January 2021
Nataliia Zolotukhina
Universitat Pompeu Fabra

SARCASM DETECTION IN TWITTER AND THE CORRELATION BETWEEN SARCASM AND SENTIMENT

Research Project

---

1. INTRODUCTION AND LITERATURE REVIEW

The boost of social media in the past years has evoked a lot of communication being held online. People use various messengers and social networks as communication channels to capture some events, share news, express opinions, exchange information and so on. Alongside the information, the users also convey their sentiments, i.e. how they feel about a certain event: positive, negative or neutral. The study of opinions can be particularly useful for different business sectors to track user experiences connected to their products and services. But, as pointed out by Wandra and Barot (2017), the task of opinion mining (or better known as sentiment analysis) faces many challenges, and among the major ones is sarcasm (p.24).

In her work "On Irony and Negation", Giora (1995) refers to irony as to a special form of negation in which an explicit negation marker is absent (p. 240). Sarcasm is a form of irony, too (Yunitasari et al., 2019, p.54). And according to Bouazizi and Ohtsuki (2015), it typically occurs in the situations when a person utters some message, but in fact means the opposite thing, i.e. conveys an implicit piece of information. Below are the examples of sarcastic messages:

- *I absolutely love when people ignore me*;
- *So glad my computer is moving so slow... not like I wanted to get some sleep tonight*.

These examples are taken from Twitter. Riloff et al. (2013) noted that on Twitter there is a common pattern in the structure of sarcastic utterances (or tweets): they often carry a positive sentiment when referring to negative situations (p.704). This observation was later supported by Joshi et al. (2016) who stated that on the surface, sarcasm has a positive sentiment, while in fact it implies a negative sentiment. And in the examples above we see how the positive verbs "love" and "glad" are used in the contexts of negative situation.

From these definitions and observations we see that sarcasm is closely related to sentiment. Specifically, it is related to sentiment incongruity (Joshi et al., 2015). That is why it becomes a real challenge for the sentiment analysis.

Extensive research has been previously done on this relation, and the general tendency shows that sarcasm detection helps to improve performance of the computational models designed for sentiment classification. For example, Yunitasari et al. (2019) reported that the accuracy of the sentiment analysis model increased for about 5.49% when sarcasm detection was performed on the initial step of Indonesian tweets. Similarly, a study by Bouazizi and Ohtsuki, T. (2015) showed that when the sentiment classificator was aware of sarcastic nature of the analyzed texts, its performance improved for about 4%, too. The aim of the present study is to look into this matter in the opposite direction. If sarcasm detection can improve sentiment analysis, will then sentiment analysis improve sarcasm detection as well? In this way, this research will focus on exploring the influence of sentiment on a sarcasm detection model. The next section will present more specific research questions and the motivation to conduct such research.

2. RESEARCH QUESTIONS AND THE MOTIVATION OF THE STUDY

Summarizing the literature review from the previous section, we see that sarcasm and sentiment are linked to each other. This connection has been studied before, but with an emphasis mainly on sentiment analysis and how awareness of sarcasm can improve it. The main concept of this study, on the contrary, will be sarcasm: the nature of sarcasm as a linguistic phenomenon, and sarcasm detection as an NLP task.

My main interest is to see what makes a statement sarcastic, that is, what linguistic tools are used to express sarcasm and irony in written speech. As Joshi et al. (2016) discuss, a person to know best whether a statement is sarcastic or not is an author of the statement, and often authors provide certain labels to emphasize their sarcastic tone (which may otherwise be not recognized as such). The most popular form of such labels is a respective hashtag, e.g. #sarcasm or #sarcastic. But if an author does not indicate it explicitly, is it still possible to tell sarcasm from a neutral narrative statement? Do sarcastic statements share any common features? Hence, the first research question of this paper is: **what are the linguistic peculiarities of sarcasm?** Since sarcasm detection models have been designed before, I hypothesize that it is possible to identify common linguistic features in sarcastic texts. To prove the hypothesis and answer this research question, I will design my own sarcasm detection model based on the hand-crafted features computed after observing the dataset.

The second part of the study will be focused on the correlation between sarcasm and sentiment. Specifically, I will address the following research question: **will sentiment as a feature improve sarcasm detection model?** From numerous definitions we have seen that sarcasm is strongly associated with sentiment and sentiment discrepancy. This suggests a hypothesis that sentiment as a feature carries relevant information and will therefore facilitate sarcasm detection. To test this hypothesis, I will compare the performance of the model without sentiment as a feature, and after adding it.

Riloff et al. (2013) explain the importance of recognizing sarcasm for natural language processing: sarcastic statements can be misinterpreted as literal (p.704). Further, it may lead to more

undesirable confusions related to semantics, and have negative impacts on other tasks. Hence, the motivation to conduct this study is to learn how to recognize sarcasm, and also to fill the research gap as to effects of sentiment features on sarcasm detection tasks.

3. RESEARCH METHODOLOGY

This section explains in details the course of the present study. Namely, it describes the used dataset, and the steps of designing a computational model for automatic sarcasm detection.

3.1 Corpus

The dataset used for this project was taken from Kaggle public resources. The dataset is called ***Tweets with Sarcasm and Irony***, it was collected by Nikhil John in November 2020 to replicate the study by Ling and Klinger (2016). The dataset was accessed on November 12, 2020 by the following URL: https://www.kaggle.com/nikhiljohnk/tweets-with-sarcasm-and-irony. This dataset consists of 2 files: train.csv and test.csv; both of them contain 2 columns:

- **tweet** (a text of a tweet),
- **class** (a tag to classify the respective tweet into one of the classes).

The original study by Ling and Klinger (2016) was aimed at separating sarcasm from irony. Therefore, this dataset has the following classes:

- **train.csv**: regular, irony, sarcasm, figurative (both irony and sarcasm);
- **test.csv**: figurative and irony.

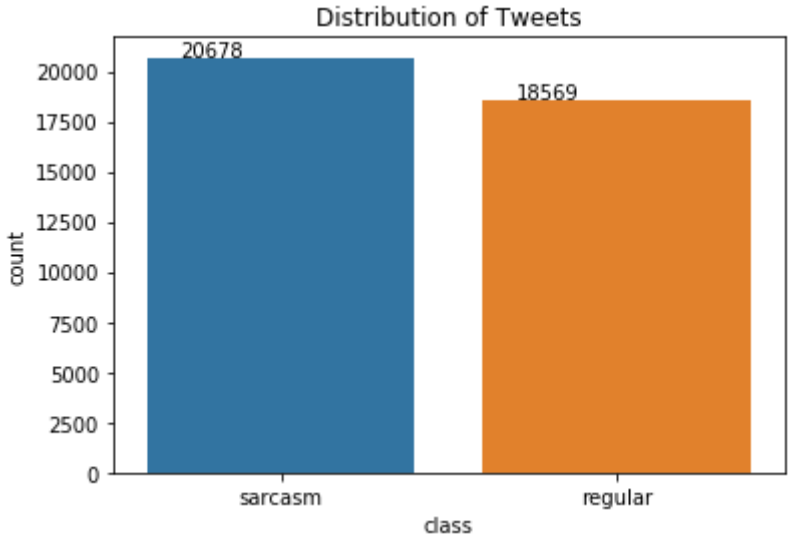
The research questions of the present study only focus on sarcasm, therefore it was decided to take only the tweets from the train.csv belonging to regular and sarcasm classes. The filtering was done in Microsoft Excel, and the working file was saved in the .xlsx format – **sarcasmdata.xlsx** (otherwise, Python did not correctly display the content of the file in .csv).

3.2 Data Exploration

The first steps to process the data file are shuffling the dataset, removing the duplicate rows (leaving only the first entries) and getting basic statistics with the **get_basic_stats()** function. All together the dataset contains 39247 tweets: 20678 sarcastic tweets and 18569 regular tweets.

```
Basic statistics for the dataset:
       count unique                                                top   freq
tweet  39247  39247  Whoa! Shonda folded Princess Diana into SCANDA...      1
class  39247      2                                             sarcasm  20678
```

The graph below visualizes this distribution:



The next step is data cleaning as it contains a lot of "noise" like user mentions, URLs, html tags, etc. This is characteristic of Twitter, but it has to be removed for proper text processing. Therefore, the function **clean_tweet()** does the following:

- fix special characters which are not displayed correctly. For example: ***donå«t*** substitute with ***don't***; but ***å*** followed by any other character just remove from the string (patterns are matched with the help of regular expressions); or substitute ***&*** with ***&***;
- remove *username mentions*, *html tags*, *URLs* and *punctuation* (everything except single and double quotes to preserve the contractions like *don't*, *we're*, *she'd*, etc.);
- remove all characters which do not belong to English alphabet, e.g. ***µ***, ***ŕ***, ***Ђ***, ***Љ***, ***Њ***, etc.;
- tokenize tweets and remove the *stopwords*;
- remove words ***"sarcasm"*** and **"sarcastic"**;
- remove single and double quotes if they are preceded or followed by a whitespace character (to avoid inaccuracies in further token counting).

The decision to remove sarcasm and sarcastic is consistent with the first research question. In the present dataset, the majority of sarcastic tweets contain *#sarcasm* or *#sarcastic* tag. However, the objective of this paper is to identify sarcasm without looking at explicit sarcasm markers. Several options as to how to remove these hashtags were considered. The major pitfall is that some tweets contain such hashtags as a part of running text. For example:

- *@electricland I find the **#sarcasm** hashtag handy in situations like this.*
- *He is a great leader and this is not **#sarcasm** at all. hidup #malaysia ! http://t.co/3nuUOpiqn5*
- ***#Sarcastic** People Are Actually #Smarter, #Sexier And More #Successful http://t.co/zOhpGmJIql*

I tried the following approach: tokenize the raw tweet and look for *#sarcasm/ #sarcastic*. If detected and followed by words (tokens which do not start with ***#***, ***@*** or ***http***), then leave it; if detected and not followed by anything (the end of the string) or followed by other hashtags/ URLs/ user mentions, then remove it. But this approach did not prove universal because of such examples:

- *Star wars marathon on a bank holiday .. Woo fun times **#sarcasm** hate being on crutches pμ□ ”*

  (this tweet is also an instance of why single/ double quote preceded or followed by a whitespace has to be removed)
- *Congrats to @CNN for announcing they'll show the shooting only once per hour. **#sarcasm** How about NO times per hour?*
- ***#sarcasm** WOW! Look at that! Another Peter Pan film! That's awesome! It's exactly the sort of creative thinking that we need right now!*

Here the target hashtags are present in the middle or even in the beginning of a string, but are not parts of running text.

In the end, it was decided to remove all instances of ***sarcasm*** and ***sarcastic*** (with or without the hashtag symbol *#*), even when it is a part of a sentence (similar to how the stopwords are removed, leaving blank spaces in the sentences).

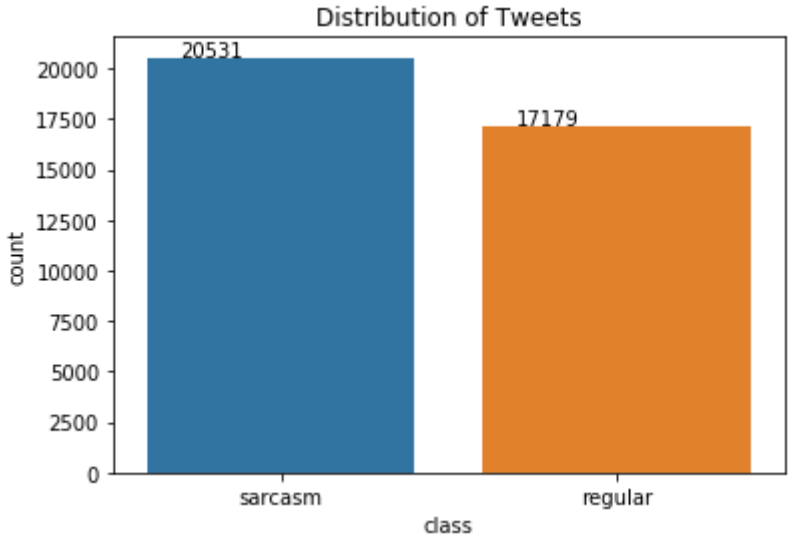
Interestingly, after cleaning the data, the same **get_basic_stats()** function shows that there are less unique values now:

	count	unique
tweet	39247	39247
class	39247	2
is_sarcastic	39247	2
hashtags	39247	17896
cleaned_tweet	39247	37710
tokens	39247	37710

Therefore, the duplicates are removed again, and we have roughly 1500 tweets less:

	count	unique
tweet	37710	37710
class	37710	2
is_sarcastic	37710	2
hashtags	37710	17885
cleaned_tweet	37710	37710
tokens	37710	37710

The final distribution of tweets per class is the following:

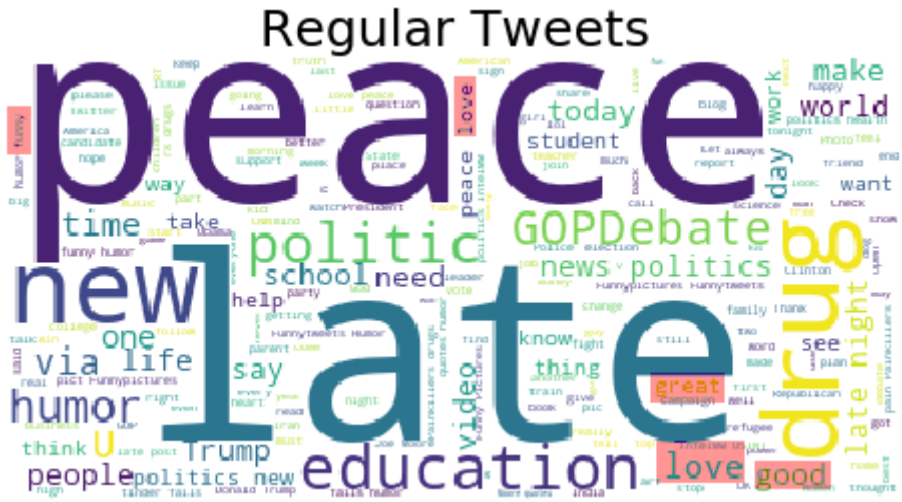


*(Execution of **get_basic_stats()** with the new columns after the data cleaning displays the results, but also raises a pandas error:*

```
  File "pandas\_libs\hashtable_class_helper.pxi", line 1652, in pandas._libs.hashtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'list'
```

Therefore, these parts are left in the code, but commented out)

The final step of data exploration is getting the word clouds of the texts, separately for regular and for sarcastic tweets.



What strikes the eyes at this point is the positive lexicon of sarcastic tweets. We see a lot of positive words like ***love***, ***great***, ***good***, ***awesome***, ***nice***, ***best***, etc. (highlighted in violet). In regular tweets, the positive lexicon is not so evident, only few words like love, good, great are marked on the WordCloud (highlighted in red).

This observation is consistent with the previous definitions of sarcasm that for expressing sarcastic tone, people choose positive words for talking about negative situations, or that people mean the opposite of what they utter in words. For example, the most striking (on the WordCloud) positive verb ***love*** does not actually mean that the authors really love what they are talking about in the tweets like these:

- *One thing I **love** is calls at 1:45am. I think it's impossible to underestimate how much I enjoy these calls. #sarcasm*;
- *Gotta **love** when you make an appointment, get there early and still waiting to be called two hours later. #sarcasm.*

Here we have instances of negative situations, and the usage of ***"love"*** when actually not loving the described situations at all.

Another noticeable thing is an abundant presence of interjections in sarcastic tweets like ***yeah***, ***lol***, ***wow***, ***oh***, ***yay*** (highlighted in green). The WordCloud of regular tweets does not contain them at all. It suggests that sarcastic tweets tend to be more colloquial, expressive and 'lively', while regular tweets are more formal and neutral.

3.3 Basic Features

Now, when the data is clean, we proceed to computing the basic features for the future sarcasm detection model. The features are resulting from observing the dataset, and at this point they do not relate to sentiment at all. There are 3 numeric features and 4 Boolean features to start with, and below they are discussed in details (including the motivation for relying on them).

The present dataset shows the tendency that on average, **sarcastic tweets** are short and to the point, contain only 1 hashtag (#sarcasm, #sarcastic), but quite often contain multiple question or exclamation marks. Examples:

- *Break his promise? Trump? Hard to imagine #sarcasm*
- *Best night ever !!!!! #sarcasm*

On the contrary, **regular tweets** are longer on average and have more hashtags (and not only at the end of a tweet, but inside it as well, as a running text). They can also contain question or exclamation marks, but usually it is less expressive (i.e. not multiple). Examples:

- *Cool off from that blazing #weekend of #partying with this #mellow selection of #late night #melodic #dnb #chillout*
- *#Disgusted to learn that @parcelforce not only collected #late yesterday but totally #failed to deliver today! Must be on site by 0800.*

These observations give the basis for the following numeric features:

1) **'n_nltk_tokens'**: number of tokens

I tried 2 different ways of tokenizing tweets: with spaCy and with NLTK (TweetTokenizer). The difference of these methods is in processing contractions with auxiliary verbs and negations. TweetTokenizer does not split them in separate tokens, but spaCy does. For instance:

tweet: Oh well he's what's best for the club hehe #Sarcasm https://t.co/a93ugOgyNt

cleaned_tweet: Oh well he's what's best club hehe

spacy_tokens: ['Oh', 'well', 'he', "'s", 'what', "'s", 'best', 'club', 'hehe'] – **9 tokens**

nltk_tokens: ['Oh', 'well', "he's", "what's", 'best', 'club', 'hehe'] – **7 tokens**

The tokens like ***is***, ***are***, ***not***, ***had***, ***will*** and others that can form contractions are in the list of the stopwords, so they have been previously removed. For the consistency reasons, there is no reason

for keeping them now as separate tokens and including them in the token count. Therefore, TweetTokenizer is prioritized for this feature.

2) **'n_hashtags'**: number of hashtags

Hashtags are collected from the raw tweets (normalized tweets do not contain # characters), #sarcasm and #sarcastic excluded.

3) **'n_!|?'**: number of question and exclamation marks

Similarly to hashtags, collected from the raw tweets.

Once these features are computed, we can get the mean values of each, per class. These values are consistent with the prior observations: on average, regular tweets have more tokens, more hashtags, but less question/ exclamation marks then sarcastic tweets.

class	n_nltk_tokens	n_hashtags	n_!|?
regular	9.505134	3.341092	0.271147
sarcasm	7.450251	0.814703	0.531273

Further, it was observed that sarcastic tweets are more colloquial and emotional in terms of language usage. For instance, as we have seen on the WordClouds, they contain a lot of interjections:

- ***Oh yeah*** *let's block someone on Twitter because that will show them. #sarcasm*
- *One time for #insomnia* ***Yay****! #sarcasm*
- *Tomorrow should be fun* ***lol*** *#sarcasm*

In some cases, to add expressiveness, some words have double/ triple/ or more letters where original spelling must have just 1 (e.g. soo, yayyy, probbbbably):

- *..****aaa****nd it's raining again after I left my umbrella in the car. Today just gets better and better. #sarcasm*
- *@Justin_Rogers @Ballin4Ever84 i smell a ch****aaa****mpi****oo****nsh****iiiiiippp****!!!! #InKellenWeTrust #Sarcasm*

In sarcastic tweets, some words are stressed by means of capitalization. This approach also adds extra expressiveness:

- *What?? It's raining* ***AGAIN*** *in Middle Tennessee?! #Sarcasm*
- ***SO EXCITED FOR TOMORROW*** *#sarcasm*

Finally, sarcastic tweets contain a lot of user mentions. In some cases, the mention appears because the tweet is a response to the mentioned user. In other cases, certain accounts are referred to/ addressed in a narrative way:

- ***@princeess_G*** *i was being #sarcastic haha*
- *thanks* ***@NetflixUK*** *for removing Friday night dinner whilst I was mid series! Much appreciated! #Sarcasm*

(however, regular tweets also have many mentions – this is characteristic of Twitter in general).

These peculiarities are implemented as Boolean features. Here it is not so relevant to count how many interjections or capitalized words are there per tweet. It is more important to know

whether in general they are present or not (with some conditions for the multiple letters and capitalized words). Therefore:

4) **'mentions'**

Hashtags are matched from the raw tweets with regular expressions.

5) **'has_interjections'**

Interjections are detected with regular expressions. The used pattern includes the following interjections: ***ye, yea, yeah, lol, lmao, oh, wow, aw, ops, woho, wtf, aha, yup, yeap, haha, hehe***. The pattern allows for any letter doubling, too. For example, **w+o+w+** will match *wow*, *wooow*, *wowww*, or any other variation of it. Some patterns, like '***aw***' or '***oh***' form parts of words, like in ***aw****esome, str****aw****berry, **Oh**io, alc****oh****olic beverage*. To avoid such matching, the pattern includes the look ahead and the look behind to match the pattern only if there is a whitespace before and after it (for this step, extra whitespaces are added in the beginning and in the end of a string for matching the interjections that open or close the tweet).

Regular expressions were prioritized over detecting interjections with the help of the POS tags because the latter method was less accurate. For example, interjections ***'lmaooo'*** and ***'yea'*** in the tweet *'@bmoe_careful lmaooo yea this look like a nigga thats beefin wit meek #sarcasm http://t.co/cPzm6lDFIE'* have been tagged as a noun and a personal pronoun: [('lmaooo', 'NN'), ('yea', 'PRP'), ('look', 'VBP'), ('like', 'IN'), ('nigga', 'NNS'), ('thats', 'NNS'), ('beefin', 'VBP'), ('wit', 'NN'), ('meek', 'NN')].

6) **'tripled_letters'**

The present dataset has some instances where a double letter is already a deviation (e.g. *s****oo*** *great*). But a lot of words in English contain double letters, too (e.g. *b****oo****k, wi****ll****, betw****ee****n*, etc.). So, detecting double letters will not bring the desired effect. Therefore, the regular expression matches only triple letters – this is already not characteristic to English (and if a letter has more than 3 repetitions, it will be matched as well).

7) **'nltk_capitalized'**

The pitfall for this feature is the presence of abbreviations in tweets which are also capitalized. Since we cannot have a list of all possible abbreviations similar to the list of the stopwords (any company name can be an abbreviation, and new companies emerge on a frequent basis), it is decided to take into account only the capitalized tokens that are longer than 3 characters. This approach is justified by the fact that the majority of wide-spread abbreviations or proper name consist of 2 or 3 letters (**US** for the United States, **GOP** for the Grand Old Party, **CNN**, **BBC**, etc.), and the majority of the short words that are not abbreviations but can be capitalized are removed as the stopwords (the, is, not).

The mean values of these features per class also prove that sarcastic tweets more often have mentions, interjections and tripled letters. However, capitalized tokens are more frequently seen in regular tweets (this can be due to the stopwords removal or the condition of the token length being more than 3 characters).

class	mentions	has_interjections	tripled_letters	capitalized_tokens
regular	0.210302	0.032377	0.017501	0.131665
sarcasm	0.409058	0.119095	0.030517	0.070638

In the course of computing these features, several columns are added to the dataframe: '**nltk_tokens'**, '**nltk_pos'**, '**hashtags'**, **'interjections'**. The **'nltk_tokens'**, **'hashtags'** and **'interjections'** columns have been already explained before along with respective features. As for the **'nltk_pos'**, I try 2 different approaches:

1) tag already normalized and tokenized tweet;
2) tag a raw tweet (tokenized with *nltk.tokenize.word_tokenize* – this method separates hashtag sign #), but then return token-tag pairs only for the tokens which are present in the normalized and tokenized version of the tweet.

These approaches give different results. Even though the majority of the tokens get the same POS tags assigned, there are some disparities, too. For example, consider below:

tweet: *@SteveinLC Good thing that bum McCullers is off the roster #sarcasm*

approach 1: [('Good', 'JJ'), ('thing', 'NN'), ('bum', 'NN'), ('McCullers', 'NNP'), ('roster', 'NN')]

approach 2: [('Good', 'NNP'), ('thing', 'NN'), ('bum', 'NN'), ('McCullers', 'NNPS'), ('roster', 'NN')]

Tokens 'Good' and 'McCullers' get different POS tags, and the correct ones are assigned in the first approach ('Good' is an adjective and not a proper noun, 'McCullers' is a proper noun in singular and not in plural).

Other examples like this have been detected across the dataset as well. Hence, the first approach (POS tagging of normalized tweets) is prioritized for the accuracy reasons.

3.4 Sentiment as a Feature

The basic features unrelated to sentiment are now implemented. The next step is to proceed to sentiment, i.e. add sentiment as another feature. For this research, I have considered two ways of obtaining sentiment:

1) use an additional dataset from the same genre (Twitter) annotated for sentiment, train a sentiment classification model on it by means of Logistic Regression with CountVectorizer, and use it to predict sentiment of the tweets in the initial dataset I work with;
2) use a pre-existing sentiment classifier, for instance TextBlob.

Both methods have been tested on an additional dataset annotated for sentiment to compare their performances. TextBlob showed lower accuracy then Logistic Regression (65% against 77%). However, in the sarcasm detection model, the coefficient of the TextBlob sentiment feature gets higher coefficient then the sentiment predicted by Logistic Regression trained on additional dataset (2.64 against 0.40). Therefore, this comparison does not give a single, definite answer as to which method is better. But we have to keep in mind that Logistic Regression in this case involves an additional dataset, and its higher accuracy might be just due to the specificities of this data, while TextBlob is presumably less subjective since it is an open tool which has been heavily tested and reported previously. Another thing to mention is that Logistic Regression in this case makes binary predictions: true for positive sentiment, and false for negative sentiment. In its turn, TextBlob gives a numeric value for sentiment ranging from -1.0 to 1.0 (meaning negative sentiment for the values below 0, and positive sentiment for the values above 0), and additionally, it gives a score of subjectivity in the range from 0.0 to 1.0 (where 0.0 is very objective, and 1.0 is very subjective).

Such numeric scores might be more informative then a simple binary classification. Hence, taking into account all these considerations, TextBlob is prioritized for this research.

'Textblob_sentiment' and **'textblob_subjuctivity'** are added as two separate features to the dataset. From their mean values per class we see that on average, sarcastic tweets tend to be more positive, but this is also a more subjective classification decision. This result is consistent with what we have seen on the WordClouds – sarcastic tweets contain more positive lexicon. But since sentiment incongruity is characteristic of sarcasm, this also leads to higher subjectivity scores.

class	textblob_sentiment	textblob_subjuctivity
regular	0.059186	0.343798
sarcasm	0.151460	0.432365

3.5 Advanced Lexicon Features

Since the issue of sentiment is not so straightforward in sarcasm, having a single sentiment feature might not be enough for the sarcasm detection model. A more sophisticated approach might prove useful, and for that, I will compute more advanced features connected to sentiment. Again, some of them will be numeric, some of them will be Boolean.

The advanced features are based on opinion lexicon and on positive/ negative scores of separate words. Before proceeding to computing the features, I obtain the lists of positive and negative words (**positive_lexicon** and **negative_lexicon**, respectively), taken from the Opinion Lexicon resource collected by Liu and Hu (2004). It was accessed on October 6, 2020 by the following URL: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html. Also, I add the **'swn_scores'** column to the dataframe with a list of tuples per tweet: each tuple in the list corresponds to each token of a normalized tweet, and contains a token itself, its POS tag, and its positive and negative scores obtained from SentiWordNet. The scores are collected from the first synset of each lemma; if lemma is not found in SentiWordNet, or if token is not a noun/ verb/ adjective/ adverb, then the positive and negative scores for this token are assigned as 0. For example, for the tweet *'I dislike logical fallacies. Therefor they are wrong! #sarcasm'*, the **'swn_scores'** content will be: [('dislike', 'RB', 0.0, -0.0), ('logical', 'JJ', 0.625, -0.0), ('fallacy', 'NNS', 0.25, -0.5), ('therefor', 'NNP', 0.0, -0.0), ('wrong', 'JJ', 0.125, -0.75)].

The observations on the dataset show some similar sentiment patterns in the sarcastic tweets. For example, when a sentence sentiment is not monochrome and there is a sudden change in positive/ negative polarity in the beginning and in the end of the sentence, it is often a sign of its sarcastic nature:

- *Thank you so much for **messing up with my life** #sarcasm #LT*
- *Couldn't imagine a better way to spend my Friday off than **waiting for hours in line at the RMV** #sarcasm*

Or when a positive verb occurs in a negative context, it is most likely sarcasm, too:

- *I just love **having a flat tire first thing in the morning**, it just makes me so happy #Sarcasm*
- *I love when **people post controversial articles from unreliable sources** #sarcasm*

Such patterns suggest ideas for the following features. Numeric:

1) **'n_pos_words'**

Number of positive words per tweet which belong to the **positive_lexicon** list.

2) **'n_neg_words'**

Number of negative words per tweet which belong to the **negative_lexicon** list.

3) **'swn_stdev'**

Standard deviation of the positive and negative scores obtained from SentiWordNet. If standard deviation is high, it means the scores are spread out and there is a lot of variation from positive to negative or from negative to positive. So, the tweet is not monochrome in terms of sentiment, and it can be the sign of its sarcastic nature.

4) **'avg_max_diff'**

This feature is also based on the list of SentiWordNet scores, and it shows the difference between the maximum value and the average value. If the maximum value is high, but the average of all the scores is quite low, it means that a certain word stands out with its sentiment. This is a scenario of a positive verb occurring in a negative situation (described with negative nouns, adjectives, etc.).

The mean values of these features show that on average, sarcastic tweets have less positive words but more negative words comparing to regular tweets. The standard deviation of SentiWordNet scores, as well as the difference between the maximum and the average score, is higher in case of sarcasm.

class	n_pos_words	n_neg_words	swn_stdev	avg_max_diff
regular	0.828433	0.433088	0.091897	0.250701
sarcasm	0.786526	0.465997	0.119544	0.293200

The Boolean features are as follows:

5) **'polarity_change_swn'**

A tweet is split into 2 halves and each is checked on the sentiment score. If the number of tokens is odd, the first half gets less tokens. The logic behind this decision is that usually the second half of a tweet is more informative: *I just love how …[description of a situation]; This is adorable when …[description of a situation]; It's so great that …[description of a situation];* etc. All positive and negative scores obtained from SentiWordNet are summed up for each half, and if one number is positive (above 0), and the other number is negative (below 0), then there is a change in polarity inside a tweet (Boolean value: True). If the 2 numbers have the same sign (both positive, or both negative), there is no change in polarity on the ends of a tweet (Boolean value: False). If one of the numbers gives a neutral result (i.e. 0), it is handled as no change in polarity (i.e. False).

6) **'polarity_change_lexicon'**

This feature is similar to the previous one, but it looks at the number of positive and negative words in the beginning and in the end of a tweet. If one half has more positive words and the other half has more negative words, there is a change in polarity.

7) **'neg_avg/pos_max'**

This feature is an extension of **'avg_max_diff'**, as it takes the same values for comparison: the maximum and the average value of the SentiWordNet scores. If the maximum is a positive value while the average is a negative value, then the output is True; otherwise False.

The mean values of these features per class show that all 3 of them are more meaningful in sarcastic tweets, therefore they will be useful for the further classification model:

class	polarity_change_swn	polarity_change_lexicon	neg_avg/pos_max
regular	0.125365	0.072629	0.151908
sarcasm	0.138010	0.082631	0.155365

4. RESULTS

All the features described in the previous section are designed and implemented in order to train a Logistic Regression model for sarcasm detection. The model is trained in 3 stages, and this section presents the results obtained at each stage.

4.1 Baseline Solution

The baseline solution is needed as a threshold for comparing it with the model performance. Sarcastic tags are assigned randomly taking into account the proportion of regular and sarcastic tweets. The accuracy of this solution is roughly 50%.

```
Baseline accuracy:  0.503765579421904
```

4.2 Basic Features

The model trained on the first 7 features (without involving sentiment) gives an accuracy of 81.58% (precision 81.40%, recall 86.11%, f1-score 83.69%):

```
Testing accuracy:   0.8158279112999601
Training accuracy:  0.8058358783694064
```

This is significantly better then the baseline solution, so the chosen features prove being relevant. The training accuracy does not outstand the testing accuracy, so the model does not overfit. The full classification report and the confusion matrix are available in the Python code file.

To see which features are more relevant than the others, we take a look at the coefficients:

```
Model coefficients:
n_nltk_tokens          -->   -0.068
n_hashtags             -->   -0.932
n_!|?                  -->    0.345
mentions               -->    0.407
has_interjections      -->    1.702
tripled_letters        -->    0.436
capitalized_tokens     -->   -0.337
```

In general, Boolean features have higher coefficients, and the highest one is assigned to the feature which tells us whether a tweet has interjections or not. Among numeric features, the number of exclamation/ question marks feature has the highest coefficient, while the number of hashtags has the lowest coefficient among all the features.

4.3 Sentiment as a Feature

When TextBlob sentiment and subjectivity are added to the model, its performance scores 81.56% on accuracy (precision 81.30%, recall 86.21%, f1-score 83.68%). Despite of my expectations, only recall shows slight improvement of 0.2%. Other measures drop down a little bit.

```
Testing accuracy:   0.8155623423184172
Training accuracy:  0.8115124153498872
```

(the model is not overfitting)

However, TextBlob sentiment and subjectivity features get almost the highest coefficients (only yielding to **'has_interjections'**, which has the highest coefficient, again):

```
Model coefficients:
n_nltk_tokens              -->   -0.074
n_hashtags                 -->   -0.931
n_!|?                      -->    0.343
mentions                   -->    0.426
has_interjections          -->    1.476
tripled_letters            -->    0.487
capitalized_tokens         -->   -0.310
textblob_sentiment         -->    0.910
textblob_subjuctivity      -->    0.727
```

4.4 Advanced Lexicon Features

Now all other sentiment-related features are added, and the model performance rises till 82.10% on accuracy. All other measures also rise comparing to the first stage (first 7 basic features): precision 82.09%, recall 86.18%, f1-score 84.09%.

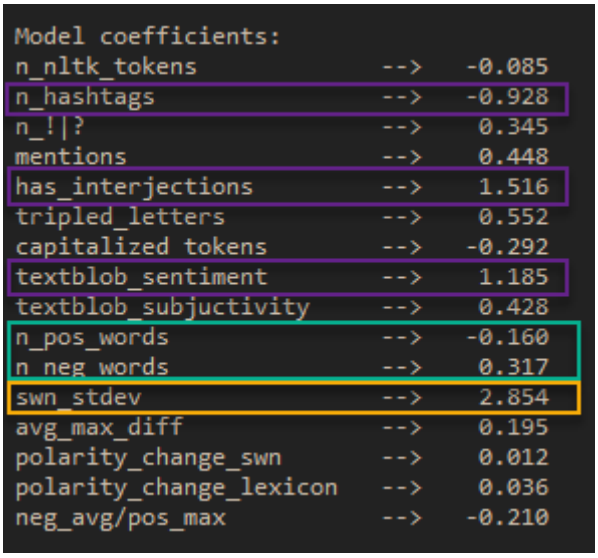
```
Testing accuracy:   0.8210065064400478
Training accuracy:  0.816126676404196
```

(the model is not overfitting)

It means that, while sentiment on its own has not improved classification decisions, a more sophisticated approach which involves features designed around opinion lexicon and positive/ negative scores of separate words has brought the desired results – the model performs better! Although, when other sentiment-related features are involved, the TextBlob sentiment importance is also weighted higher – its coefficient has increased.

But the highest coefficient on this stage is the one of standard deviation. So, it is important for the sarcasm detection model to see whether sentiments of separate tokens have little variation or, on the contrary, are spread out throughout a tweet. Interestingly, between the numbers of positive and negative words, the negative words are weighted higher, even though we have seen previously

on the WordClouds that positive lexicon is more present in sarcastic tweets. Interjections remain important as previously, only that now that yield to standard deviation. And the number of hashtags still has the lowest coefficient:



5. DISCUSSION

The results described in the previous section suggest some interesting insights as to sarcasm on its own, and also as to its correlation with sentiment. This section discusses these insights, as well as the limitations of the chosen research methodology.

5.1 Present Findings

In the course of this research I have trained a Logistic Regression model which detects sarcasm with 82.10% accuracy. This result is significant enough to give answers to the formulated research questions.

The first research question was about the linguistic peculiarities of sarcasm. We have seen how the model performs with different sets of features, and can therefore emphasize the most relevant ones to describe the linguistic peculiarities of sarcasm. First of all, the highest weight assigned to the feature of sentiment standard deviation throughout a tweet proves that **sentiment varies a lot in sarcastic speech**. The definitions mainly point out positive verbs in negative contexts. But this feature has been designed to be valid also for detecting negative verbs occurring in positive contexts, since such scenario can also mean sarcasm.

Further, the higher coefficients (compared to the other features) of interjections, tripled letters and exclamation/ question marks features reveal that **sarcastic speech is more vivid, expressive and lively, and also less formal then regular speech**. Nevertheless, it does not mean that expressive and informal speech is longer in terms of sentence lengths. Number of tokens has a negative feature coefficient, and it means that **it does not make a big difference whether a tweet is short or long**. **It can be sarcastic either way** (even though on average, we have seen that sarcastic tweets tend to be shorter). Number of hashtags features did not show a great significance

either. However, the results connected to number of tokens and number of hashtags may be not as accurate as, for example, the results for interjections and tripled letters because some hashtags and tokens have been removed in the course of text normalization. User mentions as a feature has got a comparatively high coefficient too, but I will not call it an important peculiarity of sarcasm; this is rather an important peculiarity of Twitter, and thus of sarcasm (solely) in Twitter.

Overall, the final combination of the features gives the highest performance result comparing to the other two model training stages, so they all can be considered relevant for detecting sarcasm. In this way, the first hypothesis is proved – it is possible to identify common linguistic features in sarcastic texts, implement them as hand-crafted features and train a sarcasm detection model based on them. The model can still be improved, though.

The second research question asks whether sentiment as a feature can improve sarcasm detection model. We have seen that after adding a sentiment as a single feature (or 2 features, including TextBlob subjectivity score), the model performance actually decreases. But interestingly, adding more sentiment-related features on top of that, does increase the model performance! This finding highlights the perplexing nature of sentiment in the context of sarcasm: an utterance can sound positive and have a positive literal meaning, while in fact the described thing/ action/ situation is negative and thus the implied meaning is negative, too. For this reason, **identifying an overall sentiment of an utterance (tweet, in this case) as a single characteristic is not enough for the task of sarcasm detection – it is confusing for the Logistic Regression model. But a more sophisticated approach, which involves features with opinion lexicon, sentence polarities and sentiment variations within a sentence, improves the accuracy of the model for 0.52% (precision for 0.69%, recall for 0.07%, and f1-score for 0.4%)**. Precision measure in this task is more relevant then recall, since we want to make sure that almost all the tweets identified as sarcastic are in fact sarcastic (rather than having as many sarcastic tweets as possible correctly detected)

This finding proves the hypothesis partially and suggests another look at it: sentiment really carries relevant information which can facilitate sarcasm detection model; however, it should be involved not as a single feature, but as a combination of multiple sentiment-related features.

5.2 Limitations and Pitfalls

The research designed used to conduct this study has facilitated the research process. Nevertheless, some limitations of this methodology have been encountered, too. The major one is **lack of context**. Joshi et al. (2016) highlight that context beyond text is important for sarcasm detection, and they provide references to other studies (e.g. Wallace et al., 2014) that highlight the important role of context in this task as well (p.11). Twitter has its own specific characteristics. For example, many tweets are not full sentences on their own, but just some replies to other users or comments under someone else's publications. In some cases, a part of tweet content is enclosed in links to external sources rather than in verbal utterances. All this increases the importance of context for sarcasm detection in Twitter even more.

Another limitation is connected to **the TextBlob sentiment model.** Based on my independent test involving an additional dataset annotated for sentiment, this model has not proved to be highly accurate in predicting sentiment (roughly 65%). It might be due to the peculiarities of

the additional dataset, or due to the TextBlob model itself – this issue has not been further researched in the present study. The information retrieved from TextBlob has been enough for completing this work on the initial stage. Nevertheless, this issue needs to be addressed differently in future work.

5.3 Future Work

Having mentioned the inaccuracies of the TextBlob model used in this study, I would suggest, among directions for future work, **a use of different models for sentiment prediction, or even human sentiment annotations** given the perplexing nature of the sentiment classification task in sarcasm detection. Sarcastic speech usually has different sentiments in literal and implied meanings, and human annotations will give reliable and consistent results.

Another thing to consider is **designing sentiment-related features that take into account syntax of a sentence**. In this study I haven't looked into dependencies inside sentences. But this approach might be more accurate for detecting positive verbs in negative situations then the one used in this paper (with the maximum, mean and standard deviation values of the sentiment scores). For example, if we look for a root token of a sentence and then for its children, we might get a better analysis of the context relation to the main verb. This can be done with the help of spaCy library which provides methods for assigning dependencies and detecting parent/ children tokens. In addition, some tweets contain several sentences, so it would be worth trying to implement the features which operate on a sentence level (e.g. polarity change in the beginning and in the end of a sentence) taking this into account.

6. CONCLUSION

The present study was aimed at exploring the phenomenon of sarcasm and the correlation between sarcasm and sentiment. The objective was to identify the linguistic peculiarities of sarcastic texts, and to see whether sentiment is among them. Overall, the study has been conducted successfully, the sarcasm detection model has been trained and tested, and the research questions have been answered. Recognizing sarcasm is in general important for natural language processing because it can prevent misinterpreting sarcastic statements as literal. Hence, the practical value of this research is that it suggests and tests the ideas for improving sarcasm detection involving sentiment.

One thing to keep in mind though is that I did not intent to train a universal sarcasm detection model. The used dataset is quite balanced, and so the trained model is sensitive to that. In real life, such proportion of sarcastic and non-sarcastic speech is not likely to be preserved. The objective of this research, however, was to identify common linguistic characteristics of sarcastic speech rather than training a universal classifier. And so for this objective, the balanced dataset has been suitable to work with.

REFERENCES

Bouazizi, M., Ohtsuki, T. (2015). Opinion Mining in Twitter: How to Make Use of Sarcasm to Enhance Sentiment Analysis. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 1594-1597. DOI: https://doi.org/10.1145/2808797.2809350

Giora, R. (1995). On Irony and Negation. *Discourse processes, 19* (2), 239-264. DOI: 10.1080/01638539509544916

Joshi, A., Bhattacharyya, P., & Carman, M. J. (2016). Automatic Sarcasm Detection: A Survey. *ACM Computing Surveys*, Vol. V, No. N, Article A.

Joshi, A., Sharma, V., Bhattacharyya, P. (2015). Harnessing Context Incongruity for Sarcasm Detection. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2*, 757-762.

Ling, J., Klinger, R. (2016). An Empirical, Quantitative Analysis of the Differences between Sarcasm and Irony. *European Semantic Web Conference*. DOI: 10.1007/978-3-319-47602-5_39

Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., & Huang, R. (2013). Sarcasm as Contrast between a Positive Sentiment and Negative Situation. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 704-714

Wallace, B. C., Kertz Do Kook Choe, L., Charniak, E. (2014). Humans Require Context to Infer Ironic Intent (So Computers Probably Do, Too). *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 512–516.

Wandra, K. H., Barot, M. (2017). Sarcasm Detection in Sentiment Analysis. *International Journal of Current Engineering and Scientific Research (IJCESR), 4* (9), 24-32.

Yunitasari, Y., Musdholifah, A., Kartika Sari, A. (2019). Sarcasm Detection For Sentiment Analysis in Indonesian Tweets. *Indonesian Journal of Computing and Cybernetics Systems (IJCCS), 13* (1), 53-62. DOI: https://doi.org/10.22146/ijccs.41136