Airlines tripadvisor's comments analysis using NLP

Letícia Moreira Valle - 18/0007904

***Abstract*—This paper is the final report of the class *oriented study II* and presents a study proposal of airline reviews in Portuguese, presenting wordclouds of the most influential words for either positive and negative comments using the tf-idf methodology and sentiment analysis of the comments using naive Bayes Classifier technique with some variation as negation tags and bigrams use.**

***Index Terms*—comments, nlp, tf-idf, sentiment analysis, bigrams**

I. Introduction

Social networks provide a powerful reflection of the structure and dynamics of the modern society. [1] Analyzing the data stored in social networks can help companies to understand the needs of the market and adapt their services to customers's expectations as well as direct customers to the most appropriate services to their profile.

The richest sources of service's user review are sites such as Google, Facebook, Yelp, TripAdvisor, and Foursquare. These reviews have great influence on new consumer's decision, especially when it comes to services, lodging and food. The main benefits of analyzing user's reviews are

- Improves online reputation of the company and its visibility on the internet
- Influences positively the decisions of other possible clients
- Facilitates analysis for continuous improvement of services and products
- Allows competitive analysis
- Marketing Intelligence for business

In order to benefit from review analysis, it is necessary to understand the human's natural language and how feelings are expressed through words. For this purpose, Natural Language Processing [2], usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The main objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable and usable by computers.

In parallel with the difficulty of analyzing the content, is the difficulty of obtaining data. In order to have access to comments from reviews sites in an automated way, without having to copy and paste each comment into a notepad, scientists usually use a data extraction method.

In this way, this work aims to analyze comments from airlines, a segment of the economy that has been much debated in Brazil in recent times, in an automated way using NLP applications to find the most influential words in the comments and the sentiment of the reviews, that is, whether the comment is positive or negative. To obtain the data for analysis, we will use Web scraping [6] as the data extraction method and perform a preprocessing using Python language.

A autora percente ao Departamento de Engenharia Elétrica da Universidade de Brasília. E-mail: leticia@redes.unb.br

II. Data extraction

Initially, a methodology based on Beautifulsoup python's library [3] was used but at some moment, an update occurred on TripAdvisor's website sctructure and they started using dynamic loading of HTML pages from javascript. This fact maked it necessary to use of dynamic scraping.

A. Using static scraping

Static scraping is sufficient for retrieving data from static lists [5]. Beautifulsoup library will be presented in next section.

1) Python's Beautifulsoup: Beautiful Soup is a Python library for pulling data out of HTML and XML files. It is a very powerful library that makes static web scraping by traversing the DOM (document object model) easier to implement. In this work it was used data from TripAdvisor's site. Fig. 1 shows the methodology used to extract the comments using this approach.

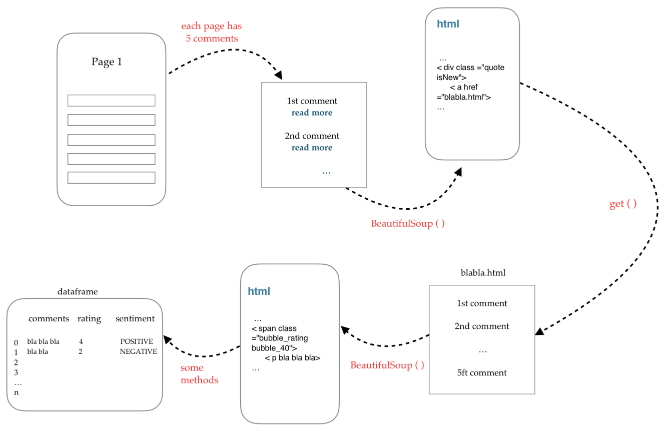


Figure 1. Method architecture using Beautifulsoup

First, we need to indicate the URL of the web page we want to scrape. In review page of TripAdvisor, the longer reviews are partially available in the final DOM. They become fully available only on clicking "More" button. As a consequence, it is necessary to open comments automatically by clicking on 'Read More'. Like BeautifulSoup does not interact with the page, thus, we have to find a html page that has all the comments already opened. The link of the fully opened html page is hiden in a *href* tab, so it's necessary to make a html request to find this *href* tag and make another requisition using this new link address. Each page shows five comments, so it's not necessary to make it for each comment but for each page. Once we have a html with all the comments completed, we use some BeautifulSoup methods to get the comments and the ratings. In this work, we consider a NEGATIVE comment the reviews with grades 1 and 2 and a POSITIVE comment the reviews with grade 3, 4 and 5.

This method worked well until TripAdvisor updates it's site and become to use dynamic html load. As BeautifulSoup does only static scraping and fetches web pages from the server without the help of a browser, it ignores JavaScript. You get exactly what you see in "view page source", and then you slice and dice it. But if you need data that are present in components which get rendered on clicking JavaScript links, dynamic scraping is needed. Next session will explain how to implement a dynamic scraping method.

B. Using dynamic scraping

One of the ways of making dynamic scraping is using the combination of Beautiful Soup and Selenium, a framework that perform various types of tasks with the browser. Another option is the utilization of dynamic web scraping solution like Web Scraper, Google's platform for scraping from browser. Both methods takes into account the Javascript of the page and enable dynamic html loading.

1) Beautilsoup with Selenium: The architecture of the solution based on the combination of BeautifulSoup and Selenium is presented in Fig. 2.

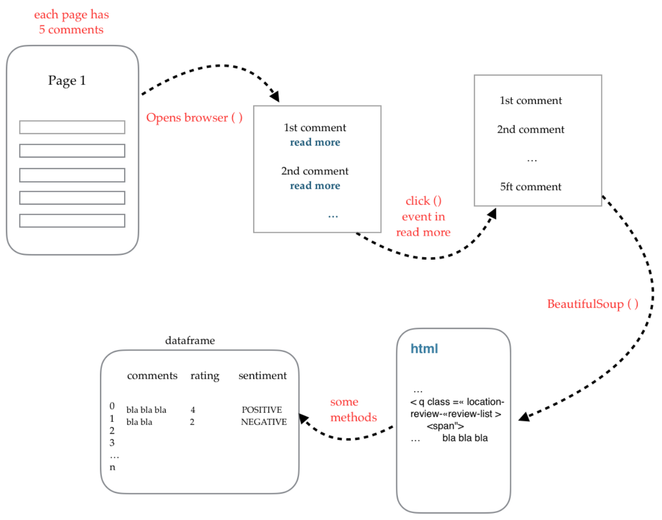


Figure 2. Method architecture using Beautifulsoup and Selenium

As before, the first step is to indicate the URL of the web page we want to scrape and then open the entire comment. For this purpose, we will use Selenium to automate the button clicks required for rendering hidden but useful data. In this step, it is necessary to open the browser, wait pages load and then click on "Read more" buttom. After open all comments, we use Beautifulsoup to get hmtl content and retrieve information for the dataframe.

Using the combination of BeautifulSoup and Selenium allows the creation of automated scripts. Because WebDriver, used by Selenium, uses a real browser to access the web site, there is no difference than browsing the web by a human and like it loads all the website resources and keeps all cookies created by the website, it is really difficult do determine whether a real person or a robot accessed the website [5]. As another advantage, selenium framework supports many browsers (Internet explorer, Chrome, Firefox, Opera, Safari).

2) Web Scraper - Google Chrome: Web Scraper [6] is a Google Chrome extension to performe web scraping by creating a plan of how a web site should be traversed and should be extracted. It can scrape multiple pages, extract data from dynamic web pages and export data as CSV.

In this work, the plan was constituted of 3 main elements: pagination, load more and wrapper. Pagination is a Link structure that makes the application search for comments on all pages of the specific airline. Load more is an element click that opens "Read more" tabs before scraping comments and wrapper is element that contains other elements. In this case, the wrapper element contains the comment text, title and the grade attributed by the user. Fig. 3 presents the selector graph used in this work.

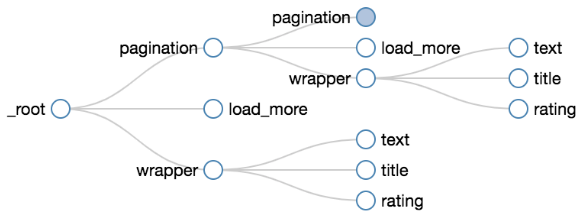


Figure 3. Selector graph used by Web Scraper.

To create a plan, we need first open Google's Chrome inspector and find tab "Web Scraper". Then, we create a sitemap and some selectors. These selectors interact directly with the web page in order to facilitate the visualization of the information that the user wants to obtain. After constructing the selectors, the user clicks in the scrape button and waits the tool to finish the job. In this step, the data can be exported in a CSV file. Fig. 4 shows the architecture used in this method.

Using a Browser extension is really intuitive way of doing web scraping. It hides the complex part of the user and allows a simple interaction with the selectors and the web page, facilitating the visualization of the data through previews. People who do not have knowledge in a specific programming languages can use this tool without difficulty.

III. Text Data Cleaning and Preprocessing

Before begining to analyse the text, we need to clean the data to get rid of the less useful parts of text through stopword

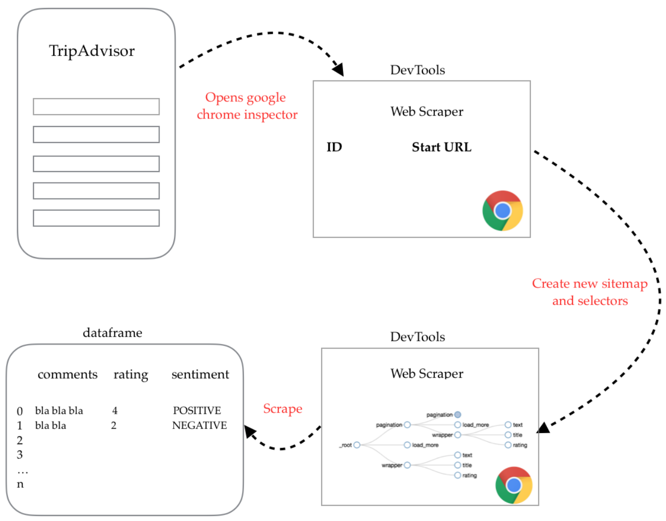


Figure 4. Architecture using Web Scraper Google's extension.

removal, dealing with capitalization or characters and other details.

A. *Lowercase*

Sometimes, the model might treat a word which is in the beginning of a sentence with a capital letter different from the same word which appears later in the sentence but without any capital latter. This might lead to decline in the accuracy. So, it is necessary to lowercase all the words in comments.

B. *Noise removal*

It's the removal of characters digits, tags and pieces of text that can interfere with the text analysis. Noise removal is one of the most essential text preprocessing steps. It is also highly domain dependent.

C. *Stop Words*

From our intuition, we think that the words that appers more frequent in text should have a greater weight in textual data analysis, but that's not always the case. Words such as 'the', 'and', 'a', 'for' — called stopwords — appear the most in a corpus of text, but are of very little significance and do not contribute much to the content or meaning of a document.

After complete the text cleaning step, we can implement NLP applications.

IV. Comment's most influential words

To find most influential words in text we will use TF-IDF [7] method. TF-IDF stands for "Term Frequency — Inverse Data Frequency". The TF part gives us the frequency of the word in each document. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. The words that appears more have higer TF.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \tag{1}$$

The term IDF, in it's turn, calculates the weight of rare words across all documents. The words that occurs rarely in the document have a higher IDF score.

$$idf_{i,j} = \log\left(\frac{N}{idf_i}\right) \tag{2}$$

The combination of these two algorithms create the TF-IDF score (w) for a word in a document. It is the product of tf and idf

$$wi,j = tf_{i,j} \times \log\left(\frac{N}{idf_i}\right) \tag{3}$$

in which $tf_{i,j}$ is the numer of occurences of i in j, df_i is the number of documents containing i and N is the total number of documents.

In this work, we want to find the most influential words for either positive and negative comments, so we need to separate the list of comments before starting the analysis using the labels applied earlier. After, for each list, we chose to ignore words that appear in more than 85% of the document and then calculate the IDF using *TfidfTransformer* from scikit-learn library. Next, we get each comment (document) we want to find the most influential words and generate the ft-idf using *transform* method and the content calculated just before. At this we have the TF-IDF scores calculated for each word in each document and we can perform a sort algorithm to sort the tf-idf vectors by descending order of scores. For last, we extract the top n to print out and make a *WordCloud* with all the words and its height. In order to find most meaningful results, we decided to work with bigrams, a pair of consecutive words, instead if always considering just one word. In this way, the algorithm will try to find the individual words and doubles of words that have more meaning in the context.

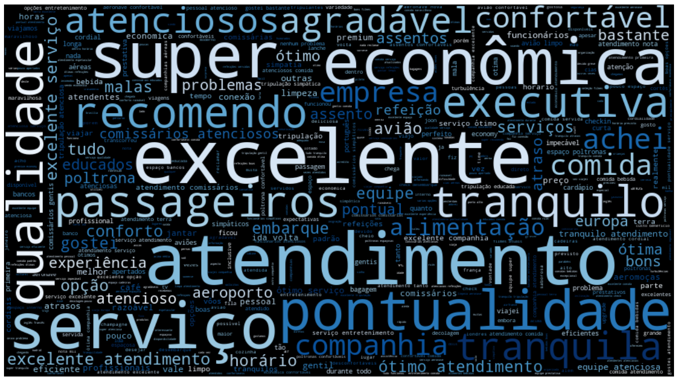


Figure 5. WordCloud with the most influnential words of positive comments.

We can see in the Fig. 5, what are the most influential words in the context of positive comments. We can infer that people finds important the punctuality, the service and the treatment. The majority of the comments were about situations including the economic class. We can also infer that the users liked the attendance of the crew, using words like "attentive commissioners", "attentive staff", "helpful staff", kind, pleasant and others. The majority of the words are positive and show which characteristics most users take into consideration when evaluating an experience.

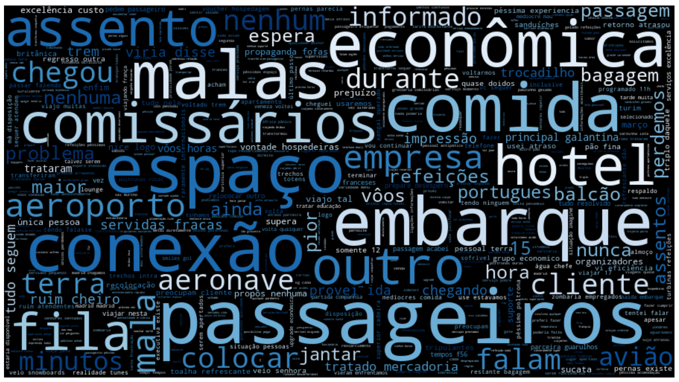


Figure 6. WordCloud with the most influnential words of negative comments.

For the negative comments, presented in Fig. 6, the most influential words are "bags", "food", "space", "boarding", "passenger", "row", "seat" and "crew". We can observe users probably had problems with the bags, did not like the food, found small the space between the chairs and etc. With key words, it is easier to understand customer needs and work to change what they dislike.

V. Sentiment Analysis

Sentiment Analysis is an extremely powerful tool and can be used in several applications in several different segments. This section is devoted to binary sentiment analysis using the Bag of Words approach and the Naive Bayes Multinomial algorithm [8]. Bag of Words, in practice, creates a vector with each of the words presented in the whole text and then calculates how often these words occur in a given sentence, then classify and train the model. Multinomial Naive Bayes classification algorithm tends to be a baseline solution for sentiment analysis task. The basic idea of Naive Bayes technique is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes. Given a feature vector table, the algorithm computes the posterior probability that the document belongs to different classes and assigns it to the class with the highest posterior probability. There are two commonly used models (i.e., multinomial model and multi-variate Bernoulli model) for using Naıve Bayes approach for text categorization [9].

Given the dependent feature vector $(x_1, \ldots, x_n)$ and the class Ck. Bayes' theorem is stated mathematically as the following relationship [10]

$$P(C_k|x_1, ..., x_n) = \frac{P(C_k)P(x_1, ..., x_n|C_k)}{P(x_1, ..., x_n)} \tag{4}$$

Since $P(x_1, \ldots, x_n)$ is constant, if the values of the feature variables are known, the following classification rule can be used [9]

$$\hat{y} = \arg\max_k P(C_k) \prod_{i=1}^{n} P(x_i|C_k) \tag{5}$$

This distribution is parametrized by vector Θ_k for each class C_k, estimated by a smoothed version of maximum likelihood. Θ_{ki} is the probability $P(x_i|C_k)$ of feature i appearing in a sample that belongs to the class C_k.

$$\hat{\Theta}_{ki} = \frac{N_{ki} + \alpha}{N_k + \alpha n} \tag{6}$$

in which Nki is the number of times feature i appears in a sample of class k and Ny is the total count of all features for class Ck. α is a smoothing priors for features not present in the learning samples. Setting $\alpha < 1$ is named Lidstone smoothing and setting $\alpha = 1$ transforms into the Laplace smoothing. Thus, the final decision rule is defined as

$$\hat{y} = \arg\max_k \left(\ln P(C_k) + \sum_{i=1}^{n} \ln \frac{N_{ki} + \alpha}{N_k + \alpha n} \right) \tag{7}$$

In this work we performed Multinomial Naive Bayes in three different approaches. First we will perform a simple model using Pipeline to make easier to work. Next, we will improve the first approach adding negations tags algorithm to the pipeline. To finish, we will use the Multinomial Naive Bayes model with bigrams. Instead of vectorizing the text "by word", we will vectorize it for each "two words" to add meaning to the model.

A. *Naive Bayes model*

First we need to equalize the amount of positive and negative comments for the algorithm to work correctly and separate the two columns we want to work with: comments text and sentiment tags. Then, we use the cleaning data algorithm discussed earlier to remain only the important and significant words in each comment.

In order to make the vectorizer => transformer => classifier easier to work with, scikit-learn provides a Pipeline class that behaves like a compound classifier [11]. Next, we will use the pipeline to perform grid search for suitable hyperparameters. We will train the model using the *fit* method and the pipeline. After implementing the model, we perform a cross-validation of the model. In this case, the model is divided into 10 parts, 9 for traing and 1 for testing.

For measuring performance, Accuracy (Acc), Sensitivity (Sn), Specificity (Sp), and the area under a Receiver Operating Characteristic (ROC) curve, also known as Area Under the Curve (AUC) are four commonly used parameters to compare the performance of the competing techniques [12]. In this work, due to the nature of the data, we will use Accuracy parameter using the *metrics.accuracy_score* method showing also the confusion matrix and ROC curve parameter using *metrics.auc*.

In sequence, to understand whats metrics like *Precision*, *Recall* and *F1-Score* means, we first have to know about the terminology for classification.

1) True positive (TP):: Correct classification of the positive class. For example, the actual class is Positive and the model predicted Positive.

2) True negative (TN):: Correct classification of the negative class. For example, the actual class is Negative and the model is classified as Negative.

3) False positive (FP):: Misclassification of the positive class. For example, the actual class is Negative and the template is classified as Positive.

4) False negative (FN):: Misclassification of the negative class. For example, the actual class is Positive and the template is classified as Negative.

The precision is calculated as follows

$$\text{precision} = TP/(TP + FP) \tag{8}$$

This means the number of times a class was predicted correctly divided by the number of times the class was predicted.

The recall is calculated as

$$\text{recall} = TP/(TP + FN) \tag{9}$$

This means the number of times a class was predicted correctly (TP) divided by the number of times the class appears in the dataset (FN).

For the F1-score, it is calculated as

$$\text{F1score} = 2 * ((\text{precision} * \text{recall})/(\text{precision} + \text{recall}) \tag{10}$$

This measure is the harmonic average between precision and recall. With this information, we can say the performance of the classifier with an indicator only.

Finally, the accuracy can be calculated as:

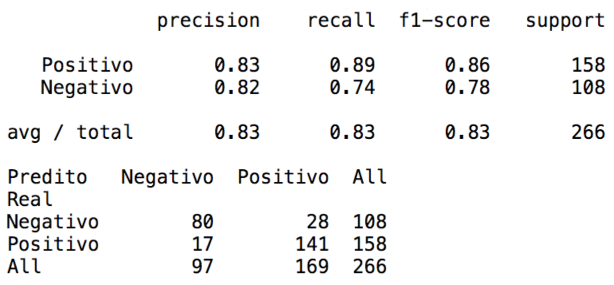
$$\text{Accuracy} = (TP + TN)/(TP + FP + TN + FN) \tag{11}$$

Accuracy shows us how the classifier performed on a general way. Fig.7 presents the metrics for the first approach.

We can see that this first approach has very good results. We can preview 83% of positive comments and 82% of negative comments.

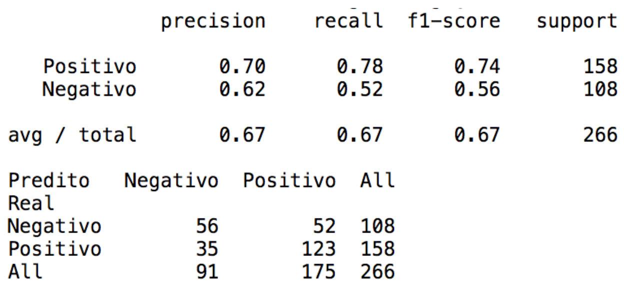
B. Naive Bayes neg-tags model

As used in the first approach, this one uses pipeline method. The difference is that this time we try to detect negative words in text, like *não, nao, not* using *CountVectorizer* and *tokenizer*.



```
             precision    recall  f1-score   support

   Positivo       0.83      0.89      0.86       158
   Negativo       0.82      0.74      0.78       108

avg / total       0.83      0.83      0.83       266

Predito   Negativo  Positivo  All
Real
Negativo        80        28  108
Positivo        17       141  158
All             97       169  266
```

Figure 7. Validation measures of the model: Accuracy and confusion matrix.



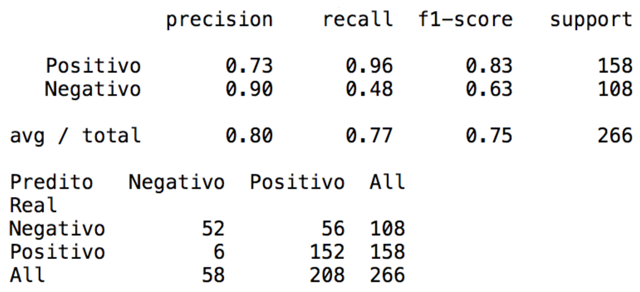
```
             precision    recall  f1-score   support

   Positivo       0.70      0.78      0.74       158
   Negativo       0.62      0.52      0.56       108

avg / total       0.67      0.67      0.67       266

Predito   Negativo  Positivo  All
Real
Negativo        56        52  108
Positivo        35       123  158
All             91       175  266
```

Figure 8. Validation measures of the model: Accuracy and confusion matrix.

Fig.8 presents the metrics. We can see that the second approach performs a little worse than the first approach. We can preview 70% of positive comments and 62% of negative comments.

C. Naive Bayes with Bigrams model

This thrid approach, instead of vectorizing the text "by word", we will vectorize it for each "two words". For this, we will use the parameter *ngram_range=(1,2)* on *CountVectorizer*.



```
             precision    recall  f1-score   support

   Positivo       0.73      0.96      0.83       158
   Negativo       0.90      0.48      0.63       108

avg / total       0.80      0.77      0.75       266

Predito   Negativo  Positivo  All
Real
Negativo        52        56  108
Positivo         6       152  158
All             58       208  266
```

Figure 9. Validation measures of the model: Accuracy and confusion matrix.

Fig.9 presents the metrics. We can see that the third approach performs a little worse than the first approach to positive comments but performs a little better than the first approach to negative comments. With this method we can preview 73% of positive comments and 90% of negative comments.

For measuring performance with another metric, we will present the ROC curve or each model. The ROC curve compares the model true positive and false positive rates to

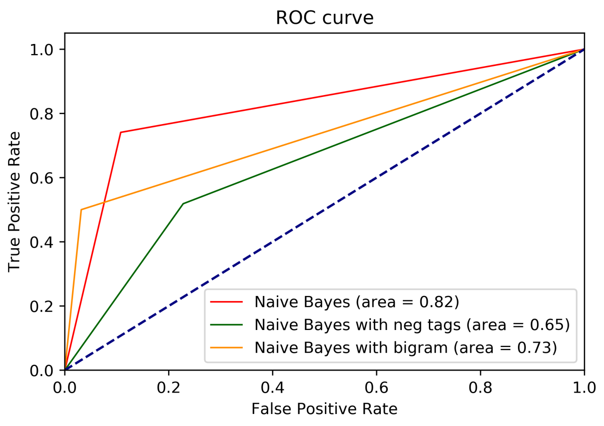


Figure 10. ROC curve for comparing models performance.

the ones from a random assignation. If the model roc is above the baseline, then the model is better than random assignation.

We can observe by using Fig. 10 that the Naive Bayes model without neg tags and bigrams performed better than the other models, having 82% chance of distinguish between positive class and negative class. This results corroborates with the results presented previously using accuracy as performance metric.

VI. Conclusion

This work presented the results of some NLP analysis on TripAdvisor's airlines reviews in Portuguese. The paper presented how data extraction was performed using either static and dynamic scraping depending on the web site implementation and how TF-IDF methodology was implemented in order to find most influential words for either positive and negative labed comments. This implementation can be very useful to industry in order to find most common problems and most positive characteristics in users' opinions.

Also, a sentiment analysis was performed in order to predict positive and negatives comments automatically, extracting subjective information in comments, and helping airlines business to understand the social sentiment of their brand.

NLP applications in Portuguese are still in development and there are few content on the internet, showing it as an area of great scientific interest for the country.

References

[1] Dunbar, Robin. (2008). Cognitive Constraints on the Structure and Dynamics of Social Networks. Group Dynamics: Theory, Research, and Practice. 12. 7-16. 10.1037/1089-2699.12.1.7.

[2] M Nadkarni, Prakash Ohno-Machado, Lucila Chapman, Wendy. (2011). Natural language processing: An introduction. Journal of the American Medical Informatics Association : JAMIA. 18. 544-51. 10.1136/amiajnl-2011-000464.

[3] Beautiful Soup Documentation. Available on: http://bit.ly/30gZRvD.

[4] Google's Chrome Web Scraper extension. Available on: https://webscraper.io/

[5] Aydin, O. (2018). R Web Scraping Quick Start Guide: Techniques and tools to crawl and scrape data from websites

[6] Mitchell, R. (2015). Web Scraping with Python, 2nd Edition.

[7] Bruno, Trstenjak Sasa, Mikac Donko, Dzenana. (2013). KNN with TF-IDF based framework for text categorization. Procedia Engineering. 69. 10.1016/j.proeng.2014.03.129.

[8] Kibriya, Ashraf Frank, E Pfahringer, Bernhard Holmes, Geoffrey. (2004). Multinomial naive Bayes for text categorization revisited. Advances in Artificial Intelligence. 488-499.

[9] Tan, Songbo Zhang, Jin. (2008). An empirical study of sentiment analysis for Chinese documents. Expert Systems with Applications. 34. 2622-2629. 10.1016/j.eswa.2007.05.028.

[10] Sentiment Analysis of Tweets using Multinomial Naive Bayes. Available on: http://bit.ly/2JeGHkm

[11] Working With Text Data. Available on: http://bit.ly/32e3bJA

[12] BahadarKhan K, A Khaliq A, Shahid M (2016) A Morphological Hessian Based Approach for Retinal Blood Vessels Segmentation and Denoising Using Region Based Otsu Thresholding. PLOS ONE 11(7): e0158996. https://doi.org/10.1371/journal.pone.0158996