Data Mining Project

Elena Veltroni, Filippo Minutella

February 2020

Contents

1 Introduction

The project born with the collaboration with the Faculty of Economy.
The main goal of the project was to implement an application in which the user can perform clustering and classification analysis on company acquisition articles.
In particular, we pre-processed the data provided by the colleagues in the Faculty of Economics, performing a data cleanup in order to prepare data to send to the classifiers and to the clustering algorithm.
The result we obtain can be useful in trading and others analysis performed by Faculty of Economy.
The users can perform the analysis with a simple web app described below.

2 Dataset

2.1 Data provided

The dataset is composed by some selected sentences in documents that talk about the company acquisitions.
The starting point are articles from the economic sector dealing with company acquisitions. The article is not taken in its totality, but it is divided into sentences, the decision to select a sentence is made by the subjective feeling which the sentence give to the user, only for these sentences a class is assigned.

Figure 1: Dataset organization

When a sentence is selected the user classified it with a label indicating if the sentence is *Negative*, *Neutral* or *Positive* according to the user feeling.
This job is done by the colleagues in the Faculty of Economy that provide us an excel file containing the labeled sentences.
In this file we had to perform some cleaning tasks, such as delete sentences written in Italian (the almost totality of the sentences was in English) and make the file interpretable for a Python script.
The first version of the dataset provided to us was highly imbalanced, this caused, in the first version of the classification algorithms, poor classification performances.
So with the objective to improve the classification performances the dataset was balanced and the current class disposition is:

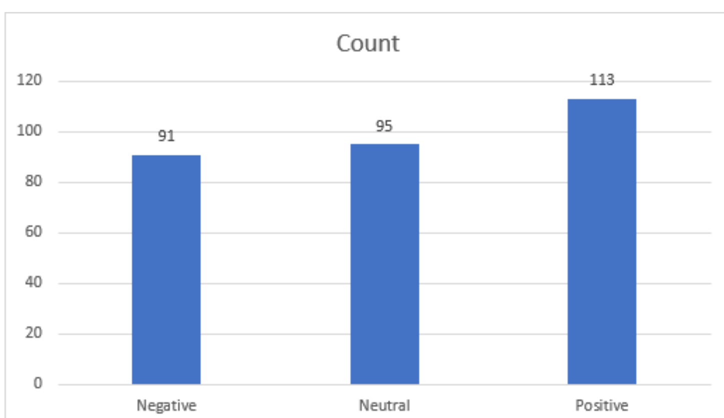


Figure 2: Dataset balanced

2.2 Data by twitter

The dataset of this application also contains a part of data retrieved from Twitter.
Twitter makes available to external users official APIs to search for tweets within the platform, restricting the search for tweets by specifying some parameters, but they show a big limit that is the possibility to retrieve only tweets published in the previous 7 days. To get around this limitation, the *GetOldTweets3* library for Python has been used, a library that allows to search by username or keyword, and allows us to specify the period of the tweets wanted.

3 Text Elaboration

There not exists algorithm that works with sentences directly so we have to get numbers from sentences that represent the sentences to fit the algorithms.

3.1 Bag Of Words

We have followed two way, the first, the simpler, uses Bag of Word (*BoW*) algorithm.

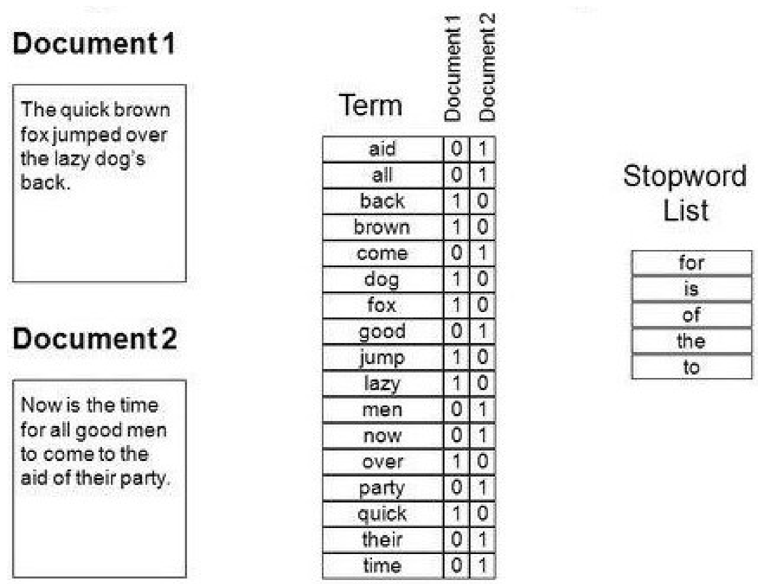


Figure 3: BoW Example

Bow works creating vector that represent the presence or absences of a word in a sentence. The words taken under consideration for presences is the K words most frequent in the dataset excluding the stop words.

3.2 Bidirectional Encoder Representations from Transformers

The second way used to elaborate the text is using Sentence-Transformers a repository that fine-tunes *BERT* / *RoBERTa* / *DistilBERT* / *ALBERT* / *XLNet* to produce semantically meaningful sentence embeddings.

BERT makes use of Transformer, a mechanism that learns contextual relations between words in a text. Transformer includes two separate mechanisms, an

encoder that reads the text input and a decoder that produces a prediction for the task.
As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once, therefore it is considered bidirectional. This characteristic allows the model to learn the context of a word based on all of its surroundings.
BERT uses two training strategies:

- The first model is called Mask Language Model (*MLM*) - and is used to predict a few words and self-check that you have actually understood what you are talking about.
  Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.

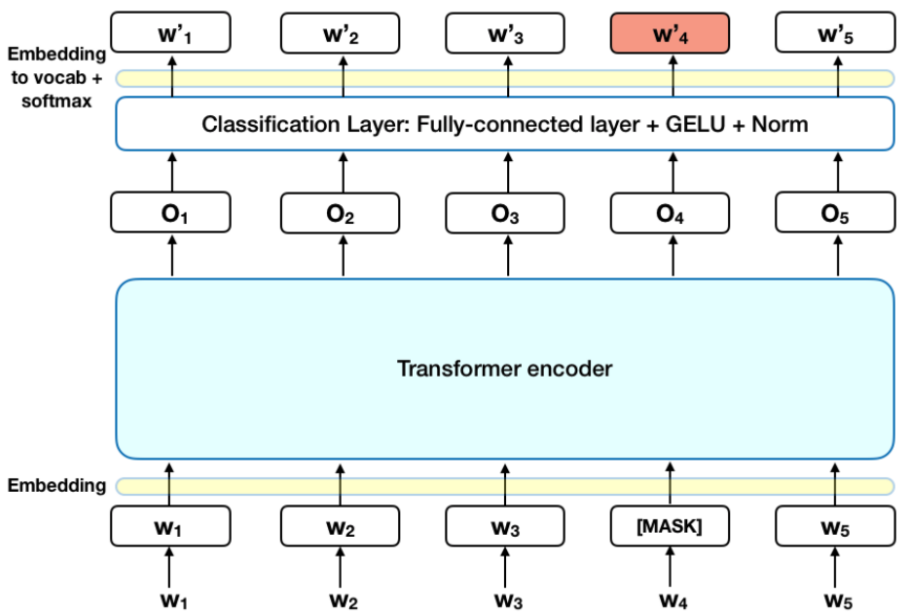


Figure 4: MLM training

- The second technique is called Next Sentence Prediction (*NSP*) and is used by BERT to relate sentences to each other. In the training process, the model receives pairs of sentences as input and learns to predict whether the second sentence of the pair is the next sentence in the original document.

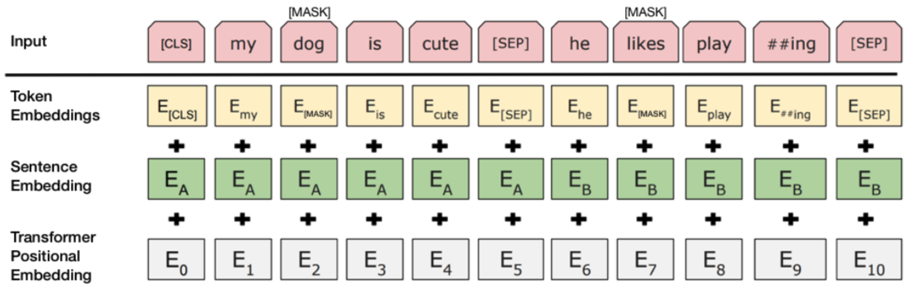


Figure 5: NSP training

4 Classification

We perform classification using 3 algorithms: K-NN, SVM, Logistic Regression. These algorithms was trained using the 768 features returned by the Sentence-Transformers (using the model bert-base-nli-mean-tokens) and the class presented in the dataset.

K-NN

	a	b	c
a = -1	69	9	13
b = 0	41	17	37
c = 1	28	26	62
Precision	0.5	0.3269230769230769	0.5535714285714286
Recall	0.7582417582417582	0.17894736842105263	0.5344827586206896
F-Measure	0.6026200873362445	0.2312925170068027	0.5438596491228069
Accuracy		0.5017241379310345	
Standard Deviation		0.05399669205919909	

Figure 6: K-NN confusion matrix

SVM

	a	b	c
a = -1	58	18	15
b = 0	26	35	34
c = 1	18	34	64
Precision	0.5686274509803921	0.40229885057471265	0.5663716814159292
Recall	0.6373626373626373	0.3684210526315789	0.5517241379310345
F-Measure	0.6010362694300517	0.3846153846153846	0.5589519650655022
Accuracy		0.5486206896551724	
Standard Deviation		0.08782098138882705	

Figure 7: SVM confusion matrix

Logistic Regression

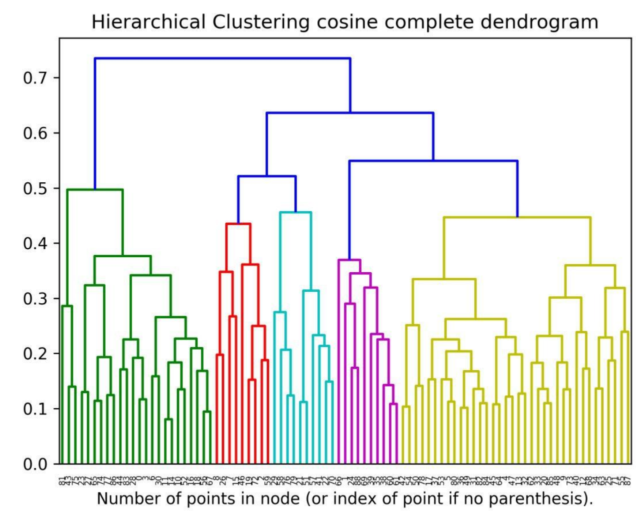
	a	b	c
a = -1	54	17	20
b = 0	20	41	34
c = 1	18	32	66
Precision	0.5869565217391305	0.4555555555555555	0.55
Recall	0.5934065934065934	0.43157894736842106	0.5689655172413793
F-Measure	0.5901639344262295	0.44324324324324327	0.559322033898305 1
Accuracy		0.5220689655172414	
Standard Deviation		0.07282096800301242	

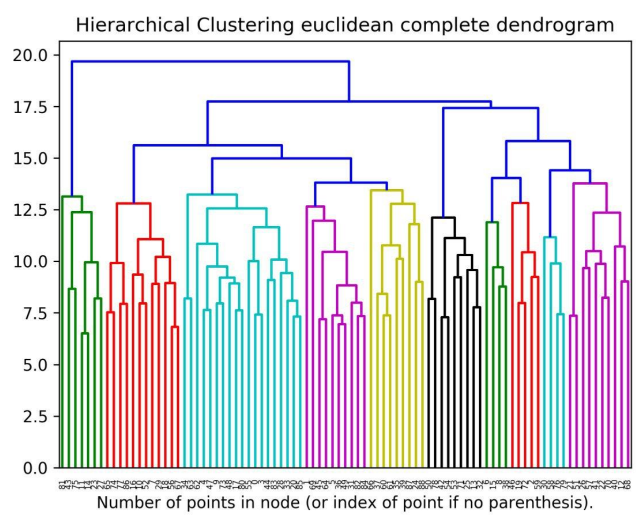
Figure 8: Logistic Regression confusion matrix

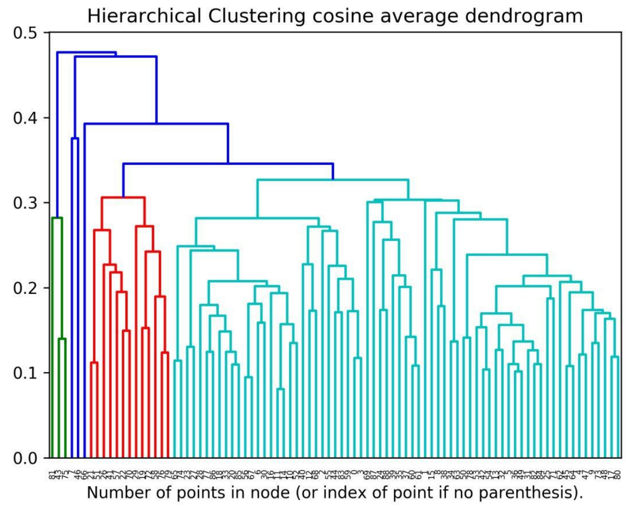
The classification performances of the three classifiers are not so good, we have low precision and recall specially for the neutral class.

5 Clustering

The clustering was performed using an agglomerative clustering with different linkage metrics and affinity measures based on the BERT returned features. The results of the different clustering are the dendograms and the cluster membership of the sentences.







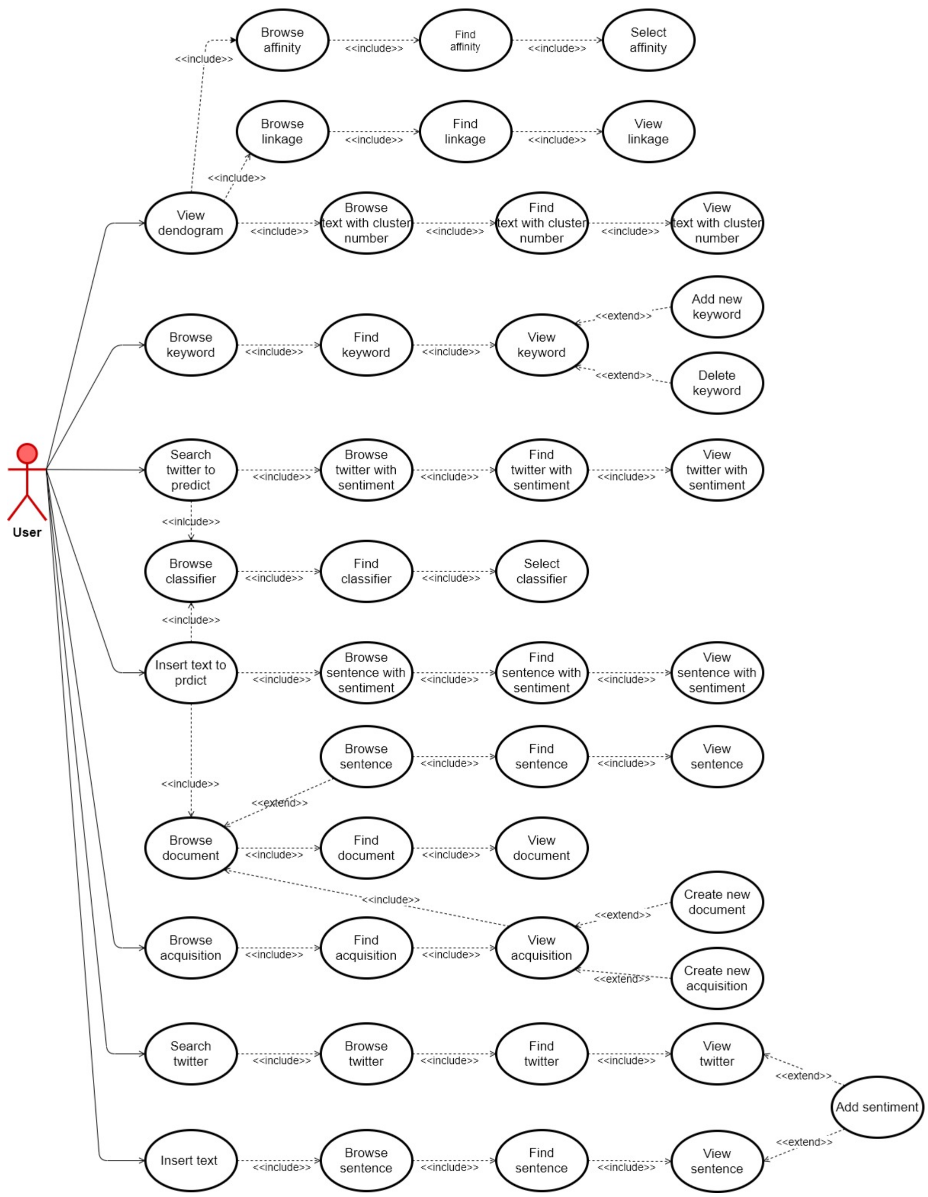
Not all dendograms have a clear organization of the cluster, in the last one is not clear how the sentences are grouped while the first two present a clearer organization so its more simple decide a threshold to cut the dendogram and get the class memberships. These clustering methods allow us to interpret how the cluster are formed and then provide the cluster membership to colleagues in the Faculty of Economy to evaluate the created cluster.

6 Application

All the functionalities seen before, are implemented in a web app developed using AngularJS for the front-end and Python for the back-end. The application uses MongoDB to store all the data used in the application.
The main actor of the application is a generic user who will use the functionalities offered by the web app.
The main actions allowed by the application are the following:

- Predict the sentiment of a sentences, choosing the classifier.
- Visualize the clustering dendogram, the text and the relative cluster number, choosing the affinity measure and linkage metric.
- Visualize the analytics, in order to see the acquisitions, documents and related sentences stored in the DB.
- Add new acquisitions and documents.
- Manage the keyword in order to improve the accuracy of filtering on the sentences to be predicted.
- Add new data to the training set to improve the accuracy of the classifier.

6.1 Use Case


User
Browse affinity
Find affinity
Select affinity
Browse linkage
Find linkage
View linkage
View dendogram
Browse text with cluster number
Find text with cluster number
View text with cluster number
Browse keyword
Find keyword
View keyword
Add new keyword
Delete keyword
Search twitter to predict
Browse twitter with sentiment
Find twitter with sentiment
View twitter with sentiment
Browse classifier
Find classifier
Select classifier
Insert text to prdict
Browse sentence with sentiment
Find sentence with sentiment
View sentence with sentiment
Browse sentence
Find sentence
View sentence
Browse document
Find document
View document
Browse acquisition
Find acquisition
View acquisition
Create new document
Create new acquisition
Search twitter
Browse twitter
Find twitter
View twitter
Add sentiment
Insert text
Browse sentence
Find sentence
View sentence
<<include>>
<<extend>>

6.2 Application manual

Once started the application the window below will be displayed.
With the tab it is possible to choose the type of data on which to make the sentiment prediction, text or twitter.
With the first tab you must enter the entire text and the document to which it belongs. The text will be divided into sentences, splitted or by dot or by a special character specified by the user, and only those containing at least one keyword, specified in the appropriate section, will be shown.

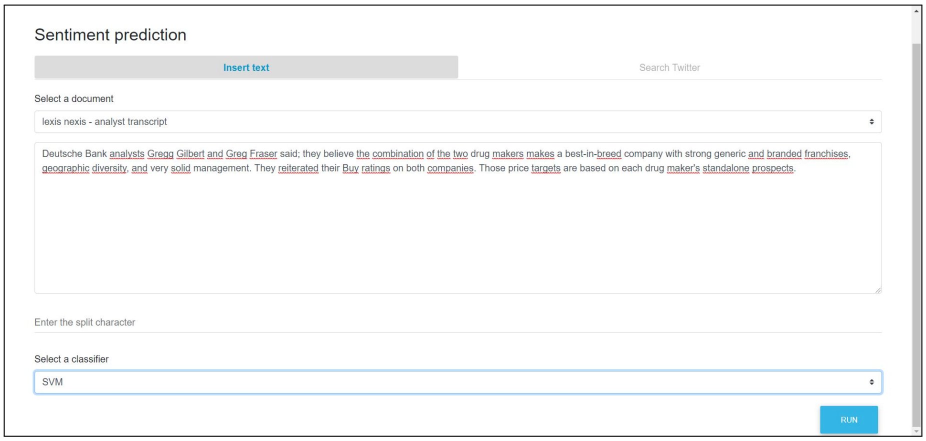


Figure 9: Home page - text

With the twitter tab you can search for keywords or username, specifying @ in front of the word, on twitter by narrowing the search to a specific period.

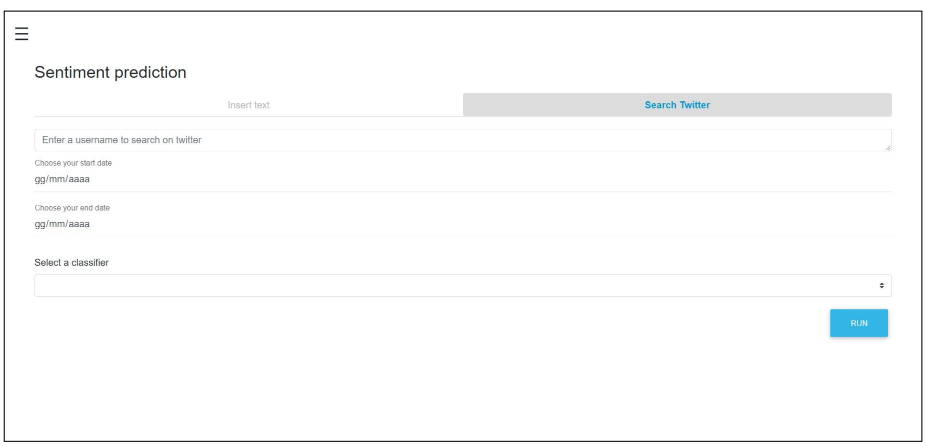


Figure 10: Home page - twitter

Before making the prediction, in both cases, it must be specify the type of classifier to use.
Once going forward, the application shows the selected phrases and the related feeling. If the prediction is good, the user has the possibility to add the result to the training set in order to improve the accuracy of the classifiers.

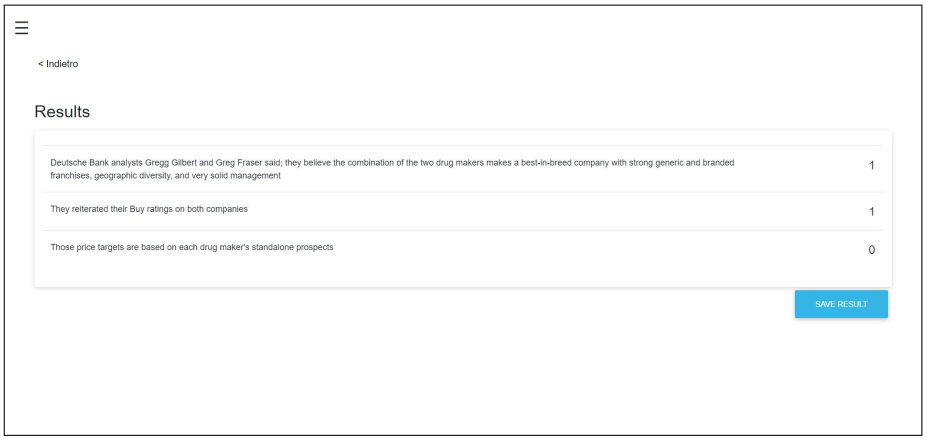


Figure 11: Prediction result

The other features of the application are listed in the side menu.

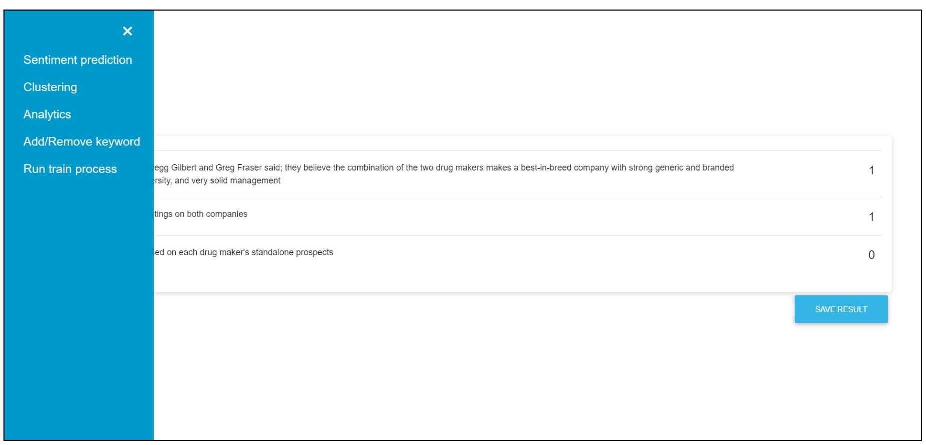


Figure 12: Menu

The other main function of the application is clustering. On the page below the user can cluster the inserted documents and see how the clusters are formed. The user must choose the linkage metrics and the affinity measure.

Figure 13: Clustering setting page

After that, the page shows the resulting dendogram and the list of documents with their cluster number.

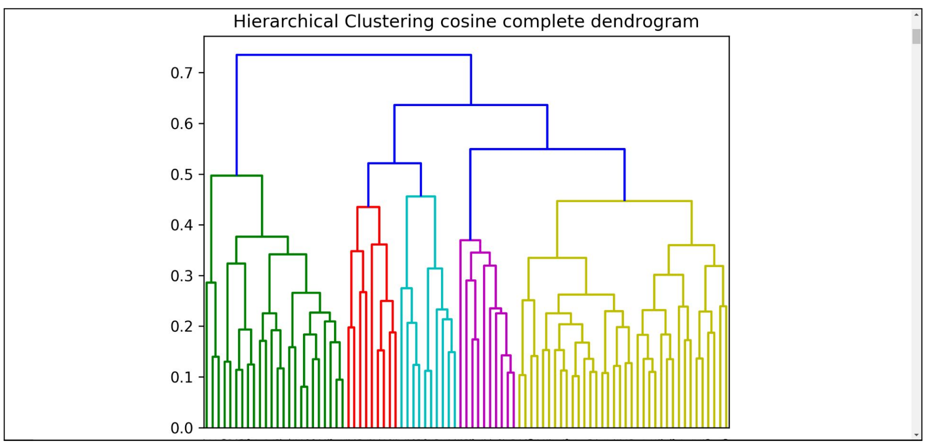


Figure 14: Clustering dendogram

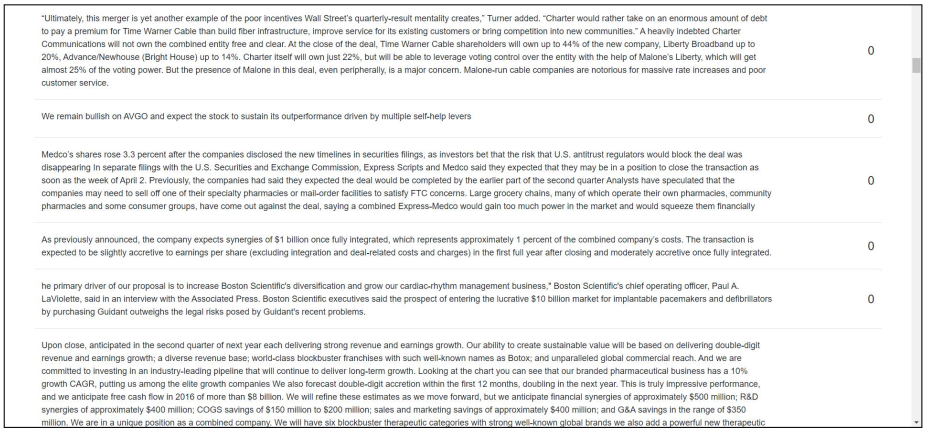


Figure 15: Clustering result

The analytics page summarizes all the acquisitions, documents and sentences with the respective sentiment saved in the DB.

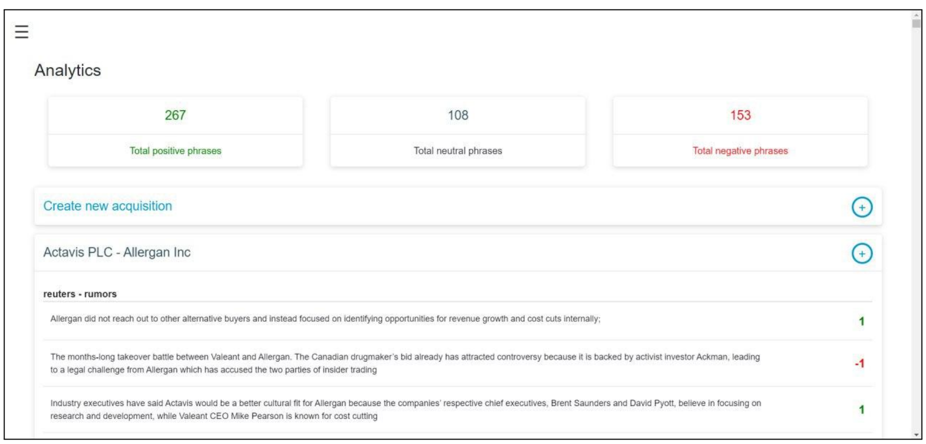


Figure 16: Analytics page

In this page the user also has the possibility to create a new acquisition or add a new document in one of the already existing acquisitions.

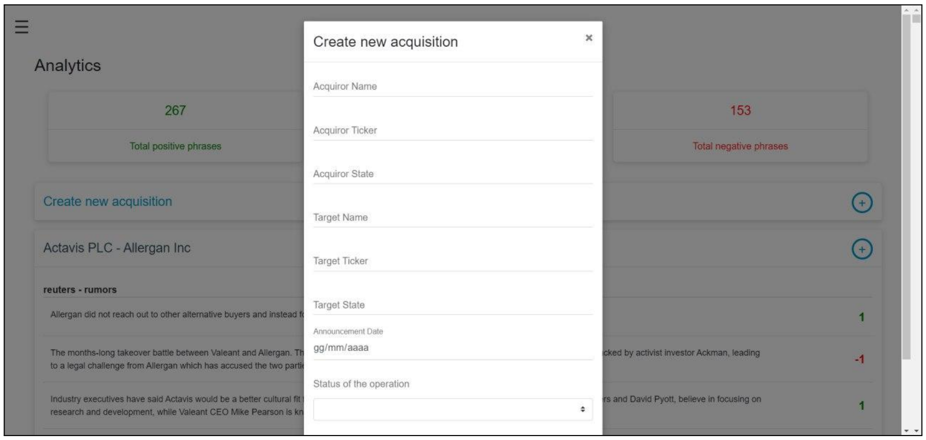


Figure 17: Add acquisition page

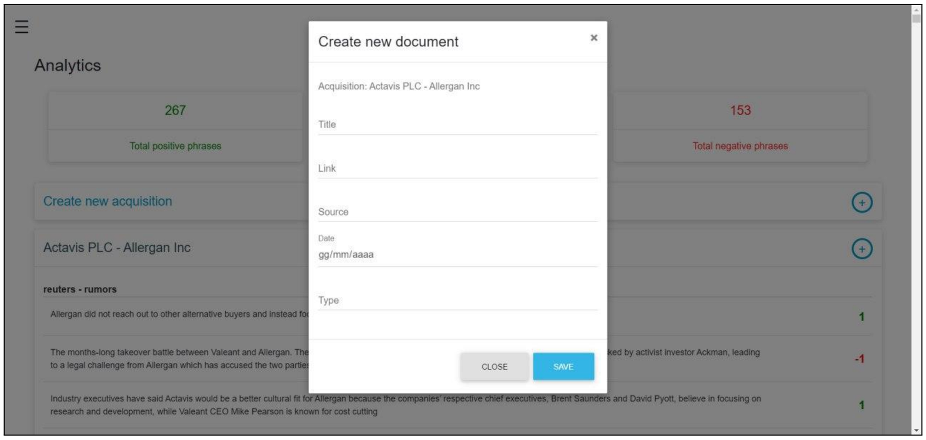


Figure 18: Add document page

The keywords section allows the user to manage them, add new ones or delete those already entered.

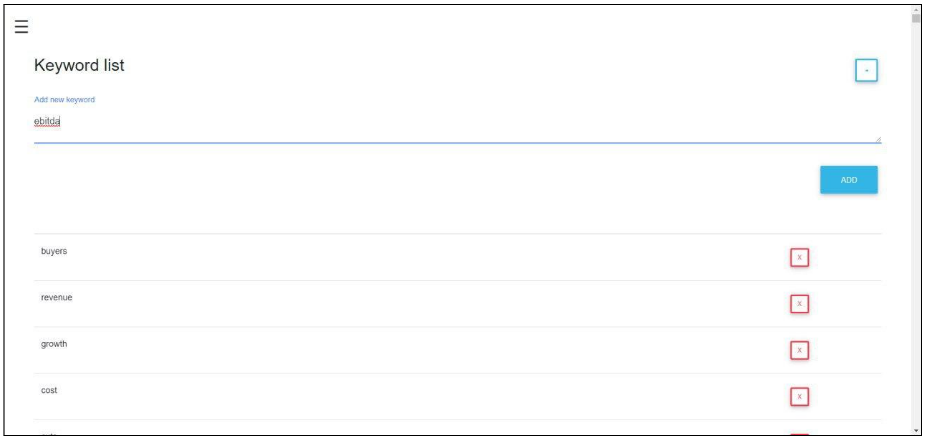


Figure 19: Add keyword

The last functionality allowed by the application is the possibility to manually add sentences and the related sentiment to the training set, in order to improve the performance of the classifiers in future predictions.
As for the prediction, you can insert phrases of a text or twitter.

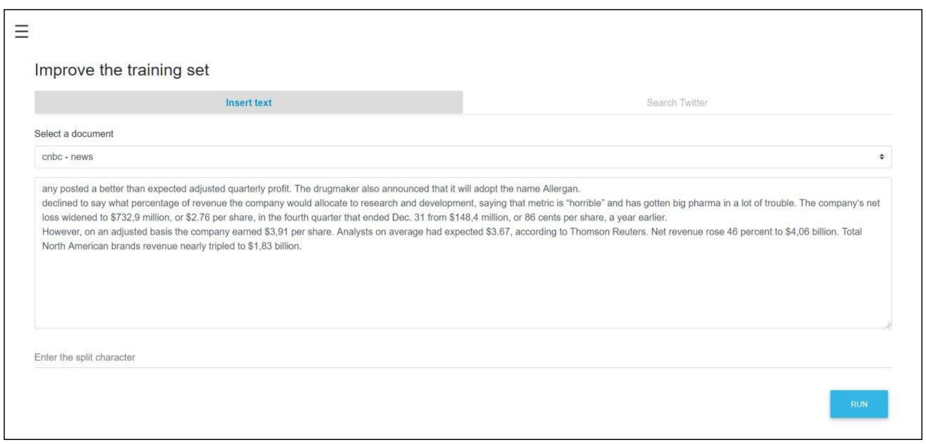


Figure 20: Add new data to training set

Once you have entered the text or made the search of the interested tweets, the page will show a list of sentences to which it is possible to assign a feeling.

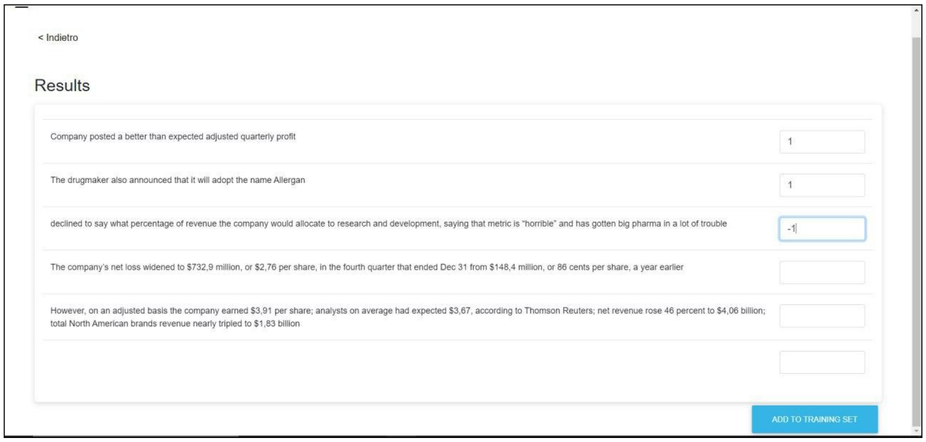


Figure 21: Add sentiment

Once the results are saved, the confusion matrices and related statistical measurements are shown to measure the quality of the classifier.

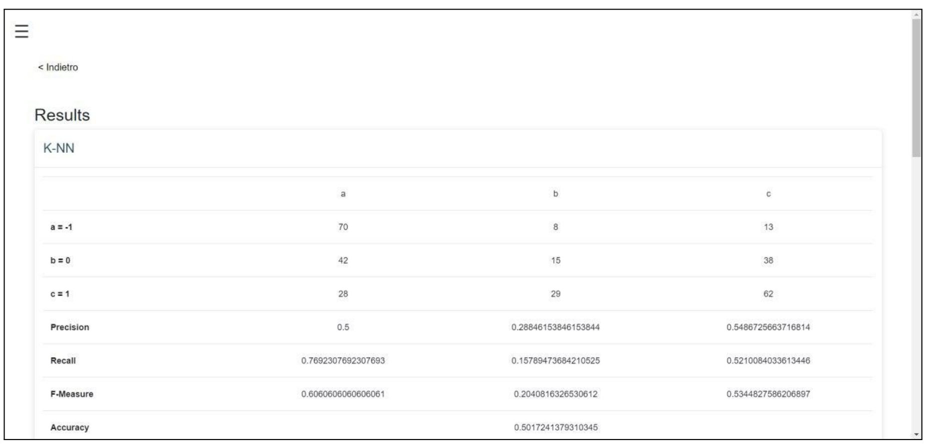


Figure 22: The train result