Sarcasm Detection on Reddit dataset

Lirida Papallazi
lirida.papallaz i@studenti.unimi.it

November 29, 2020

Chapter 1

Introduction

Analysing textual data has become more and more important for companies in order to understand customer activities, opinion, and feedback to successfully derive their business. The analysis of the tweets on Twitter is a classical example which allows to find trending news, but also help defining the social network structure while Amazon could use the review on the specific product to understand user feedback.

This project focuses on the analysis of the comments made on the data-set containing the comments made on Reddit (available on Kaggle[1]), with the aim of predicting the probability that a comment will receive a sarcastic answer. Different techniques are implemented: Naive Bayes, Logistic Regression and Random Trees, with and without Cross Validation.

Reddit is a website comprising user-generated content—including photos, videos, links, and text-based posts—and discussions of this content. As of 2018, there are approximately 330 million Reddit users, called "redditors". Another peculiarity is that the site's content is divided into categories or communities known on-site as "subreddits". Reddit's core content consists of posts from its users who can also comment on others' posts, allowing for a continuous conversation. Posts and comments can also receive positive or negative votes (respectively up-votes and down-votes), and the number of upvotes or downvotes determines the posts' visibility on the site, so the most popular content is displayed to the most people.

Chapter 2

Analytical background

Before dealing with the data-set and the analysis conducted on it, it is advisable to provide even the less experienced reader with a brief introduction on the topics covered. However, given the vastness of the same and the extent of the treaty, the reader is referred to other more complete sources. As they are often confused with each other, we will begin by clarifying the concepts of Text Mining, Natural Language Processing and sentiment analysis

2.1 Text mining

Text Data continues to grow exponentially and it is actually estimated to be 2.5 Exabytes (2.5 million of TB) a day.[2]

2.1.1 Definition and tasks

Text mining is the process of transforming unstructured text data into meaningful and actionable information. The most known and performed text mining task are:

- parsing the text;
- finding relevant information from the text;
- classification of text documents;
- information retrieval;
- performing sentiment analysis;
- topic modelling.

Text mining combines notions of statistics, linguistics, and machine learning to create models that learn from training data and can predict results on new information based on their previous experience.

2.1.2 Text Mining techniques

There are many techniques used for a Text mining analysis although this section will focus on the most commonly used.

Word frequency

As the term says this techniques consists of identifying the most recurrent terms in a text, documents or generally a data-set.

Collocation

It refers to a sequence of words that commonly appear near each other and the typical example are bigrams (namely pairs of words likely to be together).

2.1.3 Procedure

Text mining uses NLP techniques, such as tokenization, parsing, lemmatization, stemming, stopwords and punctuation removal to build the inputs for the machine learning algorithms.

2.2 Natural Language Processing (NLP)

In order to define what Natural Language Processing is it is necessary to define what Natural languages are first.

2.2.1 Natural Languages vs NLP: definition and tasks

Every language used for everyday communication is considered to be Natural language in opposition to the artificial language used by computers.

Any computation or manipulation on natural language that will give insights on what the meaning of the words and how sentences are constructed is considered natural language processing. Natural language processing needs to consider that the language used evolves, for example: old words lose popularity, new words get added, language rules may change. And the main NLP task that can be performed are:

- counting frequency of words;
- find sentence boundaries;
- tagging part of speech to a sentence;
- parsing the sentence structure;
- identification of semantic roles;
- identification of entities.

2.2.2 Text Classification and tasks

Text classification is the process that assigns categories to text data. Let's think about a news data-set, text classification could be think as the labelling of each text with a news category (such as politics, economics, technology, etc.).[3]

Topic Analysis

It allows to extract from a text the main themes in order to organize the texts.

Sentiment Analysis

Sentiment analysis is a technique that detects the underlying sentiment or polarity of a text, document or sentence. In particular it is the process of classifying text as either positive, negative, or neutral. Machine learning techniques are used to evaluate a piece of text and determine the sentiment behind it.[4]

Text Extraction

Text extraction uses machine learning to automatically scan text and extract relevant or core words and phrases from unstructured data like news articles, surveys, and customer service tickets.[5] It can be used for:

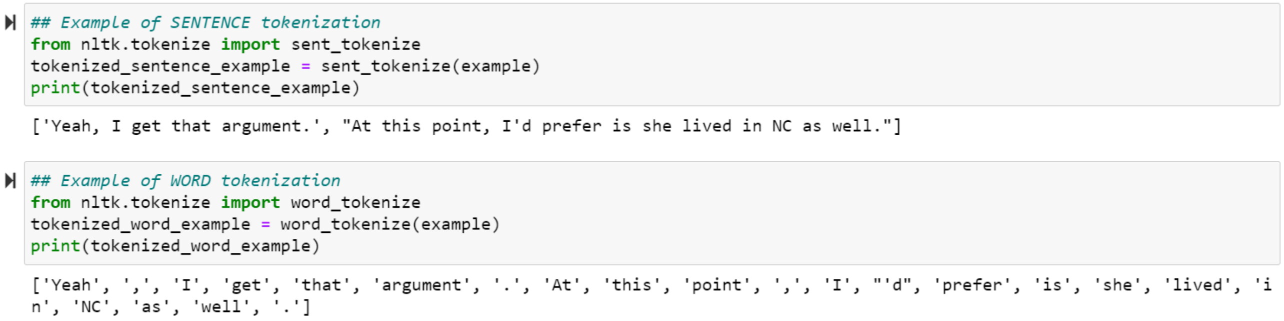
- keyword extraction;
- entity recognition;
- feature extraction.

2.2.3 NLP Techniques

A previously stated in order to apply the machine learning algorithms the textual data-set has to be pre-pocessed. The principal NLP techniques with a brief explanation will follow.

Tokenization

It is the process of breaking down a text paragraph into smaller chunks such as words or sentence.



```
## Example of SENTENCE tokenization
from nltk.tokenize import sent_tokenize
tokenized_sentence_example = sent_tokenize(example)
print(tokenized_sentence_example)
```

```
['Yeah, I get that argument.', "At this point, I'd prefer is she lived in NC as well."]
```

```
## Example of WORD tokenization
from nltk.tokenize import word_tokenize
tokenized_word_example = word_tokenize(example)
print(tokenized_word_example)
```

```
['Yeah', ',', 'I', 'get', 'that', 'argument', '.', 'At', 'this', 'point', ',', 'I', "'d", 'prefer', 'is', 'she', 'lived', 'i
n', 'NC', 'as', 'well', '.']
```

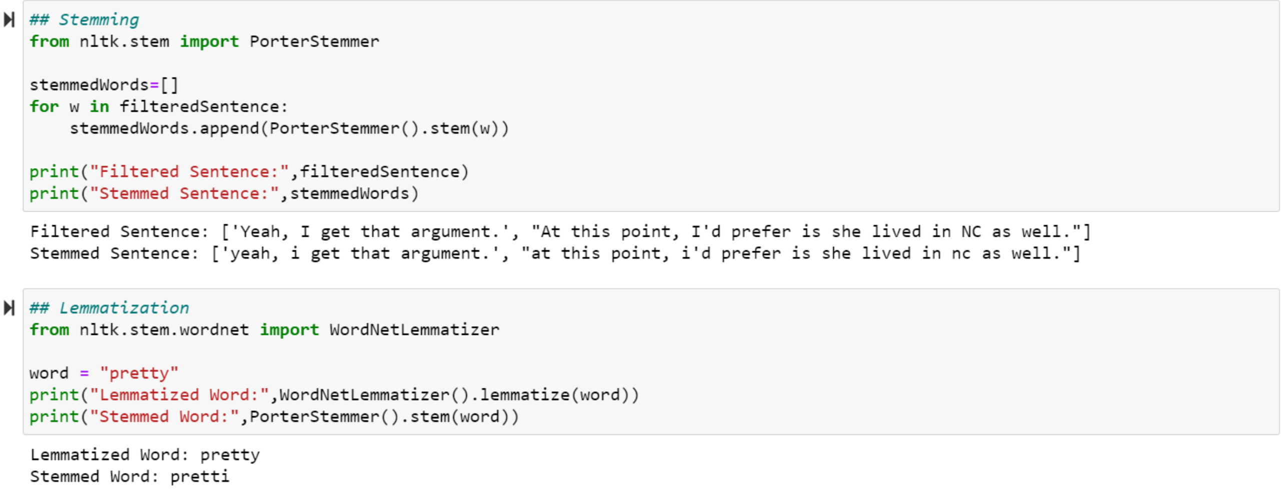
Figure 2.1: Example of sentence and word tokenization

Parsing

Also known as syntactic analysis identifies the structure of the syntax of a text and the dependency relationships between words. Diagrams called parse trees are typically employed to represent these dependencies.

Lemmatization and Stemming

To make words easier for computers to understand NLP uses lemmatization and Stemming to transform the words back to their root. When the root to which the words are reconnected is the lemma, then we will talk about lemmatization: for example am, is, are, were etc. are all replaced with the lemma 'be'.



```
## Stemming
from nltk.stem import PorterStemmer

stemmedWords=[]
for w in filteredSentence:
    stemmedWords.append(PorterStemmer().stem(w))

print("Filtered Sentence:",filteredSentence)
print("Stemmed Sentence:",stemmedWords)
```

```
Filtered Sentence: ['Yeah, I get that argument.', "At this point, I'd prefer is she lived in NC as well."]
Stemmed Sentence: ['yeah, i get that argument.', "at this point, i'd prefer is she lived in nc as well."]
```

```
## Lemmatization
from nltk.stem.wordnet import WordNetLemmatizer

word = "pretty"
print("Lemmatized Word:",WordNetLemmatizer().lemmatize(word))
print("Stemmed Word:",PorterStemmer().stem(word))
```

```
Lemmatized Word: pretty
Stemmed Word: pretti
```

Figure 2.2: Example of lemming and stemming application

Stemming instead "trims" the words to obtain the so called stems, which might not be always semantically correct.

Stopwords and Punctuation removal

Stopwords are typically considered as noise in the text and therefore removed altogether with the punctuation. A few examples (for the english language) are: 'is', 'are', 'this', 'a', 'the', etc.

2.2.4 Vectorization

The application of machine learning algorithms requires that the training data is transformed in vectors, so that the computers can understand the data. Word Vectorization is a methodology in NLP to map text to a corresponding vector of real numbers which can be used to support later automated text mining algorithms.

Terms are generic features that can be extracted from text documents and they can be single words, keywords, n-grams, or longer phrases.

Documents can be imagined as vectors of terms where each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. The computation of these values also known as (term) weights, have defined different techniques.

Bag of Words

In the Bag of words model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. It is therefore used to keep track of the occurrence of each word that is used as a feature for training a classifier. The terms mentioned above will therefore be the words (more generally n-grams) while the weights are the number of occurrences of the terms in the document.

TF-IDF

The term frequency–inverse document frequency (tf-idf) is one of the most popular term-weighting schemes today and is intended to reflect how important a word is to a document in a collection or corpus. The tf–idf value or weight increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word. Also in this case the terms can be words, n-grams, etch. while and higher weight to terms that are frequent in the document but not common in the corpus.

2.3 Machine Learning Algorithms

Different machine learning algorithms were applied to data-set in order to achieve the purpose of this paper, which will be further explored in the next chapters. A short explanation will be given for each one of them, but again the reader is advised to consult other sources in order to deepen his/her knowledge.

2.3.1 Naive Bayes

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome (assumptions which are not always true in reality). Starting from the well known:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

where:

- $P(A|B)$ is the probability of event A given that event B has already happened and it corresponds to the posteriori probability;
- $P(A)$ is the priori probability of A, namely the probability before any evidence is seen.

Let now y be the variable of the categories and X the dependent feature vector, then it is possible to obtain from the Bayes theorem the following classifier, which finds the probabilities of the inputs for all the possible values of the category variable y and picks the output which maximizes the probability:

$$y = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.2}$$

The different naive Bayes classifiers differ mainly by the assumptions made on the distribution of $P(x_i|y)$.

2.3.2 Logistic Regression

Logistic Regression actually does not perform any statistical classification since it simply models the probability of output in terms of input. It can be transformed into a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other, resulting therefore in a binary classifier. The equation of the logistic model is:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{2.3}$$

The regression coefficients are usually estimated using maximum likelihood estimation.

2.3.3 Decision Trees and Random Forest

A decision tree is a hierarchical decomposition of the (training) data space, in which a condition on the feature value is used in order to divide the data space hierarchically. Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. While the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Therefore Random forests correct for decision trees' habit of overfitting to their training set. Random forests or random decision forests are an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees.

Chapter 3

Text Analysis

The purpose of the paper is that of predicting the probability that a comment is given a sarcastic answer, given the subreddit. We decided therefore to focus on the parent comment and analyze their texts in order to understand if there are some features (namely words) that increase the chances of receiving a sarcastic answer. In this section the will be presented:

- the composition of the data-set;
- text exploratory analysis;
- text classification.

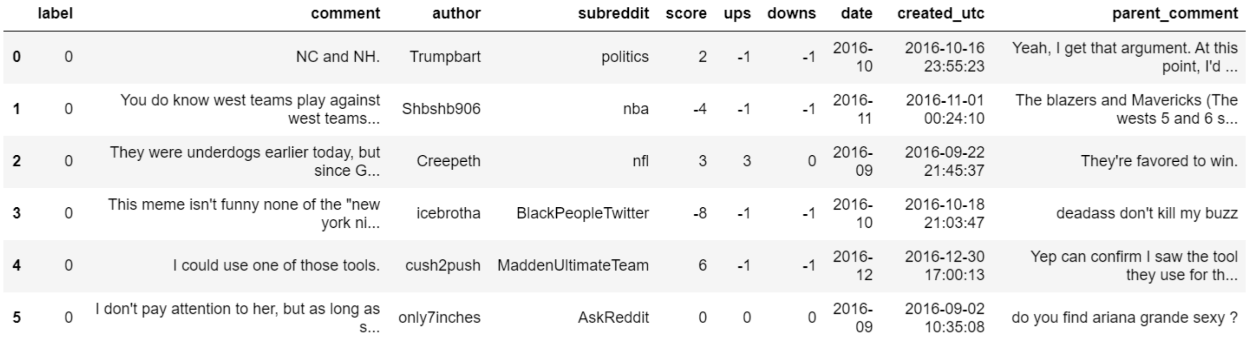
An important note is that a very small portion (0.05%) of the original dataset was used for the analysis as it was too large, which still resulted in 50539 rows and therefore comments. This choice was made also to decrease time and space needed for the computations.

3.1 Data-set composition

As it is possible to see in Figure 3.1 the Reddit data-set contains ten variables, and in particular the "comment" column contains the answers to the comments in the "parent_comment" column; the "label" will be a boolean variable that in case of sarcastic comment will be 1, and 0 otherwise. In order to achieve our purpose the logic used was that of focusing on three variables: the label, the subreddit and the parent_comment and analyze the text structure of the latter in order to undersand which features (or words) are responsible for increasing the chances of receiving a sarcastic answer.

3.1.1 Text exploratory analysis

As it is possible to observe from Figure 3.2 the data-set is completely balanced which means that it does not require any adaptation technique such as oversampling or under-sampling. In Figure 3.3 wordclouds (a novelty visual representation of text data) is used to depict the most frequently used words in the parent comment.



	label	comment	author	subreddit	score	ups	downs	date	created_utc	parent_comment
0	0	NC and NH.	Trumpbart	politics	2	-1	-1	2016-10	2016-10-16 23:55:23	Yeah, I get that argument. At this point, I'd ...
1	0	You do know west teams play against west teams...	Shbshb906	nba	-4	-1	-1	2016-11	2016-11-01 00:24:10	The blazers and Mavericks (The wests 5 and 6 s...
2	0	They were underdogs earlier today, but since G...	Creepeth	nfl	3	3	0	2016-09	2016-09-22 21:45:37	They're favored to win.
3	0	This meme isn't funny none of the "new york ni...	icebrotha	BlackPeopleTwitter	-8	-1	-1	2016-10	2016-10-18 21:03:47	deadass don't kill my buzz
4	0	I could use one of those tools.	cush2push	MaddenUltimateTeam	6	-1	-1	2016-12	2016-12-30 17:00:13	Yep can confirm I saw the tool they use for th...
5	0	I don't pay attention to her, but as long as s...	only7inches	AskReddit	0	0	0	2016-09	2016-09-02 10:35:08	do you find ariana grande sexy ?

Figure 3.1: First 5 rows of the Reddit data-set

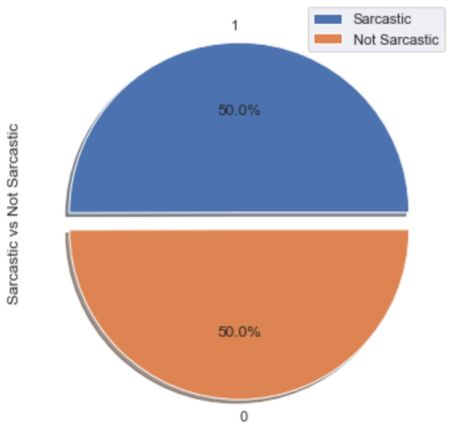


Figure 3.2: Pie chart of sarcastic answer

3.2 Pre-processing and Data preparation

In order to apply the classification algorithms it is necessary to adapt and prepare the data-set.

Train-Test-Validation split

To understand model performance the dataset is divided into a training set and a test set and to do so the train_test_split() function is used. When evaluating different settings ("hyperparameters") for estimators there is still a risk of overfitting on the test set. To solve this problem, it is advisable to hold out from the data-set also a so-called validation set. Training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.

3.2.1 TF-IDF

The following step require converting the textual data into some numbers or vectors or numbers. One approach could be Bag of Words but in this paper the TF-IDF model was used (via TfidfVectorizer() function from the sklearn library) to normalizes the document term matrix.[6]

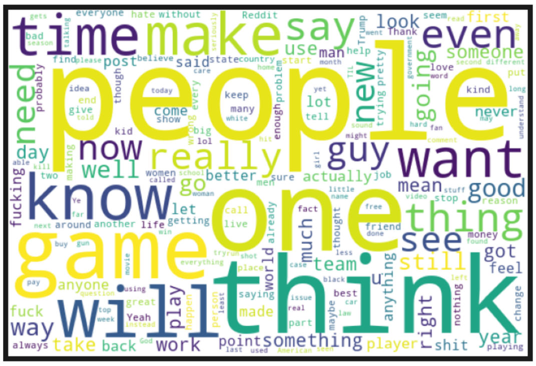


Figure 3.3: Wordcloud for parent comments that have received a sarcastic answer

3.2.2 Cross Validation

By partitioning the available data into three sets, the number of samples which can be used for learning the model will decrease, and the results can depend on a particular random choice for the pair of (train, validation) sets. A solution to this problem is a procedure called Cross Validation, which involves considering multiple choices for the pair (train, validation) sets and averaging evaluation metrics. With this methodology a test set should still be held out for final evaluation, but the validation set is no longer needed when doing cross validation. This approach, called k-fold, requires the splitting of the training set into k smaller sets: the model is at first trained using $k1$ of the folds as training data and then evaluated on the remaining part of the data.

3.3 Text classification

Finally the sentiment analysis model is built: this model will take the cleaned text of the parent comments as input and it will then come up with a prediction on whether the comment will receive a sarcastic or non sarcastic comment. Two logics were adopted and therefore two different input sets for the classification were mainly used to achieve the prefixed purpose resulting. A brief explanation on the logic behind both of them will follow along with the result of the classification models applied to them.

3.3.1 Alternative 1

The first alternative required the separation between the Tf-Idf vectorizers of the parent comments and that of the subreddits, that were afterwards joined using the hstack() which allows to stack the sequence of input arrays horizontally (namely column wise) to make a single array.

3.3.2 Alternative 2

The second alternative consisted of using both the ”cleaned” version of the parent text (namely the one obtained after applying stemming, tokenization and stopwords and punctuation removal) together with the subreddit name as input X, by simply joining the two texts.

Chapter 4

Results and Conclusion

In this section the final results and considerations are reported.

4.1 Results

Table 3.1 reports the accuracy of the different learning algorithms applied to the two alternative input data-sets as explained previously.

Table 4.1: Accuracy of the classifiers applied to the 2 alternatives

Dataset	Naive Bayes	Logistic Regression	Random Forest	Best CV
Alternative 1	57.26%	59.97%	52.94%	NB = 57.46%
Alternative 2	56.82%	60.31%	51.20%	Logit = 57.27%

4.2 Conclusion

In the first alternative the best result is achieved by the Multinomial Naive Bayes (NB) algorithm with an accuracy of 57.26% that with Cross validation reaches an accuracy of 57.46. The results of the classification performed on the second alternative are achieved by the Logistic Regression (Logit) with an accuracy that without cross validation reaches the 60%.

As it is possible to observe however these results cannot be considered the best ones: the best accuracy achieved barely reaches the 60%. This might be a consequence of three major facts: the first one is that we are using an indirect approach to discover the probability of receiving a sarcastic answer and not on the text of the comments that are actually the sarcastic answers and to which the labels refer to.The second reason could be linked to the data-set itself: analyzing some of the comments and labels assigned to them it is possible to note that in some cases some obviously sarcastic answers have not been labeled as such. It is obvious that such errors can and may have compromises the effectiveness of the classifiers decreasing their accuracy. And the last reason but probably also the most obvious one: all the analysis was performed, as previously stated, on a very small sample of the original dataset.

Bibliography

[1] Dan Ofer, Sarcasm on Reddit, https://www.kaggle.com/danofer/sarcasm

[2] Applied Text Mining in Python, University of Michigan, https://www.coursera.org/lecture/python-text-mining/introduction-to-text-mining-y5C24

[3] MonkeyLearn, "What is Text Mining?", https://monkeylearn.com/text-mining/

[4] Natassha Selvaraj, "A Beginner's Guide to Sentiment Analysis with Python" (Sep 12), A Beginner's Guide to Sentiment Analysis with Python

[5] MonkeyLearn, "Text Classification vs Text Extraction: What's the Difference?", https://monkeylearn.com/blog/text-classification-vs-text-extraction/

[6] Avinash Navlani, "Text Analytics for Beginners using NLTK" (13/12/2019), https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk