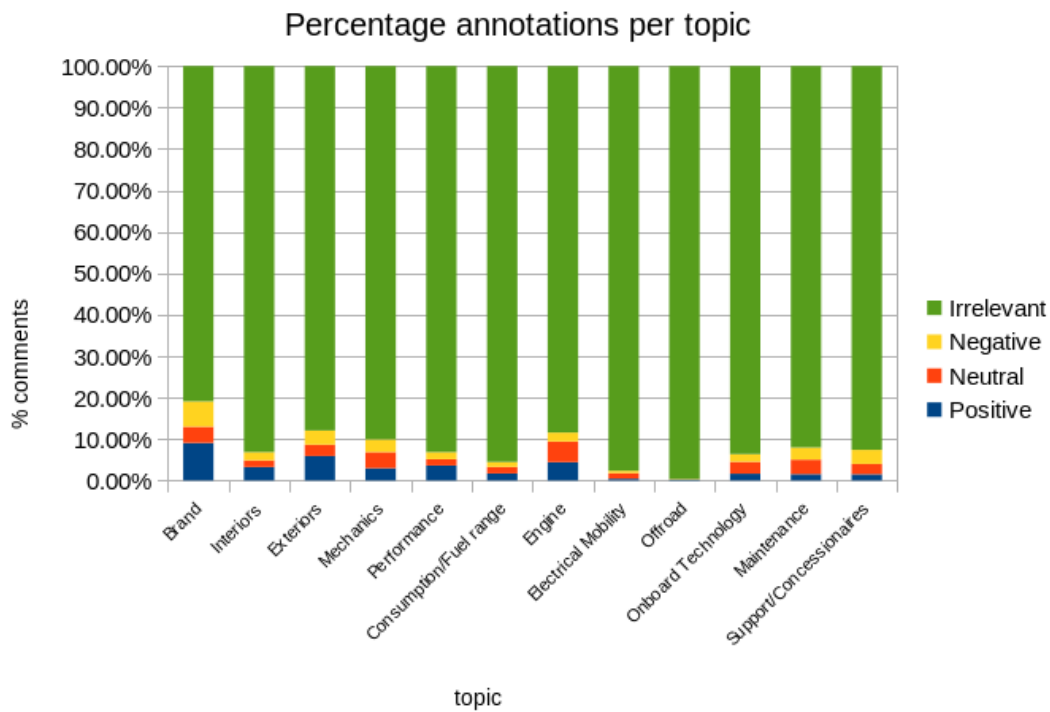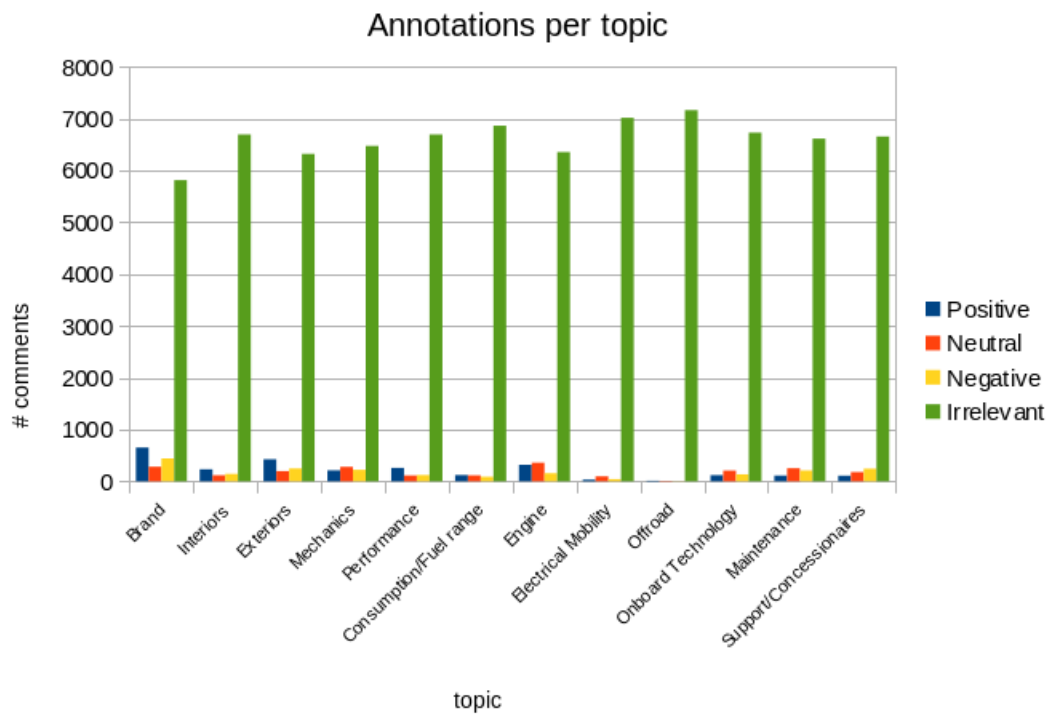
Figure 3.10: Labels' distribution
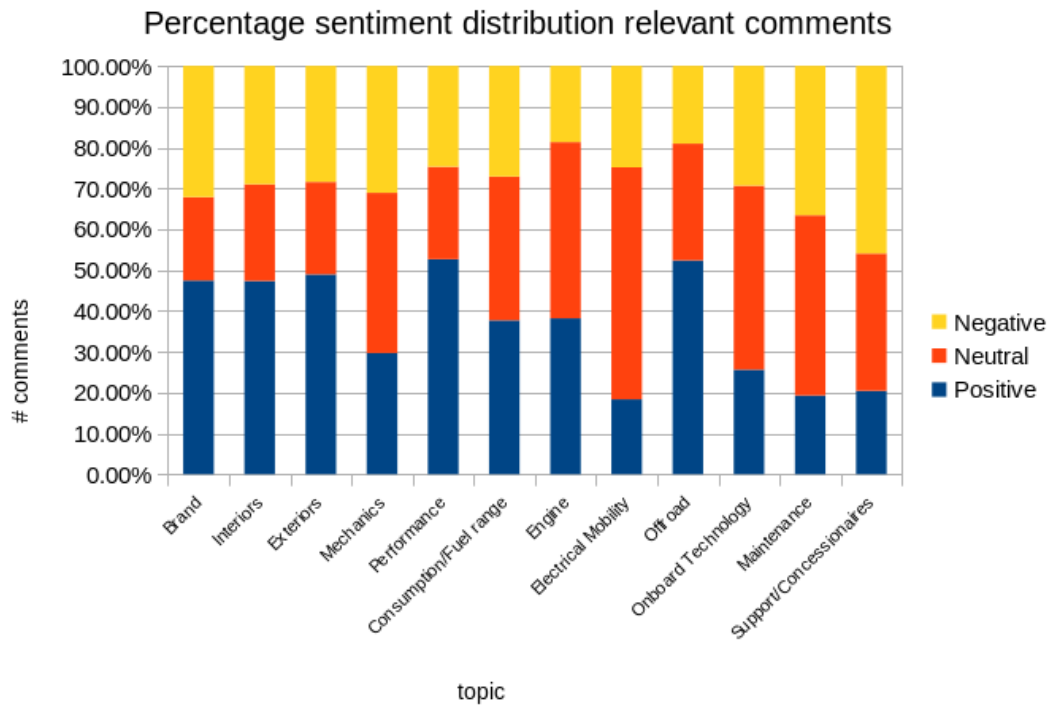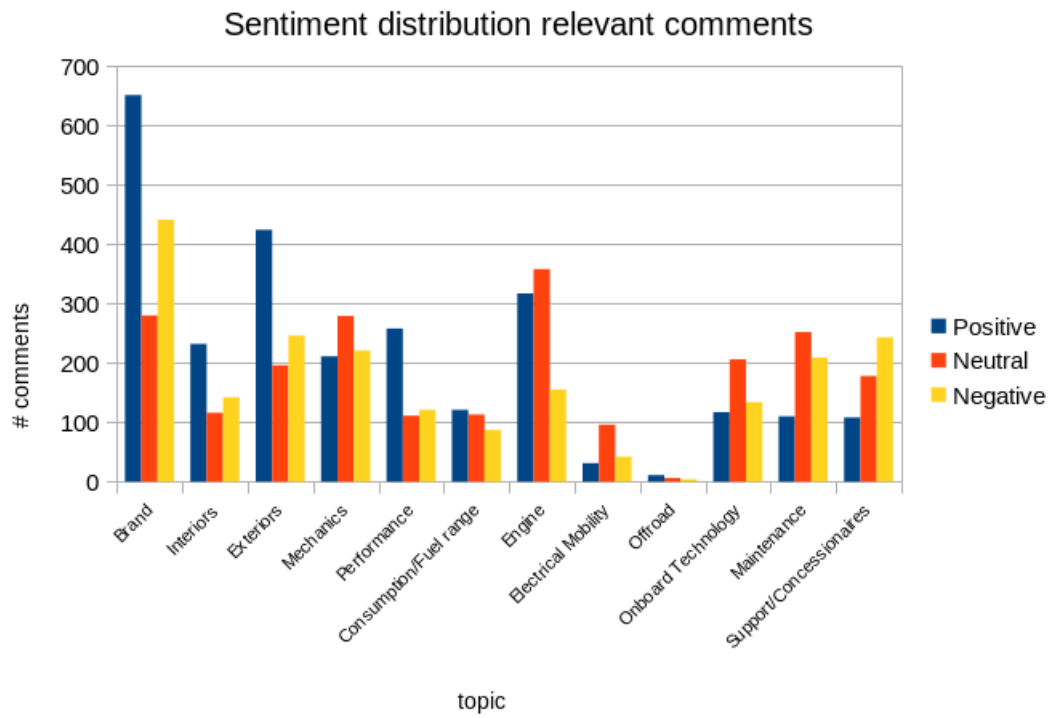
Figure 3.11: Sentiment polarity distribution per topic for relevant ones

4

Algorithms

At this point, the dataset is ready for the algorithms development. Recalling that the dataset is composed by comments labeled according to some identified topics, with a three sentiment polarity degrees, plus an additional relevancy flag, that makes the final problem a four-labels classification task. This problem has been faced in different ways, that will be explained in the Chapter 5. In this Chapter, will be argued the algorithms that will be involved in the classifications and the choices made for the development. The main pipeline remain the same of state of the art text classification problems, that has been presented in Chapter 2, but some implementation choices are needed in order to make the algorithms specific for the automotive domain. Since (almost) same algorithms can be run on different datasets, it has been made a comparison using the Twitter's Airline dataset, and the differences on implementation have been explained later on.

All code has been developed in Python 3.7 (https://www.python.org/downloads/release/python-370/) using the Jupyter Notebook (https://jupyter.org/), that is a web application that allows to create live Python code. The code is available on the repository https://github.com/elRava/master_thesis_sentiment_analysis/.

4.1 SVM and Logistic Regression Classification

The simpler algorithms implemented in this work, follow the state of the art pipeline for text classification. Since the algorithms are supposed to work on very different environments, they present some differences, especially on preprocessing stage. The same algorithm is used for both binary classification for relevance test, and multilabel sentiment classification.

4.1.1 Twitter Preprocessing

Preprocessing for Twitter's dataset handles Twitter's specific tokens and normalize them in order to treat them in a standard way. Some functions are optional in the sense that the preprocessing has been made parametric and it is possible to choose if do them or not. The steps are the followings:

- Transformation text to lowercase;

- Replace two or more dots with space, in order to remove useless punctuation;

- For every token, remove spaces, " and ';

- Optional substitution of the URLs with "URL";

- Optional replacing user mentions (@user) with "USER_MENTION";

- Optional removing "#" from hashtags (#hashtag);

- Optional removing retweet character ("RT");

- Optional grouping emoticons into positive and negative ones and replacing them with "EMO_POS" and "EMO_NEG". Positive emoticons are: :), :), :-), (:, (:, (-:, :'), :D, : D, :-D, xD, x-D, XD, X-D, <3, :*, ;-), ;), ;-D, ;D, (;, (-;, while Negative ones are: :-(, : (, :(,):,)-:, :,(, :'(, :"(;

- Removing residual multiple spaces.

4.1.2 Italian Automotive Dataset Preprocessing

As in Twitter's preprocessing, the goal is to reduce noise by handling domain specific terms. Also in this case the preprocessing has been made parametric. The steps are the followings:

- Encoding correction;

- Punctuation removal;

- Transformation text to lowercase;

- Removal of character repetitions: keep at maximum three consecutive character repeated;

- Replaced question marks and consecutive question marks with respectively "QMARK" and "MULTI_QMARK";

- Replaced exclamation marks and consecutive question marks with respectively "EMARK" and "MULTI_EMARK";

- Replacing URLs with "URL";

- Replacing HTML picture tags with "IMG";

- Stopwords removal;

- Optional replacing brands with "BRAND" and car models with "MODEL": since the classifications are considered brand independent, it comes the idea of replacing all brand names with a common token. The dictionary of all manufacturers and models has been scraped from https://www.auto-data.net, since a complete one has not be found;

- Replacing speed metrics with "SPEED": since the car independent intentions, it does not matter the actual speed values, for instance 150 km/h may be good for a city car, but not so good for a sport car. Moreover, it depends on the context, so the decision of ignore the numerical value. For the same reason also consumption metrics have been replaced with "CONSUMPTION", weights metrics with "WEIGHT" and power metrics with "POWER";

- Replaced distances with "DISTANCE";

- Replaced numbers with three or more digits with "NUMBER".

All operation have been implemented exploiting regular expressions from the re library (https://github.com/python/cpython/blob/3.7/Lib/re.py) and common natural language processing operation from nltk library (https://www.nltk.org/).

After preprocessing, it has been adopted the "snowball" stemmer from nltk library (https://www.nltk.org/_modules/nltk/stem/snowball.html), for normalization purposes.

4.1.3 CLASSIFICATION

Following the preprocessing, it is applied the stemmer. The result of the whole preprocessing phase on a tweet is:

> @united I do not see where it talks about military baggage fees. Can you please guide me. Thanks

> USER_MENTION see talk militari baggag fee pleas guid thank

and on the Italian automotive dataset is:

> Sono reali calcolati nel arco del tutto anno nel estate qualcosa in piÃ¹ causa gomme di 17" e climatizzatore nel inverno un po di meno. Per quanto riguarda le autostrade quelle che percorro io principalmente la A4 e molto congestionata cosi spesso la media e 110-115 km/h che ovviamente influisce positivamente a i consumi. Ma quello che mi piace di piÃ¹ Ã¨ assenza dei guasti. Sulla vecchia Accord il primo guasto lo ho avuto a 200000 km si Ã¨ rotto il termostato della clima. Ogni tanto faccio giro di altri forum e leggo delle turbine rotte catene di distribuzione progettate male iniettori fatti male mah nel 2015 per me sono le cose incomprensibili . Con tutti gli difetti che puÃ² avere preferisco la Honda.

> real calcol arco anno MODEL qualcos caus gomm MODEL climatizz invern po men riguard autostrad percorr principal MODEL molt congestion cos spess med MODEL SPEED ovvi influ posit consum piac assenz guast vecc MODEL prim guast DISTANCE rott termost clim ogni MODEL gir altri forum legg turbin rott caten distribu progett mal iniettor fatt mal mah NUMBER me cos incomprens difett può aver prefer BRAND

For all experiments on the Italian automotive dataset, we firstly focused on just one topic, then the same procedure has been extended to all others. Since this dataset has not just one text, but it contain also the quote, the classification should involve both texts. It has been implemented joining the texts and, in a parametric way, adding a suffix to each word like "_TEXT" or "_QUOTE" weather a word is belonging to the text or to the quote.

For vectorization it has been chosen the TF-IDF method considering both unigrams and bigrams. Classification involved either Support Vector Machine classificator and Logistic Regression classificator for different experiments, both from scikit-learn library (https://scikit-learn.org/).

An important task for model selection consists on hyperparameters optimization. Hyperparameters are model parameters that must be chosen before the learning process begins. In general, different algorithms require different hyperparameters, and some others also none. The search of the optimum parameters has been implemented using a grid search, which requires a proper list of parameters and a list of testing values. It searches among all, the parameters the ones that, when the model is trained with those, reaches the best score. The reference score, of course, must be defined previously and depends on the classification problem. Feature selection has been performed in the same way for both SVM and Logistic Regression. After a first optimization involving all features, found weights of the classifier are ranked with respect to the absolute value. Then, after some tries it is defined an adequate cutoff value which selects the features to keep. From scikit-learn library it is possible to achieve the weights of the binary classifiers that compose the one-versus-one multiclass classifier, so for a three-labels classifier, there are three sets of weights. From the different sets, the goal is to find the most relevant features for all binary classifiers, from here the choice to act in this way: for every set of weights, apply the cutoff to select the most important features in the binary case, and then make the union of all sets of selected features in order to have just one set of features for every classifier.

The final model is obtained by train again the algorithm keeping only the selected features.

4.2 BPEF Revisitation

The second algorithm employed for text classification is a revisitation of the BPEF algorithm described in Chapter 2. It has been chosen this algorithm because of its proved good performance and stability in Twitter sentiment analysis, and above all, the capability to adaptation to slightly different problems. The adaptation procedure will be described in the followings paragraphs.

4.2.1 Parametric Model

As shown in Section 2.4.1, Bootstrap Ensemble Framework is based upon a parametric model that involves dataset, feature and classifier parameters. Every type of parameters had been modified from the original paper and it has been adapted for the problem of this work.

Dataset Parameters

In the original BPEF algorithm, the target dataset is integrate with other dataset of similar domain. The purpose is to enrich the dictionary of common saying and common patterns that are used for sentiment classification in order to make the classifier more stable. Actually, the Italian automotive dataset already contains a comments with a large variety of expressions, and most important, it can be considered also an union of different datasets, in fact it contain comments from very different car brands, that cause variety on subjects and patterns. For instance, suppose that the target classification is a the set of comments of sports cars, but since the dataset contain comments of every type of automotive vehicles, then off-topic ones can be considered as an integration, that in BPEF algorithm consisted in the aggregation of multiple datasets. In conclusion, because of these reasons, dataset parameters have just been removed.

Feature Parameters

AS described in Chapter 2.4.1, BPEF algorithm every sentence is processed in four parallel ways, defined by the feature parameters that are: "word", "POS", "semantic" and "SWNt". For semplicity and lack of Italian resources, in the BPEF revisitation have been implemented "word", "POS" and "SWNt" features. For POS fea-

tures it was exploited Python's "spacy" library (https://spacy.io/models/it) that also supports Italian, while for what concerns words' sentiment polarities, SentiWordNet does not support Italian, so it has been utilized "OpeNER" (https://dspace-clarin-it.ilc.cnr.it/repository/xmlui/handle/20.500.11752/ILC-73) that is a list of words that has been semi-automatically labeled with ItalWordNet v.2, starting from a list of 1000 manually labeled words.

On top of these feature groups, it has been applied the words summarizing that has been presented in Section 4.1.2. Finally, the stemmer is applied for normalization.

An example of the different preprocessings is:

Original : *Sono reali calcolati nel arco del tutto anno nel estate qualcosa in piÃ¹ causa gomme di 17" e climatizzatore nel inverno un po di meno. Per quanto riguarda le autostrade quelle che percorro io principalmente la A4 e molto congestionata cosi spesso la media e 110-115 km/h che ovviamente influisce positivamente a i consumi. Ma quello che mi piace di piÃ¹ Ã¨ assenza dei guasti. Sulla vecchia Accord il primo guasto lo ho avuto a 200000 km si Ã¨ rotto il termostato della clima. Ogni tanto faccio giro di altri forum e leggo delle turbine rotte catene di distribuzione progettate male iniettori fatti male mah nel 2015 per me sono le cose incomprensibili . Con tutti gli difetti che puÃ² avere preferisco la Honda.;*

word : *real calcol arco anno MODEL qualcos caus gomm MODEL climatizz invern po men riguard autostrad percorr principal MODEL molt congestion cos spess med MODEL SPEED ovvi influ posit consum piac assenz guast vecc MODEL prim guast DISTANCE rott termost clim ogni MODEL gir altri forum legg turbin rott caten distribu progett mal iniettor fatt mal mah NUMBER me cos incomprens difett può aver prefer BRAND;*

SWNt : *POS_2 calcolati NEU_2 anno MODEL qualcosa POS_5 gomme MODEL climatizzatore inverno po meno riguarda autostrade percorro principalmente MODEL molto congestionata cosi spesso NEU_2 MODEL SPEED ovviamente influisce POS_10 consumi piace NEG_6 guasti vecchia MODEL POS_6 NEG_5 DISTANCE NEG_5 termostato clima ogni MODEL NEU_2 altri forum leggo turbine rotte catene NEU_5 progettate NEG_6 iniettori*

fatti NEG_6 mah NUMBER me cose incomprensibili difetti può NEG_5 preferisco BRAND;

POS : *ADJ VERB ADJ NOUN PROPN PRON NOUN ADJ PROPN NOUN ADJ VERB ADV VERB NOUN ADJ ADV PROPN ADV VERB ADJ ADV ADJ PROPN PROPN ADV VERB ADV NOUN VERB NOUN NOUN VERB PROPN ADJ NOUN PROPN VERB ADJ NOUN DET PROPN NOUN DET NOUN DET NOUN VERB VERB NOUN VERB ADV NOUN VERB ADV NOUN PROPN PRON NOUN ADJ NOUN AUX VERB NOUN PROPN adj_real verb_calcol adj_arc noun_ann PROPN_MODEL pron_qualc noun_caus adj_gomm PROPN_MODEL noun_climatizz adj_invern verb_p adv_men verb_riguard noun_autostrad adj_percorr adv_principal PROPN_MODEL adv_molt verb_congestion adj_cos adv_spess adj_med PROPN_MODEL PROPN_SPEED adv_ovv verb_influ adv_posit noun_consum verb_piac noun_asst noun_guast verb_vecc PROPN_MODEL adj_prim noun_guast PROPN_DISTANCE verb_rott adj_termost noun_clim det_ogn PROPN_MODEL noun_gir det_altr noun_forum det_legg noun_turbin verb_rott verb_caten noun_distribu verb_progett adv_mal noun_iniettor verb_fatt adv_mal noun_mah PROPN_NUMBER pron_m noun_cos adj_incomprens noun_difett aux_pu verb_av noun_prefer PROPN_BRAND*

Classifier Parameters

In the learning stack are included some commonly used classification algorithms: Support Vector Machines, Logistic Regression, Naïve Bayes and Random Forest, that work in an ensemble, making the model more stable.

Contraction: Step-Wise Model Selection

Original BPEF model essentially consists in an ensemble model, made by a number of classificators that is the product of dataset, features and classification parameters. Since in the original paper this number was excessive, it was applied the Step-wise Iterative Model Selection, that is an heuristic iterative algorithm used for selecting the best subset of classifiers. In this work the number of classifiers is not so high, in fact there are no dataset parameters, 6 feature parameters

("word", "POS" and "SWNt" with and without words summarizing) and 4 classifier parameters, counting a total of 24 models, so the selection is not implemented.

4.2.2 CLASSIFICATION

In BPEF model, feature selection is performed using Information Gain heuristic, and must be necessary applied before classification, since it is based upon probability characteristics of the dataset, rather than classifiers' weights. IG features selection has been implemented using "info-gain" library (`https://pypi.org/project/info-gain/`), that is based on discrete probability distributions, so the vectorization consisted on Term Frequency, instead of TF-IDF, considering both unigrams and bigrams. Feature selection consisted into ranking all features based on their IG, and then selecting a reasonable cutoff value that eliminates the feature with lower IG.

For every classifier it is necessary the hyperparameters optimization process, that has been implemented with a grid search, where each model validation is performed with a 5-folds cross validation, searching for the model that gives the best score.

4.3 CASCADE CLASSIFICATION

A text classifier can be used for all relevance classification, sentiment classification and for the whole four-labels classification. In this way the dual goal of classification, that is the topic detection and sentiment classification, can be faced in two different ways: a one-step classification with a 4-labels classifier (Figure 4.1), or a two-steps one with a cascade classifier. The second option consists in a cascade of two classifiers, where the first one deals with relevance targeting, while the second one with the sentiment classification, as shown in Figure 4.2. This may improve the final performance because both classifiers are thought for different purposes, while having only one model may have difficulties on both topic relevance and sentiment.
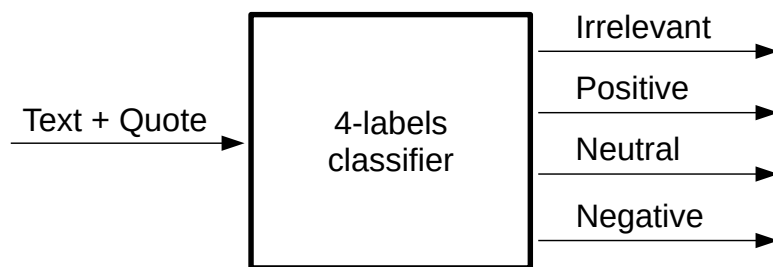
Figure 4.1: 4-labels classifier
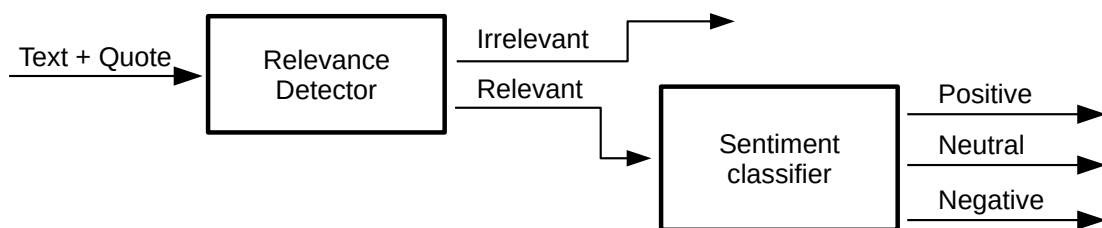
Figure 4.2: Cascade classifier

90

5
Experiments

Summarizing the work done until now, it has been created a sentiment analysis dataset based on Italian automotive forums by crawling web resources and then scraping HTML pages. Then, the dataset has been labeled by many workers, obtaining a sufficiently large amount of data to be processed. The next phase consisted in defining some machine learning models in order to make both topic detection and sentiment classification. In this Chapter, will be firstly presented, as a benchmark, results obtained running the models on a Twitter's dataset, and then the results obtained on the created dataset.

Every test will be presented with a common framework:

- Results obtained with a classification before features selection;

- Features selection and most important features;

- Results obtained after features selection;

Classification results will be presented both visually using confusion matrices and using numerical scores.

Since used algorithms don't require huge amount of computational power, runs have been made on a 2019 consumer technology with the following specifics:

CPU	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 4 cores, 8 threads
RAM	16GB DDR4
O.S.	Ubuntu 18.04

Where it has been installed a Python 3.7 release on a Jupyter Notebook.

5.1 Experiments on Twitter's Airline Dataset

The first series of experiments concerns the Twitter's Airline dataset described in Section 2.1.6. The purpose of these tests is to verify the goodness of the models in a dataset that is plenty of well labeled comments. The reference result on tweets sentiment classification can be extracted from [1], where BPEF algorithm reached the best results with an average accuracy of 71.38% on sentiment classification on five different datasets, but average performance of all models stay around 65%, so a similar result will be positive.

The dataset consists in 14640 labeled tweets, divided into 3099 positives, 2363 neutrals and 9178 negatives. Due to the plenty of data, all classes were balanced, with the number of comments of the minority class. Successively, the dataset were divided into training set and test set, respectively the 80% and the 20%. Moreover, the training set were further divided into actual training set and validation set, again respectively the 80% and the 20%. The distribution of the datasets is:

	Positives	**Nautrals**	**Negatives**	**Total**
Training	960	960	960	2880
Validation	240	240	240	720
Test	300	300	300	900

Due to the class balance, also accuracy score is meaningful, so it is presented along with F1-macro and F1-micro scores.

5.1.1 Sentiment Classification with SVM

After preprocessing and vectorization with TF-IDF method on the training set, involving both unigrams and bigrams, the outcome dimension counts 22.486 features. Data are stored in a sparse matrix 2880x22.486 with 50.006 stored elements, so as expected, it is actually very sparse.

A first model train is performed involving all features, optimizing the regularization parameter C from 7 candidates exponentially equally spaced from 10^{-3} to 10^3, searching for the best F1-macro score.

The selected model is the one with $C=1$, and the classification results obtained with the validation set are:

Predicted value

	Positive	Neutral	Negative
Positive	182	36	22
Neutral	26	163	51
Negative	18	39	183

Actual value

F1-macro	0.734
F1-micro	0.733
Accuracy	0.734

It is possible to see that the baseline method without feature selection reaches good results, aligned with state of the art. From the confusion matrix, diagonal values (which are the correctly classified) are in fact the majority, and also the accuracy of 70,4% reflects the visual considerations.

The sorted weights of the three binary classifiers that constitute the one-versus-one multiclass classifier are shown in Figure 5.1.
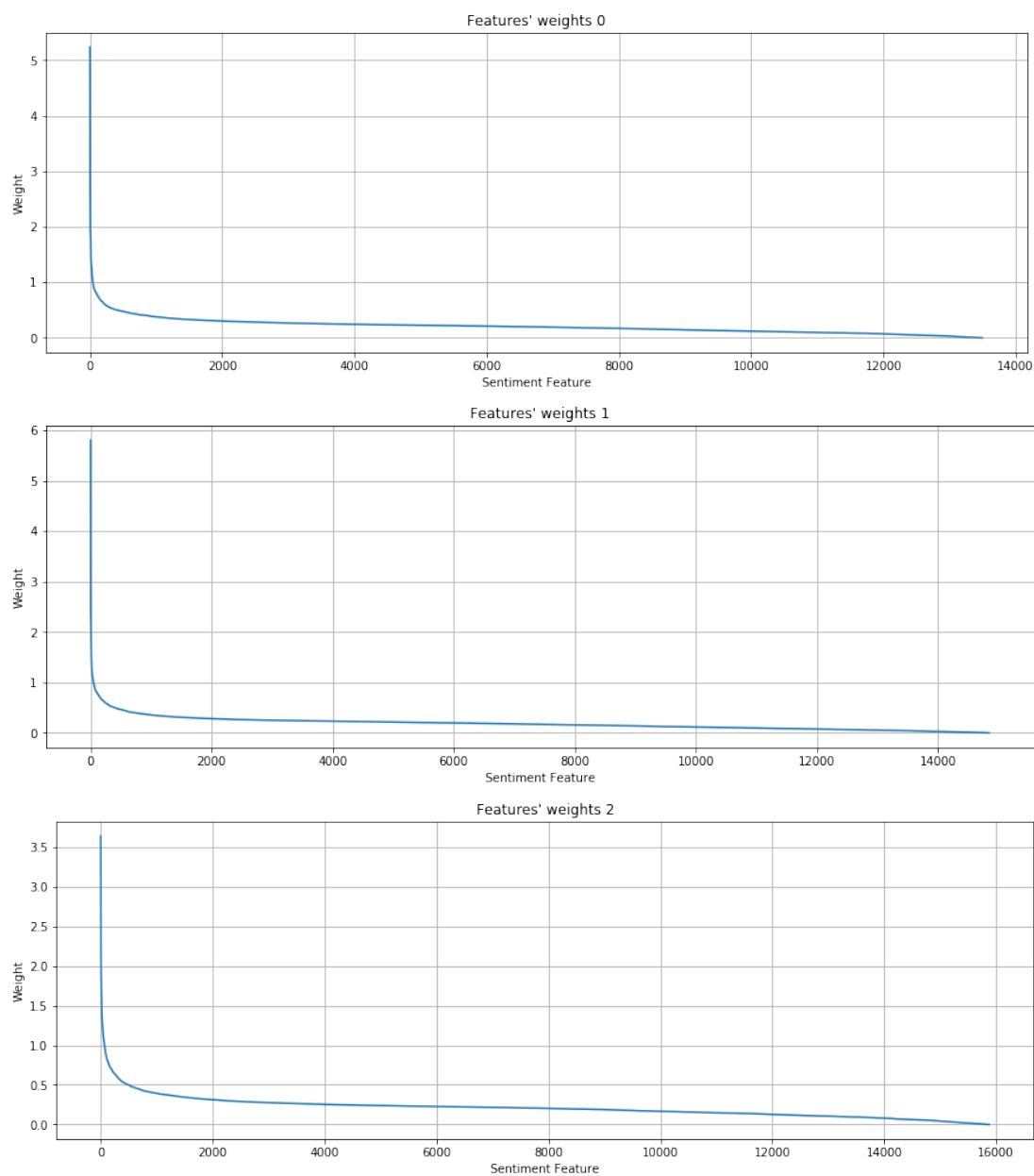
Figure 5.1: Features' weights of the three binary classifiers of the one-versus-one multiclass classifier

After some manual tries, the selected cutoff values are respectively 0.3, 0.3 and 0.2, obtaining 9564 final selected features. Retraining the model using just selected features, the results on the validation set are:

Predicted value

	Positive	Neutral	Negative
Positive	194	33	13
Neutral	50	142	48
Negative	33	36	171

Actual value

F1-macro	0.702
F1-micro	0.704
Accuracy	0.704

After feature selection it is possible to notice a loss of performance due to the removal of some relevant features. However, features selection has the goal to make the model more stable, rather that more performing. The final metrics still highlight performance close to state of the art models.

5.1.2 SENTIMENT CLASSIFICATION WITH REVISED BPEF

At this point the goal is to enhance the previous model with the revised BPEF ensemble. In this model, feature's relevance is calculated using the information gain metric, and since it is based on dataset's distribution instead of model's weights, it is possible to calculate it as the first phase, for every combination of the model's features parameters.

The outcome of features ranking is very similar for all features parameters and has the trend shown in Figure 5.2.
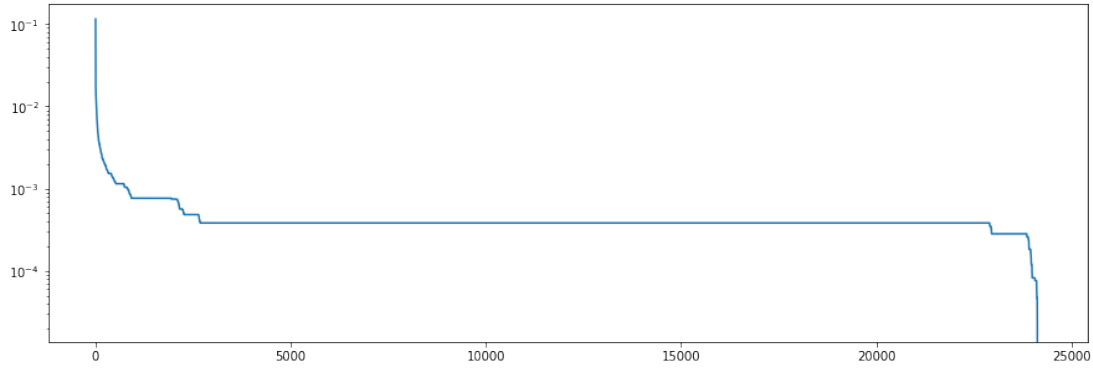
Figure 5.2: Information Gain trend of the features.

It is possible to notice that there are just few features with higher information gain, while most have a flat trend. The strategy for features selection was to set the cutoff value in correspondence of the starting of the flat curve, in order to keep just features with higher information gain.

Before features selection it were counted the following number of features:

Feature name	Word summarizing	# Features
word	false	24.206
word	true	22.485
pos	false	30.282
pos	true	28.318
swnt	false	22.789
swnt	true	21.113

Recalling that the involved algorithms are SVM, Logistic Regression, Raïve Bayes and Random Forest, they are all trained optimizing with a grid search on the followings parameters:

- SVM: C from 10^{-3} to 10^3 with 7 exponentially evenly spaced values;

- Logistic Regression: C from 10^{-3} to 10^3 with 7 exponentially evenly spaced values;

- Random Forest:

  - Number of estimators: [201, 501]

96

- Maximum of looked features on split: [auto, log_2]
- Maximum depth of the tree: [10, 100]
- Split criterion: [gini, entropy]

The classification without features selection on validation data gives the following results:

Predicted value

	Positive	Neutral	Negative
Positive	181	43	16
Neutral	19	180	41
Negative	18	57	165

Actual value

F1-macro	0.732
F1-micro	0.731
Accuracy	0.731

The results are very similar to the ones reached with SVM classificator without features selection. Setting a cutoff value for feature selection equal to 4×10^{-4}, the number of selected features becomes:

Feature name	Word summarizing	# Features
word	false	2.665
word	true	2.622
pos	false	3.181
pos	true	3.057
swnt	false	2.657
swnt	true	2.612

Which are less than one third than the previously case. Training again the model optimizing the same hyperparameters with the same grid search, the obtained results are the followings:

Predicted value

		Positive	Neutral	Negative
Actual value	Positive	190	40	10
	Neutral	20	187	33
	Negative	18	58	164

F1-macro	0.753
F1-micro	0.751
Accuracy	0.751

Looking at the scores, it is easy to notice the better performance of the BPEF revisitation against the SVM model, even considering very less features. Moreover, due to the strong cut of the features and the optimal results, it is possible to suppose that the feature selection method returns an actual set of relevant features, that combined with the stability of the ensemble lead to prefer this model against the previous one.

5.1.3 Sentiment Classification with Test Data

SVM Classifier

Predicted value

	Positive	Neutral	Negative
Positive	235	44	21
Neutral	36	196	68
Negative	27	42	231

Actual value

F1-macro	0.735
F1-micro	0.736
Accuracy	0.736

Revised BPEF Classifier

Predicted value

	Positive	Neutral	Negative
Positive	190	40	10
Neutral	20	187	33
Negative	18	58	164

Actual value

F1-macro	0.734
F1-micro	0.732
Accuracy	0.732

5.2 Experiments on Italian's Automotive Dataset

Now that some results are available, it is possible to state the effectiveness of the implemented algorithms. In this section the same models for sentiment analysis will be executed on the Italian automotive dataset. Moreover, have been designed models for relevance detection that will be implemented for the cascade classifier. For all classification have been involved "Text" and "Quote" columns of the dataset, in the way that was mentioned in the previous chapter: both techniques for texts combinations were tested, and similar but better results were obtained without adding the suffix that subdivides text's words against quote's words.

Relevance Detection

Before sentiment classification, it has been studied the task of relevance detection, where the goal consists on identifying weather a comment talks about a given topic. In the dataset, relevance is identified in the presence of a sentiment expression, whether it is "positive", "negative" or "neutral", while the irrelevance is obviously identified in the "irrelevant" label. The problem is then summarized in a binary text classification problem, that will constitute the first block of the cascade classifier.

Relevance is studied one topic at time, in fact the whole study of the models was made on the class "Engine", and then exported for all others. For the class "Engine" the dataset is constituted of 7183 comments, 866 of which are classified as "relevant", so as said, unbalance is really visible. The dataset is splitted into training data and test data, respectively the 80% and 20%. Then, the training dataset has been further splitted into actual training and validation sets, respectively the 80% and 20%. The distribution of the datasets is:

	Relevant	Irrelevant	Total
Training	532	4.064	4.596
Validation	133	1.017	1.150
Test	166	1.271	1.437

Training of the models has been made again optimizing the hyperparameters with a grid search, maximizing the F1-macro score, which is more informative for this imbalance case, in fact small changes on minority class classification give appreciable differences on the macro score, while considering the micro score, the final output is predominated by the values of the majority class. However, for comparison, if two models obtain similar F1-macro scores, the better choice falls on the one which gives the best recall, which means that it catches better the topic relevance, even returning more false positives.

SENTIMENT CLASSIFICATION

Sentiment classification involves the same techniques used for the same task with the Twitter's Airline dataset presented in the preceding section. Even for sentiment classification the algorithms were tested on the "Engine" class, but just considering the relevant comments. For this reason the dataset became similar to an usual sentiment analysis's dataset. It contains 830 total comments, divided into 317 positives, 358 neutrals and 155 negatives. Also for this phase, the dataset has been divided into training set and test set, respectively the 80% and 20%, and training set has been splitted into actual training set and validation set, again respectively the 80% and 20%. The distribution of the datasets is:

	Positives	Neutrals	Negatives	Total
Training	204	228	100	532
Validation	51	57	25	133
Test	62	73	30	165

Since classes are largely unbalanced, undersampling the dataset for train the classifier on balanced data may lead to underfitting, so the difficulty is to control the bias due to the presence of the large "irrelevant" class.

5.2.1 RELEVANCE DETECTION WITH SVM

The whole dataset is preprocessed involving words summarizing and vectorization with TF-IDF method for unigrams and bigrams. The outcome's dimension count 156.369 features. The number of features is a lot larger than the one seen for twitter's dataset, even in obtained with the same method. This is a consequence

of the scarcity of words and expressions on Twitter's comments, and the opposite on Forums, where texts usually are more articulated. Since the number of features is very large, and the data are just few hundreds, the features selection technique may be essential for improve the stability picking just the features that are actually useful for the classification. All trainings have been made optimizing the parameter C with a grid search on the values from 10^{-3} to 10^{3} with 7 exponentially evenly spaced values. A first training considering all features gives the following results:

Predicted value

		Irrelevant	Relevant
Actual value	Irrelevant	969	48
	Relevant	62	71

F1-macro	0.563
Recall	0.534
Precision	0.597

The best classifier, obtained with C=1000, gives the weights shown in Figure 5.3, that follow the usual trend.
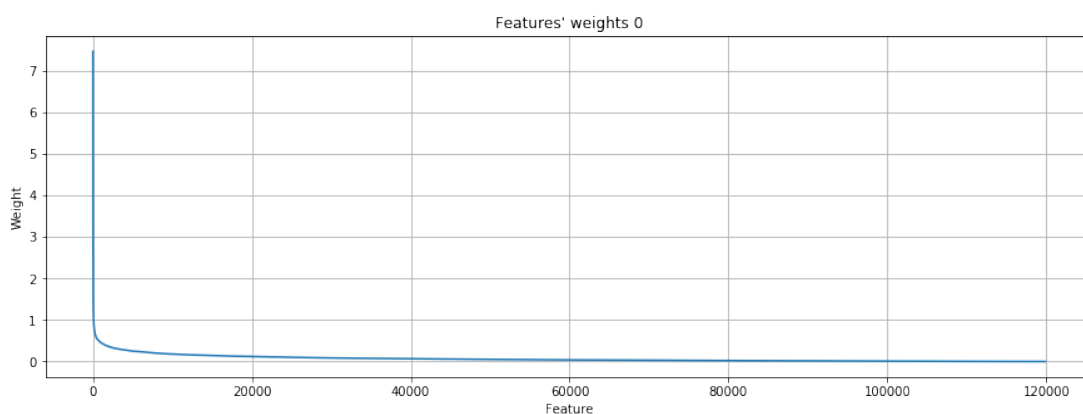
Figure 5.3: Features' weights of SVM classifier for relevance detection before features selection

102

The cutoff value for features selection has been set to 0.2, reducing the final features' dimension to 8234. Training again the classifier on selected features, it reaches the following results:

Predicted value

	Irrelevant	Relevant
Irrelevant	962	55
Relevant	60	73

Actual value

F1-macro	0.559
Recall	0.549
Precision	0.570

The most relevant features, obtained by the 25 heaviest weights of the SVM classifier, are shown in Figure 5.4, where it is interesting to notice some actual reference about engines, especially for red values that have the weight concordant to the relevance.
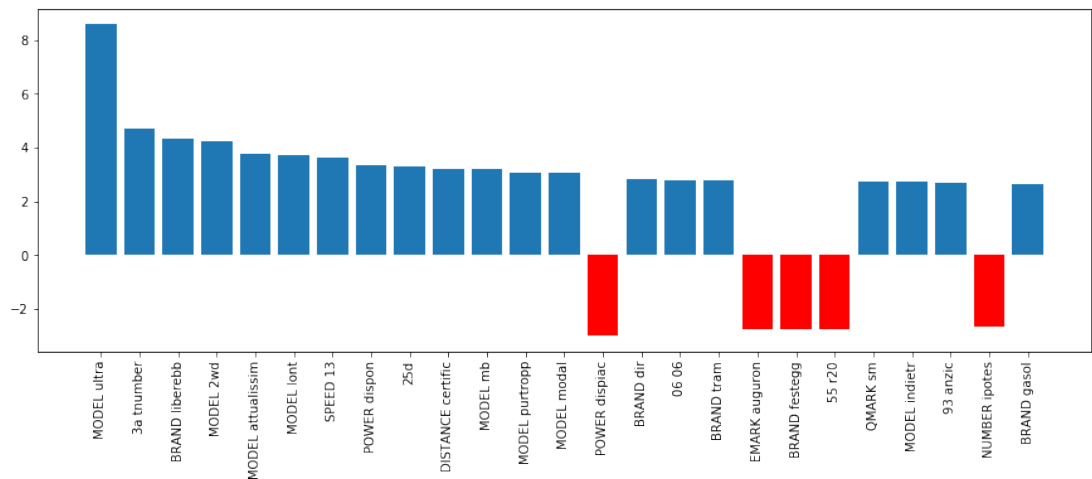
Figure 5.4: 25 most relevant features of SVM relevance classifier after features selection

The final model was the one with the parameter $C=100$. Just few considerations about the previous results: Even if the task is different from the Twitter

sentiment analysis, the many more difficulties are reflected in the final results, in fact performances are significantly lower. After features selection the F1-macro score obtained slightly poorer results, but the recall's improvement and the supposed improved stability makes the second model a bit better.

In the following paragraph, the SVM relevance detection classifier will be compared to a Logistic Regression model with the same final purpose.

5.2.2 Relevance Detection with Logistic Regression

The same process for relevance detection seen in the previous paragraph, has been applied involving the Logistic Regression classifier. The procedure was the same: a first naïve classification for empirically find the most important features, then the cutoff of the less relevant ones, followed by the final training that defines the actual model. also the grid for hyperparameter's optimization was the same of the previous case.

Without features selection, the classifier with optimal $C=100$, obtained the following scores:

Predicted value

	Irrelevant	Relevant
Irrelevant	962	91
Relevant	49	84

Actual value

F1-macro	0.545
Recall	0.631
Precision	0.480

Features' selection has been made exploiting L1-regularization on model training. The effect is shown in Figure 5.5, where it is possible to verify the early discussed characteristic: L1-regularization selects almost automatically most important features nullifying less important ones. Setting the cutoff value to 0.1,

the method returns 1349 final values.
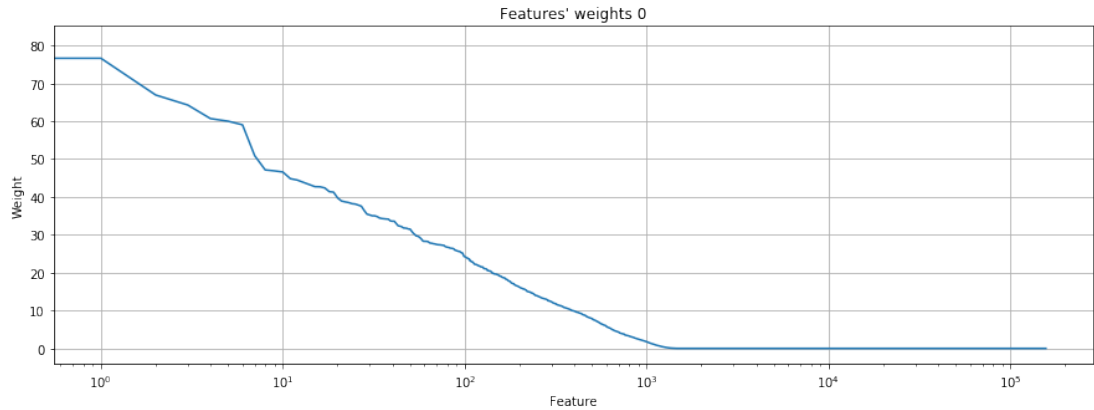
Figure 5.5: Features' weights of Logistic Regression classifier for relevance detection before features selection

After features selection the scores, obtained with the model with $C{=}1000$, become:

Predicted value

	Irrelevant	Relevant
Irrelevant	933	84
Relevant	50	83

Actual value

F1-macro	0.553
Recall	0.624
Precision	0.497

As in the previous situation, are shown the 25 most relevant features, obtained from the most heaviest weight of the Logistic Regression classifier. As before, the red weights are the ones concordant with the class "relevant". It is curious that most important features are different between SVM and Logistic Regression classifiers, and that there is no an effective interpretation of the features.
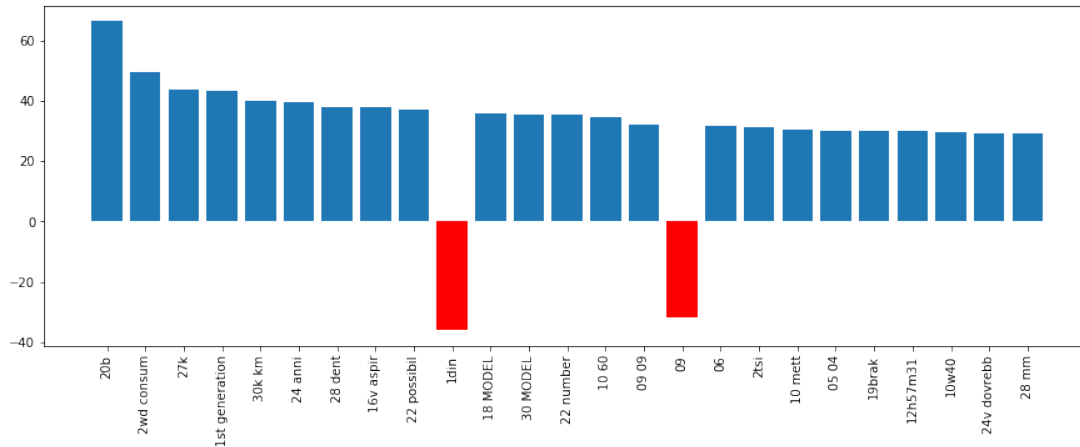
Figure 5.6: 25 most relevant features of Logistic Regression relevance classifier after features selection

From the confusion matrix it is possible to notice that the features selection improves mostly the precision, in fact it reduces the number of true positives from 91 to 84, while the recall remains similar.

Comparing the results with the SVM classifier, both reach similar results for what concerns the F1-macro score, but the Logistic Regression one gets a better recall, that makes it the preferable one.

5.2.3 Sentiment Classification with SVM

For sentiment classification it has been followed an approach similar to the one used for relevance detection: the same preprocessing and vectorization techniques have been applied, so the same is the dimension of the features space. The only thing that has been changed was the classifier, that became multiclass instead of binary.

A first result on validation data, obtained with the hyperparameter $C=10$, gave the following scores:

Predicted value

	Positive	Neutral	Negative
Positive	35	16	0
Neutral	28	28	1
Negative	10	12	3

Actual value (left vertical label)

F1-macro	0.422
F1-micro	0.496

For simplicity in Figure 5.7 it has been shown just one of the three classifiers' weights that compose the one-versus-one multiclass classifier for sentiment classification. All graphics actually have the same shape, which is typical. Setting the cutoff values, respectively to 0.15, 0.1 and 0.1, the number of selected features become 9657, which is again much lower than initial dimension.
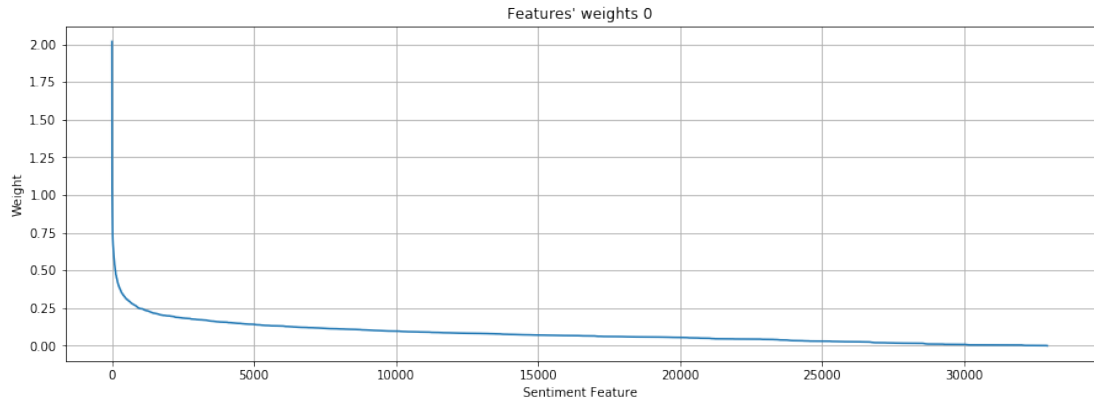
Figure 5.7: Features' weights of SVM classifier for sentiment classification before features selection

After features selection, the optimized model with $C=10$, the scores become:

107

Predicted value

	Positive	Neutral	Negative
Positive	35	15	1
Neutral	24	33	0
Negative	13	9	3

(rows labeled as *Actual value*)

F1-macro	0.451
F1-micro	0.534

The scores notify a little improvement on the performance, that is easily seen in the classification of neutral class. Again, the scores are very different from the one seen for the Twitter's dataset, and the motivation have been already presented. Moreover, for this classification, the bias inducted for the negative class imbalance is visible on the confusion matrix: while positive and neutral classes perform quite well, for what concerns the negative class, the classifier tends to predict positive or neutral. This issue will be present in all other models, and it is characteristics the fact that the dataset is unbalanced.

This model has been used as baseline, in order to compare the effectiveness of the revised BPEF model, which results will be presented in the following paragraph.

5.2.4 SENTIMENT CLASSIFICATION WITH REVISED BPEF

As mentioned, one goal was to improve the SVM classifier using the revised BPEF algorithm. The process was somehow the same as the one seen for Twitter sentiment classification. As previously discussed, features selection is based on information gain, and it has to be calculated before the classification, since it is not based on the classifiers' weights. Thus, for every feature parameter it has been calculated the information gain. All information trends actually have mostly the same graphics, so an example is shown in Figure 5.8.
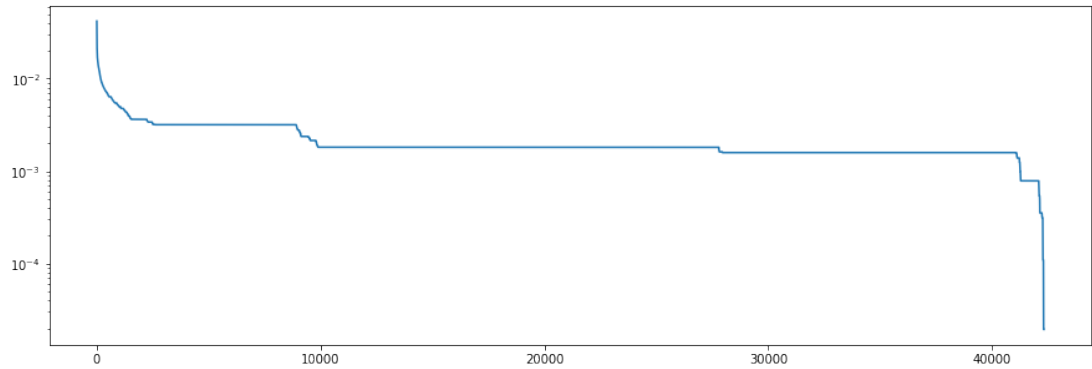
Figure 5.8: Information Gain trend of the features.

All BPEF models have been trained optimizing the following hyperparameters:

- SVM: C from 10^{-3} to 10^3 with 7 exponentially evenly spaced values;

- Logistic Regression: C from 10^{-3} to 10^3 with 7 exponentially evenly spaced values;

- Random Forest:

  - Number of estimators: [201, 501]
  - Maximum of looked features on split: [auto, log_2]
  - Maximum depth of the tree: [10, 100]
  - Split criterion: [gini, entropy]

A first model training without features selection gives the following results:

Predicted value

F1-macro	0.465
F1-micro	0.534

It is possible to notice that even the model without features selection gives better results than the best SVM model, so this is actually an improvement.

Features selection has been made for each feature parameter selecting a cutoff value for information gain. The cutoff value, after some experiments was set to 0.0019, that is the value just above the flat area, and the number of selected features, which is different for each feature parameters is:

Feature name	Word summarizing	# Features
word	false	9.860
word	true	9.433
pos	false	11.580
pos	true	10.887
swnt	false	9.082
swnt	true	8.547

After features selection the scores become:

Predicted value

		Positive	Neutral	Negative
Actual value	Positive	29	22	0
	Neutral	15	42	0
	Negative	7	15	3

F1-macro	0.467
F1-micro	0.556

With features selection it is possible to notice a little improvement of the performance, even if the bias inducted by the class unbalances is still present. However,

the most important improvement is against the SVM sentiment classifier, in fact the overall F1-macro score improves by 0.016, improvement seen especially in the neutral classification. The final outcomes make this last model the preferred one for sentiment classification on the Italian automotive dataset.

For results evaluation, below are reported some mistakes on classification (between squared brackets the quotes):

- **Positive predicted Neutral**: [*@Matric: Ma l'hai ordinata come la volevi tu? Io mi sono dovuto accontentare di quello che c'era in arrivo (alla fine cambia solo il colore e motorizzazione e allestimento è quello voluto) ed è passato un mese e ancora nulla.*] *Si come l'ho ordinata io con tutti gli accessori comunque il motore è molto silenzioso sembra a benzina*

- **Negative predicted Neutral**: [*sicuro da giocarmi la mia GTA con il busso?la macchina nn me la giocherei nemmeno se mi dovessi indovinare la risp a questo quesito: chi è più BONA la Hunziker o tua nonna?*] *ha fatto un passo indietro perchè va più veloce? la 159 temo che nemmeno con un motore sbarbato da un cofano maserati che a loro volta l'hanno dovuto rubare da dentro il centro ricerche ferrari riuscirà ad essere competitiva sul serio... qua dentro sembra un elefante la m3 figuriamoci una 159 gta... la prossima generazione di alfa magari sarà all'altezza del nome che porta ma per ora...*

- **Negative predicted Neutral**: *@Touareg 2.5 2.0 MultiJet II Aut. 4WD (170Cv) : Trailhawk : 32.800*

From these example come some difficulties and mistakes on manual labeling: In the second case, it is not explicit the reference of the engine of subject, in fact it is written that the car has poor performance, but the matter is not exactly the engine, so the correct label may be "neutral". In the third case, the comment is surely an error, since there is no sentiment polarity expressed by the user, so the correct label is "neutral". For both examples the classifier predicted the correct label (in the first it depends on the interpretation), even if the manual annotation was wrong, and it is a great result.

5.2.5 4-labels Classification with SVM

At this point, are available two classifier: the one responsible to relevance detection, and the one responsible to sentiment classification. Now the goal is to

select the best model that best classifies all the four labels. The tested approaches involve the SVM classifier and the cascade classifier, built with the relevance detector and the sentiment classifier.

The first case follows the same approach seen for all SVM classifier, the only difference is that the output has four labels. The first result, obtained without features selection, with the optimal parameter C=1000, is the following:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	997	12	8	0
Positive	32	16	3	0
Neutral	36	11	10	0
Negative	19	3	3	0

Actual value

F1-macro	0.385
F1-micro	0.890

As usual, features selection consisted in the selection of the cutoff values, that have been chosen, for each of the six binary classifiers that compose the one-versus-one multiclass classifier, with respect to the features' weights that have the usual trend shown in Figure 5.9.
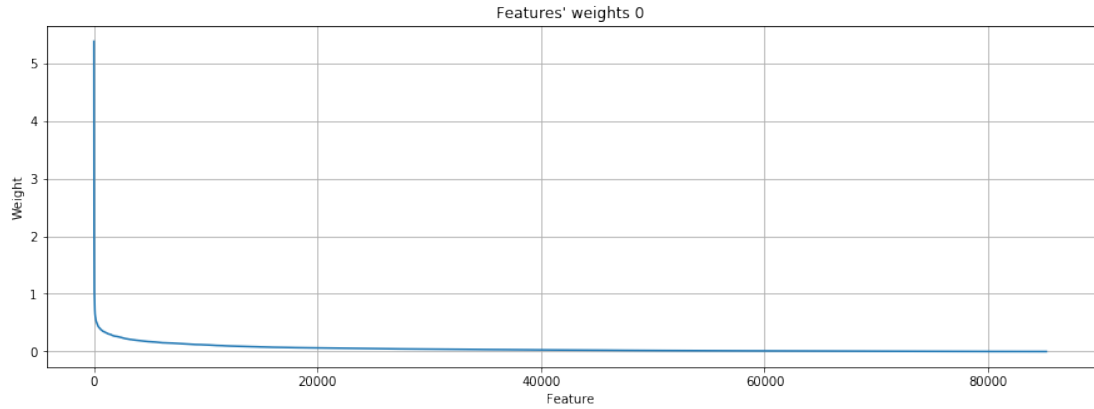
Figure 5.9: Features' weight of the four labels SVM classifier.

After some tries for selecting the cutoff values, the total selected features become 45.088, which are a lot with respect to all the other features selection results. A new training with just the selected features gave the following results:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	1000	11	6	0
Positive	32	15	4	0
Neutral	38	10	9	0
Negative	20	3	2	0

Actual value

F1-macro	0.378
F1-micro	0.890

As it is possible to see, the performances seem quite poor, especially for the "negative" class, that is definitely ignored. The causes are surely related to the high imbalance across the classes, which is heavier than the past cases, in fact

113

most errors are predicted as "irrelevant", that is a proof of the inducted bias. This confusion matrix is also a good example about why the micro score is not good to evaluate this particular problem, in fact the 89% F1-micro score is misleading, because the actual performance are quite poor.

This result was taken as baseline in order to compare the results obtained with the cascade classifiers that will be presented in the following paragraphs.

5.2.6 4-labels Classification with Cascade Classifier

At this point, all blocks needed for build the cascade classifier are actually ready to be assembled. As previously said, the chosen classifier for relevance detection was the one that used Logistic Regression algorithm, because it was the one with better recall score, which means that it tends to detect better the topic treated on the comment. For comparison, on the second stage of the cascade classifier have been employed both SVM and the revised BPEF sentiment classifiers, trained after features selection.

Cascade Classifier with SVM Sentiment Classifier

Below are shown the scores on validation data obtained by the cascade classifier composed by the Logistic Regression relevance classification, and the SVM sentiment classifier.

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	933	36	43	5
Positive	19	24	8	0
Neutral	20	18	19	0
Negative	11	10	3	1

Actual value

F1-macro	0.409
F1-micro	0.850

The results show an improvements with respect to the baseline classifier, especially for what concerns the negative class, that is no more completely ignored. Even positive and neutral classes have shown some improvements, while the irrelevant class shows decreased performance.

Cascade Classifier with Revised BPEF Sentiment Classifier

Below are shown the scores on validation data obtained by the cascade classifier composed by the Logistic Regression relevance classification, and the revised BPEF sentiment classifier.

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	933	18	61	5
Positive	19	23	9	0
Neutral	20	4	33	0
Negative	11	3	3	8

Actual value

F1-macro	0.556
F1-micro	0.867

The scores present a huge improvements with respect to the both baseline classifier and the previous cascade classifier. The "negative" class now gives good results and may be considered reliable, while "positive" and "neutral" classes show better classification performance. It is also important to see that "big mistakes" don't happen, for instance just few negatives have been predicted positives and

no positives has been predicted negatives. The presented scores lead to define this last classifier the preferable for topic relevance and sentiment classification, and it will be used for classify all the other categories.

5.2.7 4-LABELS CLASSIFICATION WITH TEST DATA

At this point, when the models have been chosen, it is a good practice to test them using new data, not yet involved in the model definition procedure. This data has been kept apart at the beginning, and consists in the actual test dataset. The following tests are used to verify that the models don't overfit validation data, and show how the model is going to work on real new comments.

CASCADE CLASSIFIER WITH SVM SENTIMENT CLASSIFIER

The cascade classifier composed by the Logistic Regression relevance detector and SVM sentiment classifier gives the following results:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	1153	52	62	4
Positive	28	29	5	1
Neutral	32	24	16	0
Negative	11	11	8	1

Actual value

F1-macro	0.375
F1-micro	0.834

The scores highlight a little decrease of the performance, but absolutely tolerable. It is possible to conclude that this model does not suffer from overfitting.

The same test has been made for the cascade classifier that involves the revised BPEF. It is important to get good scores because since it is the best model, a poor result will make the model unusable. The obtained scores are:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	1153	28	86	4
Positive	28	22	12	1
Neutral	32	2	38	0
Negative	11	2	10	8

Actual value

F1-macro	0.503
F1-micro	0.850

It is possible to conclude that, even if the scores are a little lower than the ones obtained with validation data, the model has still good values to be considered a good model for both relevance and sentiment classification.

5.2.8 CLASSIFICATION WITH OTHER CLASSES

All the process seen for the classification of the class "engine" have been employed also for the training of the models for the classification of the classes "brand", "exteriors" and "support". Obviously it can be further extended for all the other topics. In this section, for sake of brevity, only main results will be presented along with some considerations. For every class, the subdivision of the comments between training, validation and test has been made maintaining the original distribution, subdividing the 80% training and 20% test, and the training was further divided into 80% actual training and 20% validation sets.

The class "brand" was the one responsible to classify the sentiment about the manufacturers' perception. Its distribution about relevance is shown in Figure 5.1.

	Relevant	Irrelevant	Total
Training	880	3716	4.596
Validation	220	930	1.150
Test	274	1.163	1.437

Table 5.1: Relevance distribution for class "brand".

Again, imbalance is really heavy, so the same issues found for the "engine" class are expected. The distribution of the relevant comments is shown in Table 5.2.

	Positives	Neutrals	Negatives	Total
Training	416	180	284	880
Validation	104	45	71	220
Test	130	56	88	274

Table 5.2: Relevance distribution for class "brand".

It is possible to see again imbalance in favor of the "positive" class, and lack of comments of the "neutral" class.
Starting from the final results, obtained on validation data with the cascade classifier, composed by the Logistic Regression relevance detector and the revised BPEF sentiment classifier, it is possible to notice lower performance than the one reached with the "engine" class, but still acceptable.

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	771	89	14	56
Positive	62	30	1	11
Neutral	24	6	10	5
Negative	42	4	1	24

(Actual value)

F1-macro	0.417
F1-micro	0.726

The lower results comes mostly from the relevance detector, that was not very good to interpret comments belonging to the class "brand". In a way it is understandable, because sometimes comments that contain an opinion to a brand, express the subject implicitly, and this is a further difficulty for the classifier. The performance of the Logistic Regression relevance classifier on validation data are:

Predicted value

	Irrelevant	Relevant
Irrelevant	838	92
Relevant	134	86

(Actual value)

F1-macro	0.391
Recall	0.418
Precision	0.367

As discussed, it presents performance lower than the one reached with the "engine" class. The second stage of the cascade classifier consisted in the revised BPEF, and it performance on the validation dataset are:

Predicted value

	Positive	Neutral	Negative
Positive	96	1	7
Neutral	35	4	6
Negative	49	5	17

Actual value

F1-macro	0.386
F1-micro	0.532

From the confusion matrix it is possible to conclude that the classifier is affected by some bias, due to the imbalance to the positive class.

Finally, the classification with all four labels on test data gave the following results:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	957	120	15	71
Positive	71	44	3	12
Neutral	33	4	12	7
Negative	54	8	1	25

Actual value

F1-macro	0.412
F1-micro	0.722

The performance are still affected by the relevance detector that doesn't reach the comments with brand reference. However, bias on "positive" sentiment seems reduced, and a simple exploitation may be that the comments that were going to be classified with the bias, were before classified "irrelevant".

In conclusion, even if lower than the engine classifier, the performance still may be considered reliable, also because performance on test and validation data were comparable, so no overfitting seems to be present.

CLASSIFICATION WITH "EXTERIORS" CLASS

The class "exteriors" was the one responsible for the sentiment classification about vehicles' aesthetics and all comments with exteriors references. The distribution about comments' relevance is:

	Relevant	Irrelevant	Total
Training	554	4.042	4.596
Validation	139	1.011	1.150
Test	173	1.264	1.437

Table 5.3: Relevance distribution for class "exteriors".

In general, even if the imbalance is still important, there are many comments that are relevant about this class. Continuing, the sentiment distribution of relevant comments is shown in Table 5.4.

	Positives	Neutrals	Negatives	Total
Training	272	124	160	556
Validation	68	31	40	139
Test	85	39	50	174

Table 5.4: Relevance distribution for class "exteriors".

The distribution shows an imbalance on the "positive" class, but the metric that may gives more problems is supposed to be the actual number of comments,

which are not plenty.

The best results comes from the cascade classifier with Logistic Regression relevance detector and BPEF sentiment classifier. The scores on validation data are:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	934	31	24	22
Positive	32	26	4	6
Neutral	15	2	13	1
Negative	16	3	4	17

Actual value

F1-macro	0.517
F1-micro	0.861

The results show performance that are close to the one reached with the class "engine", and actually good. Some problems comes again from the relevance classification, which performance on validation data are:

Predicted value

	Irrelevant	Relevant
Irrelevant	934	77
Relevant	63	76

Actual value

F1-macro	0.521
Recall	0.547
Precision	0.497

The second stage with revised BPEF model gave the following results:

Predicted value

	Positive	Neutral	Negative
Positive	62	1	5
Neutral	26	1	4
Negative	33	0	7

(Actual value)

F1-macro	0.322
F1-micro	0.504

The classifier actually suffers from the imbalance to the "positive" class, in fact most "negatives" and "neutrals" are classified as "positive". Even in this case the fact that poor results on classification are translated into good ones on the four label case can be explained in the fact that comments that were going to be classified wrongly, were before annotated as "irrelevant". These facts may be considered lucky cases, but another point of view may be that these comments are actually ambiguous or sufficiently "dirty" to be classifies "irrelevant".
Finally, classification on test data remarks the good final performance, that are:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	1187	43	14	20
Positive	50	27	4	4
Neutral	26	1	10	2
Negative	28	2	3	16

Actual value

F1-macro	0.477
F1-micro	0.863

The results are comparable to the ones reached with validation data, so the results can be considered out of overfitting and most importantly reliable.

Classification with "Support" class

The last class to be handled was the "support" one. It consists on the comments that regard reference about the customer care. The relevance class distribution is shown in Table 5.5.

	Relevant	Irrelevant	Total
Training	344	4.294	4.638
Validation	86	1.063	1.149
Test	108	1.288	1.396

Table 5.5: Relevance distribution for class "support".

The sentiment distribution of relevant comments is shown in Table 5.5.

	Positives	Neutrals	Negatives	Total
Training	68	116	160	344
Validation	17	29	40	86
Test	22	36	50	108

Table 5.6: Sentiment distribution of relevant comments for class "support".

For this class is visible an imbalance to the "negative" sentiment polarity, and again a lack of relevant comments.
the best performance were reached again with the cascade classifier with Logistic Regression relevance detector, and the revised BPEF sentiment classifier. The scores on validation data are:

F1-macro	0.435
F1-micro	0.899

That are obtained with the Logistic Regression relevance detector that gave the following scores on validation data:

Predicted value

Irrelevant Relevant

		Irrelevant	Relevant
Actual value	Irrelevant	1007	56
	Relevant	42	44

F1-macro	0.473
Recall	0.512
Precision	0.440

And the revised BPEF that gave the following scores on validation data:

Predicted value

Positive   Neutral   Negative

		Positive	Neutral	Negative
Actual value	Positive	3	3	11
	Neutral	0	12	17
	Negative	1	13	26

F1-macro	0.420
F1-micro	0.477

The mentioned imbalance on the "negative" polarity inducted a bias that has repercussions on the classification, in fact most errors fall on the negative side of the confusion matrix. The cascade classifier on new test data gave the following results:

Predicted value

	Irrelevant	Positive	Neutral	Negative
Irrelevant	1226	1	17	44
Positive	12	0	3	7
Neutral	18	0	9	10
Negative	22	0	0	27

Actual value

F1-macro	0.406
F1-micro	0.907

That again seem to be out of overfitting, having the scores similar to the one reached on validation data. Even for this case the model can be considered reliable.

6
Industrial Use Case

6.1 Data Visualization

7
Conclusions and Future Works

References

[1] D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, "The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 2, pp. 5:1–5:29, Aug. 2018. [Online]. Available: http://doi.acm.org/10.1145/3185045

[2] A. Bermingham and A. Smeaton, "On using twitter to monitor political sentiment and predict election results," in *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, Nov. 2011, pp. 2–10. [Online]. Available: https://www.aclweb.org/anthology/W11-3702

[3] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, 2011. [Online]. Available: http://scholar.google.de/scholar.bib?q=info:7jxZLbM1mzwJ:scholar.google.com/&output=citation&hl=de&as_sdt=0&ct=citation&cd=54

[4] H. Rui, Y. Liu, and A. Whinston, "Whose and what chatter matters? the effect of tweets on movie sales," *Decis. Support Syst.*, vol. 55, no. 4, pp. 863–870, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2012.12.022

[5] F. Jurado and P. Rodriguez, "Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues," *Journal of Systems and Software*, vol. 104, pp. 82 – 89, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215000485

[6] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland:

Association for Computational Linguistics, Jun. 2014, pp. 1555–1565. [Online]. Available: https://www.aclweb.org/anthology/P14-1146

[7] A. Montejo-Ráez, E. Martínez-Cámara, M. Martín-Valdivia, and L. López, "Ranked wordnet graph for sentiment polarity classification in twitter," *Computer Speech & Language*, vol. 28, 01 2013.

[8] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14 – 46, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705115002336

[9] M. Taboada, "Sentiment analysis: An overview from linguistics," *Annual Review of Linguistics*, vol. 2, 02 2016.

[10] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[11] K. Anbananthen, J. Krishnan, M. Shohel Sayeed, and P. Muniapan, "Comparison of stochastic and rule-based pos tagging on malay online text," *American Journal of Applied Sciences*, vol. 14, pp. 843–851, 09 2017.

[12] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, "Text classification algorithms: A survey," *CoRR*, vol. abs/1904.08067, 2019. [Online]. Available: http://arxiv.org/abs/1904.08067

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[14] V. Saifee and T. Jay, "Applications and challenges for sentiment analysis: A survey," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, 2013.

[15] F. Grässer, S. Kallumadi, H. Malberg, and S. Zaunseder, "Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning," in *Proceedings of the 2018 International Conference on Digital Health*, ser. DH '18. New York, NY, USA: ACM, 2018, pp. 121–125. [Online]. Available: http://doi.acm.org/10.1145/3194658.3194677

[16] A. Rane and A. Kumar, "Sentiment classification system of twitter data for us airline service analysis," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, July 2018, pp. 769–773.

[17] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.* Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[18] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* New York, NY, USA: Cambridge University Press, 2014.

[19] What is underfitting and overfitting in machine learning and how to deal with it. [Online]. Available: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a9

[20] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation.* Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565

[21] Intro to machine learning part 2: Binary classification with perceptron. [Online]. Available: https://waterprogramming.wordpress.com/2018/09/28/intro-to-machine-learning-part-2-binary-classification-with-perceptron/

[22] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of Machine Learning Research*, vol. 2, pp. 499–526, 06 2002.

[23] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[24] G. Melki, "Fast online training of l1 support vector machines," Ph.D. dissertation, 04 2016.

[25] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic, "Feature selection using support vector machines," 2002.

[26] J. Pereira, M. Basto, and A. Ferreira-da Silva, "The logistic lasso and ridge regression in predicting corporate failure," *Procedia Economics and Finance*, vol. 39, pp. 634–641, 12 2016.

[27] V. Fonti and E. N. Belitser, "Paper in business analytics feature selection using lasso," 2017.

[28] D. Roobaert, G. Karakoulas, and N. V. Chawla, "Chapter 22 information gain, correlation and support vector machines."

[29] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 7 1948. [Online]. Available: https://ieeexplore.ieee.org/document/6773024/

[30] H.-A. Park, "An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain," *Journal of Korean Academy of Nursing*, vol. 43, pp. 154–164, 04 2013.

[31] D. Fradkin and I. Muchnik, "Support vector machines for classification," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 01 2006.

[32] L. Pinault and L. A. Belanche, "Modelling with kernels based on gower similarity," 2015.

[33] L. Bottou and C. jen Lin, "Support vector machine solvers," 2006.

[34] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*, 1st ed.   Athena Scientific, 1996.

[35] Support vector machine: Kernel trick; mercer's theorem. [Online]. Available: https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4

[36] Dynamics of neural networks. [Online]. Available: https://neuralrant.wordpress.com/2017/05/04/neural-networks-dynamics/

[37] A. Cutler, D. Cutler, and J. Stevens, *Random Forests*, 01 2011, vol. 45, pp. 157–176.

[38] H. Sharma and S. Kumar, "A survey on decision tree algorithms of classification in data mining," *International Journal of Science and Research (IJSR)*, vol. 5, 04 2016.

[39] R. Bedi, "Review of decision tree data mining algorithms : Id 3 and c4.5," 07 2015.

[40] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.*   Monterey, CA: Wadsworth and Brooks, 1984, new edition [**?**]?

[41] S. Taheri and M. Mammadov, "Learning the naive bayes classifier with optimization models," *Int. J. Appl. Math. Comput. Sci.*, vol. 23, no. 4, pp. 787–795, Dec. 2013. [Online]. Available: https://doi.org/10.2478/amcs-2013-0059

[42] M. Aly, "Survey on multiclass classification methods," 2005.

[43] G. Canbek, S. Sagiroglu, T. Taskaya Temizel, and N. Baykal, "Binary classification performance measures/metrics: A comprehensive visualized roadmap to gain new insights," 10 2017, pp. 821–826.

[44] M. Hossin and S. M.N, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, pp. 01–11, 03 2015.

[45] A. Hassan, A. Abbasi, and D. D. Zeng, "Twitter sentiment analysis: A bootstrap ensemble framework," *2013 International Conference on Social Computing*, pp. 357–364, 2013.

[46] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *LREC*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds. European Language Resources Association, 2010. [Online]. Available: http://nmis.isti.cnr.it/sebastiani/Publications/LREC10.pdf

[47] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *CoRR*, vol. abs/1308.6242, 2013. [Online]. Available: http://arxiv.org/abs/1308.6242

[48] T. Ali, D. Schramm, M. Sokolova, and D. Inkpen, "Can I hear you? sentiment analysis on medical forums," in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan: Asian Federation of Natural Language Processing, Oct. 2013, pp. 667–673. [Online]. Available: https://www.aclweb.org/anthology/I13-1077

[49] D. D. Wu, L. Zheng, and D. L. Olson, "A decision support approach for online stock forum sentiment analysis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 8, pp. 1077–1087, Aug 2014.

[50] M. S. Neethu and R. Rajasree, "Sentiment analysis in twitter using machine learning techniques," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, July 2013, pp. 1–5.

[51] The document object model. [Online]. Available: http://www.technologyuk.net/computing/website-development/world-wide-web/document-object-model.shtml

[52] S. Chatterjee and A. Nath, "Auto-explore the web – web crawler," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, pp. 6607–6618, 05 2017.