Sarcaasm Detection

Presented by Group 4

August 19th, 2018

What’s Sarcasm?

Sarcasm: *a nuanced form of language where usually, the user explicitly states the opposite of what she implies.*

Why detect Sarcasm?

- It can be dangerous because if somebody is sarcastic and you dont know that they are sarcastic you might be lead to believe what is not true. *It's so bad that North Korean Government has banned sarcasm all together*

- Most large companies have dedicated social media teams providing real-time assistance to consumers.

Problems with Sarcasm Detection on Twitter

- Sarcasm is ambiguous
- No voice (no tone or gestures like rolling the eyes)
- No background information, such as who said it, or how they feel when they said it. Eg. I love being rich
- Lots tweets without #sarcasm
- Presence of URLs, ‘@‘ sign, ‘#’ tags
- Multiple languages
- Length of the tweets can be very short. It’s almost impossible to be sarcastic in 2 words
- Contradiction in polarity of the sentiment eg. I love being poor!

Problem Statement

*Given an unlabeled tweet **t** from user **u** along with a set of **u**'s past tweets **T**, a solution to sarcasm detection aims to automatically detect if **t** is sarcastic or not.*

Dataset: This file contains three columns -

○ ID - ID for each tweet
○ tweet - contains the text of the tweet
○ label - the label for the tweet (‘sarcastic’ or ‘non-sarcastic’)

Milestones

- Exploratory Data Analysis
- Data Pre-processing and Feature Selection
- Model Building

Exploratory Data Analysis

Our Data

	ID	tweet	label
0	T000452358	b'oh yea that makes sense '	sarcastic
1	T000452359	Estas enfermedad a un cargo poltico tu como pb...	sarcastic
2	T000452360	@alleygirl2409 until i\'m and all the old men ...	sarcastic
3	T000452361	b"@sarinas it had been chanted peacefully you ...	sarcastic
4	T000452362	b"there's nothing like being on vacation and h...	sarcastic

We cleaned the data & removed the ID column since it wasn't adding direct value to our problem.

And label encoded our target variable

	tweet	label
0	oh yea that makes sense	1
1	Estas enfermedad a un cargo poltico tu como pb...	1
2	@alleygirl2409 until i'm and all the old men w...	1
3	@sarinas it had been chanted peacefully you ca...	1
4	there's nothing like being on vacation and hav...	1

	tweet	label
51300	new blog post is up to put a smile on your fac...	0
51301	i got my flight details and i can't wait to se...	0
51302	did i hear correctly on we have a new album on...	0
51303	rt can not wait for wales vs nz tonight at ede...	0
51304	weekend spent with some of my	0

Data Preparation

Data Preparation 1: HTML decoding/souping

Reasoning:HTML encoding has not been converted to text

Data Preparation 2: ‘@’mention

Reasoning: Even though @mention carries a certain information this information doesn’t add value to build sentiment analysis model.

Data Preparation 3: URL links

Reasoning:The third part of the cleaning is dealing with URL links for sentiment analysis purpose, these can be ignored

Data Preparation 4: hashtag / numbers

Reasoning:Sometimes the text used with hashtag can provide useful information about the tweet. It might be a bit risky to get rid of all the text together with the hashtag. We decided to leave the text intact and just remove the ‘#’

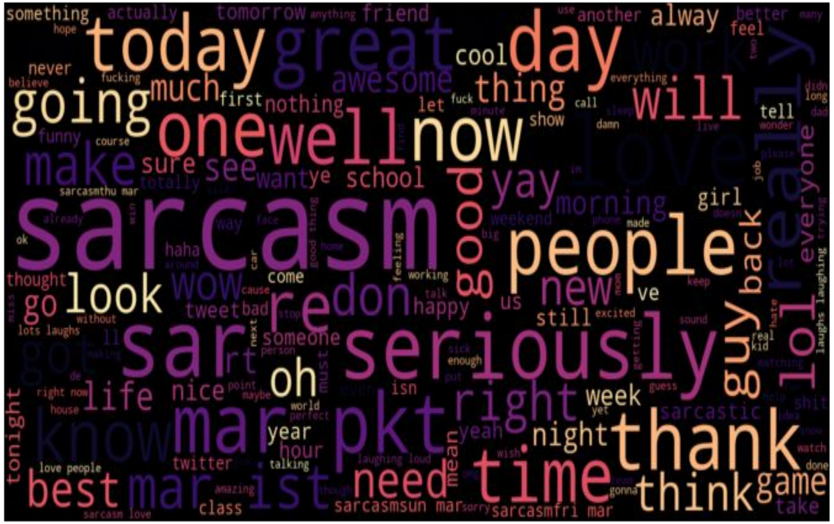
Data Preparation 5:Word Punct Tokenizer (Tokenize a text into a sequence of alphabetic and non-alphabetic characters)

Reasoning: In order to perform the cleaning, remove symbols, numbers, and unnecessary white spaces

Below is the updated data cleaning function. The order of the cleaning is

1. Souping
2. url address(‘http:’pattern), twitter ID removing
3. url address(‘www.'pattern) removing
4. lower-case
5. negation handling
6. removing numbers and special characters
7. tokenizing and joining

Sarcastic vs non-sarcastic wordclouds

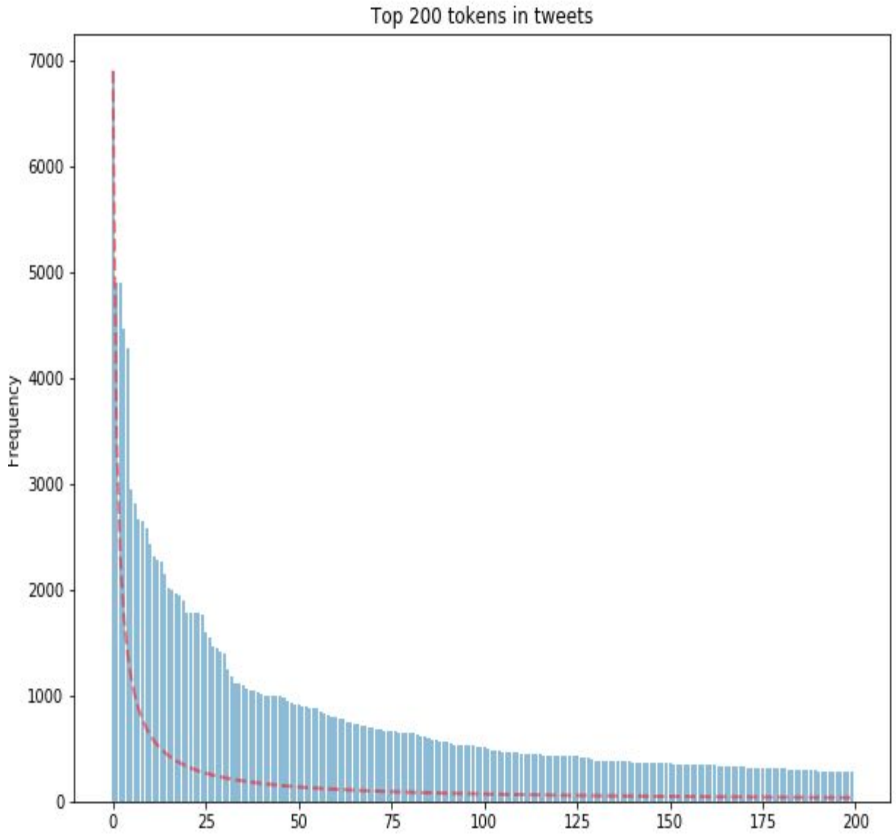


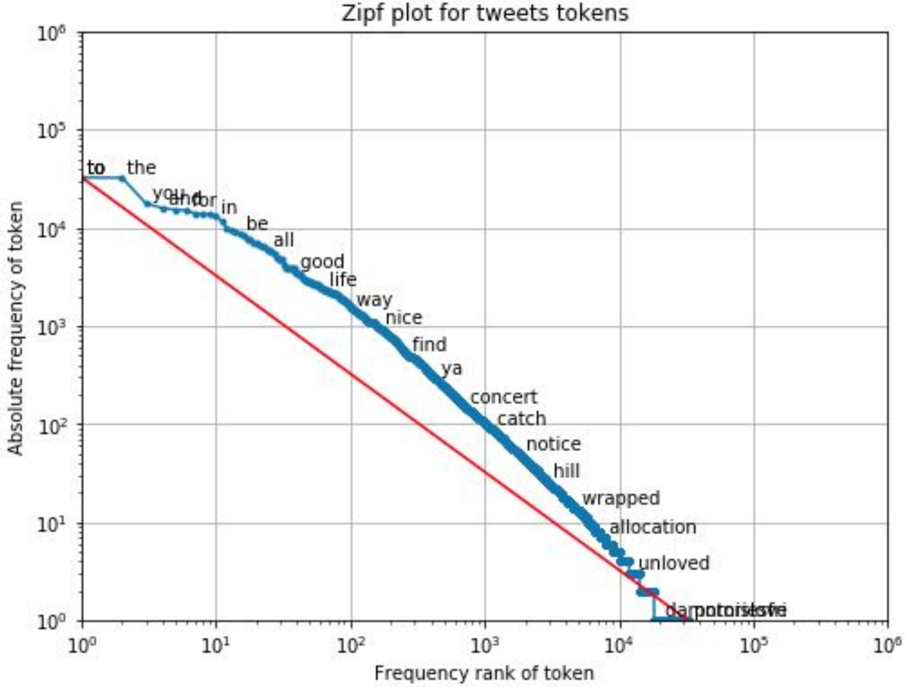
Zipf’s Law

Zipf’s Law states that a small number of words are used all the time, while the vast majority are used very rarely. We know that we use some of the words very frequently, such as “the”, “of”, etc, and we rarely use the words like “aardvark”.

	non-sarcastic	sarcastic	total
to	14348	18093	32441
the	13152	19048	32200
you	7050	10658	17708
and	6899	8849	15748
my	6655	8566	15221
for	7647	7315	14962
not	1092	12819	13911
of	5167	8538	13705
is	5148	8545	13693
in	5649	7462	13111

To see zipf’s law in action we need term frequency data. What kind of words are used in the tweets, and how many times it is used in entire corpus. We used countvectorizer to calculate the term frequencies.





We proved that even the tweet tokens follow “near-Zipfian” distribution

Tweet Tokens Visualisation

After having seen how the tokens are distributed through the whole corpus.

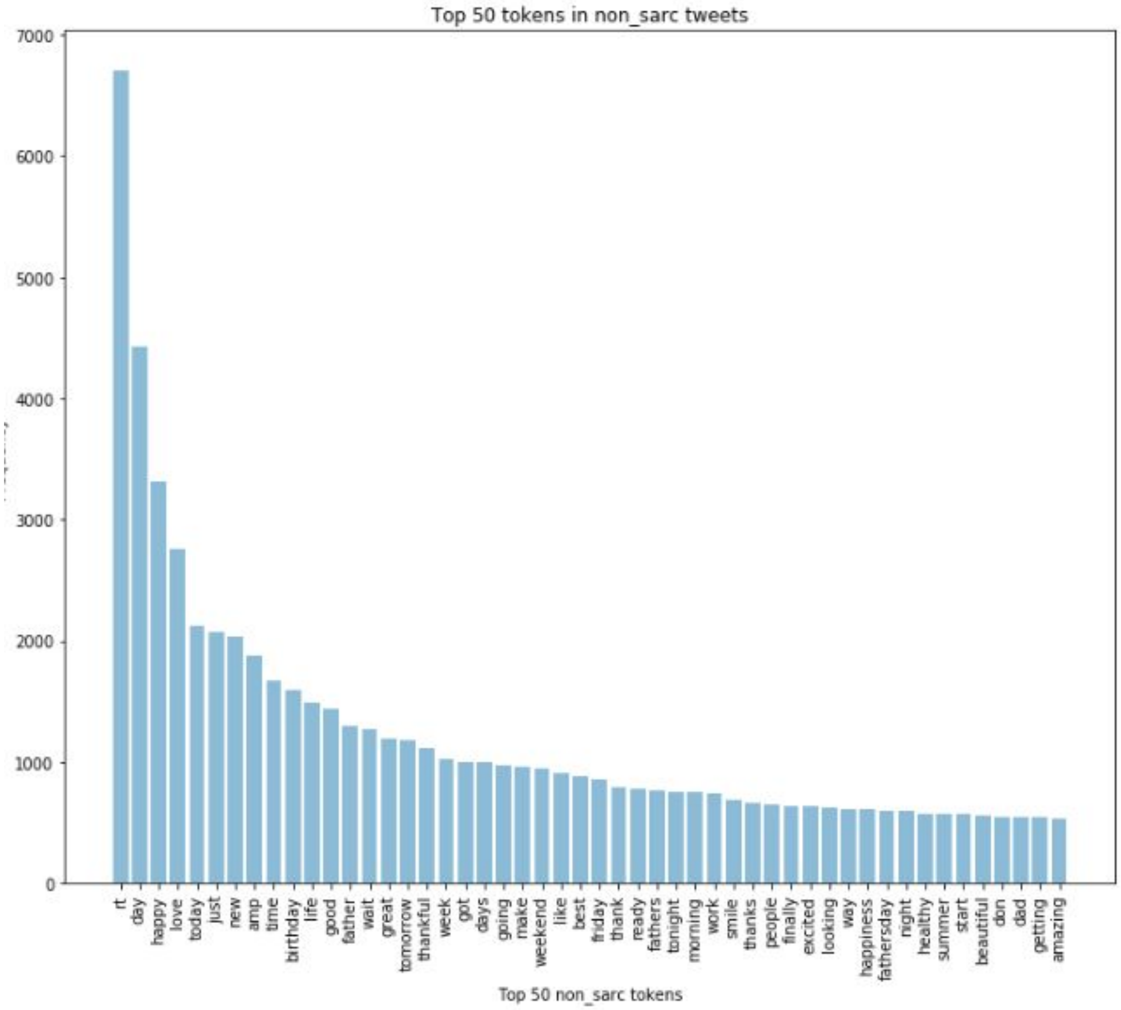
We saw how different are the tokens in two different classes(positive, negative).

The stop words do not help much, because the same high-frequency words (such as "the", "to") will equally frequent in both classes.

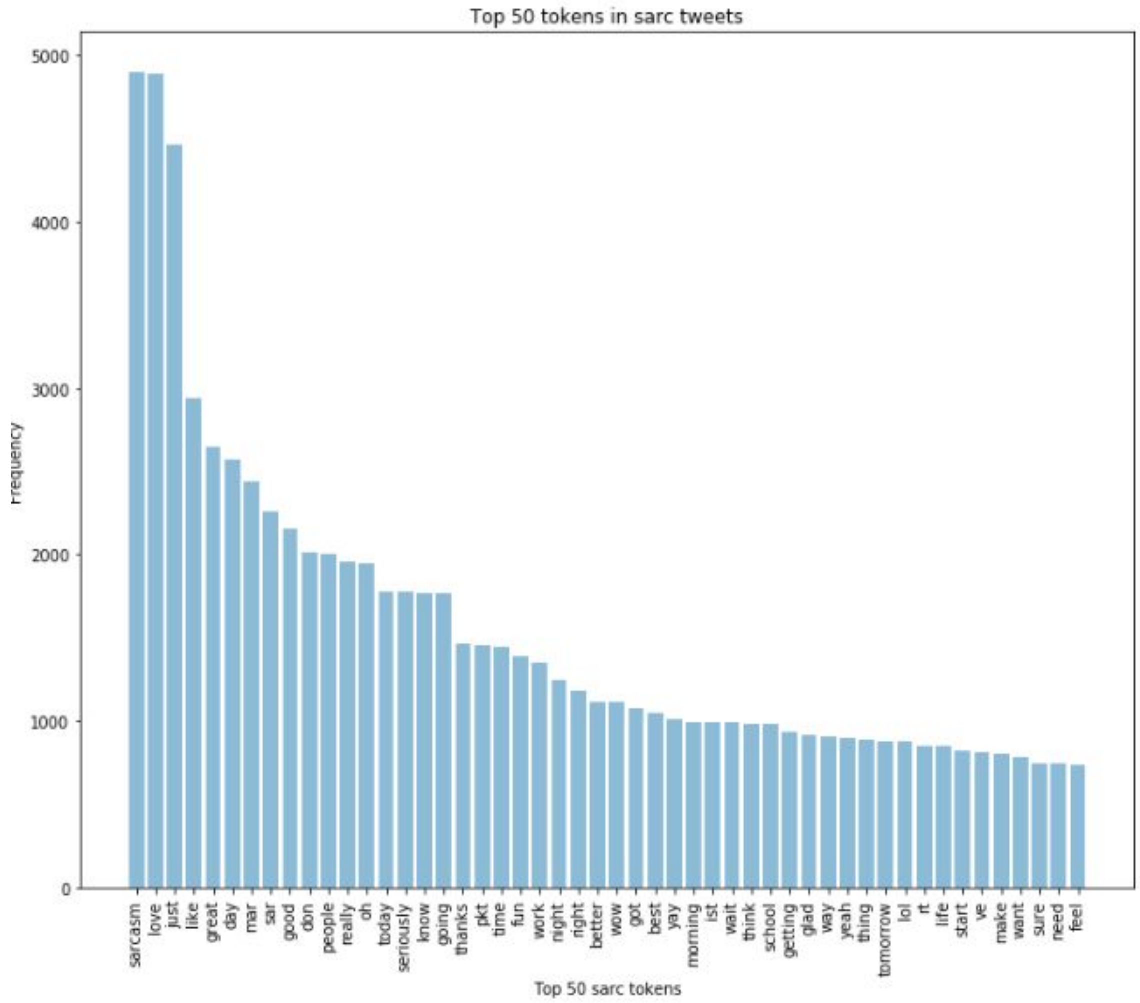
If these stop words dominate both of the classes, we wont get a meaningful result. So we remove stop words, and also will limit the max_features to 5,000 with countvectorizer.

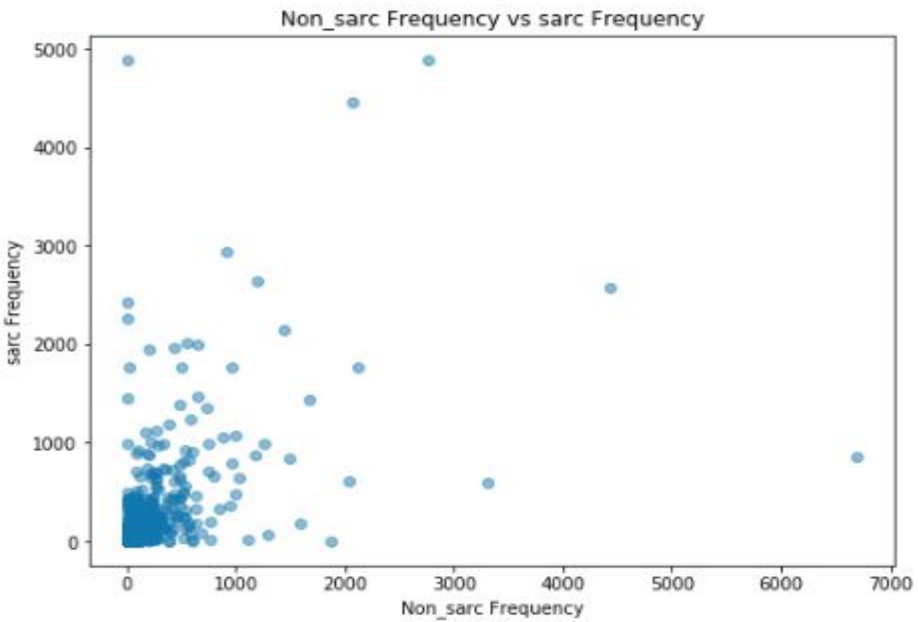
Term frequency after stop word removal

	non_sarc	sarc	total
love	2763	4888	7651
rt	6701	852	7553
day	4425	2572	6997
just	2075	4464	6539
sarcasm	0	4895	4895
happy	3308	591	3899
today	2117	1775	3892
like	907	2944	3851
great	1191	2648	3839
good	1440	2155	3595



Top 50 tokens in non_sarc tweets
7000
6000
5000
4000
3000
2000
1000
0
rt
day
happy
love
today
just
new
amp
time
birthday
life
good
father
wait
great
tomorrow
thankful
week
got
days
going
make
weekend
like
best
friday
thank
ready
fathers
tonight
morning
work
smile
thanks
people
finally
excited
looking
way
happiness
fathersday
night
healthy
summer
start
beautiful
don
dad
getting
amazing
Top 50 non_sarc tokens


Top 50 tokens in sarc tweets
Frequency
0
1000
2000
3000
4000
5000
sarcasm
love
just
like
great
day
mar
sar
good
don
people
really
oh
today
seriously
know
going
thanks
pkt
time
fun
work
night
right
better
wow
got
best
yay
morning
ist
wait
think
school
getting
glad
way
yeah
thing
tomorrow
lol
rt
life
start
ve
make
want
sure
need
feel
Top 50 sarc tokens

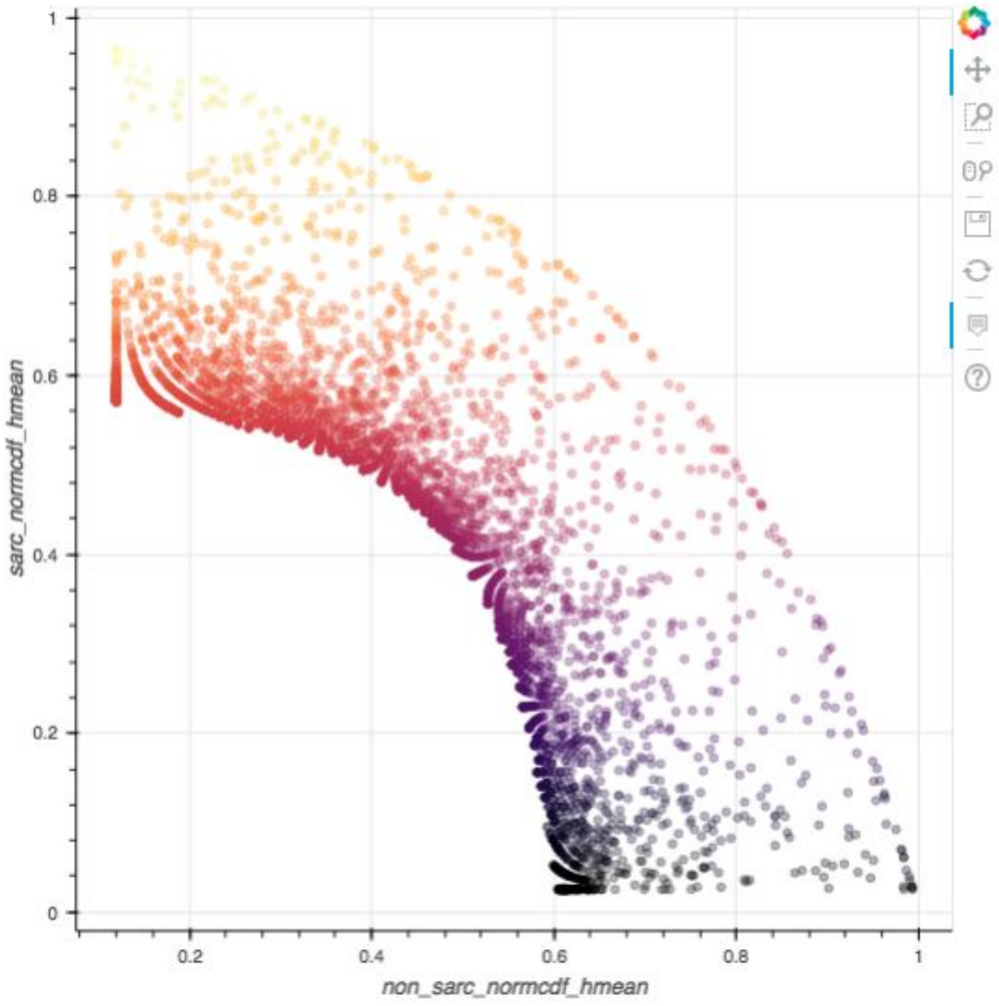


This graph does not help us characterise important tokens in each class we therefore make a interactive plot with Bokeh.

With below Bokeh plot, you can see what token each data point represents by hovering over the points.

For example, the points in the top left corner show tokens like “sarcastic”, “loud”, etc.

And some of the tokens in bottom right corner are “father”, “thankful”, “happiness”, etc. And the color of each dot is organised in “Inferno256” color map in Python, so yellow is the most sarcastic, while black is the least negative, and the color gradually goes from black to purple to orange to yellow, as it goes from non-sarcastic to sarcastic.



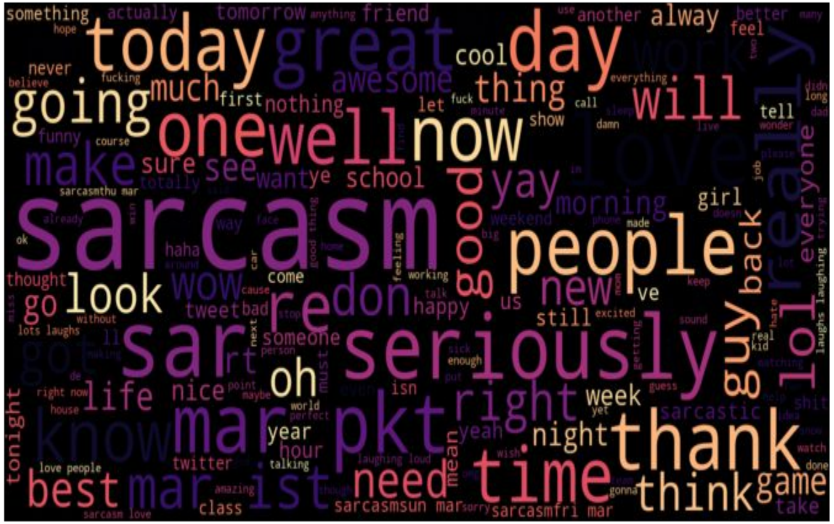
sarc_normcdf_hmean
non_sarc_normcdf_hmean
0
0.2
0.4
0.6
0.8
1

Data Preprocessing & Feature Selection

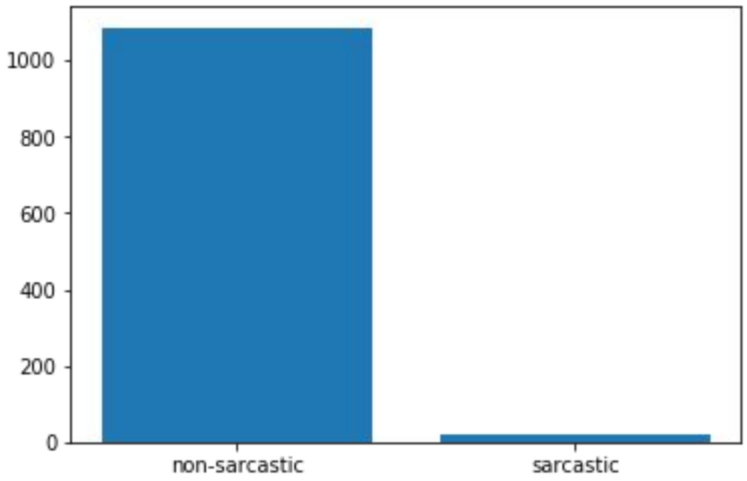
Features

- Word Unigrams and bigrams using TF-IDF
- Tweets with Links
- UpperCase Tweets
- Language
- Retweets
- Hashtags
- Words with highest frequencies (only for base-line model)

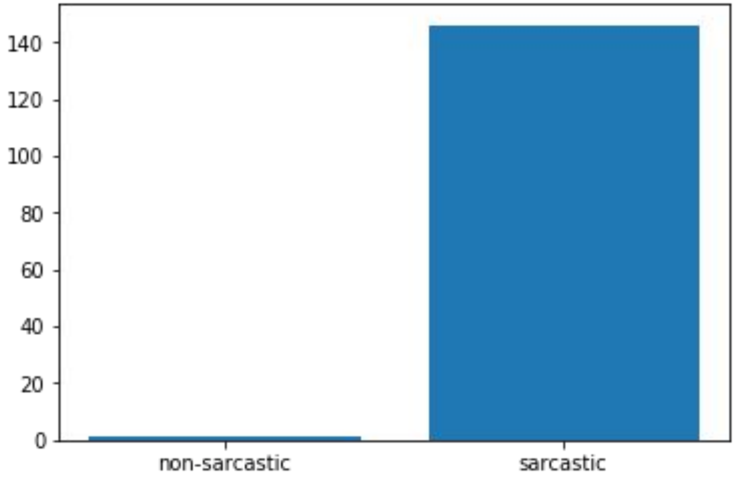
Terms with highest frequency



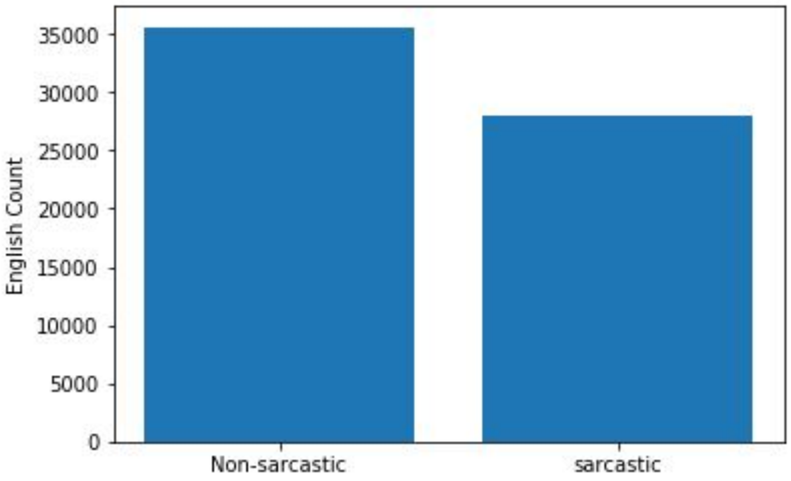
Tweets With Links

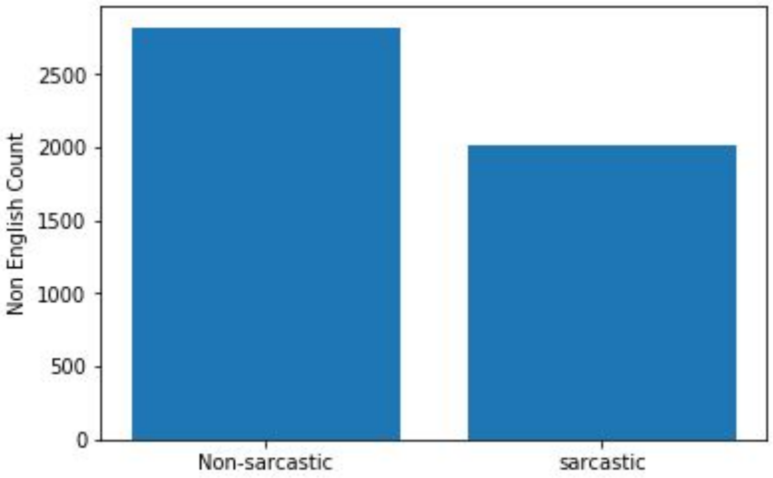


Tweets In UpperCase



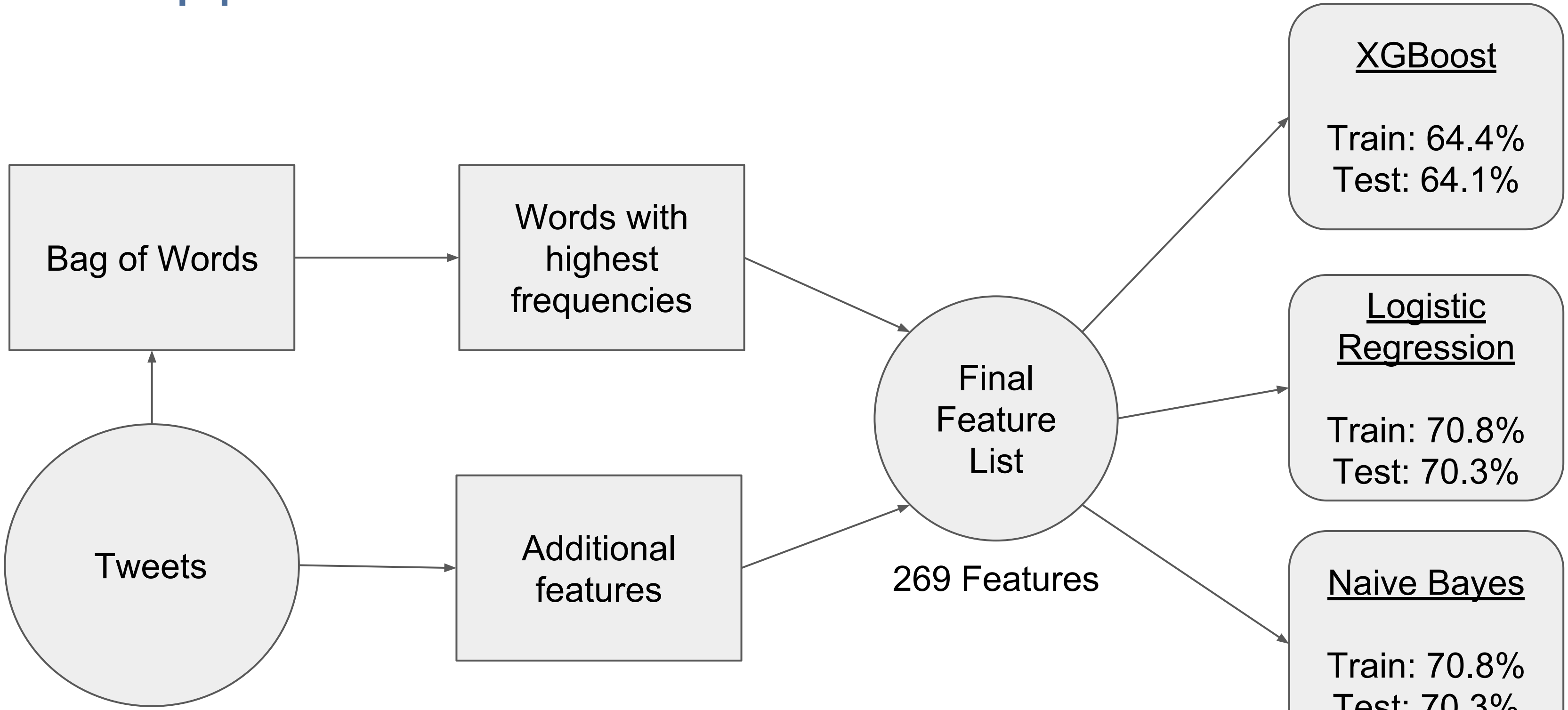
Language of the tweets



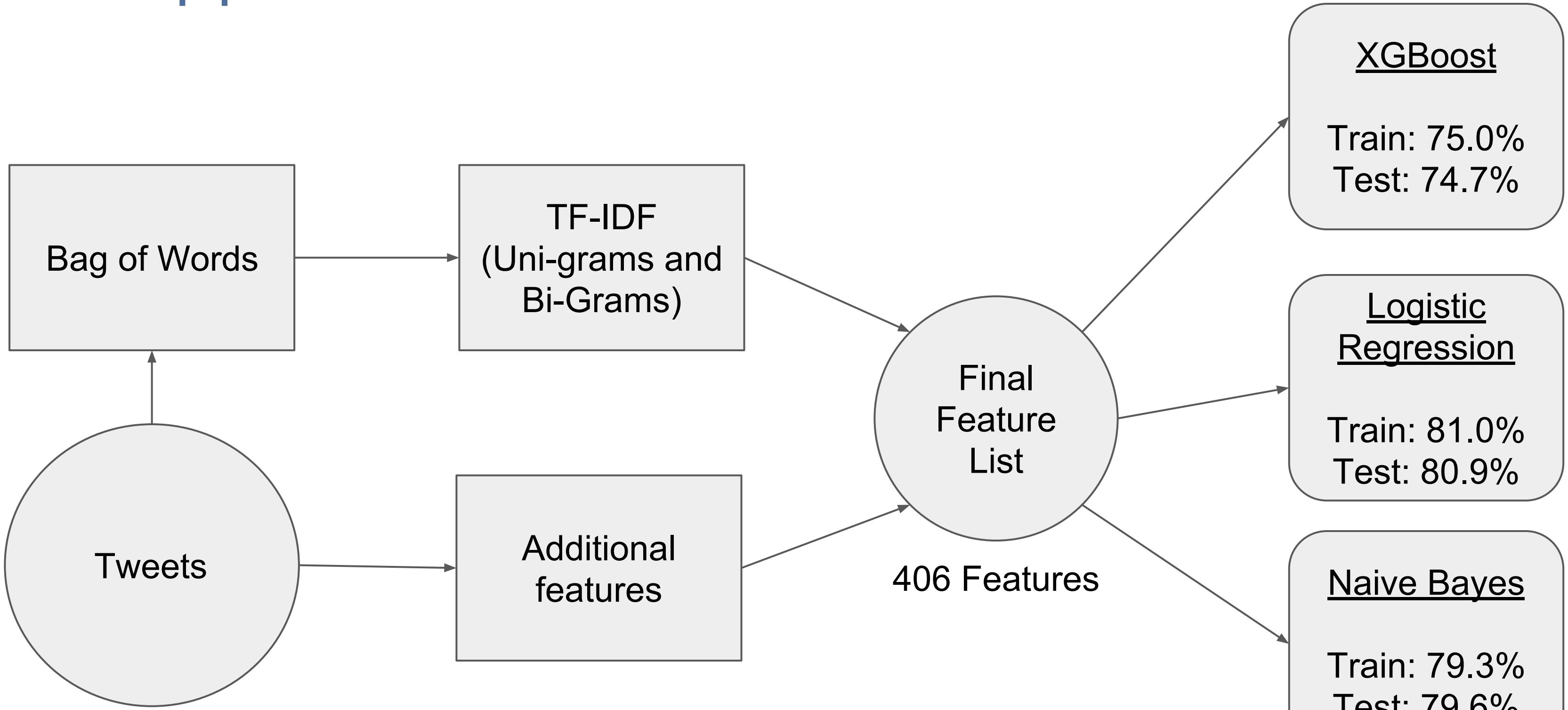


Building Models

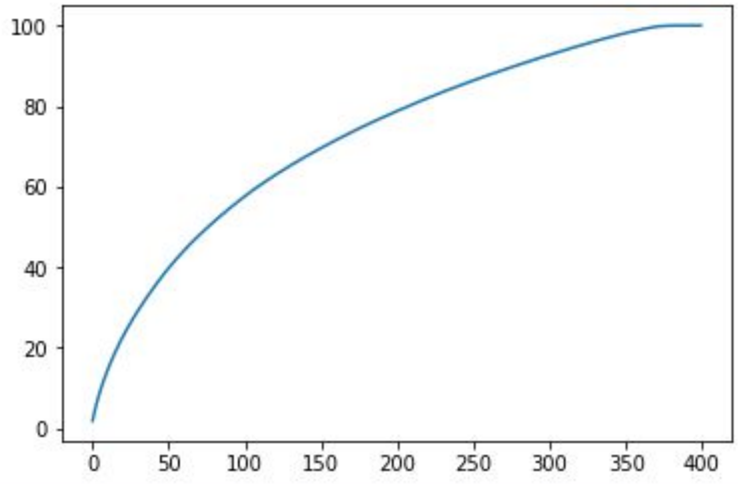
Our Approach



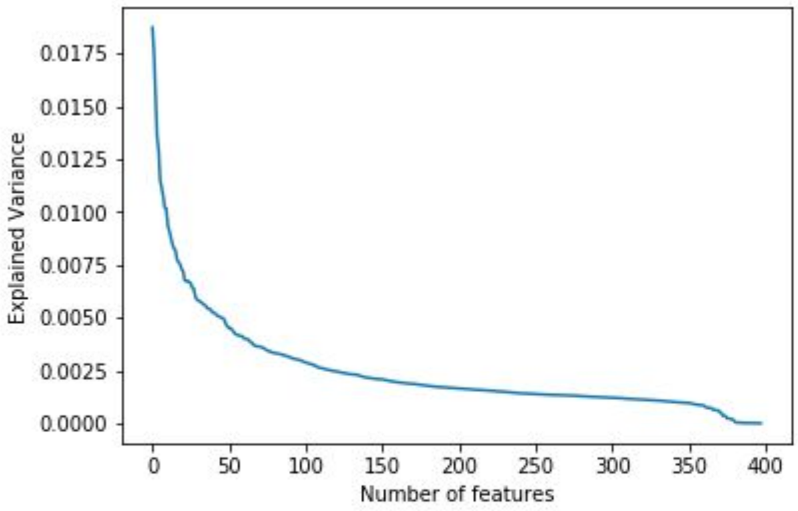
Our Approach



PCA Dimensionality Reduction



SVD Dimensionality Reduction



Future Scope

- Behavioral modeling using social graphs, past tweets and profile bio details
- Context-based tweet classification
- Using Sentiment as a feature
- Deep learning methods eg. LSTM

ThankYou