School of Electronic Engineering and Computer Science

Final Report

Programme of study:
Computer Science

<u>Project Title:</u> Laughter Classification From Tweets

Supervisor:
Professor Matthew Purver

Student Name:
Mursalin Habib

Final Year
Undergraduate Project 2022/23

Date: 9th May 2023

Queen Mary
University of London

Acknowledgements

I would like to thank my supervisor Professor Matthew Purver for his invaluable feedback and guidance, which helped me immensely in completing this project.

Abstract

Laughter classification is an important task in NLP. Identifying laughter in text is as important as identifying laughter in audio and video. Laugher classification system can be used for sentiment analysis, humour detection, social interaction analysis and other psychological research and so on. Laughter is often used as an indicator of emotion. laughter classification model can be used for analysing social media posts or customer feedback of particular products by identifying whether the laughter is associated with positive or negative sentiment. Laughter is an important social cue and can be used in identifying humour in conversations and dynamics of group conversations. Widespread use of social media platforms makes classification laughter an even more important task.

In this project I categorised laughter from tweets into three classes: Joyous laughter, sarcasm or ironic laughter and schadenfreude laughter. I experimented with different machine learning algorithms and used TF-IDF Vectorizer for feature engineering, I experimented with different pre-processing steps and n-grams to come up with the best version of the model. The final model achieved an accuracy of 73%.

Contents

Chapter 1: **Introduction**

1.1 Background

The world we live in is changing rapidly and unpredictably. With that is changing the way we communicate with each other. Computers now-a-days are essentially part of our life and most of our conversations are done over computer devices through phone calls, texts, emails or even engaging with other persons social media content. Today emails and text conversations are considered to be more convenient methods of communication than making phone calls. Understanding emotions behind text is key to having an effective communication online as words may mean different things in different context. In a face-to-face conversation we look at body language, facial expression and tone of voice to understand emotions behind spoken words, to express those emotions in writing is difficult, so we use punctuation marks, emojis, write detailed description of the topic, use all capital letters to emphasis and to give more context.

Laughter is one of the key ways humans communicate with one another. We use laughter to express humour, enjoyable experiences, to show our comfort and agreeableness, as well as discomfort and nervousness. In the ICSI meting corpus Laskowski and Burger reported 9.5% of the total verbalizing time being laughter (Laskowski and Burger, 2007; Vogel et al., 2013). People laugh on their own, with someone or in a group. Mutual laughter is a sign of rapport and consensus. And people often laugh alone in a conversation and not at things considered funny. Analyses show that, in dialogue, we often laugh alone and not always at things considered particularly funny. Mutual laughter is a sign of rapport and consensus. (Adelswärd, V., 1989). Laughter is also used to degrade and demean someone. Different types of laughter express different sentiment. Detecting and categorising laughter in a text conversation is an important research area as it can help us define emotions behind written words.

1.2 Problem Statement

Laughter is generally considered caused by or related to something humorous, funny or amusing. But sneer, giggle or snigger are also type of laughter which are used to express attitudes and moods instead. Four different laughter types has been described in one paper as voiced, chuckle, breathy and nasal (Campbell et al, 2005). Laughter is generally described as a means keying into a playful frame (Goffman, 1974). But laughter is also used to mock or bully someone. Even in the case of bullying the laughter is still considered playful by the person laughing, but at least one of the participants does not wish to engage or believe this to be playful. Physical laughter is used to mediate conversation such as marking the beginning and end of utterances, turns and topics. This makes laughter a very complex way of expressing different sentiments and studying laughter a very important research area. Laughing sounds are universally recognised and easier to detect than typed laughter, which makes detecting laughter in text a more challenging task (McKay, 2020).

In a face-to-face conversation we seem to intuitively understand meaning behind laughter, mostly because of facial expression, tone of voice and body language. But none of these features are present in text conversation. To express laughter in text communication we use Typed Laughter Derived Expressions (TLDEs) such as haha, hehe, lol, lmao, lmfao, 😂 (Unicode: 1f602), 🤣 (1f923), and 😆 (1f606). TLDEs are also complex as they too can be used to express different sentiments. Like physical laughter TLDEs are also used to manage text conversation e.g., topic shifting or turn taking in a conversation (Petitjean and Morel, 2017) as nonverbal aspect of conversation is not present. Studies shows how TLDEs are used to portray a conversation as playful, flirty or non-serious (Petitjean and Morel, 2017; Dresner and Herring, 2010). So, in order to mitigate the lack of nonverbal aspect of conversation TLDEs are often used even if the content of the conversation is not humorous or amusing. Therefore, categorising laughter has always been a research interest.

1.3 Aim

The aim of this project is to classify different types of laughter in text communication by analysing tweets which can help understand how people express laughter on twitter. The goal is to build a machine learning model that will be able to identify laughter from a tweet where the person tweeting intended to laugh and categorise laughter accordingly.

1.4 Objectives

- Review articles and books to understand different categories of laughter and role of laughter in a conversation.
- Perform background research to identify emojis and phrases that are used to express laughter.
- Learn to develop, train and test machine learning model.
- Build a model that can correctly identify laughter.
- Find datasets for training and testing the model.
- Train the model to obtain high accuracy.
- Analyse the result to draw conclusion.

1.5 Research Questions

- Can a classifier be built to accurately predict different laughter types?
- Can laughter from twitter communication be categorised, if so how should it be categorized?
- Can the classifier be used to classify other text conversation outside twitter?
- Do different experiments show any significant improvement or decrease the model's performance?

1.6 Report Structure

- **Introduction**: This chapter gives a brief introduction to the research, discusses the problems, aims and objectives of the research. It also defines the Questions the paper wishes to get answered.

- **Literature review:** This chapter gives an overview of all other related published research paper that I have studied.

- **Dataset and methodology:** This chapter explains how I collected data and created dataset for the model and the process of building the model.

- **Experiments:** In this chapter I explained how I experimented with different algorithms and steps to evaluate and improve performance of the model.

- **Evaluation:** In this chapter I evaluate and analyse results found in the experiments.

- **Conclusion:** This chapter discusses the end result of the research and possible further works

- **References:** The reference section contains a list of references to other researches and works, to validate any mention to those works I have made in this paper. I have used Harvard referencing because it's one of the standard referencing systems.

Chapter 2: **Literature Review**

This section describes all the research papers I have reviewed on laughter and on machine learning models.

2.1 Physical Laughter Categories

Many studies have been done to categorise laughter based on laughter sound, facial expression etc. Laughter has been classified based on sound as 4 types: ‘mirthful’, ‘polite’, ‘derisive’, and ‘others’ in a study on classification of laughter in social context (Tanaka and Campbell, 2013). Earlier study by Campbell (Campbell et al, 2005) described hearty and amused laughter type which were both included as mirthful due to being hard to distinguish the sound. Mirthful laughter is associated with joy and amusement, while polite laughter is used to show politeness or agreement. Derisive laughter is a form of mockery or contempt, and others type does not fit neatly into any of these categories.

To recognise facial expression, laughter has also been classified into 4 types such as ‘joyful’, ‘intense’, ‘schadenfreude’ laughter, ‘grinning’ (Ruch, Hofmann and Platt, 2013). Joyful and intense laughter has been associated with positive emotions. Duchenne Display is said to be a recognising factor for joyful and intense laughter. Duchenne Display is the involuntary contraction of the muscles around the eyes that signals genuine enjoyment. Clear Duchenne Display was not seen in both schadenfreude and grinning laughter. Schadenfreude laughter is laughing due to a feeling of pleasure attained from the misfortunes of others and grinning laughter is a forced or insincere laughter or smile.

Another similar study which looks at laughter through the lens of underlying emotion rather than facial expression, categorised laughter as ‘joy’, ‘tickle’, ‘schadenfreude’ and ‘taunt’ (Szameitat et al, 2009). Joyful laughter was associated with positive emotions, tickle laughter with physical sensations and schadenfreude laughter with a feeling of pleasure derived from the misfortunes of others. While taunt laughter is to provoke or insult others.

2.2 Audio Laughter Detection

Kennedy and Ellis built a system that detects laughter events in a meeting environment. They defined laughter event as a point where at least more than one participant laughs simultaneously (Kennedy and Ellis, 2004). They implemented their system using Support Vector Machine classifier which was trained on three different features and then they analysed performance to compare the results. The system achieved 87% correct acceptance rate.

Support Vector Machine is a machine learning algorithm used for classification and regression of linearly separable data. SVM uses supervised learning which means it needs labelled data to be trained. The algorithm draws a line known as hyperplanes to separate different data groups. SVM can eventually perform unsupervised learning after processing numerous training examples. The algorithm tries to find the best separation by maximizing distance between hyperplane and nearest point of each group of data. (Cortes and Vapnik, 1995).

2.3 Hashtag to Detect Emotion

People often use hashtags to notify others of the emotions associated with their tweets, which may or may not have been obvious from the tweet itself. SVM was used in a research project to capture emotions using hashtags (Mohammed and Kiritchenko, 2015). It creates an emotion corpus and lexicon from tweets using hashtags and named them Hashtag Emotion Corpus and Hashtag Emotion Lexicon respectively. After refining 21,000 tweets were used to create the Hashtag Emotion Corpus. Twitter Search API was used to collect data for the corpus.

Pointwise Mutual Information (PMI) was used to create the Hashtag Emotion Lexicon. PMI measures how much more likely the two words are to appear together than if they were independent. A high PMI score indicates a strong association between the two words, while a low PMI score indicates a weak or no association. PMI is used to find how frequently a word is associated with an emotion.

Strength of Association (SoA) between an n-gram w and an emotion e is:

$$\text{SoA}(w, e) = \text{PMI}(w, e) - \text{PMI}(w, \neg e)$$

where, $\text{PMI}(w, e) = \log_2 \frac{freq(w,e)*N}{freq(w)*freq(e)}$

If n-gram associated with an emotion occurs less than 5 times, it is ignored for low frequency e.g., word is unlikely associated with then emotion and if n-gram which has a higher chance of being associated with an emotion occurs in same sentence SoA is positive i.e., greater than zero.

2.4 Emojis to Detect Sentiment

Using emojis to detect sentiment is one of the very complex task. Unlike hashtags emojis are not exact labelling of emotions. Multiple emojis can express same or similar sentiment or a single emoji can express multiple sentiment too. Using positive emoji with a seemingly negative text can give the text a positive meaning. Also use of emoji can give meaning to ambiguous sentences. (Kunneman et al, 2014)

Wiegand and Ruppenhofer used emojis to create a lexicon of abusive language detection (Wiegand and Ruppenhofer, 2021). They used PMI, project-based induction and recall based expansion by label propagation and compared the lexicons. Project-based induction is represented by embeddings, so it can have larger vocabulary than PMI. Recall base expansion takes high rank induction methods and applies label propagation. Twitter streaming API was used to collect 100,000 tweets containing abusive emojis for the experiment.

The DeepMoji is a model trained on 1246 million tweets with known emotional content, such as tweets labelled with emojis. By analysing the text and identifying patterns, the model is able to learn which words and phrases are most closely associated with specific emotions. In pre-processing stage tweets containing URLs were removed and tokenization was done for generalization. Also special tokens were used for all usernames. The deep learning model uses four layer. Embedding layer maps data into vector spaces for the model to understand relationships between words in a meaningful way. Bidirectional Long Short-Term Memory (BiLSTM) layer a recurrent neural network that is commonly used in sequence modelling tasks. It can process text input in both forward and backward directions, which allows the model to capture dependencies in the data over longer distances. Attention layer allows the model to focus on the most important parts of the input data when making predictions. Softmax layer converts a vector of scores into a probability distribution over the possible classes which allows the model to make prediction about class in which input text belongs to. The model uses transfer learning approach called chain-thaw where weights of layers are frozen first. Then they unfreeze and finetune a single layer at a time. (Felbo et al, 2017)

Emojis were used to detect irony in Persian language in an experiment where a neural network model was first pretrained on a large unlabelled dataset containing Persian tweets and then was finetuned on a manually labelled dataset (Golazizian et al, 2020). After prepossessing a total of 4,463,430 tweets were used for the experiment. The experiment uses a single task learning model and a multitask learning model. Multi-task learning models can learn to perform several related tasks with a single model, whereas single task learning models perform only one specific task (Collobert and Weston, 2008). Similar to DeepMoji model four layer layers has been used. The model had an 83.2% accuracy rate. Experiments were done through removing different steps and layers which showed significant decrease in accuracy which proves all the components to be essential for the model to attain best performance.

2.5 Summary

Many attempts to categorise physical laughter has been done based on different categories. General assumption of laughter being only an expression of humorous events is hardly true. Beside humour laughter is used to express joy, to make fun of others, to be polite and agreeable, to scorn or look down on someone, to insult or provoke someone and many more. Among works done on laughter are laughter detection from audio and video using Support Vector Machine. But I haven't been able to find any categorisation of text based laughter, which motivated me to undertake this project.

Research that used hashtags and emojis to detect sentiment has used SMV or deep learning for the experiments. In all cases twitter data has been used for creation of dataset. Like all social media sites, twitter has a large active user base, but in twitter communication is mainly text base, which makes it a reliable data source for NLP tasks. Given users often use hashtags to express their sentiment on twitter it makes collecting relatable tweets quite easy.

Chapter 3: **Dataset and Methodology**

Following I explained in detail how I created the dataset and the classification model. I described what hashtags I used and how I filtered tweets to collect English tweets related to laughter. Then I explained each step of creating, training and testing of the machine learning model.

3.1 Dataset

I have considered using an existing dataset for the model but was unable to find a dataset consisting of laughter data that is labelled accordingly. So I have decided to build a dataset myself. I collected data from twitter to create the dataset because it's easy to find relatable tweets to use as dataset from twitter with hashtags and emojis. Twitter users often use hashtags in their tweets to categorize their content and make it easier for other users that are interested in the topic to discover their tweets. Emojis are often used in twitter to express emotions, convey humour or indicate the subject of a tweet. So Using emojis as a search term can help narrow down the tweets that are relevant to a particular topic. Two most common ways of collecting tweets from twitter are through stream API and search API. These APIs allows one to search tweets based on keywords, hashtags, and emojis. This makes it easy to find tweets that are related to a specific topic to use them in dataset. Both search API and stream API returns JSON object as a response containing all information about the tweets or simply the text of the tweets could be collected. I decided to use the search API because I found it to be simple to implement. The stream API requires to be run on a server virtually over a long period of time whereas the search API can instantly collect a large number of tweets. I collected tweets separately based on category on many CSV files and labelled them accordingly and then I have merged the CSV files and shuffled the data to create the final dataset. I used Python's 'pandas' library to create and merge the CSV files. Then I used the 'random' library to shuffle the rows in the csv file.

I decided to categorise the tweets into 3 different laughter types as 'joyous' laughter, 'schadenfreude' laughter and 'sarcasm' laughter. Joy and schadenfreude laughter has been used in earlier research on physical laughter and they are easier to distinguish compared to other types (Ruch, Hofmann and Platt, 2013; Szameitat et al, 2009). Irony and sarcasm are generally associated with laughter (Tepperman, 2006) and has been a research interest in NLP field. Therefore, I decided to use these three categories and labelled the dataset as such.

While collecting tweets I filtered out all retweets to avoid having same tweet appear multiple times in the dataset. I also filtered for tweets that are only in English language, but collected tweets from all geographical locations. I used hashtags and emojis related to laughter and hashtags related to the each of the categories in different scripts and labelled the tweets as I collected them. Initially with only the three labels (joy, schadenfreude and sarcasm) as hashtags I managed to collect a handful of tweets. So I looked for other hashtags that are related to the labels to collect more tweets. Emojis and hashtags I used to collect laughter tweets based on labels are listed on table 1. I have collected in total of 4,009 tweets containing laughter of which 1,500 tweets were 'joyous' laughter, 1,509 were 'schadenfreude' laughter and 1,000 were 'sarcasm laughter.

Laughter emojis	• U+1F480: "Skull" emoji 💀 • U+1F638: "Cat Face" emoji 😸 • U+1F639: "Cat Face with Tears of Joy" emoji 😹 • U+1F923: "Rolling on the Floor Laughing" emoji 🤣 • U+1F602: "Face with Tears of Joy" emoji 😂 • U+1F606: "Laughing Squinting Face" emoji 😆
Laughter hashtags	#Hilarious, #ROFL, #lol, #lmao, #lmfao, #looool, #hahaha
Joy hashtags	#happyness, #happiness, #happy, #joy, #laughter, #fun, #Celebrate, #Celetration, #cute, #Joyful, #Blessed, #Greatful, #GoodVibes, #enjoy, #Enjoying, #funTime
Schadenfreude hashtags	#schadenfreude, #karma, #MiseryLovesCompany, #Roasting, #HumbleBrag, #KarmaIsABitch, #failarmy, #Fights, #brawls, #SchoolFights, #ByeFelicia, #EvilLaugh, #InstantKarma, #HumbleBrag, #lessonLearned, #Revenge
Sarcasm hashtags	#Sarcasm, #Irony, #SarcasmAlert, #SarcasmDetected, #SarcasmOverload, #NotReally, #JustKidding, #NotSerious, #TongueInCheek, #SarcasmByAssociation, #ironic, #BeingSarcastic, #SorryNotSorry, #ISaidWhatISaid

Table: 1: Hashtags and emojis used to create dataset.

3.2 Methodology

3.2.1 Cleaning and pre-processing of data

After building the dataset I have done some cleaning to standardise the data and make data reliable for pre-processing. Using Python’s ‘re’ library I converted all text to lower case, removed punctuation marks, hashtags, emojis and whitespaces. If hashtag appears in the middle of a sentence, I kept the word and only removing hashtag from the front. I converted all usernames into a single token ‘username’. And I also did the same for any links in the dataset by converting them to a single token ‘http’. To conduct different experiments, I sometimes removed usernames and http links or kept question marks and exclamation marks. Then I lemmatized the words using NLTK’s WordNetLemmatizer. Lemmatization is the process of reducing a word to its base form. For example, ‘running’, ‘runs’ and ‘ran’ becomes run. Another way of reducing a word to its base from is stemming, but with stemming a lot of the times meaning of the root word gets lost e.g. ‘caring’ gets reduced to ‘car’ rather ‘care’. So, I decided to use lemmatization rather than stemming.

For feature extraction I used TF-IDF Vectorizer. I experimented with different n-grams, starting with unigram up to trigram and combination of all. In NLP feature extraction is the process of transforming raw text data into a format that can be used by machine learning algorithms by extracting relevant features from the text data. The TF-IDF

Vectorizer transforms text data into numerical values. It is calculated by multiplying term-frequency with inverse document frequency. Term-frequency is number of times a word appears in the document divide by the total number of words in that document and inverse document frequency is calculated by taking the logarithm value of total number of documents in the corpus divided by number of documents containing the word. This ensures words that are common in the document but rare in the corpus will have a high TF-IDF score and words that are common in both the document and the corpus will have a low TF-IDF score. (Kumar and Subba)

TF-IDF = (TF * IDF)

Where TF = Number of times the term appears in the document / Total number of terms in the document

And IDF = $\log_2 \frac{total\ number\ of\ documents\ in\ the\ corpus}{number\ of\ documents\ that\ contain\ the\ term}$

3.2.2 Implementation

Naïve Bayes, Logistic Regression and Support Vector Machine are well known algorithms for text classification problems. These algorithms are simple but quite fast and efficient. Naïve Bayes is a probabilistic algorithm that was widely used for NLP tasks like spam detection and sentiment analysis. Logistic Regression estimates the probability of a binary outcome based on a set of features. It is a simple and effective algorithm that can achieve high accuracy with relatively little training data making it an ideal choice for the task. SVM as mentioned in chapter 2 tries to find the best separation by maximizing distance between hyperplane and nearest point of each group of data. It's a widely used algorithm for NLP tasks and many others. I ran experiments using these three algorithms and made different versions of the model. I then compared different versions and picked the best one from them. I have found Logistic Regression and SVM had performed much better than Naïve Bayes classifier and SVM performed relatively better than Logistic Regression. So I decided to continue further experiments with SVM.

3.2.3 Testing

I used 5-fold cross validation to test the model's performance. Cross-validation is a method for training and testing machine learning models which involves splitting the data into multiple equal sized subsets or folds. One subset or fold is used for testing and the rest is used for training. With 5-fold cross validation 4 folds are used for training the model and one for testing its performance. This process is then repeated multiple times with different folds as testing set, and the results are averaged to estimate the model's accuracy. Because the model is tested on the whole of dataset, it gives more accurate results and reduces the risk of overfitting.

5-fold cross-validation is a widely accepted and very common benchmark for evaluating and comparing machine learning models in NLP. It provides a good balance between computation time and accuracy and reduces the risk of overfitting (Kohavi, 1995).

3.2.4 Comparing results

At the end I compared results from all experiments and discussed which was the best and most accurate model and possible reasons for it. I used confusion matrix, precision and recall valuers, F-1 score and accuracy score to compare and evaluate performance of different versions of the model.

A confusion matrix is a table consisting actual and predicted classes, that is used to evaluate performance of a classification model. Each row of the table here represents actual classes and each column represents predicted classes. Table 2 displays an example of a two-class confusion matrix, classes being ‘Positive’ and ‘Negative’. Confusion matrix is used to evaluate how many accurate and inaccurate predictions the model is making. Accurate predictions are when the model predicts items in positive class to be positive (true positive or TP) and items in negative class to be negative (true negative or TN) and inaccurate predictions are when the model predicts items in positive class to be negative (false negative or FN) and items in negative class to be positive (false positive or FP).

	Predicted			
	Classes	Positive (1)	Negative (0)	Total
Actual	Positive (1)	TP = 20	FN = 5	25
	Negative (0)	FP = 10	TN = 15	25
	Total	30	20	50

Table 2: Example of a two-class confusion matrix (Grandini et al, 2020)

Precision is the measurement of the accuracy of true predictions made by the model. Precision for positive class is calculated using following formula:

Precision = TP / (TP + FP)

Recall is the ability of the classifier to correctly classify items in positive class as positive. Recall for positive class is calculated using following formula:

Recall = TP / (TP + FN)

Low precision score means the model makes a lot of wrong predictions and low recall score means the model makes less predictions but most of which being accurate. So for model to be accurate both precision and recall score needs to be high.

F-1 score is used to evaluate model’s performance in both precision and recall. A higher F-1 score indicates the model is performing well in both precision and recall. Formula for F-1 score is the following:

F-1 score = 2 * (precision * recall) / (precision + recall)

Accuracy is the proportion of correct prediction made by the model. Accuracy is calculated using following formula:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

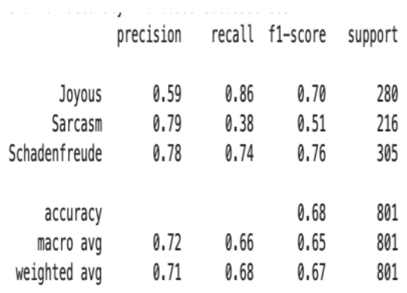
Chapter 4: **Experiments**

Below I conducted 5 experiments and compared different versions of the model in order to select the one that performs best. The first experiment was done to find the best performing classification algorithm for the model, the second was to find the best n-grams for feature selection, the third experiment was to see difference in performance with and without removing usernames and links, the fourth was to see if question marks and exclamation marks are important features and finally the fifth experiment was to see how stop words removal impacts the model's performance.

4.1 Experiment-1: Classifier selection

In the first experiment, I implemented 3 different classification algorithms: MultinomialNB, Logistic Regression and Support Vector Classifier. I started with Naïve Bayes classifier due to algorithm being simple and well-known for NLP classification tasks and then I experimented with logistic regression and SVC classifier to see if I can improve performance. I then analysed the results and selected the best performing algorithm for the further experiments. For this experiment I kept the n-gram range constant (ngram_range = (1,3)) for all 3 classifiers. Results found are in the figures below:

MultinomialNB



```
              precision    recall  f1-score   support

      Joyous       0.59      0.86      0.70       280
     Sarcasm       0.79      0.38      0.51       216
Schadenfreude       0.78      0.74      0.76       305

    accuracy                           0.68       801
   macro avg       0.72      0.66      0.65       801
weighted avg       0.71      0.68      0.67       801
```

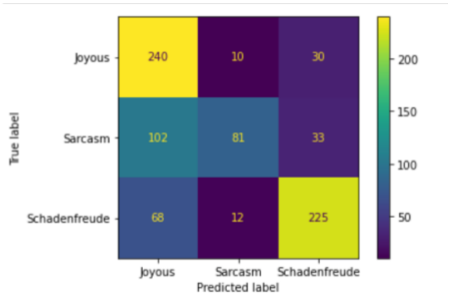
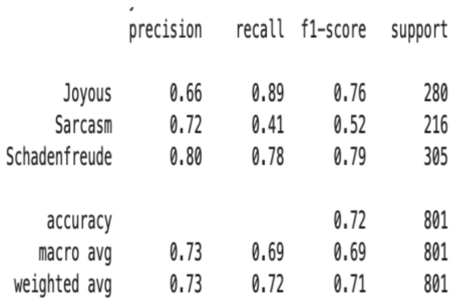


Figure 1: results from Naïve Bayes algorithm

The accuracy score for MultinomialNB was 68% and average precision and recall values (macro average) are 72% and 66%. Sarcasm had a very low recall value which means less sarcasm laughter tweets were being predicted. The confusion matrix on the right shows most sarcasm laughter tweets (102 tweets) have been predicted as joyous laughter and a large number of schadenfreude laughter (68 tweets) has been classified as joyous laughter.

Logistic Regression



```
              precision    recall  f1-score   support

      Joyous       0.66      0.89      0.76       280
     Sarcasm       0.72      0.41      0.52       216
Schadenfreude       0.80      0.78      0.79       305

    accuracy                           0.72       801
   macro avg       0.73      0.69      0.69       801
weighted avg       0.73      0.72      0.71       801
```

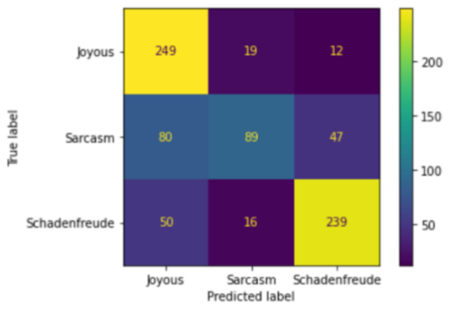
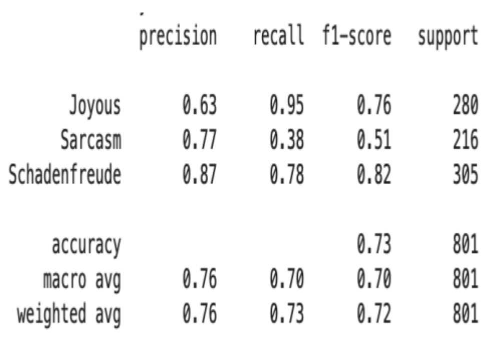


Figure 2: results from Logistic Regression

Logistic regression has an accuracy score of 72% which is significantly higher than multinomialNB. Average precision, recall and f1-score values are also higher. Clearly showing Logistic Regression outperforming MultinomialNB. Although still quite low, the recall value for sarcasm has improved.

SVM



```
               precision    recall  f1-score   support

      Joyous        0.63      0.95      0.76       280
     Sarcasm        0.77      0.38      0.51       216
Schadenfreude       0.87      0.78      0.82       305

    accuracy                            0.73       801
   macro avg        0.76      0.70      0.70       801
weighted avg        0.76      0.73      0.72       801
```

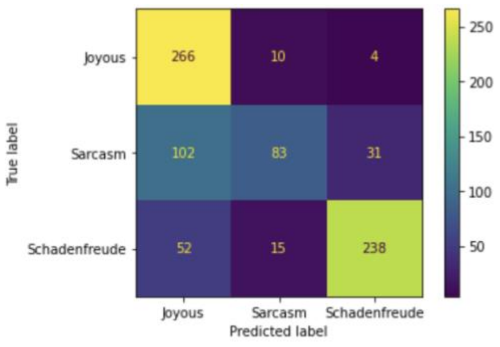


Figure 3: results from SVM

The SVC has an accuracy of 73% with highest average precision, recall and f1-score among all three. Still like MultinomialNB, most of Sarcasm laughter tweets were being classified as Joyous laughter and a high number of schadenfreude tweets were being classified as joyous laughter.

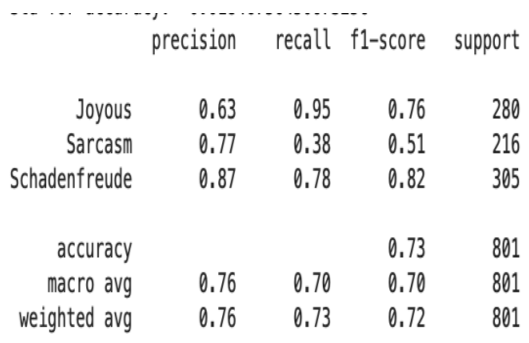
Summary

Overall precision has improved with logistic regression and SVM. Recall scores were too better in logistic regression and SVM. Although average recall is higher in SVM, logistic regression scored better in recall value for the sarcasm class compared to the other two classifier, which means it was making more sarcasm predictions. But because precision score is higher in SVM this means many sarcasm laughter prediction made by Logistic Regression algorithm were wrong. From looking at the confusion matrix in figure 2 and 3, prediction for sarcasm laughter has increased as well as some joyous laughter tweets were predicted to be sarcasm. The sarcasm laughter has been most misclassified tweets with all three algorithms. Increasing max_features threshold improved performance in all algorithms. Given SVM scored higher on average in precision, recall and F1-score it is clearly the best performing algorithm. So I used SVM for the following experiments.

4.2 Experiment-2: Different n-grams

Here I have experimented with different n-grams to get best possible feature extraction with TF-IDF Vectorizer. I experimented with unigram, bigram, trigram and combination of the three and compared average precision, recall and f1-score.

ngram_range = (1,3)



```
              precision    recall  f1-score   support

      Joyous       0.63      0.95      0.76       280
     Sarcasm       0.77      0.38      0.51       216
Schadenfreude       0.87      0.78      0.82       305

    accuracy                           0.73       801
   macro avg       0.76      0.70      0.70       801
weighted avg       0.76      0.73      0.72       801
```

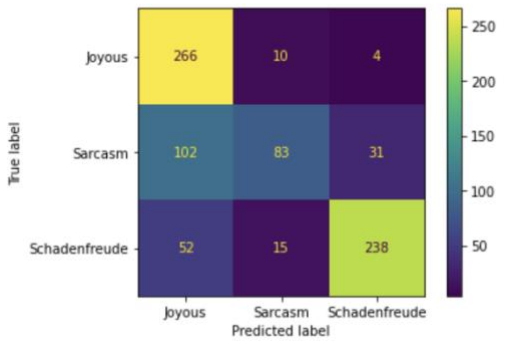
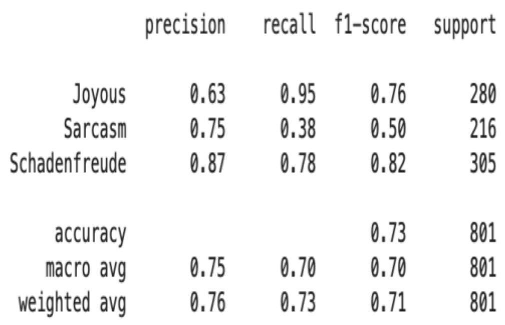


Figure 4: ngram_range = (1,3)

N-gram range in experiment 1 was 1-3 so the results here is same as results of SVM classifier in experiment 1. Accuracy rate was 73% and macro average of precision, recall and f1-score being 76%, 70% and 70%.

ngram_range = (1,2)



```
              precision    recall  f1-score   support

      Joyous       0.63      0.95      0.76       280
     Sarcasm       0.75      0.38      0.50       216
Schadenfreude       0.87      0.78      0.82       305

    accuracy                           0.73       801
   macro avg       0.75      0.70      0.70       801
weighted avg       0.76      0.73      0.71       801
```

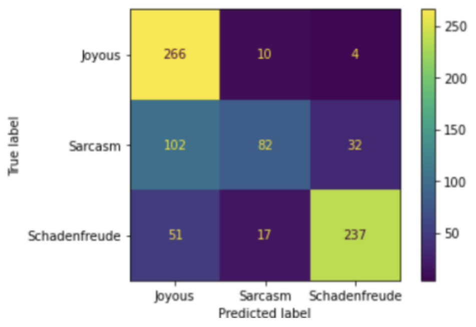
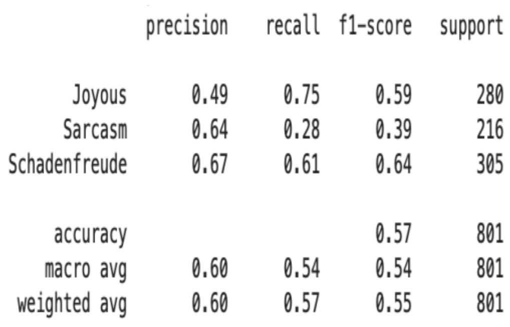


Figure 5: ngram_range = (1,2)

Experiment with unigram and bigram gave an almost identical result with averages being exactly same but slightly bad at precision for sarcasm therefore reducing macro average for precision by 1%. Most of Sarcasm laughter tweets were classified as Joyous laughter and a high number of schadenfreude tweets were classified as joyous laughter.

ngram_range = (2,3)



```
              precision    recall  f1-score   support

      Joyous       0.49      0.75      0.59       280
     Sarcasm       0.64      0.28      0.39       216
Schadenfreude       0.67      0.61      0.64       305

    accuracy                           0.57       801
   macro avg       0.60      0.54      0.54       801
weighted avg       0.60      0.57      0.55       801
```

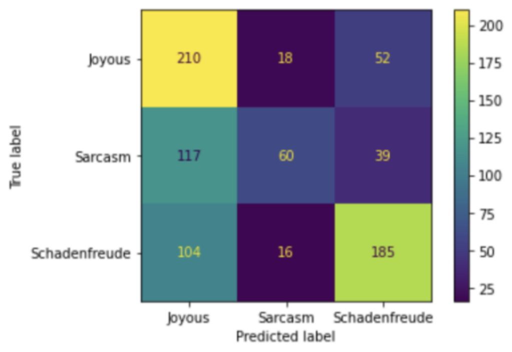


Figure 6: ngram_range = (2,3)

Overall performance has dropped quite a lot when experimenting with bigram and trigram. Both precision and recall values for all three class has dopped significantly which reduced F-1 score for all three classes too. Macro average of precision, recall and f1 score being 60%, 54% and 54% with an accuracy rate of 57%. The confusion matrix shows large number of sarcasm and schadenfreude laughter being misclassified as joyous laughter, many joyous laughter has also been misclassified.

ngram_range = (1,1)

```
              precision    recall  f1-score   support

      Joyous       0.63      0.94      0.76       280
     Sarcasm       0.77      0.38      0.50       216
Schadenfreude      0.86      0.79      0.82       305

    accuracy                           0.73       801
   macro avg       0.75      0.70      0.69       801
weighted avg       0.76      0.73      0.71       801
```

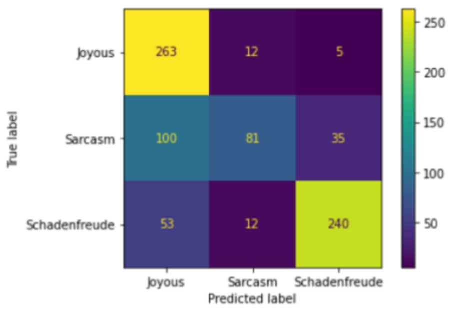


Figure 7: ngram_range = (1,1)

Experiment with unigrams only increase performance again making it nearly identical to n-gram range of 1-3 and 1-2. Average precision, recall and f1 score being 75%, 70% and 69% and accuracy being 73%. As seen in the confusion matrix true label and predicted label for all three classes were also similar.

ngram_range = (2,2)

```
              precision    recall  f1-score   support

      Joyous       0.49      0.73      0.59       280
     Sarcasm       0.60      0.26      0.37       216
Schadenfreude      0.65      0.62      0.64       305

    accuracy                           0.56       801
   macro avg       0.58      0.54      0.53       801
weighted avg       0.58      0.56      0.55       801
```

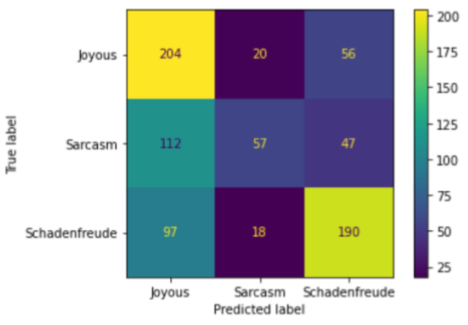


Figure 8: ngram_range = (2,2)

Performance for bigram model was very low, a 56% accuracy and macro average of precision, recall and f1-score are 58%, 54% and 53%

ngram_range = (3,3)

```
              precision    recall  f1-score   support

      Joyous       0.38      0.94      0.55       280
     Sarcasm       0.54      0.06      0.11       216
Schadenfreude      0.72      0.22      0.34       305

    accuracy                           0.43       801
   macro avg       0.55      0.41      0.33       801
weighted avg       0.55      0.43      0.35       801
```

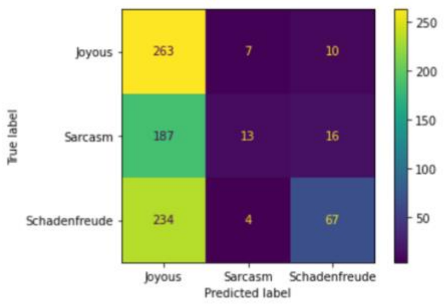


Figure 9: ngram_range = (3,3)

The trigram model performed the worse from all of them with a 43% accuracy and macro average of precision, recall and f1-score being 55%, 41% and 33%

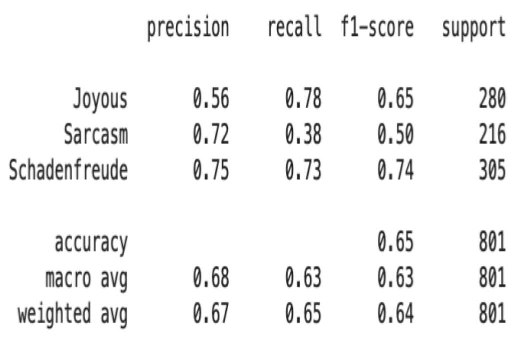
Summary

The unigram model has performed quite well and performance dropped gradually for bigram and trigram. Reason could be with increase in n-gram, the features get rarer. When individual words are used as features, they are much more likely to be found many times in the dataset than when two or three words are used as features, which may exist only once in the dataset. Therefore, bigger n-grams leads to decrease in performance. Unigram model performing better suggests there could be certain words that are important in categorising laughter into three classes. But using unigram only may sometimes cause meaning of sentence to be lost. Because to capture context of a word, the word before and after are often important. This may explain why when unigram is used with bigram and trigram the performance was slightly better (macro average f1 score was 69% for unigram and 70% for ngram_range = (1,2) and ngram_range = (1,3)) (see figure 4, 5 and 7). From confusion matrix sarcasm laughter could be seen to be the worst performing class in all experiments, in trigram model most of schadenfreude laughter was also classified as joyous laughter.

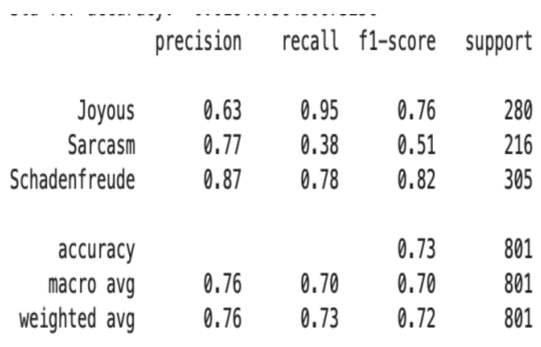
Both ngram_range = (1,2) and ngram_range = (1,3) has identical precision and recall score except ngram_range = (1,3) has 1% increase in precision for sarcasm class (see figure 4 and 5). Given n-gram range 1-3 performed little better than n-gram range 1-2, for further experiments I continued to use the n-gram range of 1-3.

4.3 Experiment-3: Tokenising usernames and links

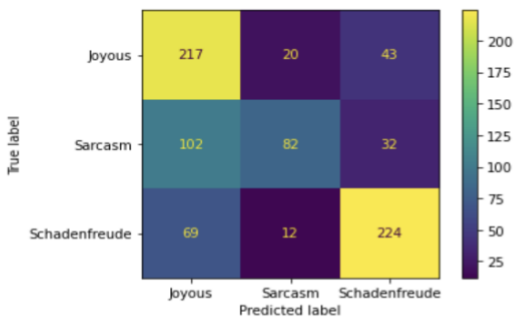
During the pre-processing steps I had tokenised all usernames as one token 'username' and all links into one token 'http' so that the classifier treats all usernames as singular feature and all links as same. Here I experimented to see how useful the pre-processing step of tokenising usernames and links were. I ran an experiment without tokenising all usernames and links to see if there was any significant drop in model's performance or did it perform better.



```
              precision    recall  f1-score   support

      Joyous       0.56      0.78      0.65       280
     Sarcasm       0.72      0.38      0.50       216
Schadenfreude      0.75      0.73      0.74       305

    accuracy                           0.65       801
   macro avg       0.68      0.63      0.63       801
weighted avg       0.67      0.65      0.64       801
```



```
              precision    recall  f1-score   support

      Joyous       0.63      0.95      0.76       280
     Sarcasm       0.77      0.38      0.51       216
Schadenfreude      0.87      0.78      0.82       305

    accuracy                           0.73       801
   macro avg       0.76      0.70      0.70       801
weighted avg       0.76      0.73      0.72       801
```



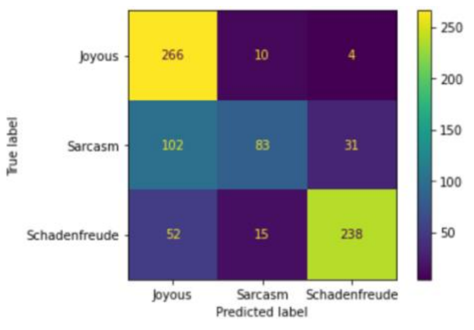
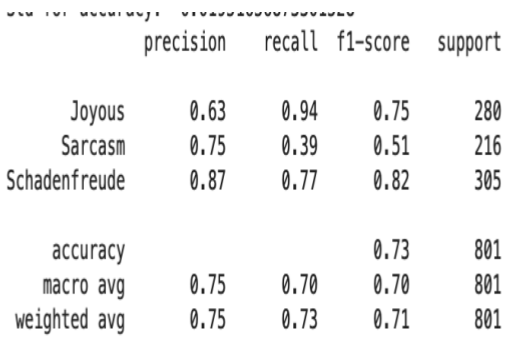


Figure 10: without tokenising username and links (left), with tokenising (right)

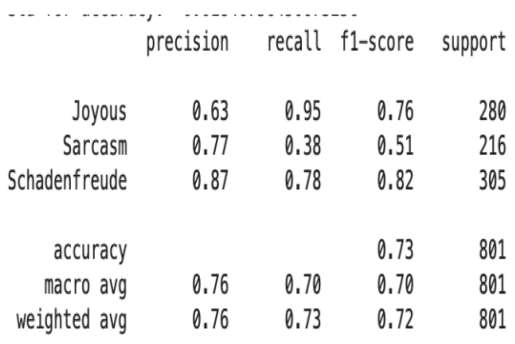
From figure 10 significant decrease in performance is clearly visible. Macro average of precision, recall and F-1 score without tokenising was 68%, 63% and 63%. The confusion matrix shows increase in misclassification of joyous laughter and schadenfreude laughter, but sarcasm laughter remained same. Precision, recall and F-1 score has dropped significantly for joyous and schadenfreude laughter, for sarcasm laughter precision was quite low but recall score has remained the same. This shows making single tokens for all usernames and all links was an important step.

4.4 Experiment-4: Keeping Question marks and Exclamation marks

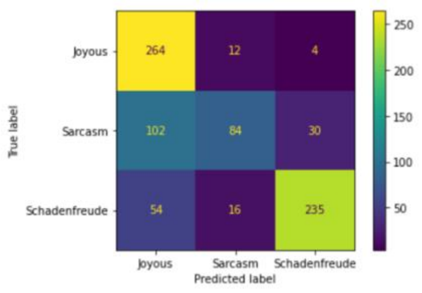
One assumption I made while making the dataset was that question marks and exclamation marks would be important for classification of laughter, because they are important emotion markers for excitement. When I removed punctuation marks at pre-processing steps, I removed question marks and exclamation marks too. In this experiment I kept question marks and exclamation marks to see if there was any increase or decrease in performance of the model.



```
[illegible]
               precision    recall  f1-score   support

       Joyous       0.63      0.94      0.75       280
      Sarcasm       0.75      0.39      0.51       216
Schadenfreude       0.87      0.77      0.82       305

     accuracy                           0.73       801
    macro avg       0.75      0.70      0.70       801
 weighted avg       0.75      0.73      0.71       801
```



```
[illegible]
               precision    recall  f1-score   support

       Joyous       0.63      0.95      0.76       280
      Sarcasm       0.77      0.38      0.51       216
Schadenfreude       0.87      0.78      0.82       305

     accuracy                           0.73       801
    macro avg       0.76      0.70      0.70       801
 weighted avg       0.76      0.73      0.72       801
```



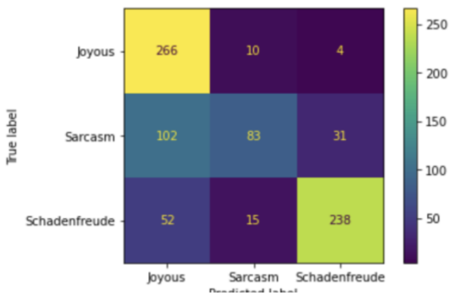
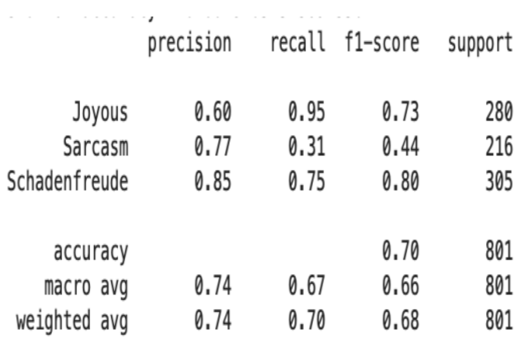


Figure 11: with (left) and without (right) question marks and exclamation marks

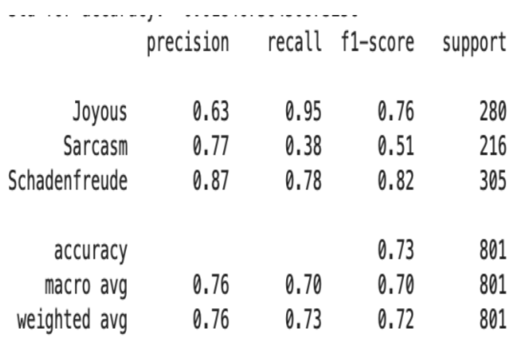
After comparing both side in figure 11, no significant difference has been observed. There has been 2% decrease precision score for sarcasm but 1 % increase in recall score. And 1% decrease in recall score for other two classes. The confusion matrix for both side looks quite similar. This proves my earlier assumption to be wrong as there hasn’t been any significant improvement in performance, rather there was slight decline in overall precision score.

4.5 Experiment-5: Stop-word removal

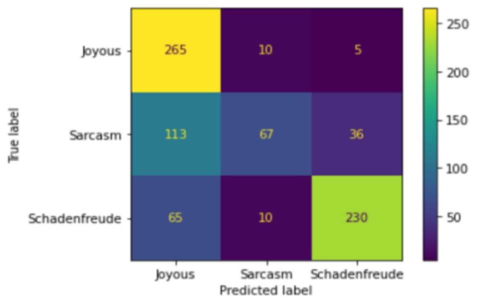
Another assumption I made was for schadenfreude and sarcasm laughter individual subject is quite important as this two laughter are generally directed at others. So I didn’t remove stop words as that would remove the be verbs and important information may get lost. So in this final experiment I removed stop words to see what difference it makes in all three classes, specially sarcasm and schadenfreude.



```
               precision    recall  f1-score   support

       Joyous       0.60      0.95      0.73       280
      Sarcasm       0.77      0.31      0.44       216
Schadenfreude       0.85      0.75      0.80       305

     accuracy                           0.70       801
    macro avg       0.74      0.67      0.66       801
 weighted avg       0.74      0.70      0.68       801
```



```
               precision    recall  f1-score   support

       Joyous       0.63      0.95      0.76       280
      Sarcasm       0.77      0.38      0.51       216
Schadenfreude       0.87      0.78      0.82       305

     accuracy                           0.73       801
    macro avg       0.76      0.70      0.70       801
 weighted avg       0.76      0.73      0.72       801
```



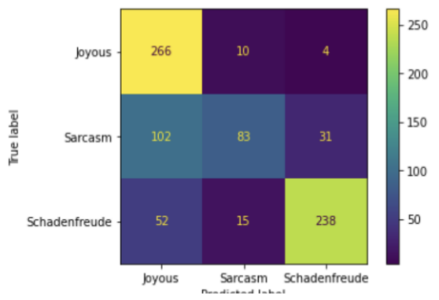


Figure 12: removing stop-words (left), without removing stop-words (right)

From figure 12 it is evident that removing stop-words has reduced overall performance of the model for all three classes. For joyous laughter precision value has decreased and recall value remained same. For sarcasm, precision value remained same but recall score decreased significantly, which means the model is making far less sarcasm predictions. And for schadenfreude both precision score and recall score dropped by 2%. Therefore, macro average for precision and recall also decrease precision score being 74% and recall 67%. Given recall score for sarcasm and schadenfreude laughter has decreased, this means the model is making fewer predictions for sarcasm and schadenfreude laughter. This proves my assumption of important information being lost with stop-words removal to be correct.

Chapter 5: **Evaluation**

In first experiment I checked to see which classifier is better suited for the laughter classification task. I experimented with Multinomial Naïve Bayes, Logistic Regression and Support Vector Machine. SVM performed better overall then the other two classifier. SVM had higher accuracy rate and higher average for precision, recall and F-1 score. Therefore, I selected SVM as the classifier for the model.

In next experiment I checked which combination of n-grams were better for feature extraction with TF-IDF Vectorizer. The n-gram range of 1-3 and n-gram range of 1-2 has performed nearly the same, but n-gram range of 1-3 has performed slightly better overall (1% better in average precision score). The third best performing n-grams was unigram which was also quite close to first two in terms of performance. This suggests some words have high weights for all three classes. Further experiments could be done to find the words. And the rest of the n-grams had performed significantly low.

In third experiment I checked if converting all usernames and links into single tokens had made any significant difference or not. Without tokenising the performance had dropped significantly, which proves converting all usernames and all links into single tokens to be an important pre-processing step.

In forth experiment I tested my assumption that question marks and exclamation marks will make a difference in performance of the model. This hasn't had any significant difference, rather this reduced performance of the model slightly.

For the final experiment I tested another assumption that stop word removal may lead to lose of important information for sarcasm and schadenfreude laughter as this two laughter are generally directed to others. Recall for both sarcasm and schadenfreude laughter class has decreased after removal of stop words. Which means the classifier was making less predictions for these two classes. The overall performance of the model also decreased which proved my assumption to be correct.

So SVM has been used as the classification algorithm for the final model and for feature selection TF-IDF Vectorizer with n-gram range of unigram, bigram, trigram and combinations of the three has been used. All usernames and links were made into single tokens, punctuation marks including question marks and exclamation marks were removed and stop words has not been removed. The sarcasm laughter class has a low recall score meaning the classifier makes less predictions for sarcasm class, instead most sarcasm laughter has been predicted to be Joyous laughter. This could be due to the dataset having 500 less tweets for the sarcasm laughter (1,000 tweets) class compared to the other two classes (1,500 and 1,509 tweets). The joyous laughter class had highest recall score and lowest precision score which means the model was making lots of predictions for joyous laughter class but lots of them being wrong as a large number of sarcasm laughter (102 tweets) tweets were classified as joyous and a lot of schadenfreude laughter (52 tweets) were also classified as joyous laughter. The schadenfreude laughter had the highest precision rate, meaning the model was making mostly accurate predictions for this class compared to other two classes.

Chapter 6: **Conclusion**

The final model had an overall accuracy of 73%. This could be improved with a larger dataset. Most of the sarcasm laughter tweets were wrongly predicted to be in other classes, mostly in joyous laughter class. Having a balanced dataset may had improved number of sarcasm tweets correctly predicted as sarcasm but this would mean using 500 less Joyous laughter tweets and 509 less schadenfreude laughter tweets to make all three classes have 1,000 tweets. But this means 1,000 less overall tweets in the dataset. This would mean reduced overall performance of the model, and if performance improved this can mean increased risk of overfitting. This led to me conducting the experiments with imbalanced dataset.

One future experiment can be done using a balanced dataset to see if that improves recall for sarcasm laughter class i.e. more sarcasm predictions. If balanced dataset doesn’t improve recall value for sarcasm laughter, then further works should focus on improving recall value of sarcasm laughter class. This could mean having relatively more sarcasm laughter tweets in the dataset or different ways of feature extraction like using a different word embedding system than TF-IDF Vectorizer.

More laughter tweets could be collected if the data was not labelled (i.e. without using hashtags for specific laughter types), but that would mean having to hand label all data which would be very time consuming or using different algorithms like K-means Clustering to categorise tweets. This approach can produce more labels to work with and can be experimented with in future.

References

Laskowski, K. and Burger, S., 2007, August. Analysis of the occurrence of laughter in meetings. In *INTERSPEECH* (pp. 1258-1261).

Adelswärd, V., 1989. Laughter and dialogue: The social significance of laughter in institutional discourse. *Nordic Journal of Linguistics*, *12*(2), pp.107-136.

Petitjean, C. and Morel, E., 2017. "Hahaha": Laughter as a resource to manage WhatsApp conversations. *Journal of Pragmatics*, *110*, pp.1-19.

Dresner, E. and Herring, S.C., 2010. Functions of the nonverbal in CMC: Emoticons and illocutionary force. *Communication theory*, *20*(3), pp.249-268.

Goffman, E., 1974. *Frame analysis: An essay on the organization of experience*. Harvard University Press.

Kennedy, L.S. and Ellis, D.P., 2004. Laughter detection in meetings.

McKay, I., 2020. Some distributional patterns in the use of typed laughter-derived expressions on Twitter. *Journal of Pragmatics*, *166*, pp.97-113.

Mohammad, S.M. and Kiritchenko, S., 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, *31*(2), pp.301-326.

Kunneman, F.A., Liebrecht, C.C. and van den Bosch, A.P.J., 2014. The (un) predictability of emotional hashtags in twitter.

Wiegand, M. and Ruppenhofer, J., 2021, April. Exploiting emojis for abusive language detection. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (pp. 369-380).

Golazizian, P., Sabeti, B., Asli, S.A.A., Majdabadi, Z., Momenzadeh, O. and Fahmi, R., 2020, May. Irony detection in Persian language: A transfer learning approach using emoji prediction. In *Proceedings of The 12th Language Resources and Evaluation Conference* (pp. 2839-2845).

Felbo, B., Mislove, A., Søgaard, A., Rahwan, I. and Lehmann, S., 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Collobert, R. and Weston, J., 2008, July. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167).

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine learning*, *20*, pp.273-297.

Campbell, N., Kashioka, H. and Ohara, R., 2005. No laughing matter. In *Ninth European Conference on Speech Communication and Technology*.

Tanaka, H. and Campbell, N., 2014. Classification of social laughter in natural conversational speech. *Computer Speech & Language*, *28*(1), pp.314-325.

Szameitat, D.P., Alter, K., Szameitat, A.J., Darwin, C.J., Wildgruber, D., Dietrich, S. and Sterr, A., 2009. Differentiation of emotions in laughter at the behavioral level. *Emotion*, *9*(3), p.397.

Kohavi, R., 1995, August. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, No. 2, pp. 1137-1145).

Tepperman, J., Traum, D. and Narayanan, S., 2006. *" Yeah right": sarcasm recognition for spoken dialogue systems*. UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES.

Grandini, M., Bagli, E. and Visani, G., 2020. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Kumar, V. and Subba, B., 2020, February. A TfidfVectorizer and SVM based sentiment analysis framework for text data corpus. In *2020 National Conference on Communications (NCC)* (pp. 1-6). IEEE.

26