Machine Learning with Python: NLP and Text Recognition

In this article I apply a series of natural language processing techniques on a dataset containing reviews about businesses. After that, I train a model using Logistic Regression to forecast if a review is "positive" or "negative".

Roberto Sannazzaro Follow
Feb 3 · 6 min read

No, I will not talk about drones in this article.

The natural language processing field contains a series of tools that are very useful to extract, label, and forecast information starting from raw text data. This collection of techniques are mainly used in the field of emotions recognition, text tagging (for example to automatize the process of sorting complaints from a client), chatbots, and vocal assistants.

The dataset

A condensed version of the yelp dataset will be used. This version contains a collection of 1000 observations, originally in JSON format, then converted into `.csv`.

The review dataset being used:

	business_id	cool	date	funny	review_id	stars	text	useful	user_id
0	iCQpiavjjPzJ5_3gPD5Ebg	0	2011-02-25	0.0	x7mDIiDB3jEiPGPHOmDzyw	2.0	The pizza was okay. Not the best I've had. I p...	0.0	msQe1u7Z_XuqjGoqhB0J5g
1	pomGBqfbxcqPv14c3XH-ZQ	0	2012-11-13	0.0	dDl8zu1vWPdKGihJrwQbpw	5.0	I love this place! My fiance And I go here atl...	0.0	msQe1u7Z_XuqjGoqhB0J5g
2	jtQARsP6P-LbkyjbO1qNGg	1	2014-10-23	1.0	LZp4UX5zK3e-c5ZGSeo3kA	1.0	Terrible. Dry corn bread. Rib tips were all fa...	3.0	msQe1u7Z_XuqjGoqhB0J5g
3	elqbBhBfElMNSrjFqW3now	0	2011-02-25	0.0	Er4NBWCmCD4nM8_p1GRdow	2.0	Back in 2005-2007 this place was my FAVORITE t...	2.0	msQe1u7Z_XuqjGoqhB0J5g
4	Ums3gaP2qM3W1XcA5r6SsQ	0	2014-09-05	0.0	jsDu6QEJHbwP2Blom1PLCA	5.0	Delicious healthy food. The steak is amazing. ...	0.0	msQe1u7Z_XuqjGoqhB0J5g

A glimpse of the dataset.

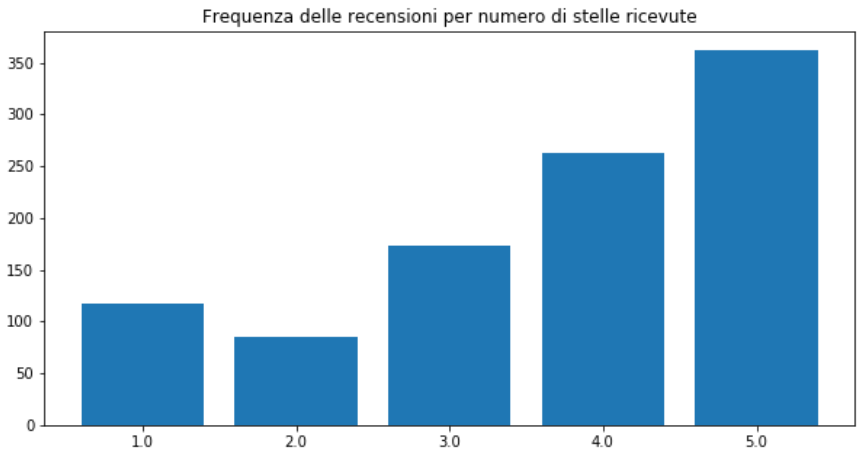
Made out of 9 features ('business_id', 'cool', 'date', 'funny', 'review_id', 'stars', 'text', 'useful', 'user_id') this dataset contains a collection of reviews made by users from yelp, for each review a user gave a score from 1 to 5 stars. In order to create an efficient model to forecast if a review is "positive" or "negative", we start from a model that takes the `text` variable as a predictor and the `stars` variable as the target.

```
'I love this place! My fiance And I go here atleast once a week. The portions are huge! Food is amazing. I love their
carne asada. They have great lunch specials... Leticia is super nice and cares about what you think of her restauran
t. You have to try their cheese enchiladas too the sauce is different And amazing!!!'
```

An observation from the 'text' variable.

Data preprocessing and explorative analysis

Once the dataset is reduced to 2 columns it is possible to conduct a small explorative analysis. It is important to know which distribution the target variable (stars received) follows, in this way it is possible to understand if there is a bias in the dataset—imbalance between positive or negative reviews. This influences the results of the model, giving the propensity to predict outcomes that are more present in the training set.



As we can see from the plot, there is a major component of positive reviews (5 stars), which creates an imbalance or bias.

In order to be able to obtain useful results, it is necessary to reduce the complexity of the problem, an efficient way to do so is to divide the reviews into positive and negative, using this division as the dependent variable.

	text	categoria
0	The pizza was okay. Not the best I've had. I prefer Biaggio's on Flamingo / Fort Apache. The chef there can make a MUCH better NY style pizza. The pizzeria @ Cosmo was over priced for the quality and lack of personality in the food. Biaggio's is a much better pick if youre going for italian - family owned, home made recipes, people that actually CARE if you like their food. You dont get that at a pizzeria in a casino. I dont care what you say...	False
1	I love this place! My fiance And I go here atleast once a week. The portions are huge! Food is amazing. I love their carne asada. They have great lunch specials... Leticia is super nice and cares about what you think of her restaurant. You have to try their cheese enchiladas too the sauce is different And amazing!!!	True
2	Terrible. Dry corn bread. Rib tips were all fat and mushy and had no flavor. If you want bbq in this neighborhood go to john mulls roadkill grill. Trust me.	False
3	Back in 2005-2007 this place was my FAVORITE thai place EVER. I'd go here ALLLLL the time. I never had any complaints. Once they started to get more known and got busy, their service started to suck and their portion sizes got cut in half. I have a huge problem with paying MORE for way less food. The last time I went there I had the Pork Pad se Ew and it tasted good, but I finished my plate and was still hungry. I used to know the manager here and she would greet me with a "Hello Melissa, nice to see you again, diet coke & pad thai or pad se ew?" Now a days, I know she still knows me but she disregards my presence. Also, I had asked her what was up with the new portion sizes and she had no answer for me. Great food but not worth the money. I havent been back in over a year because I refuse to pay $10-15 for dinner and still be hungry after. Sorry PinKaow, you are not what you used to be!!	False
4	Delicious healthy food. The steak is amazing. Fish and pork are awesome too. Service is above and beyond. Not a bad thing to say about this place. Worth every penny!	True

Before proceeding with any other visualization, it is mandatory to apply some preprocessing procedures very common in NLP:

- Remove any non-useful characters (slashes, punctuation, HTML tags, question marks, etc.)
- Convert the whole text to lowercase characters

```
def tokenize(s):
    """Converte una string in minuscolo and la divide in parole singole, ignorando la
       punteggiatura.
    """
    word_list = re.findall(r'\w+', s.lower())
    filtered_words = [word for word in word_list if word not in stopwords.words('english')]
    return filtered_words


def count_ngrams(lines, min_length=2, max_length=4):
    """Itera in una lista di testo, tokenizza ogni parola e ritorna un lista contenente un
        dizionario con ogni parola e la frequenza con la quale occorre.
    """
    lengths = range(min_length, max_length + 1)
    ngrams = {length: collections.Counter() for length in lengths}
    queue = collections.deque(maxlen=max_length)
```

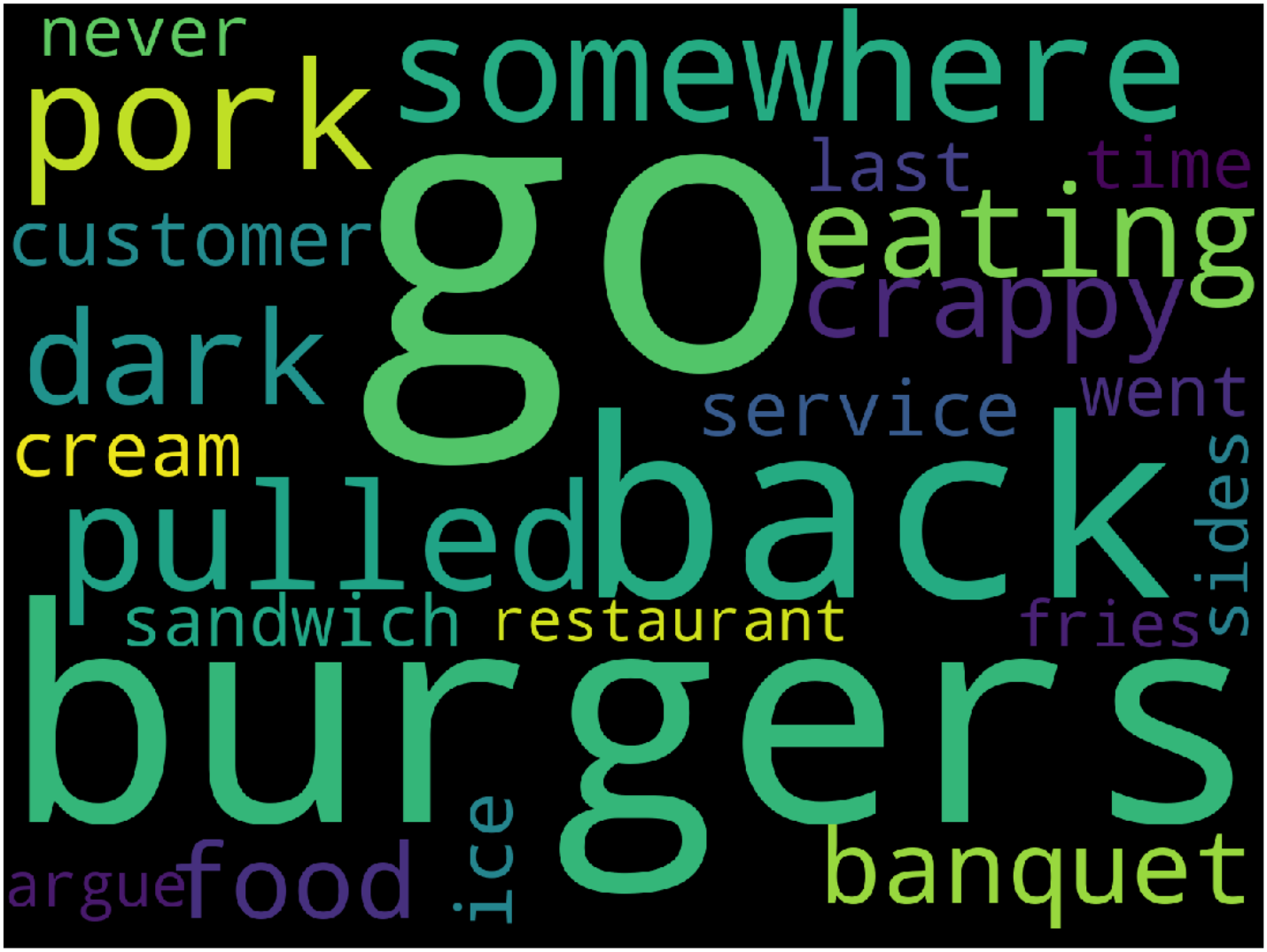
Those two `def` functions will be very useful while preprocessing the text as described before. From here it is possible to determine which single words and a combination of words (bigrams) are more common:

```
----- 10 most common 2-word phrase -----
customer service: 48
ice cream: 46
go back: 43
highly recommend: 41
food good: 26
come back: 25
first time: 25
great place: 25
even though: 24
gluten free: 24

----- 10 most common 3-word phrase -----
indie coffee passport: 11
would go back: 10
would highly recommend: 9
highly recommend place: 7
extra soy milk: 7
great customer service: 6
prairie girl bakery: 6
pretty cool place: 6
pulled pork sandwich: 6
would recommend place: 6
```

Combinations of words made out of 2 and 3 words.

After a small indexing adjustment we can create a bubble chart displaying the most common words in positive and negative reviews:



Never trust the 'burgers'.

And for the positive reviews:

The indie atmosphere is always appreciated.

After this short but interesting insight, we can proceed into the next phase: model creation.

The model

A very simple, fast to train, and very efficient algorithm is Logistic Regression. The `scikit-learn` library provides a tool that helps to build this model, but before doing this and before doing the classical splitting between train and test set, it is mandatory to perform few steps like stemming, vectorization, and removal of stopwords:

- Stemming allows us to reduce every word to its root. This procedure avoids ‘dispersion’ in the text. For example, conjugation of the verb ‘to be’ like: ‘am’, ‘are’, ‘is’ are converted into its root form ‘be’.
- The removal of the stopwords consists of removing every word like ‘the’, ‘that’, ‘of’ that would cause a decrease in the model accuracy.
- Vectorization transforms every observation (review) in the dataset into a numerical representation. This phase is mandatory, and for every machine learning algorithm we would like to train, it is necessary to input numerical data. Vectorization provides the ability to translate text into a numeric representation of itself.

Let’s take a look at a review before and after applying stemming and stopwords removal:

```
The pizza was okay. Not the best I've had. I prefer Biaggio's on Flamingo / Fort Apache. The chef there can make a MU
CH better NY style pizza. The pizzeria @ Cosmo was over priced for the quality and lack of personality in the food. B
iaggio's is a much better pick if youre going for italian - family owned, home made recipes, people that actually CAR
E if you like their food. You dont get that at a pizzeria in a casino. I dont care what you say...
```

Before stemming

```
pizza okay best prefer biaggio flamingo fort apach chef make much better ny style pizza pizzeria cosmo price qualiti
lack person food biaggio much better pick your go italian famili own home made recip peopl actual care like food dont
get pizzeria casino dont care say
```

After applying stemming the results look much more ‘raw’, but at the same time, it is still understandable as its original version.

Now it is possible to proceed with the text vectorization. The `sklearn.feature_extraction.textCountVectorizer` class offers a tool that is very simple to use. This tool needs to be initialized with the `max_features` argument which establishes the max length of the dictionary that will be created in order to represent the text. For example, after choosing 1500 as number of features the algorithm will create a dictionary based on the 1500 (features) with the highest amount of frequency, so each review in the dataset will be represented by a list containing 1500 elements. Each element represents a feature of the dictionary created previously, with the number assigned matching the number of times a word occurred in the observation (review).

Let’s check this example:

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

For each observation (Doc 1, Doc 2, Doc n..) a number represents the occurrences of this feature (word) in the observation (review).

To implement this in technique in Python, only two lines of code are necessary:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
```

It is now possible to split the dataset into a training set and test set:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Then to train the Logistic Regression model with 10 folds:

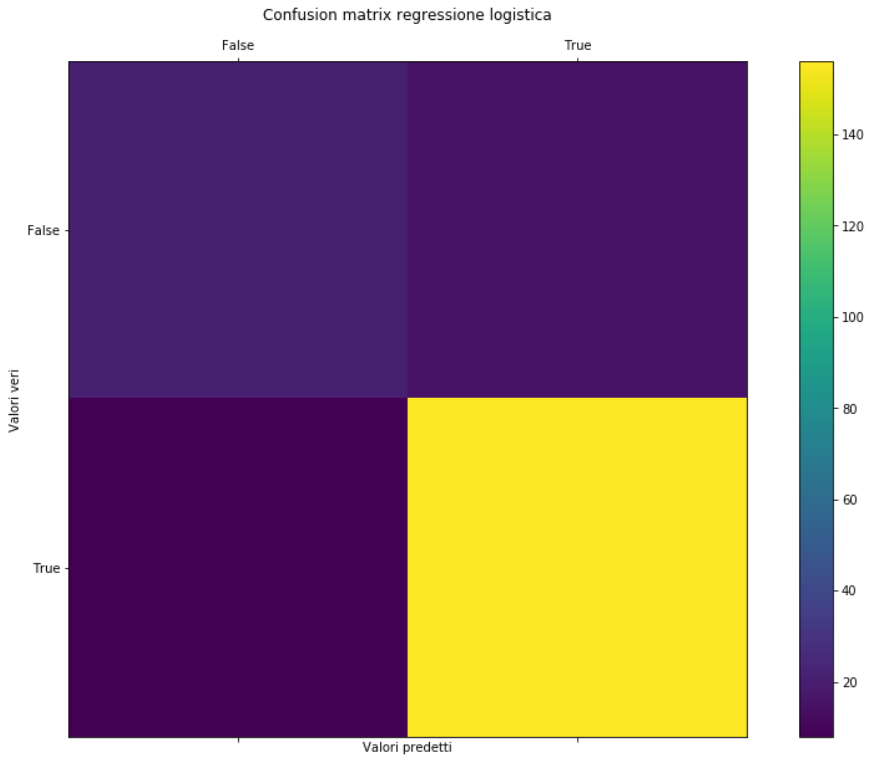
```
from sklearn.linear_model import LogisticRegressionCV
logreg = LogisticRegressionCV(cv=10)
logreg.fit(X_train, y_train)
```

	precision	recall	f1-score	support
False	0.72	0.58	0.65	36
True	0.91	0.95	0.93	164
avg / total	0.88	0.89	0.88	200

```
In [526]: logreg.score(X_test, y_test)
Out[526]: 0.885
```

It is possible to understand from the report that the accuracy is 88.5%, and the bias toward positive reviews is quite evident as the accuracy in predicting positive reviews is much bigger than the accuracy in predicting negative reviews. That means that given a block of text, we can predict that if it is 'positive' or 'negative' with an accuracy of 88.5%.

This difference is more evident in the confusion matrix:



Wrap Up

This model is not perfect, but it does his job. As it was mentioned before the bias toward positive reviews is quite big. To improve this model there are some possible solutions, for example:

- Increase the number of observations (the golden rule)
- Use a different algorithm, like Naïve Bayes, decision trees or some RNN, CNN or HAN.
- Use a different stemming technique
- Use a different stopwords collection

After manually modifying some parameters like the class_weights, it is possible to slightly improve the score. This practice is certainly not the best, but knowing that the model is biased toward positive reviews, a change in the class weights by decreasing the weight for positive reviews and increasing the weights for negative can lead to a (slightly) higher accuracy.

```
In [574]: logreg_2.score(X_test, y_test)
Out[574]: 0.89
```

. . .

Would like the notebook? Just tell me in the comments ⬇️ ⬇️

. . .