Using Machine Learning Techniques for Sentiment Analysis

Òscar Romero Llombart

Resum– El processament de llenguatge natural és la disciplina que estudia com fer que les màquines aprenguin a llegir i interpretar el llenguatge que usem les persones, el llenguatge natural. Però, en el món de la computació, les paraules no existeixen i són representades per seqüències de números que a l'hora de mostrar-los per pantalla són convertits en lletres. L'anàlisi de sentiments és el nom que obté el problema que donada una sentència o text una computadora sigui capaç d'avaluar-lo i predir amb la màxima precisió possible el sentiment que obtindria una persona en llegir-lo o l'opinió contextual en vers a alguna cosa. Aquest article pretén mostrar el que es pot obtenir, en aquest àmbit, usant les eines d'aprenentatge automàtic més usades.

Paraules clau– Anàlisi de Sentiments, Mineria de Dades, Aprenentatge Automàtic, Llenguatge Natural, Màquines de Vectors de Suport, Arbres de Decisió, Xarxes Neurals Recurrents, Naive Bayes

Abstract– The Natural language processing is the discipline that studies how to make the machines read and interpret the language that the people use, the natural language. But in the machines world, the words not exist and they are represented by sequences of numbers that the machine represents with a character when displaying them on screen. The Sentiment Analysis is the name of the problem that with a sentence or text the machine gets capable to analyze and predict with the maximum precision possible the sentiment that will be obtained by a person when reads it or the contextual opinion related to something. This document wants to show what we can obtain using the most used machine learning tools.

Keywords– Sentiment Analysis, Data Mining, Machine Learning, Natural Language, Support Vector Machines, Decision Trees, Recurrent Neural Networks, Naive Bayes

1 Introduction

With the emergence of the social media, the high availability of the information on Internet and the users that have become prone to share on Internet its feelings about products, movies or wherever they want to share, for example in Twitter or Facebook where the people shares how they are feeling today or if its new car is good or not. The ability to process this information has become important because, for example, we can introduce a new product to the market and then wait to the feelings of the people on Internet, extract them in a useful form, and decide the future viability of this new product.

Most of this information aren't classified or rated in any kind of classification range that can be easy used and is hard to classify at massive scale with humans or normal tools. For this reason, the development of tools that can learn to read texts and extract the feelings is important to the future.

The natural language processing (NLP) is the discipline in the computer science, the artificial intelligence and linguistics, that pursue give the capacity to the machines to understand the people language, like English for example. Inside the NLP there is the field of Sentiment Analysis that studies how to use the machines to process texts and give to each one a kind of classification that we can understand and use. This field uses language processing algorithms for extract features, like words frequency, and supervised machine learning algorithms that learns from an initial set of data initially classified by a human.

Sadly, the machines are limited and we need to be careful on the type of texts that we want to process because it has a high impact on the size of the vocabulary that the algorithm needs to learn and the size of the text that needs to be processed. For example, *microblogging* sites like Facebo-

• Contact Email: scarxx@gmail.com
• Intensification: Information Technologies
• Supervised work by: Jordi Duran (DEIC)
• Course 2016/17

ok or Twitter uses relative short sentences and the language that the people use on these sites is open and informal, the same word or meaning can appear with lots of different representations. From another side, the sites like Imdb that holds movie reviews are large texts and have a more formal language. In this document, I use two different datasets, the movie reviews from Pang et. al.[11] and the Stanford Twitter Sentiment[1] that are used by another's researchers to get comparable results.

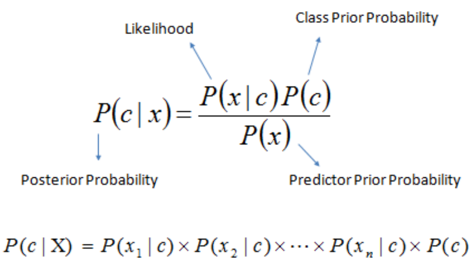
I use different machine learning techniques that have different ways to work and, consequently, learns different things for the same data. This differences on the process of learning can have a huge impact on the performance of the final software that will use the methods. In this document, I compare the performance of this methods and different types of data.

2 Machine Learning Methods

I used five different machine learning methods that has been used with a good performance in the sentiment analysis problem [5][4][10][8], where two of them are based on neural networks explained in section 2.4.

2.1 Naive Bayes

Naive Bayes (NB) is a simple method based on the Bayes rule. The probability each feature contributes independently to the final probability to be a class, each one has its distribution. In a real problem, this independence is rare. To avoid this, I use Multinomial Naive Bayes, provided by SciKit-Learn[12], that models the same probability but it uses a multinomial distribution[3].



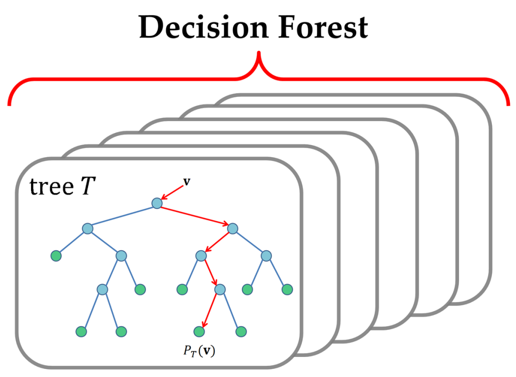
$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

2.2 Random Forest

Random Forest (RF) is a method that trains multiple decision trees. Each tree is trained using a random subset of the vector features. The decisions of each tree are combined using a voting algorithm that gives the result. The sequence of features and the value of the feature generates the path to a leaf that represents the decision. While training, the values of the intermediate nodes are updated to minimize a cost function that evaluates the performance of the trees. The objective is minimize a cost function that evaluates the performance of the trees. I use the implementation provided by SciKit-Learn.

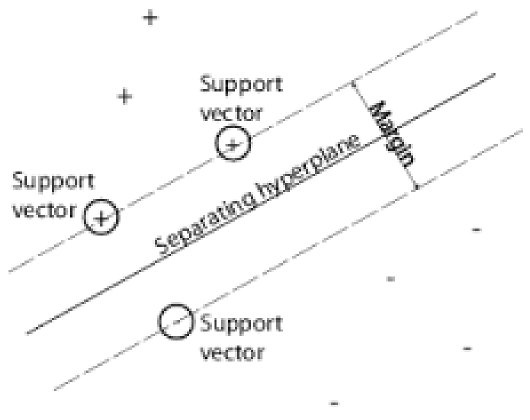
[1]https://nlp.stanford.edu/sentiment/treebank.html



2.3 Support Vector Machines

Support vector machines (SVM) is a method that considers that each set of features represents a position inside a hyperspace then the SVM tries to divide it using a hyperplane maximizing the distance between this hyperplane and each vector, minimizing the objective function. This space division is hard to accomplish, and sometimes impossible, for this the SVM can use a margin that allows to misclassify some examples but increases the overall performance.

For this document, I use the implementation provided by SciKit-Learn using Liblinear. I also tried with Libsvm implementations but tends to be slower and give worst results.

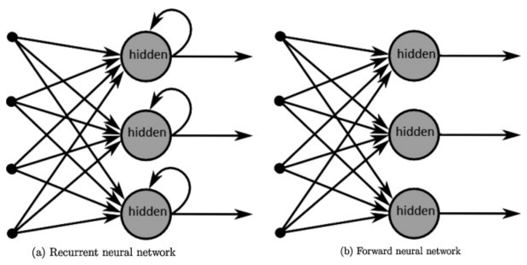


2.4 Neural Networks

Neural Networks is a method that tries to optimize some weights, the body of the neuron, that are multiplied by the vector of features, the dendrites. The result of this multiplication is the prediction made by this neuron, the axon terminal. It can be used as a result or as a feature to the next set of neurons, called Multi-Layer Neural Networks(MLP). The objective is train the internal weights using the Gradient Descent and Back-propagation[14] method where a cost function is computed and the result is propagated back to the neurons weights that gets updated to minimize the objective function on each round.

The text classification problem can be viewed as a temporal distribution of features, characters or words. In this document, I also consider the use of Recurrent Neural Networks(RNN) that is a special case of Neural Networks that uses an internal memory on each neuron that represents the intermediate understanding between features that can be accumulated or forget[6] by the neuron. With MLP and

RNN I have used our implementation using Tensorflow[1] and Long-Short Term Memory(LSTM) cell.



3 Objectives

As I said before, there is a lot of important data in Internet that, actually, is hard to use. Process this data can give the power to predict future products, economy trends or social facts by processing the people feelings that are shared on Internet. The objective is learn how to process this type of massive data and build a system that can exploit this data. This project is divided in three phases:

- **Research**: Investigate different methods and algorithms that exists to do Natural Language Processing, more concretely Sentiment Analysis.
- **Build a *workframe***: Build a system that can concatenate transformations to the data that can be concatenated and applied to any machine learning methods.
- **Build and Train models**: Train different combinations of transformations and models to record the effects.
- **Evaluate**: Evaluate the trained models and compare them with a reference baseline from the state of art.

To accomplish this objective, I work on each machine learning method independently. For each one I apply the set of text transformations, optimize its hyper-parameters to have the maximum performance on each round, using a test driven methodology explained in section 4, and when the method works correctly I evaluate with the reference dataset and record its results for compare all of them, as can be seen at figure 1.

4 Methodology

One of the most important things that happens on machine learning is that the algorithms can memorize the data and when we want to use it with a new data it has a poor performance, this behavior is called *overfit*. To avoid this problem, I work with test driven methodology. Each dataset is divided in three random parts and each part in three more divisions:

- **Train(60%):** Used to feed the machine learning algorithm on the learning process
- **Test(20%):** Used to see if our algorithm is *overfitting* or not.
- **Validation(20%):** Used to evaluate the performance of the algorithm

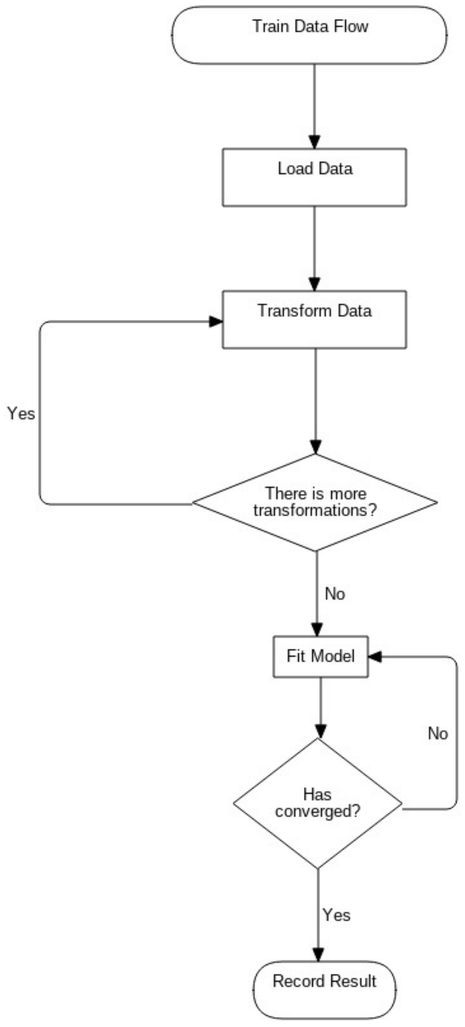
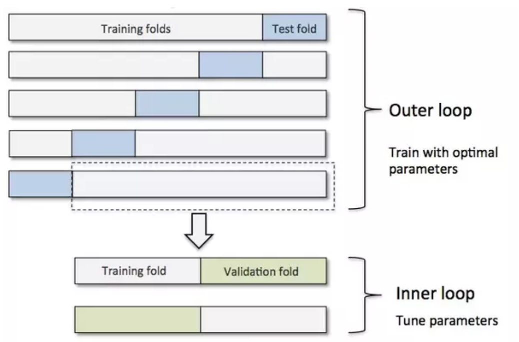


Fig. 1: Training Process Flow

Then I use each method with these parts, evaluate its performance and record a table of performances, process called cross validation. This way, I can tune the hyper-parameters of each algorithm and decide when the algorithm has been optimized to work best and generalize.

At last, when I was sure that the algorithm works with my splits I test the method with the original reference splits to have an objective idea of the performance that I have accomplished and compare my results with some reference work on the Sentiment Analysis field, because different data splits can give different results.



5 Planning and Development

To develop this project, I divided it into different phases, as mentioned in section 3. The first phase is focused on investigate, acquire data and build the base of the project that can handle the flexibility needed to build multiple machine learning models over it. The rest of the project are focused on build and train multiple systems, record its performances and compare the results. During the first phase of the project an investigation about what is Sentiment Analysis, what is the stat of art of the current solutions, what data transformation methods are used, an investigation about implementations of machine learning methods and acquire reference datasets.

5.1 *Workframe* design and implementation

During this phase of the project I designed the *workframe* that can handle the machine learning models, store and restore them. Also, the *workframe* needs to have the flexibility to change quickly the models and transformations because big part of the time was reserved to train models.

Finally, a design inspired by SciKit-Learn[2] was used, as can be viewed at figure 2. There are three important modules:

- **DataInputInterface**: Represents a dataset that can be loaded in memory to be processed. This one processes the data into arrays and generates the splits.
- **TransformationInterface**:Represents each transformation that are mentioned in section 8. Can be stored into a file, if needed.
- **IAInterface**: Represents a machine learning model and must be implemented by each one. Can be stored into a file, if needed.

With the idea to process the data efficiently, a special class exists, *Pipleline*, that can concatenate transformations with a final IAInterface and store them with the same order to reproduce the same transformations in the future, because the same transformations that are used to train the model must be used while predicting.

5.2 Train models and collect results

Each machine learning method requires a time to develop and integrate inside the system. This process consists in three steps:

- **Develop the Model**: Integrate it in the system and concatenate the transformations each round.
- **Test and Optimization**: As I say in section 4 each machine learning method and text transformations pair must be tested that works correctly and optimize the hyper-parameters of the model to archive the maximum performance.
- **Collect Results**: After the optimization was done. Three more trains and test has been done with the original data splits of the reference documents to have an objective idea of the general performance accomplished.

After repeating this process several times, I collected four tables of results that can be seen at A.1, A.2, A.3 and A.4. These results can be compared with the reference baselines established by other researchers and state of art documents.

6 Environment

The machine learning methods are resource hungry algorithms, especially while training them. I use an AMD FX^{TM}-8350 Eight-Core Processor at 4.00 Ghz. with 24 GB of RAM and Linux kernel 4.8 with Python 3.5 to train the models. The neural networks are a special case that can be computed using GPU's. To train the neural networks I use an NVidia GTX 1060 with 6 GB GDDR and 1280 Cuda cores at 1708 Mhz. Anyway, the machine learning methods mentioned on this document are more efficient while doing predictions and all of them can be trained in a high-performance computer and exported to low machines.

7 Datasets

There is a lot of data around Internet. But to face the problem, I need categorized or rated data to be used to feed the machine learning methods and comparable data to have a reference of the performance I accomplish.

There is a different categorizations or rates that can be applied to the data. For example, some opinion websites have surveys with a 5-star rating. From the point of view of machine learning the number of classes influences the quantity of data that the algorithm needs to learn and makes the process of learning more complicated. In this document, I use a two-class classification because is out of the scope of this project to see if is better to use two or more classes and I'm interested on classify the data as positive or negative. Anyway, classify the data with more than two classes like positive, negative and neutral, for example, can make sense because there is data that doesn't apply any specific sentiment when we read them, but add more than three classes can tend in a more relative opinions and classifications and let the algorithm training forever trying to learn to reproduce different opinions and points of views.

7.1 Movie reviews

This data was used on Pang et. al.[11] and published by Cornell University[2]. Is a part of the data from the Internet Movie Database. They have been selected only the reviews where the author rating was expressed with stars or any kind of classification convention that can be automatically processed, approximately 752 negative and 1301 positive documents with 122 authors represented. Ratings were automatically extracted and converted into one of three categories: positive, negative, or neutral. As the original paper, I'm only discriminating between positive and negative sentiment and I tried to avoid the class unbalance.

7.2 Standford Twitter Sentiment

This data was used by Go et. al.[4] and at Stanford University. Is a series of messages recorded at the Twitter so-

[2]http://www.cs.cornell.edu/people/pabo/movie-review-data/

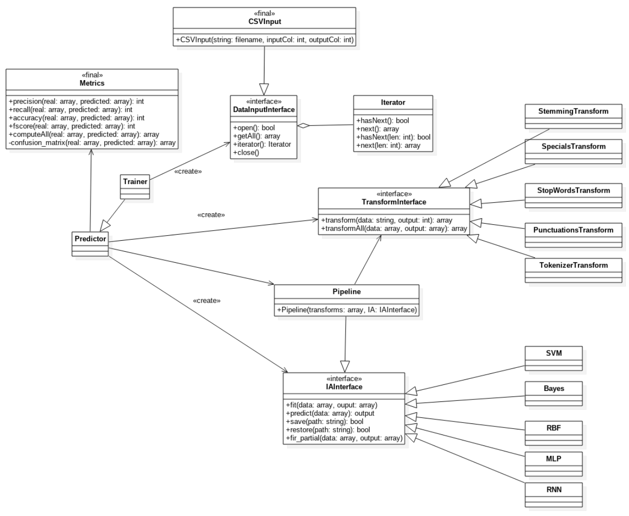

«final»
CSVInput
+CSVInput(string: filename, inputCol: int, outputCol: int)
«final»
Metrics
+precision(real: array, predicted: array): int
+recall(real: array, predicted: array): int
+accuracy(real: array, predicted: array): int
+fscore(real: array, predicted: array): int
+computeAll(real: array, predicted: array): array
-confusion_matrix(real: array, predicted: array): array
«interface»
DataInputInterface
+open(): bool
+getAll(): array
+iterator(): Iterator
+close()
Iterator
+hasNext(): bool
+next(): array
+hasNext(len: int): bool
+next(len: int): array
Trainer
«create»
Predictor
«create»
«interface»
TransformInterface
+transform(data: string, output: int): array
+transformAll(data: array, output: array): array
StemmingTransform
SpecialsTransform
StopWordsTransform
PunctuationsTransform
TokenizerTransform
Pipeline
+Pipeline(transforms: array, IA: IAInterface)
«create»
«interface»
IAInterface
+fit(data: array, ouput: array)
+predict(data: array): output
+save(path: string): bool
+restore(path: string): bool
+fir_partial(data: array, output: array)
SVM
Bayes
RBF
MLP
RNN

Fig. 2: Class diagram

cial network (called ”tweets”). This dataset plays with the idea that some *emoticons* like *”:)”* or *”:(”* is an explicit declaration of a positive or negative feeling, then they use the presence of the emoticon to automatically categorize the tweets. This days, Twitter have a very high activity and its API gives us access to collect thousands or millions of tweets with emoticons easy. Again, as the original paper, I only consider the use of positive or negative examples. This dataset contains 1.6 million of categorized tweets but our computational power is limited because 1.6 million of tweets with 140 characters each one converted into 32-bit representation oversize's the 24 GB of ram of my machine, and, need remember that I'm using techniques that generates a feature vector with all words on all texts, for example. To avoid this problem, I only use the first and last 100 thousand tweets where half of them are positive and the rest negative, to avoid the class unbalance that is a problem outside the scope of this work.

8 Transformations

Machine learning methods have limitations and, actually, can't work at a character level like humans. To improve the performance of the methods I have used some transformations on the original data that makes it easier to be processed by the machine learning methods. Each sentence is converted into a vector of features, for example a sequence of numbers representing the words. Some of them are focused to generate new feature vectors that tries to represent the data in a more compact way and the rest are focused on modify the original data to reduce the size of the feature vector. Need to keep in mind that any type of modifications that are made to the data have any kind of information loose.

8.1 Lower case

One transformation that has used on all the data is the conversion to lower case of the data. This is because the character “A” and “a” have a different representation inside the machines memory and represented with a different number, but we already know that the word *Play* and *play* is the same. All the machine learning methods are based on numerical computation. Transforming the data to lower case have a positive impact, as we can see in section 9, but also have negative effects, for example ”Apple”, referencing the business, is not the same as ”apple”the fruit.

8.2 Punctuations

Punctuations like admiration(!) or question(?) marks have only a little effect on the sentiment meaning of the word and can be irrelevant. Because the fact of be or not a question or admiration does not have any effect on the meaning of have a negative or positive sentiment. With the same way, the dot(.) can means that the current feeling has ended and a new one is starting.

Another punctuation's, like accents, was converted into a similar form. For example, my own name “Óscar”, in Spanish, can be writed as “Oscar” without confusion. But, in a similar way, another words like “más” and “mas”, in Spanish, has a different meaning and can be confused.

8.3 Bag of words

The bag of words are methods that ignores the order of the words and generates a reduced form of the sentence containing the number of occurrences or the frequency of each word in the texts.

8.3.1 Count vector

The general idea behind the count vector is that a sentence can be represented by the words and the number of occurrences of each word in the document, generating a bag of word counts for each sentence.

For example:

- “Im happy” can be represented as the vector
  (Im: 1, happy: 1, you : 0, today : 0) = (1, 1, 0, 0)

The problem is that each vector will contain all words in all the documents. It's important to cut the size in the vector, for example using only the more common words[11], explained at section 8.9 .

8.3.2 Time Frequency and Inverse Document Frequency

The Time Frequency and Inverse Document Frequency(TF-IDF) value increases proportionally to the number of times a word appears in dataset which helps to adjust for the fact that some words appear more frequently in general. The idea behind is that the most used words have highest values than the others less used.

$$TF = \frac{Number\ of\ times\ that\ the\ word\ appears}{Number\ of\ words\ in\ the\ text}$$

$$IDF = \frac{Number\ of\ texts}{Number\ of\ words\ in\ the\ text}$$

$$TF - IDF = TF * IDF$$

8.4 Dictionary of Words

With this method, a dictionary containing all words in the texts is created and then each word in the text is converted to the index of the word inside the dictionary. This method does not break the words order and, also, does not group the words. This transformation can generate a huge dictionary because the machine is case sensitive, for example, and can generate a different index for the same word. Also, is hard to have a dictionary with all words on any language and all of its forms, formal or informal.

This method can make a good performance in machine learning methods that can work with time-dependent data like Recurrent Neural Networks and Decision Trees.

8.5 Stemming

The idea behind stemming is that in the natural language there is a lot of morphologies of the same word. The computer will see each morphology as a different word. Stemming cuts this words with a *common root* that reduces the number of representations of the same word. For example, *argue*, *argued*, *argues* and *arguing* will be reduced to *argu*. The problem with stemming is that different words can be reduced to the same word, for example *catastrophe* and *cats* can have the same root *cat* depending on the heuristics used to generate these *common roots*.

8.6 Lemmatization

Is a similar idea than stemming but tries to use the natural word root or its base form, called "lemma'‘. For example, the word *meeting* has the lemma *meet*. This way tries to solve the collisions that can happens on stemming. The main problem is that is hard to have a dictionary for every word and casual representations of the words are unknown and can be considered individuals. For this task, I used the parser and the dictionary provided by the Natural Language Toolkit[7].

8.7 Stop-words

In some problems, like search engines, removing the words that does not have a special meaning by itself, like connectors, can be helpful and reduce the number of features of the final vector and improve the performance. But needs to keep in mind that removing these words can lead to another expression that have another meaning. For example, "not" is considered a stop word but if we delete it in the sentence "Im not happy" the meaning has changed to the inverse "Im happy" and make a wrong classification.

8.8 Unigrams, bigrams or n-grams

The N-grams are collections of words grouped N by N. In the bigrams case, they are grouped by 2. And the unigrams reference the single words, grouped by 1. The use of n-grams has been used, with results, by Pang et. al.[11], Go et. al.[4] and others[9]. For example:

- **Unigrams:** The sentence "Im not happy" is converted to the vector ("Im", "not", "happy")
- **Bigrams:** The sentence "Im not happy" is converted to the vector ("Im not", "not happy")

This type of transformation makes the relation between words more explicit but one thing to be careful is that the usage of n-grams bigger than two can have a huge impact on the number of features and, consequently, an increase of the size of the vocabulary and the feature vectors.

8.9 Feature Selection

As I said before, the vocabulary size has a high impact on the memory footprint and the computational cost of the machine learning methods. To reduce the vocabulary, after applying the other transformations only the first 50 thousand words, ordered by the most used first, was used to feed the machine learning methods.

This transformation has a high loss of information and features but can have a high positive impact on the final performance because the machine learning method can learn only the words that are important, like "‘happy"’ and skip other words like onomatopoeias that are rare.

8.10 A Special case: Twitter

In the Twitter datasets, we must consider that the texts have some special characters used by the social platform to make relation between texts. With a similar way than Go et Al.[4], Vosoughi et al. [15] and Pak et al.[9] I used some rules with these special words:

- **Hashtags(starts with #):** I delete all the hash of the hashtags and leave the word. For example, "Im so #happy" gets converted to "Im so happy"
- **Usernames(starts with @):** I delete all the user-name because it hasn't any special effect on the sentiment analysis. Only references an individual.
- **URL, Links, Images..** All these things get deleted because they not have any special meaning by itself.

9 RESULTS

For each dataset, I execute all the machine learning methods and in each round, I apply different transformations with the objective to see if have a positive effect on the results. On each round, I optimize the method to archive the maximum performance.

9.1 Results with Movie Reviews

I have archived on this dataset better results than the established baseline by Pang et. al.[11] and a similar performance with Mullen et. al.[8]. One curious thing is if we look on the F-Score, in general, the capacity to predict positive is better than classify the negative examples. This can be because things like sarcasm, where we want to say a negative thing instead a positive or positive instead of negative, are more often used with a negative sentiment.

9.1.1 Unigrams

As we can see in the table A.1 with unigrams the effect of apply TF-IDF have a little better effect than count vectors. Also, apply lemmatization was better than apply Stemming. One thing is that the best results has been obtained without doing any additional transformations like remove the stop-words or the punctuations. This have a logical sense as we said in 8.7.

On the machine learning methods side the RNN and the SVM have the best performance followed by the Random Forests. This can be because the RNN is a special case of Neural Networks for time based problems and the SVM are easy to be tuned with less hyper-parameters, and the way that they work. The MLP also gives a good result with 2 layers with 1024 and 512 neurons. Must be mentioned that the MLP and RNN have a lot of hyper-parameters and with more time more accurate configuration can be studied.

9.1.2 Bigrams

In the table A.2 we can see similar results but with a different effect. Apply Count or TF-IDF gets more remarkable than apply another transform. This have the meaning that the stop-words and the punctuations can have any type of impact on the sentiment analysis.

One thing is that the direct conversion from unigrams to bigrams was impossible because it multiplies the number of features by 4, on this dataset. A similar way than Pang et. al.[11] was used to select the most useful features, as described in section 8.9. With the machine learning methods, the MLP and RNN archived similar performance and the SVM stays with the same performance. The Naive Bayes method now gets better results because in the bigrams the words relations gets more explicitly declared, for example the pairs "not happy" causes a negative relation and the sentiment is not only dependent by the word "happy".

9.2 Results with STS

With this dataset, I archived similar performance than Go et. al.[4] and Saif et. al.[13] but less than Vosoughi et. al.[15]. This is because the reduction of the dataset I applied and the transformations mentioned at 8.10.

9.2.1 Unigrams

As we can see on the table A.3 in this case the effect to delete the stop words, the punctuations and apply lemmatization or stemming have a positive effect on the results. This is because tweeter generates a huge vocabulary and apply these transformations can reduce it.

For the methods, we can appreciate better results with Decision Trees and Naive Bayes, compared to movie reviews. Some methods have problems to converge with this *dataset* and they required a special attention to finalize or the training process becomes infinite jumps between local minimums of the cost function. The RNN continues to be the best machine learning method.

9.2.2 Bigrams

On the table A.4 we can appreciate the same effect as the unigrams with the transformations. But, in general, the performance gets worst because the effect to generate pairs of words makes the vocabulary bigger.

On the methods side, all the methods have less performance in general. The worst drop in performance have been archived by SVM. The Naive Bayes, Decision Trees and RNN have resisted better.

10 Conclusions

I build, test and evaluate several machine learning methods for the Sentiment Analysis task. I learned a lot of things about how to face a machine learning problem and how to do data analysis to make the work easier to the machine to learn. I see that one of the most important things when we are facing a text classification problem is the type of text and the words that we can see in the data. This is because it has an important impact on the number of words that the machine learning methods will learn and, in consequence, the final number of features.

In this document, we can see that the effect of apply transformations on the data can improve the performance of the classification methods but the type of transformations depends on the dataset and the type of the language it has. In consequence, look the data, make a feature selection, apply transformations and filter the data that have less importance and information can make the machine learning method learn more efficiently and generalize better, because this days the machines have limitations and can't handle all the data without any kind of prior process.

In general, the machine learning methods tends to give similar results and, again, the results depend on the type of the data. A special mention to the Recurrent Neural Networks that the capacity to learn temporal dependences between words gives better results on all data that can be seen on this document.

10.1 Future Work

I suggest that a future work on the Recurrent Neural Networks can give better results. Because, they suggest that is one of the better methods for this type of problem. Also, with the emergence of specialized hardware and the ability to train with large datasets that does not fit in memory, using Gradient Descend, makes the neural networks the most promising method.

To improve the general performance more investigation must be done on learn with less transformations. This is because, all the text transformations have a loss on the quantity of information we process, as more transformations less information, and if we have less information the possibility to do a wrong classification augments.

Finally, work at character level can make a high impact on the vocabulary size because can be reduced from 50 thousand to, nearly, 3 hundred characters that can represent the most words in the language. Also, is the way that the humans read. The problem is that, today, the machine learning methods have problems processing the intermediate meaning behind the time dependency between words and, in the same way, the dependency between characters. In Twitter, for example, a sentence has around 20 words that means that the machine needs to learn the dependence between 20 time steps and interpret the hidden sentiment while reading. If we work at character level the machine needs to learn more than 300 hundred time steps. More investigation is needed to make the machines learn like humans reading the sentences at character level.

11 Acknowledgments

At first, thanks to Jordi Duran Cals to guide me with this final project and the opportunity to have all the freedom that I want to decide and work on this project with the objective to do my own investigation on the machine learning methods to process language and apply them to a real data.

To all my friends that I met during this four years Alejando Herrera, Ignacio Gordo, Waziri Ajibola, Francisco Gutiérrez, Aitor Herrera and more with whom we have always helped to achieve all the things. Also, a special mention to Hector Herranz he was my special comrade during this four years and he always help me a lot with all the things we do during this years.

In the life, there is always a person that means something special, that always stays near when you are in troubles, that you can always trust. I want to say thanks to Ruben Guix to help me during this years he was always an example of constancy and determination for me.

To my family, especially my couple Ariadna Ujaque and my little boy Carles Romero. They always stay with me and bring me up when I was down. The patience they had this four years of joys and sorrows. The way that they calm me when I was nervous. I will be always with them.

Thank you all.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[3] P. Raghavan C.D. Manning and H. Schuetze. Introduction to information retrieval, 2008.

[4] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[5] Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728, 2014.

[6] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, 2015.

[7] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[8] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418, 2004.

[9] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, 2010.

[10] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[11] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. 2013.

[14] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[15] Soroush Vosoughi, Helen Zhou, and Deb Roy. Enhanced twitter sentiment classification using contextual information. *arXiv preprint arXiv:1605.05195*, 2016.

APPENDIX

A.1 Results for unigrams and Movie Reviews

Model	Transforms	Acc	F1
RNN		87	87
SVM	TF-IDF	86	87
SVM	TF-IDF Lemmanize, remove StopWords Punctuations	86	87
RNN	Lemmanize, remove StopWords Punctuations	86	86
RBF	Count Lemmanize, remove StopWords Punctuations	86	85
Mullen et al 2004		**86**	
RNN	Stemming, remove StopWords Punctuations	85	85
MLP	TF-IDF	84	85
MLP	Count Lemmanize, remove StopWords Punctuations	84	85
RBF	CountVectorizer, unigram,200 estimators	84	84
SVM	TF-IDF Stemming, remove StopWords Punctuations	84	84
SVM	Count Stemming, remove StopWords Punctuations	84	84
MLP	TF-IDF Lemmanize, remove StopWords Punctuations	84	84
MLP	CountVectorizer, unigram	83	84
MLP	Count Stemming, remove StopWords Punctuations	83	84
SVM	CountVectorizer, unigram	83	83
MLP	TF-IDF Stemming, remove StopWords Punctuations	83	83
SVM	Count Lemmanize, remove StopWords Punctuations	83	83
Pang et al 2002		**82,9**	
RBF	Count Stemming, remove StopWords Punctuations	82	83
RBF	TF-IDF	82	82
RBF	TF-IDF Stemming, remove StopWords Punctuations	82	82
NB	CountVectorizer, unigram	81	81
NB	TF-IDF Stemming, remove StopWords Punctuations	80	80
NB	Count Stemming, remove StopWords Punctuations	80	80
RBF	TF-IDF Lemmanize, remove StopWords Punctuations	80	80
NB	TF-IDF Lemmanize, remove StopWords Punctuations	79	79
NB	Count Lemmanize, remove StopWords Punctuations	79	79
NB	TF-IDF	78	78
NB	Raw data, Char Level	55	51
RBF	Raw data, Char Level	54	54
SVM	Raw data, Char Level	53	53
MLP	Raw data, Char Level	51	42

A.2 Results for Bigrams and Movie Reviews

Model	Transforms	Acc	F1
Mullen et al 2004		**86**	
MLP	CountVectorizer, bigrams	85	85
MLP	TF-IDF, bigrams	85	85
RNN		85	84
SVM	TF-IDF, bigrams	84	85
RNN	Lemmanize, remove StopWords Punctuations	84	84
- RNN	Stemming, remove StopWords Punctuations	84	84
NB	CountVectorizer, bigrams	83	83
NB	TF-IDF, bigrams	83	83
Pang et al 2002		**82,9**	
SVM	CountVectorizer, bigrams	82	82
MLP	Count Stemming, remove StopWords Punctuations, bigrams	82	82
NB	Count Stemming, remove StopWords Punctuations, bigrams	81	81
NB	Count Lemmanize, remove StopWords Punctuations, bigrams	81	81
NB	TF-IDF Lemmanize, remove StopWords Punctuations, bigrams	80	81
SVM	TF-IDF Lemmanize, remove StopWords Punctuations, bigrams	80	81
NB	TF-IDF Stemming, remove StopWords Punctuations, bigrams	80	80
SVM	TF-IDF Stemming, remove StopWords Punctuations, bigrams	80	80
MLP	TF-IDF Stemming, remove StopWords Punctuations, bigrams	80	80
MLP	TF-IDF Lemmanize, remove StopWords Punctuations, bigrams	79	79
SVM	Count Stemming, remove StopWords Punctuations, bigrams	78	78
SVM	Count Lemmanize, remove StopWords Punctuations, bigrams	77	77
MLP	Count Lemmanize, remove StopWords Punctuations, bigrams	76	76
RBF	Count Lemmanize, remove StopWords Punctuations, bigrams	75	75
RBF	TF-IDF Stemming, remove StopWords Punctuations, bigrams	74	74
RBF	CountVectorizer, bigrams	73	73
RBF	TF-IDF, bigrams	73	69
RBF	TF-IDF Lemmanize, remove StopWords Punctuations, bigrams	72	72
RBF	Count Stemming, remove StopWords Punctuations, bigrams	71	70

A.3 Results for Unigrams and STS

Model	Transforms	Acc	F1
Vosoughi et al 2015		**86,2**	
RNN	Lemmanize, remove StopWords Punctuations	84	84
RNN	Stemming, remove StopWords Punctuations	84	84
RNN		83	84
Go et al 2009		**83**	
RBF	Count Stemming, remove StopWords Punctuations	81	81
Saif et al. 2013		**80,1**	
RBF	Count Lemmanize, remove StopWords Punctuations	80	80
RBF	CountVectorizer, unigram,200 estimators	78	79
RBF	TF-IDF Stemming, remove StopWords Punctuations	78	79
NB	Count Lemmanize, remove StopWords Punctuations	78	78
NB	TF-IDF	78	78
RBF	TF-IDF Lemmanize, remove StopWords Punctuations	77	78
NB	CountVectorizer, unigram	77	78
NB	Count Stemming, remove StopWords Punctuations	77	77
NB	TF-IDF Stemming, remove StopWords Punctuations	76	77
SVM	TF-IDF	76	76
NB	TF-IDF Lemmanize, remove StopWords Punctuations	76	76
RBF	TF-IDF	74	75
MLP	TF-IDF Stemming, remove StopWords Punctuations	74	74
SVM	TF-IDF Lemmanize, remove StopWords Punctuations	73	73
MLP	Count Lemmanize, remove StopWords Punctuations	72	73
SVM	TF-IDF Stemming, remove StopWords Punctuations	72	72
MLP	CountVectorizer, unigram	72	72
SVM	CountVectorizer, unigram	71	71
MLP	Count Stemming, remove StopWords Punctuations	71	71
SVM	Count Lemmanize, remove StopWords Punctuations	70	70
MLP	TF-IDF	70	69
SVM	Count Stemming, remove StopWords Punctuations	69	69
MLP	TF-IDF Lemmanize, remove StopWords Punctuations	69	69
RBF	Raw data, Char Level	52	50
NB	Raw data, Char Level	50	50
SVM	Raw data, Char Level	47	46

A.4 Results for Bigrams and STS

Model	Transforms	Acc	F1
Vosoughi et al 2015		**86,2**	
Go et al 2009		**83**	
Saif et al. 2013		**80,1**	
RNN		80	81
RNN	Stemming, remove StopWords Punctuations	79	78
RNN	Lemmanize, remove StopWords Punctuations	78	79
RBF	Count Stemming, remove StopWords Punctuations	78	78
RBF	CountVectorizer	76	76
NB	Count Lemmanize, remove StopWords Punctuations	76	76
RBF	TF-IDF Stemming, remove StopWords Punctuations	75	75
NB	TF-IDF	75	75
NB	CountVectorizer,	74	74
NB	Count Stemming, remove StopWords Punctuations	74	74
RBF	Count Lemmanize, remove StopWords Punctuations	73	73
RBF	TF-IDF Lemmanize, remove StopWords Punctuations	73	73
NB	TF-IDF Stemming, remove StopWords Punctuations	73	73
NB	TF-IDF Lemmanize, remove StopWords Punctuations	73	73
SVM	TF-IDF	72	72
RBF	TF-IDF	72	72
MLP	TF-IDF Stemming, remove StopWords Punctuations	71	71
SVM	TF-IDF Lemmanize, remove StopWords Punctuations	70	70
MLP	Count Lemmanize, remove StopWords Punctuations	70	70
SVM	TF-IDF Stemming, remove StopWords Punctuations	70	70
MLP	CountVectorizer	69	69
MLP	Count Stemming, remove StopWords Punctuations	69	69
SVM	CountVectorizer	69	69
SVM	Count Lemmanize, remove StopWords Punctuations	68	68
MLP	TF-IDF	67	67
SVM	Count Stemming, remove StopWords Punctuations	66	66
MLP	TF-IDF Lemmanize, remove StopWords Punctuations	65	65