Natural Language Processing

Coursework 1

Part I

Simone Zanetti

MSc Data Science

The first part of the coursework has regarded the necessity of working with the Regular Expression tagger, as well as on the Unigram and Bigram tagger, with the goal of identifying the model which guarantees the most accurate result in terms of tagging the famous Treebank corpus, a parsed text corpus that annotates syntactic or semantic sentence structure.

Regular Expression tagger

With the regular expression tagger, the task regarded the necessity of building a series of pattern, which could allow the machine to assign specific tags to each token, based on the correspondence to that pattern. In this context, the following patterns were already provided:

```
patterns = [
     (r'.*ing$', 'VBG'),                   # gerunds
     (r'.*ed$', 'VBD'),                    # simple past
     (r'.*es$', 'VBZ'),                    # 3rd singular present
     (r'.*ould$', 'MD'),                   # modals
     (r'.*\'s$', 'NN$'),                   # possessive nouns
     (r'.*s$', 'NNS'),                     # plural nouns
     (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),      # cardinal numbers
     (r'.*', 'NN')      ]
```

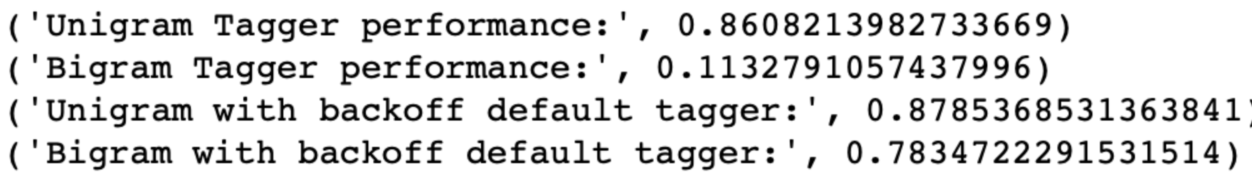
In this context, the first step has been the one of defining a function called *findtags(tag)* which returns the frequency distribution of each word belonging to the specified tag.
This function had a precious role since it allowed me to visualise the words sharing the same tag, giving me the chance to look for patterns between them. Moreover, I defined a second function, called *go( )*, which automatically run the function *nltk.RegexpTagger(patterns)* and its evaluation towards the already labelled/tagged corpus *treebank_tagged_sents*, which allowed to verify the accuracy of the model. As soon as I defined the two functions, I moved towards the direction of verifying the different tags, in order to analyse the possibility of discovering patterns. The following tags have been analysed:

TO	\b[Tt][oO]\b	identifying the word *to* and any capitalisation of it. This is the only word belonging to this category
RB	.*ly$'	words ending in 'ly'
WP	^[Ww]ho.*'	Whom, whose, who
DT	\b[Tt]he\b’ ‘\b[aA]\b' '^[Tt]h.*'	# the: since it is so current it is important to identify it precisely # a: since it is so current it is important to identify it precisely # this, that, these, those, ..
PRP	^.?he.?$	Personal pronoun
, (comma)	` , `	Highly frequent, it highly increased the accuracy
. (period)	.’	Highly frequent, it highly increased the accuracy
-NONE-	^*.*	es. *T*, *-2
WDT	^.hat$ ^which.*	that, what which, whichever
JJ	\bother\b'	# expression matching only one word, since it occurs 134 times, otherwise it would be insert in the wrong label (the one below)
JJR	‘.+er$'	ex. faster, slower, etc.
JJS	'.+est$'	ex, largest, lowest, etc,
NNP	'^[A-Z].*'	Proper nouns, it highly increased the accuracy
IN	'\bof\b| \bin\b| \bfor\b| \bon\b'	Thanks to the findtags(tag) function I could verify that these words are so highly recurrent that it is important to match them
IN	^..$	Lots of preposition are two letters only
IN	'^be.*'	# because, before, between, below
NNS	'.*\'s'	Words ending with plurals
CC	'..t$'	Words ending with ’t’ (but, yet)
CC	'..r$'	Words ending with r (or, either, nor)

The ending pattern allowed to obtain an accuracy of 61%, representing an increase of almost 200% from the initial pattern, whose accuracy was around 21 %. It is important to observe that the research of pattern was not exhausted at the moment. A deeper research towards more subtile patterns would have slightly increased the accuracy. In particular, I regard that it would be important to identify the most recurrent words through a frequency distribution and considering the idea of directly identify them alone, without the necessity to insert them into pattern. In fact, this task could be quite simple and it would give an increased result. On the other side, the regular expression tagger may not be the most state-of-the art system of tag identification, which means that in the end the task proposed of tagging of individual words may not be so that necessary.

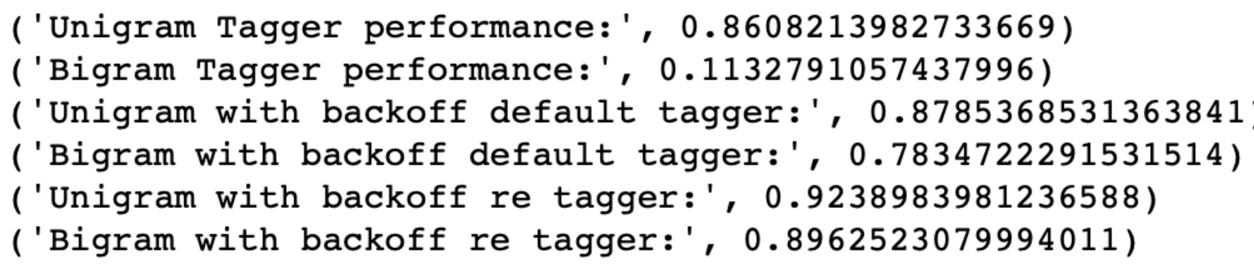
Default tagger, Unigram, Bigram and Backoff

With the second part of the task I worked with different tagger. In particular, I identified the most common POS tag in the corpus, which revealed to be the 'NN' tag (Noun). The reason of this research was to set the Default tagger to be used as backoff with the Unigram and Bigram. In particular, firstly I performed the Unigram and Bigram methods independently, and after I used those methods with Default tagger set to 'NN' as backoff.
By doing this I obtained different levels of accuracy, which are summarised below:



```
('Unigram Tagger performance:', 0.8608213982733669)
('Bigram Tagger performance:', 0.1132791057437996)
('Unigram with backoff default tagger:', 0.8785368531363841)
('Bigram with backoff default tagger:', 0.7834722291531514)
```

It is possible to notice how the Bigram tagger performance resulted in a very low accuracy. The reason for this can be identified in the so-called *Sparse data problem*. In fact, as the size of the n-gram gets larger, the context increases in terms of specificity, with the consequence that the possibility that the training data do not contain the context of the data we are attempting to tag. On the other side, the accuracy of the other taggers seem to be pretty high. The possibility to combine the unigram and bigram with the *Regexp_tagger* obtained in the previous task (Regular Expression tagger) allowed to obtain even better results, by reaching the accuracy of 0.92. All the results are shown and compared in the figure below:



```
('Unigram Tagger performance:', 0.8608213982733669)
('Bigram Tagger performance:', 0.1132791057437996)
('Unigram with backoff default tagger:', 0.8785368531363841)
('Bigram with backoff default tagger:', 0.7834722291531514)
('Unigram with backoff re tagger:', 0.9238983981236588)
('Bigram with backoff re tagger:', 0.896252307994011)
```

The -None- tags

A particular category regard contained into the Treebank corpus is the so-called *-None-* tag. With the purpose of the analysis, they have been extracted and observed in the portion of the text they are found. This possibility allowed me to have a preliminary observation of the context in which they were present in the text. By inserting the tag -NONE- to the function *findtags(tag)* expressed above, I could obtain a list of the frequency distribution of the 50 most frequent strings linked to this particular tag, as shown below:

```
-NONE- [('*-1', 1123), ('0', 1099), ('*', 965), ('*T*-1', 806), ('*U*', 744), ('*-2', 372), ('*T*-2', 345), ('*-3', 1
30), ('*T*-3', 97), ('*ICH*-1', 70), ('*?*', 45), ('*-4', 34), ('*ICH*-2', 34), ('*RNR*-1', 34), ('*EXP*-1', 33), ('*
T*-4', 23), ('*ICH*-3', 13), ('*-5', 11), ('*EXP*-2', 8), ('*T*-5', 5), ('*ICH*-4', 5), ('*RNR*-2', 5), ('*-6', 4), (
'*-7', 4), ('*-52', 4), ('*-25', 3), ('*T*-51', 3), ('*EXP*-3', 3), ('*-64', 3), ('*-73', 3), ('*-80', 3), ('*PPA*-3'
, 3), ('*T*-6', 2), ('*T*-7', 2), ('*T*-8', 2), ('*-8', 2), ('*-9', 2), ('*T*-9', 2), ('*T*-10', 2), ('*-10', 2), ('*
T*-11', 2), ('*-11', 2), ('*-12', 2), ('*T*-12', 2), ('*T*-13', 2), ('*T*-14', 2), ('*T*-15', 2), ('*RNR*-4', 2), ('*
-13', 2), ('*-14', 2)]
```

While a preliminary visualisation of the context of these strings was useful, a review of the Treebank documentation allowed me to conclude that these tag are not actual words, but instead non words identifying trace elements, identified with ‘ * ‘ as initial. In particular each trace represents a placeholder in the syntax.

Natural Language Processing

Coursework 1

Part II

Introduction

The goal of the present task is to develop a sentiment classifier for movie reviews, by using a corpus individually obtained from the BBC archive of film reviews (Bbc.co.uk, 2019[1]). In this context, the primary task has been represented by the necessity of scraping each review from the website with the associated score, graded from 1 to 5. In particular, this phase will represent the object of the first section of the report. Secondly, all the reviews have been processed in order to be made feasible for the task proposed, through a series of transformations which include the creation of a new binary variable assuming positive and negative value and representing the target of the analysis, the necessity to balance their presence in the corpus, and so on. The second section of the report will provide more details about it. The third part of the presentation will describe the attempt of performing the sentiment analysis, by using a series of Machine Learning algorithms such as Naive Bayes, Decision Tree, and Logistic regression. Following this, the goal is the one to show the different attempt done in order to optimise the accuracy obtained, by applying a series of techniques, which wide from the modification of the size of the feature sets, the attempt of filtering stop words, in order to avoid common words such as *the*, *is*, *which*, to have influence in the model. These techniques, together with another series of attempt, will be described in the forth section.

Obtaining the Corpus: scraping the web

The first task of this project has regarded the necessity of obtaining a corpus from the *BBC archive* of film reviews, which consists in a series of reviews related to different movies, and the score associated to them, which can range from 1 to 5. For this purpose, the goal was the one of scraping from the *html* file of the web page of each review, only the content related to the review itself and its score. In order to do this, I took advantage from the fact that the reviews are set in alphabetical order. Specifically, for each letter there is a web page containing the list of links related to movies starting with it. By collecting a list of links for each letter, I could create a *for loop* which scraped from the web page of each letter all the links present in there, and store them in a new list. This process has allowed me to have all the necessary links to perform the above mentioned scraping for each review, through a new *for loop.* Although it seems a complicate process to understand, the codes, as well as a direct observation of the website, can ease the process of understanding (*www.bbc.co.uk/films/gateways/az/review/cinema/a.shtml*).
Once the reviews have been collected, they have been stored into a dataframe and modified by eliminating the escape sequences symbols, such as *\n* and *\s*, turning the score variable into numeric type, filtering the rows containing *Nan's*, and so on.

Prepare the Corpus

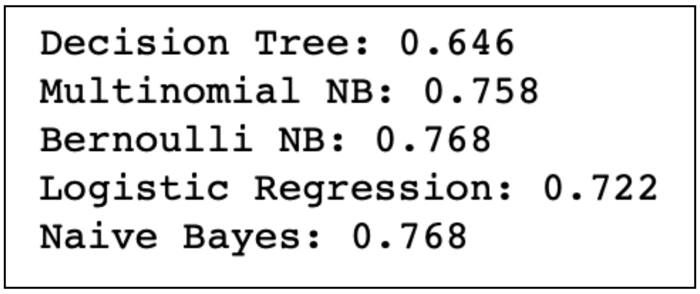
The second phase has regarded the necessity of preparing the corpus for the sentiment analysis. The first step has been the one of creating a new variable *sentiment,* which would represent the output of the later analysis. This variable only assume *positive* or *negative* value. In particular, for reviews whose score had a value above 3, the sentiment has been set as positive, while for those with a value below 3, the sentiment has been set as negative. On the other side, all the reviews owning a score equal to 3 have been dropped by the dataset, in order to balance the number of positive and negative reviews in the corpus. Successively, a variable called documents, including a list of tuples containing each review tokenised and the score related to it, has been created. Then, a list containing all the tokenised words of each reviews called *rev_words* has been created. Moreover, frequency distribution of all the words included in the list has been obtained, and filtered including only the 500 most frequent words. The variable *featuresets* which I created evaluated for each word in the corpus whether this is part of the most frequent words or not. Finally, the variable *featuresets* has been split into train set and test set, with a proportion of roughly 75% and 25% between each other.

Apply Machine Learning classifiers

The third part of the process has regarded the attempt to perform the sentiment analysis by applying different classifiers to it, with the goal of obtaining the best accuracy. In this context, the objective of performing a sentiment analysis can be interpreted as a binary classification task, in which positive and negative represent the output.
A series of different Machine Learning classification algorithm has been applied. Specifically:

- Decision Tree
- Naive Bayes, from the *nltk* package
- Multinomial Naive Bayes
- Bernoulli Naive Bayes
- Logistic regression

With the first attempt, the number of most common words was set to 500, and the tokens included punctuation. In this context, the performance measured on the test set for each classifier is summarised below.



```
Decision Tree: 0.646
Multinomial NB: 0.758
Bernoulli NB: 0.768
Logistic Regression: 0.722
Naive Bayes: 0.768
```

Attempts to improve the accuracy

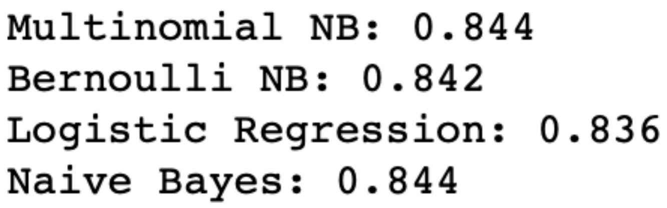
This section deals with the attempt performed during the analysis to improve the accuracy of the model, by applying a series of actions to the dataset. In particular, this attempts regarded the following efforts:

A- Remove punctuation
B- Vary the size of the Feature set
C- Eliminate/filter stop words from the set
D- Modify the size of the words

Each of the above mentioned attempts have been combined together, with the goal of increasing the accuracy.

A - Remove punctuation

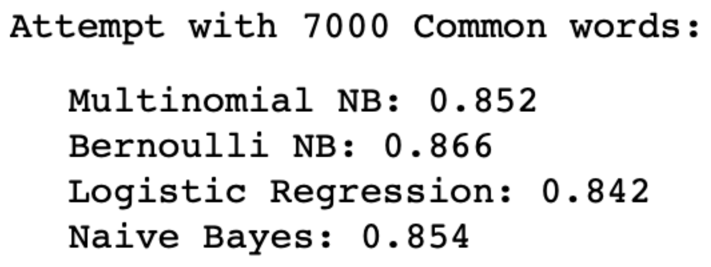
The first and probably most significant modification has been the one to remove the punctuation. A regular expression set in order to find and keep only the words and digits (removing the punctuation) allowed to run the classifier on a modified corpus. The result, as shown in the image below, has been surprising. In fact, the accuracy has improved of almost *7/8 %*. It is important to observe that Decision Tree algorithm is not shown anymore, since it has been dropped from the analysis. This is due to an excessive slowness of the algorithm to run on my personal laptop, as well as the fact that the accuracy provided by it was lower than any other model.



```
Multinomial NB: 0.844
Bernoulli NB: 0.842
Logistic Regression: 0.836
Naive Bayes: 0.844
```

B - Vary the size of the Feature set

The second attempt done in order to increase the accuracy has been the one of modifying the size of the Feature set. From this point of view, the number of the most common words initially set to 500 has been increased to different values in order to verify if/ how the performance could be modified. With this purpose, a function which could automatically obtain return the train and test data by only inserting the parameter of the size chosen has been created (named *attempt(x)*), and a *for loop* which could iterate the function adding 1000 most common words over a range included from 1000 to 8000.
While the output of the different values of x is shown in the python file, the following figure shows the value of x which obtained the best results, which is x = 7000.



```
Attempt with 7000 Common words:

   Multinomial NB: 0.852
   Bernoulli NB: 0.866
   Logistic Regression: 0.842
   Naive Bayes: 0.854
```

C - Eliminate stop words from the set

The possibility to modify stop words raises from the fact that there are a series of words commonly used that may not add any value to the sentiment analysis, in terms of distinction between a positive review and a negative review. From this point of view, an attempt of filtering the so-called (english) *stop words* has been done. With this purpose, I considered important to verify again the possibility to modify the size of the feature sets, since after the process of filtering the stop words their amount is changed, and this can leave space to new attempts to 'play' with the dimension of the feature set.
Just like previously done, a function named *attempts_stopwords(x)* accepting the size of the feature set as parameter has been created, and a *for loop* over values within the range (4000, 12000) iterated. However, in this case the function has been tested only on the Naive Bayes classifier from the nltk package, and the Bernoulli Naive Bayes from the ScikitLearn package. For the result, see the python document.

D - Modify the size of the words

Another attempt has been performed in relationship with the length of the words belonging to the set of most common words. Despite several attempts have been made, the analysis only shows the effect of dropping every word with length above 6, and the one of dropping every word with length below 3. While the first case did not provide any particular result, the second case obtained a performance of 0.856 on the Bernulli Naive Bayes, which is closer to the bests results in terms of accuracy. With this regards, it is interesting to verify the features identified as the most informative in the context of words with length above 3.

```
Most Informative Features
       contains(unfunny) = True              neg : pos    =     22.1 : 1.0
       contains(creates) = True              pos : neg    =     17.9 : 1.0
      contains(physical) = True              pos : neg    =     17.3 : 1.0
     contains(affecting) = True              pos : neg    =     13.1 : 1.0
  contains(unconvincing) = True              neg : pos    =     12.9 : 1.0
```

Conclusion

Several attempts have been done in order to improve the accuracy of the model created on this Sentiment Analysis task. From this point of view, a remarkable result has been obtained when removing the punctuation from the tokens, increasing the accuracy from an average of 0.75 to one of 0.84. In this context, varying the size of the feature set to 7000 has allowed to obtain the maximum performance over the analysis, with a value of 86.6% of accuracy on the Bernoulli Naive Bayes classifier. On the other side, the attempt of filtering stop words, as well as filtering words based on their length has not lead to further improvement of the performance.

Bibliography

[1]Bbc.co.uk. (2019). BBC - Films - Site Index. [online] Available at: http://www.bbc.co.uk/films/archive.shtml?film_reviews [Accessed 26 Feb. 2019]

SENTIMENT ANALYSIS:
A short Report

INTRODUCTION

Sentiment analysis represents the process of extraction of subjective information from texts with the goal of obtaining informations related to their orientation. The possibility to obtain results from this analysis relies on the potentiality of computational techniques such as Natural Language Processing, and Text analysis. Sentiment analysis is applied in a wide range of context, as well as for a numerous amount of goals. From this point of view, the next section will describe the most common application of them, while the last section will focus on the methods which allow sentiment analysis to be a powerful tool nowadays, principally Machine Learning based and Lexicon-based.

APPLICATIONS OF SENTIMENT ANALYSIS

Monitoring the voice of people, in general: Reputation management

The possibility for business to monitor their reputation over the public is vital nowadays more than any other epoch . In fact, the opportunities granted by the society in which we are living, where every person can be instantaneously connected with each other all over the world and access to the web universe so easily, leads businesses to run the risk of their reputation being ruined by bad reviews or news about them. The majority of us check social media reviews as well as review sites before making a purchase decision or go eating in a restaurant, and these are only an example of the way consumers habits has changed over the time. In particular, it is important to notice how one bad review can have influence people more than 10 good reviews, and potentially burn down an entire brand. In this context, the possibility of companies to monitor social media, online reviews and news combined with sentiment analysis can represent a vital source for their surviving, allowing companies to obtain useful insights about how people feel about their brand and their activity,

Monitoring the voice of costumers: Business decisions

The possibility of monitoring the response of consumers around a product or a service, can allow companies to apply decisions as a consequence. From this point of view, companies have the possibility to verify the quality of their product and eventually combine it with other information such as location or demographic data of the consumer (when legally allowed).

In some cases they can intervene in real time to fix eventual issues, or decide about modification of their activity, such as advertisement investment or the distribution of their product in specific area.

Monitoring the satisfaction of employees

Sentiment analysis is more and more used by HR directors to monitor the satisfaction of their employer towards their job and the company itself. In fact, thanks to this method, companies are being able to improve work conditions and increase worker satisfaction, which in most cases leads to an improvement in their productivity.

METHODS

MACHINE LEARNING METHODS

Sentiment analysis widely relies on the concept of classification, which is defined as the process of extraction of useful features from single observation, which can thereby allow classifying the observation into one of a set of discrete classes. In this context, the basic application of sentiment analysis can be managed as a binary classification task, where the output of the analysis can assume either positive or negative value. In this context, the ending goal is the one to be able to map the correct output 1 - 0 (pos - neg) from a new observation. As a consequence, supervised machine learning task represents one of the ways in which sentiment analysis is performed. In fact, the possibility of relying on probabilistic classifier can guarantee the analysts to obtain the probability of the observation being in one of the classes. Machine learning presents a series of different algorithms. A simple yet powerful application is the Naive Bayes algorithm. In particular, its name derives from two assumptions which tends to simplify the analysis. This regards the way features interact between each others.
1- Words within a text are considered to be uncorrelated, which means that words are not analysed within the context they are in, but the corpus of analysis is considered as an unordered set of words, discarding their position within the text.

2 - The second is commonly called the naive Bayes assumption: this is the conditional independence assumption that the probabilities $P(f_i|c)$ are independent given the class *c.* *(Speech and Language Processing.* [online] Available at: https://web.stanford.edu/~jurafsky/slp3/)

LEXICON-BASED METHODS

While machine learning methods in relationship with Sentiment analysis are Supervised methods, and as such they necessity to have a label which can identify their output, Lexicon based methods are Unsupervised learning methods. These are defined as methods which adopt a lexicon which count the the words related to the specific sentiment (weighted per sentiment) that gave been previously evaluated and tagged. There are different ways of working with a Lexicon based approach. These can be identified as following:

. Manual approach

. Corpus-based approach.

. Dictionary-based approach

The dictionary-based approach rely on resources available, which can help to expand the first seed of dictionary manually created containing sentiment words and their orientation, by find similarities with it, such as synonyms, and so on. An example of lexicographical resources is WordNet or HowNet. This allow to obtain new words related to the sentiment and eventually extract the specific sentiment from the corpus analysed.