Report

Team Name: Sarcastic Monkeys

Task: Sarcasm Recognition using Twitter data. There are two parts in this task, one being the Sarcasm detection which is an classification problem and the next one being Sarcasm extraction which is an Information extraction problem.

Dataset: Our Dataset consists of 2980 manually tagged tweets labelled as Sarcastic and Non sarcastic. The percentage of sarcastic tweets in our entire dataset is 23.053% while the percentage of non sarcastic tweets is 76.946%. We divide the dataset such that 60-40 train-test split using stratified sampling such that class distributions are preserved in both the training and the test datasets and no one class dominates the other thus skewing the results.

Technical Approach: Before we dig deeper into the technical approach, below is the general outline followed in the project

- **Data cleaning and preprocessing:** The first step in Data cleaning is the removal of "#sarcasm" tag from all of the tweets. Then we removed other hashtags if any from the tweet and stored them in another separate list.

- **Hashtag expansion:** Since Hashtags are groups of meaningful words , we need to have a way to split them into words that make semantic sense. We tried two ways to expand hashtags, one is our own dynamic algorithm that takes a large english word list of more than 3,50,000 words and then expands the hashtags as a combination of those words. Even though it performed well for short hashtags, it was giving bad results for longer hashtags and hence we used a library which uses of Microsoft N-gram data and hence gave us better results than our model.

- **Spellcheck:** Since, Tweets often contains misspelled words, we developed a two pass algorithm that in first pass removed multiple character occurrences within a word and check after the removal if the word exists in our comprehensive english word list containing over 3,50,000 English words. After pass 1, we then checked the spellings of all the words (except nouns, since nouns can be of different forms) using TextBlob library's spelling correction feature.

- **Tweet splitting:** Our aim was to divide a tweet into multiple parts using a specific set of operators like . , ... ! ? and because etc. We splitted a tweet into multiple parts using all of the above operators and chose which minimizes the

split_criterion score.Split criterion score is the sum(Absolute value(**(1/number of parts the tweet is splitted)** - **number of words in the current part of the split/total number of words in the tweet**)) . Basically, it favors the operators which split the tweet such that all the parts have the same number of words. We chose this because this results in the most informative splits of the tweet.

- **Feature extraction:** Initially, we extracted some high level syntactic features such as presence of Question mark, Exclamation mark, HTTP, Presence of hashtags other than "#sarcasm", presence of capitalized words in the tweet.

- **Tf-Idf:** We tried using Tf-Idf for unigrams, bigrams and trigrams and found out that bigrams gave us the best score.Using just Tf-Idf alone with logistic regression, we got a F1 score of 0.44 which is not very good but, when when we reduced the dimensionality of the Tf-Idf features using SVD with grid search and cross validation and combined the result of SVD with the syntactic features obtained above our F1 score went to 0.539. SVD is trained using only the training data . Below are some most important words as reported by our algorithm. Great,fun, love ,today, cancelled, yay, yay for, school tomorrow, thanks so, love being.

- **POS-Tagging:** For POS tagging we tried different taggers starting with TextBlob tagger (which we think used NLTK POS tagger under the hood), CMU tweet NLP and having our own POS features by combining groups of coherent tags together to one parent tag like all verb tags "VB","VBZ","VBG","VBD","VBN" are mapped to one "V" tag. We used the same approach for Nouns, adverbs and adjective tags. We used Tf-IDF for POS tags after mapping and found out trigrams were most helpful. SVD actually is not needed since, the dimensionality is lower (~400). The mapping approach worked best for our case as there was a considerable difference of 0.6 in F1 score using NLTK tagger and mapped tags.

- **Topic Modelling:** Before we performed topic modelling, we stemmed each word so that the topic distributions and document distributions wouldn't include same words with minute differences. We selected optimal parameters for the LDA algorithm using Cross validation with grid search and found out that we got best results for num_topics 200. Below are the two most important topics and the five most important words in each of those topics. TOPIC-1 : Day, great, start, today, exact and TOPIC-2: Thought, late, would, never, wait

- **Sentiment Analysis:** We use the extra hashtags list we computed in the data cleaning phase as one of our many features that uses sentiment. Here are the sentiment features used in our model. Polarity of each tweet using TextBlob, the subjectivity of each tweet using TextBlob, Afinn sentiment score, Sentiment shift

between different parts of tweet (using the splitted tweets, TextBlob and Afinn), Polarity of hashtag, Subjectivity of hashtag, Sentiment contrast between the tweet and the hashtag (using TextBlob and Afinn)

- **Word2Vec features:** We ran word2vec model on the sarcastic tweets in the train dataset and on a much more huge sarcasm data obtained from Mr. Mathieu Cliche to obtain more reasonable word vectors. We then clustered the word vectors using k-means clustering to obtain 500 clusters. Below are some of the words that are present in the cluster containing the word "love" Wrong,saying,interviewer,offline,counselling, reap , realize, frankly, wake, awake, judgements. As we can see some if not most of the words make sense in the context of sarcasm. We then created a bag of centroids model i.e we represented each tweet as a 500 dimensional vector where each element "i" of the vector is the number of words of that tweet present in the "ith" cluster. We then used a Logistic regression model to predict sarcasm using these features.

- **Stacking:** Stacking involves training a learning algorithm to combine the predictions of other learning algorithms. We reported stacking with Logistic regression in our final results table and as expected stacked model gave us the best F1 score than any of the individual model. However, we found some interesting results when we tried stacking with Gradient Boosting Trees (GBT). When optimized for only precision score using GBT we got a very high test precision score of 0.75, but the recall score was very low at 0.05. We then tried a different approach for stacking, we performed a two level stacking. At level one, we didn't consider Tf-Idf model predictions and then used GBTs on the predictions of all other models and then we combined the outputs of this GBT model with the Tf-Idf model to obtain a precision of 0.59 and a recall of 0.45.

- **Sarcasm Extraction:** Upon analysis of multiple sarcasm sentences, we understood that there is always a sentiment semantic drift between phrases or words which denotes a "Sarcastic" remark. A sarcastic remark contains in all forms a very strong positive sentiment phrase or word and also a negative or neutral sentiment phrase or word. With the following observation we extracted phrases using GenSim with the a criteria of more than 3 occurrences and a threshold of 0.7 for the possibility of the phrase occurring in the entire corpus. While reporting the phrases for each tweet we found the most positive sentiment word in the tweet and reported that too. We also tried phrase extraction using conditional probability of a POS n-gram occurring and the tweet being sarcastic. We used CMU's TweetNLP , a tweet specific POS tagger to tag the tweets in our corpus. We extracted bigrams, trigrams, 4-grams, 5-grams and calculated the probabilities.We used a threshold of 1 for 4-grams and 5-grams while 0.8 for bigrams and trigrams.

Team Member Contributions:

Nishant Agarwal- Data Cleaning, Feature extraction ,pos tagging, sarcasm extraction using n-gram probability and phrase extraction using gensim & Topic modelling.
Murali krishna teja Kilari- Preprocessing, Hashtag expansion, Tweet spellcheck, Tweet splitting, TF-Idf, Word2Vec, Sentiment Analysis & Stacking.

EXTERNAL RESOURCES:

1. TextBlob – POS tagging, Spell Check, Sentiment Analysis https://textblob.readthedocs.org/en/dev/index.html
2. TweetNLP – Unique POS tagger for tweets http://www.cs.cmu.edu/~ark/TweetNLP/
3. GenSim – Phrase Extraction, Word2Vec, Topic Modelling https://pypi.python.org/pypi/gensim
4. Afinn – Sentiment strength score https://github.com/fnielsen/afinn
5. Scikit learn – TF-IDF, ML algorithms such as clustering, SVD, GridSearch and CV. http://scikit-learn.org/stable/
6. WikiWords – List of most frequent English words (approx 300,000 words). https://github.com/krishnateja614/SarcasmDetection
7. Tweet hashtag expansion: We used the following library to expand tweets into words . https://github.com/piyushbansal/hashtag-segmentation
8. Additional sarcasm tweets for Word2Vec: We obtained extra tweets needed to train our Word2Vec model from the following repository after obtaining permission from the owner https://github.com/MathieuCliche/Sarcasm_detector

EVALUATION METRICS AND RESULTS : Our evaluation metric is the F1 score since it is better for unbalanced classification problems compared to accuracy. We also reported the precision and recall scores below for both train and test sets and for each of our logistic regression models trained on the respective feature sets

Model	Train F1 score	Train Precision	Train Recall	Test F1 score	Test Precision	Test Recall
Baseline Model Tf-Idf (with SVD) + 5 syntactic features	0.6312	0.5240	0.796	0.539	0.470	0.6327
POS bigrams (No SVD)	0.424	0.323	0.614	0.3969	0.305	0.567

Topic Modelling (200 Topics)	0.4353	0.3630	0.5436	0.416	0.3764	0.465
Sentiment features + Word2Vec	0.5569	0.4546	0.7188	0.4809	0.4029	0.5963
Stacking	0.66535	0.5596	0.820	0.569	0.4933	0.6727

UNSUCCESSFUL SARCASM DETECTION EXAMPLES: Below are some tweets where we mistakenly predicted that it was sarcastic when the true label was non sarcastic.

1. Thanks man! @jason_van_wyk @Joni_JPL @cat2martin @arnyblackhole @BlackHoleRec @7Skies @DaveTritonal @Arksun @DerekTheBandit @SusanaVocalist'
2. Glad that person was in my dream #sarcasm #goingtobeaweirdday'
3. 'done what i had to do outside of the house....now to do stuff inside the house!!!! #ilovehousework #sarcasm'

SUCCESSFUL SARCASM DETECTION EXAMPLES: Below are some tweets where we correctly predicted sarcasm.

1. '@dunlop27 I just love TSwift so much!! #sarcasm #bleedingears #lookslikearat'
2. "Can't wait until Friday when we get to wear our band uniforms in 90 degree weather #sarcasm ??"
3. 'I love doing economics first thing in the morning #sarcasm'

SARCASM EXTRACTION EXAMPLES:

Glad person was in my dream	glad,my_dream
I love doing economics first thing in the morning	i_love,first_thing
I am so not a morning person have a good day everybody	Have_a, good_day
Lol, can’t wait to read the complaints of how you need more sleep in case you can’t tell yes that is	lol,can’t_wait,to_read,need_more,you_can’t

going from one babysitting job to another babysitting job, then another babysitting job today. This will be fun.	babysitting_job,another_babysitting,this_will, be_fun
Love getting home from work knowing that in less than 8 hours you're getting up to go back there again	love_getting,knowing_that,you're_getting,to_ go

We noticed that as the context window of n-gram grew there were higher number of unique POS tag sequences which occurred for sarcastic tweets. This did not work out very well as it was not a definite representation of sarcasm and gave multiple or no phrase at all.
Good Extraction:
Tweet - @Giants_101 Well that's just fantastic. #thisactuallysucks
Phrase Extracted - that's just fantastic . #thisactuallysucks
Some non productive extractions were -
Tweet - Gee wiz what a great day so far. You can cut the tension with a knife and they think im faking #great.

Phrases extracted - “Gee wiz what a”, “wiz what a great”,“day so far . You”, “Gee wiz what a great”, “and they think im faking” , “wiz what a great day”, “think im faking #great “., “knife and they think im”

LESSONS LEARNT: As we can see from the results table above, even though there is a notion that sarcasm detection has more to do with semantics of the sentence rather than the syntax, Tf-Idf model which captures only syntactic features seems to able to do a reasonable job of predicting sarcasm. We are particularly proud of how helpful sentiment shifts and word2vec features turned out to be and if we had more time we would have experimented with RNNs ,LSTMs as it seems that they are better suited to this task than the traditional ML algorithms. In hindsight, we think sarcasm detection is a very complex problem, requiring the understanding of more important feature like the context of the tweet and more details in order to predict it with high F score.

Looking at the above examples of Sarcasm phrases extracted, we can see that it is very difficult to determine which of the extracted phrases would be correct or in some cases no phrase is extracted. We believe that more work and research is required in understanding what constitutes a “Sarcasm”. In future we look at trying new approaches and also read a couple of already published papers to get a base footing to start building up with new novel ideas.