# Identification of Misinformation in COVID-19 Tweets

*Thesis submitted by*

**Ankit Kumar (2017MT10727)**
**Naman Jhunjhunwala (2017MT10737)**

*under the guidance of*

**Prof. Niladri Chatterjee**

*in partial fulfilment of the requirements*
*for the award of the degree of*

**Bachelor of Technology**

**Department Of Mathematics and Computing**
**INDIAN INSTITUTE OF TECHNOLOGY DELHI**

**May 2021**

# THESIS CERTIFICATE

This is to certify that the thesis titled **Identification of Misinformation in COVID-19 Tweets**, submitted by **Ankit Kumar and Naman Jhunjhunwala**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by them under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Prof. Niladri Chatterjee**
Department of Mathematics
Indian Institute of Technology, Delhi

# ACKNOWLEDGEMENTS

We would first like to express our profound gratitude and indebtedness to Prof. Niladri Chatterjee and Raksha Agarwal, Department of Mathematics and Computing at Indian Institute of Technology, Delhi for their guidance, constructive criticism and valuable suggestions throughout the duration of this project. They consistently allowed this project to be our own work but steered us in the right direction whenever we needed help. We would also like to thank our families and friends for providing us with their unfailing support and continuous encouragement throughout the years and during the process of writing this thesis. This would not have been possible without them. Thank you.

Ankit Kumar
Naman Jhunjhunwala

# Contents

# ABBREVIATIONS

| | |
|---|---|
| **BERT** | Bidirectional Encoder Representations from Transformers |
| **ELMo** | Embeddings from Language Models |
| **FEVER** | large-scale dataset for Fact Extraction and VERification |
| **GloVe** | Global vectors for word representation |
| **GPT** | Generative Pre-trained Transformer |
| **LGBM** | Light Gradient Boosting Machine |
| **LSTM** | Long Short-Term Memory |
| **NER** | Named Entity Recognition |
| **POS** | Part Of Speech |
| **RoBERTa** | Robustly Optimized BERT Pretraining Approach |
| **SVM** | Support Vector Machines |
| **XLM** | Cross-lingual Language Models |
| **XLM-R** | XLM-RoBERTa |

# Chapter 1

# Introduction

Rapid propagation of social media has revolutionized the way information is consumed by general public. The ability of web platforms, such as Twitter, Instagram and Facebook, to quickly and broadly disseminate huge volumes of information has encouraged any user to be a (super) conduit of information. This can be helpful for problem solving in stressful and uncertain circumstances. However, this has also raised serious concerns about the disability of naive internet users in distinguishing truth from widespread misinformation.

As the world reacts to the COVID-19 pandemic, we are confronted with an overabundance of virus-related material. Some of this knowledge may be misleading and dangerous. The wildfire of Fake News in the times of COVID-19 has been popularly referred to as an 'infodemic' by the WHO chief, and in literature, we also see terms such as 'pandemic populism' and 'covidiocy' [1]. Distorted facts and figures formed by drawing false equivalence between scientific evidence and uninformed opinions and doctored videos of public figures have flooded the online space since the onset of COVID. In order to ensure safety and well being of online information consumers, it is crucial to identify and curb the spread of false information. Twitter should mark content that is demonstrably inaccurate or misleading and poses a serious risk of damage (such as increased virus transmission or negative impacts on public health systems). Hence, developing and improving classification methods for tweets is need of the hour.

In the present work, Fighting with Covid19 infodemic dataset [6] comprising English tweets about COVID-19 has been utilised for identifying false tweets.The goal of this task is to build prediction models for seven features using only text of tweet as input. Many Deep Learning models have been trained to predict several properties of a tweet, such as whether it is harmful, whether it contains a verifiable claim, whether it may be of interest to the general public, whether it appears to contain false information, etc.

# Chapter 2

# Related Work

Classification of tweets has been studied widely by many researchers. Most of the methods use traditional Machine Learning classifiers on the features extracted from individual tweets, such as POS, unigrams, bigrams. [3] built a Naive Bayes classifier for detecting sentiment of tweets. They considered Lemmas, Polarity Lexicons, and Multiword from different sources and Valence Shifters as input features to the classifier.

With the advancement of Modern NLP based on Deep Learning, models like LSTM, Bert, GPT etc. have taken precedence over more conventional classifiers like Naive-Bayes, SVM etc. Classification problems have primarily been tackled in two ways - Feature based and Fine-tuning based. Feature based approach uses word-embeddings such as GloVe [2], ELMO etc. and feed them into some Deep Learning model to perform downstream task. While fine-tuning based approach fine tunes all the pre-trained parameter on downstream tasks. Since fine-tuning based approach modify the parameter based on down-stream task, it generally performs better than feature based approach. We have experimented with both these approaches.

Recently, language models such as BERT [5], pre-trained on large amount of unlabelled data and fine tuned on downstream task, have given state-of-the-art results in numerous NLP tasks. BERTweet [7] is one such model which is pre-trained on English tweets.

It has been found that BERTweet outperforms other state-of-the art language models, e.g RoBERTa, XLM-R [8] with respect to several NLP tasks, viz. text classification, NER etc. This motivates us to use BERTweet based approach for this task.

# Chapter 3

# Dataset Description

## 3.1 Features

The dataset [6] contains numerous tweets, and the corresponding labels (which are mainly "yes"/"no"), that are the answers to the following questions:

1. **Does the tweet contain a verifiable claim?** A verifiable factual claim is a sentence that asserts what is true and can be supported by factual, verifiable evidence such as facts, detailed examples, or personal testimony.

2. **Does the tweet contain any false information?** It's possible that the reported assertion is incorrect. False information can be used on social media sites, websites, and news stories with the intent of intentionally misleading or deceiving people.

3. **Will the tweet be of any interest to the public?** The majority of the time, people do not make insightful statements that can be confirmed using our common knowledge. "The sky is blue," for example, is a claim, but it is uninteresting to the general public. The general public is most interested in issues such as healthcare, political reporting and observations, and current affairs.

4. **Can the claim made be harmful to society?** The goal of this query is to see whether the tweet's content seeks to or has the potential to harm society as a whole, individual people, companies, or products, or spread gossip about them. The knowledge would be harmed or weaponized as a result of the material. A rumor involves a form of a statement whose veracity is not quickly or ever confirmed.

5. **Does the claim need any verification?** It is important to have a competent fact-checker validate a credible claim that could damage society, individual people, companies, products, or government agencies. However, since fact-checking is a time-consuming process, not all accurate statements are relevant or worthwhile to be fact-checked by a competent fact-checker.

6. **Is the tweet harmful or misleading the society?** The purpose of this question is to categorize if the content of the tweet is intended to harm the society or weaponized to mislead the society.

7. **Should the govt pay any attention to the tweet?** People frequently use Twitter to blame officials, provide advice, and/or make calls for action. The information could be helpful to any government agency in developing a strategy, responding to it, or reacting to it. The goal of this query is to categorize this type of data. It's important to remember that not all knowledge demands a government entity's attention.

As per the dataset specifications, Q2 to Q5 will be NaN if and only if Q1 is "no". Further, Q1, Q6 and Q7 are never supposed to be NaN. However, there are a few instances in the dataset where this condition is violated (for instance, tweet number 56 in the train dataset). To tackle this problem, we have dropped the corresponding tweets (independently for all the questions) during training or validation. Finally, for predictions for cases other than Multi-Label classification (discussed later in Chapter 4), we first obtain the predictions for Q1, and the tweets are checked for the labels Q2 to Q5 only when Q1 is "yes". The approach for Multi-Label classification is described in the corresponding sections (4.5 and 5.2.5).

## 3.2 Examples

The following table shows a few example tweets from the dataset:

| Examples | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 |
|---|---|---|---|---|---|---|---|
| (335) If you didn't blame Obama for approx. 12,000 American swine flu deaths (and I didn't), but want to turn around and somehow pin approx. 22 corona virus on Trump/Pence, then YOU are the freaking problem with political discourse in this country. Full stop. | no | nan | nan | nan | nan | yes | no |
| (345) Just got a look of guidelines for testing COVID-19 at a San Diego hospital and they are...really bad! In order to get a test the patient has to have had close contact w someone w a confirmed infection or travelled to an infected area. But that's totally insufficient | yes | no | yes | yes | yes | yes | yes |
| (851)Three nations today receive #MadeInIndia Covid-19 Vaccines. Planes carrying consignments of #COVID19Vaccine doses landed in #Jamaica, #Tajikistan and #Belize. #VaccineMaitri https:/t.co3htiNmg5iC | yes | no | yes | no | no | no | no |

Table 3.1: Example tweets and their corresponding labels from dataset

## 3.3 Dependencies Among Questions

By the definitions of questions, it is evident that Q4 & Q6 and Q5 & Q7 have some dependence on each other. This can be seen in the dataset labels as well, because Q4 & Q6 have the same label for 87.6% of the tweets. Similarly, Q5 and Q7 have the same label 83.3% of the times. In the below figure, 00 and 11 denote the percentage of labels when both Q4 and Q6 (or Q5 and Q7) attain the same labels `no` and `yes` respectively. On the other hand, 01 and 10 denote the percentage of labels when Q4 and Q6 (or Q5 and Q7) attain the opposite labels.

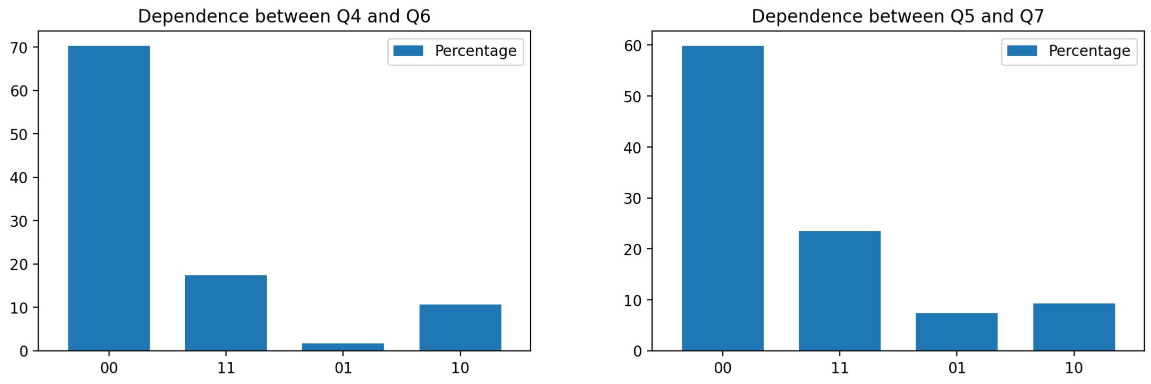

Figure 3.1: Dependency between Q4 and Q6 & Q5 and Q7

This motivated us to experiment with multi-label classification, which is explained in Section 4.5.

# Chapter 4

# Model Description

A vast number of Language Models have been developed in the last decade. We used a number of them to solve the given problem, and they are described in the following sections.

## 4.1 BERTweet

### 4.1.1 BERT

One of the major task in NLP is to convert a sentence into different sequence. For instance, in machine translation, we need to convert a sequence of words into different language. Prior to the discovery of transformer, seq2seq model based on LSTM gave the best performance on this task. A seq2seq model has two parts:

1. Encoder, which maps the sequence in higher dimension.
2. Decoder, which converts the higher dimension sequence to our desired output.

One choice for both encoder and decoder is to use an LSTM based model for each task.

One of the key concepts used in transformers is attention, which is introduced in [17]. The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important. Attention layer used in transformer takes three matrices $Q$, $K$, $V$ as input and compute output using below eqn:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Here, $d_k$ is equal to the number of columns in $Q$.

Researchers have shown that an architecture with only attention-mechanisms without any RNN (Recurrent Neural Networks) can improve on the results in translation task as well on various other tasks.

Figure 4.1 shows the architecture of the transformer. It contains two parts encoder on left and decoder on right. Many of the NLP architectures giving state of the art results on various tasks are based on transformers.

One of the advantages of transformer based model compared to RNN model is that RNN based models are very slow to train because we cannot parallelise it while the transformer based models can be parallelised.

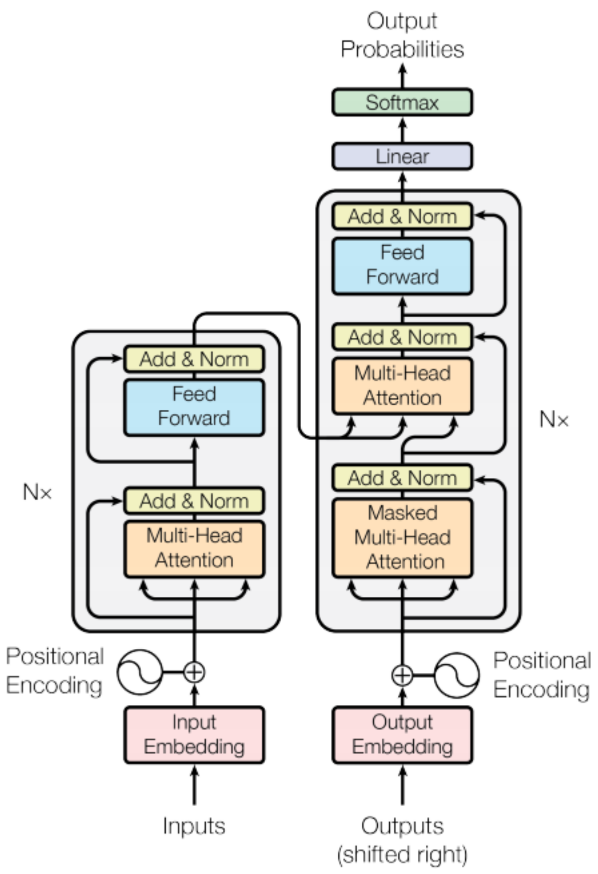

Figure 4.1: Transformer architecture

BERT (Bidirectional Encoder Representations from Transformers) [5] is a language representation model, which is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. It is basically a trained transformer encoder stack. The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks without too much task-specific architecture modifications.

BERT is basically a trained Transformer Encoder stack. It has many variants, but the ones relevant to the given task are:

1. $\text{BERT}_{\text{base}}$ – It has 12 encoder modules, and is comparable in size to the OpenAI Transformer
2. $\text{BERT}_{\text{large}}$ – It has 24 encoder modules, and is a ridiculously huge model which achieved the state of the art results reported in the original paper BERT paper.

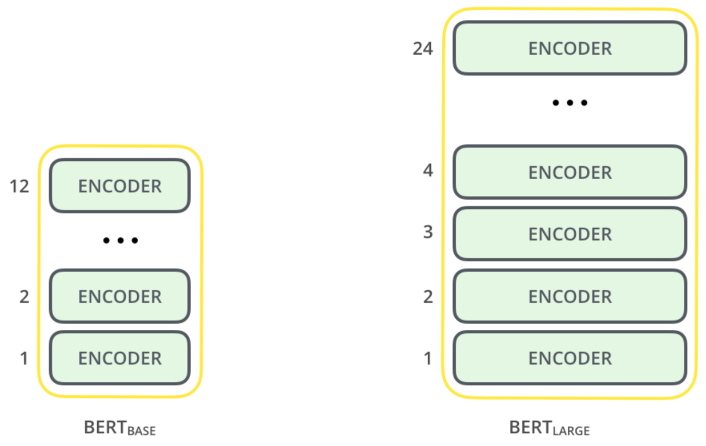

Figure 4.2: $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ variants of BERT

### 4.1.2 BERT to BERTweet

BERT and its variants have successfully produced state-of-the-art performance results for various NLP tasks. BERTweet [7] is one such variant. It is the first public largescale pre-trained language model for English Tweets. It has the same architecture as $\text{BERT}_{\text{base}}$ and is trained using the RoBERTa pre-training procedure. It outperforms $\text{RoBERT}_{\text{base}}$ [9] and $\text{XLM-R}_{\text{base}}$ [8], thus giving better results than the previous state-of-the-art models on Tweet NLP tasks: classification, Part-of-speech tagging and Named-entity recognition. It has three variants, that differ on the data they are trained on:

1. **Base:** This model has been trained on 845M (cased) English tweets along with 5M COVID-19 tweets.
2. **Cased:** It has been trained on additional 23M COVID-19 (cased) English Tweets
3. **Uncased:** It has been trained on additional 23M COVID-19 (uncased) English Tweets

### 4.1.3 Transfer Learning in BERTweet

Since BERTweet was trained on a different dataset (and not the one described in chapter 3), using the pre-trained embeddings provided by BERTweet may not give the best results. A Transfer Learning based approach, however, can improve the performance significantly. Transfer Learning is a technique in which a model trained on different task (large dataset) is used to perform the similar task.

Thus, in order to fine-tune the model for this task, the BERTweet model is plugged to a fully connected neural network. We vary the number of hidden layers, optimization algorithm (Adam and AdaFactor), learning rate and the number of epochs. For each iteration over the hyperparameters, we pick the weights which lead to the minimum cross-entropy loss over the validation dataset. For each label, we try all three of the BERTweet variants, and choose the best one depending upon the F1-score obtained.

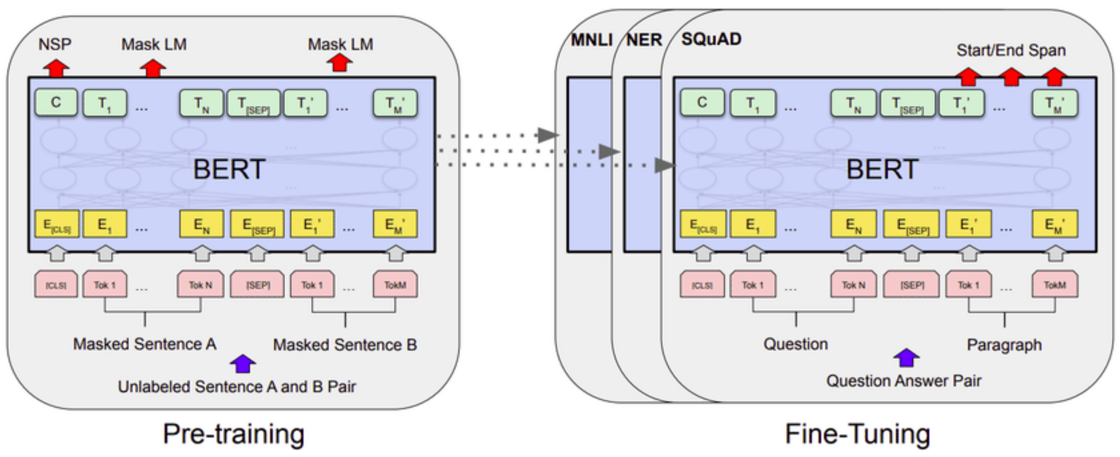

Figure 4.3: The figures shows the overall pre-training and fine-tuning procedures for BERT. Note that apart from output layers, both procedures use the same architectures.

The following pseudocode describes the Adam optimizer algorithm:

```
Algorithm 1: The Adam algorithm
Input: step size $\epsilon$, exponential decay rates for moment estimates - $\rho_1$ and $\rho_2$ in
  $[0, 1)$, small constant $\delta$ (to avoid division by zero)
Initialisation:
Initial parameters theta
1st and 2nd moment variables $s = 0$, $r = 0$
time step $t = 0$
while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $x^{(1)}, ..., x^{(m)}$ with
      corresponding labels $y^{(i)}$.
    Compute gradient: $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $s \leftarrow \rho_1 s + (1 - \rho_1)g$
    Update biased second moment estimate: $r \leftarrow \rho_2 r + (1 - \rho_2)g \cdot g$
    Correct bias in first moment: $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$
    Correct bias in second moment: $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$
    Compute update: $\Delta\theta = -\epsilon\frac{\hat{s}}{\delta+\sqrt{\hat{r}}}$ (operations applied element-wise)
    Apply update: $\theta \leftarrow \theta + \Delta\theta$
end
```

Adafactor on the other hand, is a stochastic optimization method based on Adam that reduces memory usage while retaining the empirical benefits of adaptivity. By tracking moving averages of the row and column sums of the squared gradients for matrix-valued variables, we are able to reconstruct a low-rank approximation of the exponentially smoothed accumulator at each training step that is optimal with respect to the generalized KL divergence. For an $n * m$ matrix, this reduces the memory requirements from $O(nm)$ to $O(n + m)$.

## 4.2 GloVe

GloVe [2] is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. The two main highlights of GloVe are:

1. **Nearest Neighbours:** The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words.

2. **Linear substructures**: Using vector arithmetic, GloVe provides analogies between different words. The most famous example to think about words and how they can be added and subtracted like vectors is: king – man + woman = queen. In other words, adding the vectors associated with the words king and woman while subtracting man is equal to the vector associated with queen.

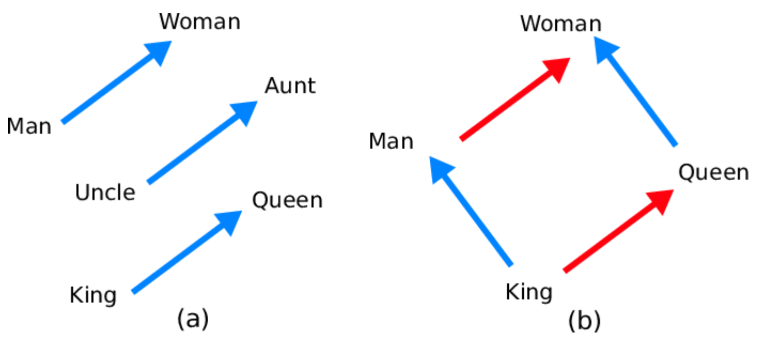

Figure 4.4: Analogies using GloVe

### 4.2.1 Training Procedure

GloVe uses a weighted least squares regression model. It introduces a weighting function $f(X_{ij})$ into the cost function, giving the model:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \bar{w}_j + b_i + \bar{b}_j - \log X_{ij})^2$$

where $V$ is the size of the vocabulary. The weighting function should obey the following properties:

- 1. $f(0) = 0$. If $f$ is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x)\log^2 x$ is finite.
- $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.
- $f(x)$ should be relatively small for large values of $x$, so that frequent co-occurrences are not overweighted.

### 4.2.2 Variants

There are multiple variants of GloVe available which have been trained on different datasets:

1. Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocabulary, max 300d vectors)
2. Common Crawl (42B tokens, 1.9M vocabulary, 300d vectors)
3. Common Crawl (840B tokens, 2.2M vocab, 300d vectors)
4. Twitter (2B tweets, 27B tokens, 1.2M vocabulary, upto 200d vectors)

Since our problem deals with tweets, we have used the GloVe Twitter embeddings. To obtain the embeddings for the entire tweet, we have taken average of the embeddings of the words present in the tweet. The subsequent model used is the same as that for BERTweet.

## 4.3 ELMo

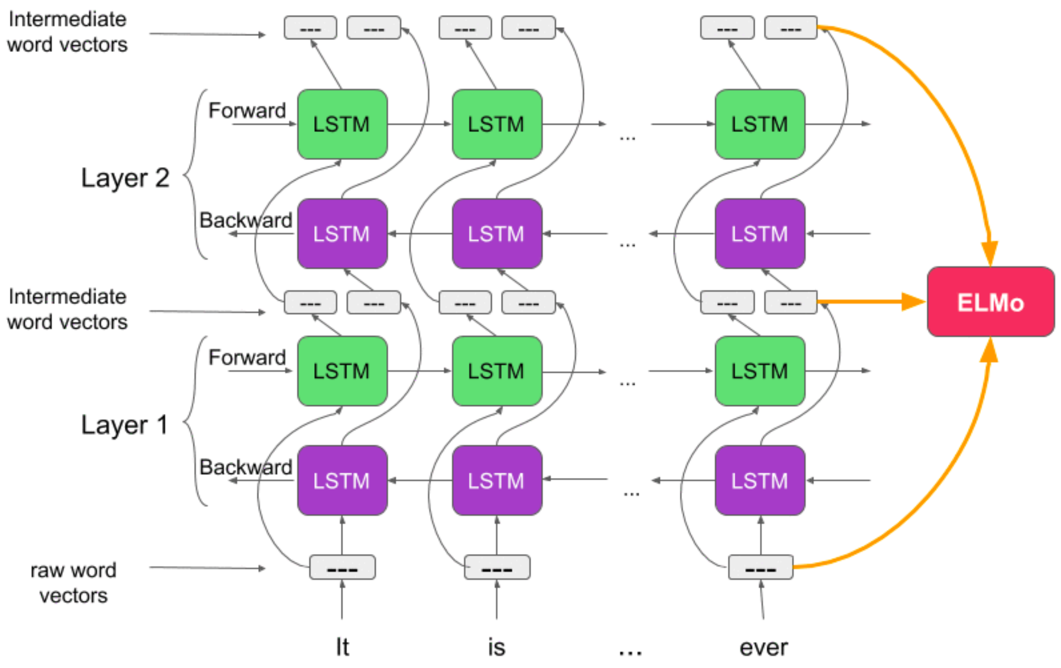

Figure 4.5: Architecture of ELMo

ELMo (Embeddings from Language Models) [4] was created by AllenNLP in 2018, and goes beyond standard embedding techniques. To generate word representations, it employs a deep, bi-directional LSTM model. Rather than a dictionary of words and their vectors, ELMo examines words and the sense in which they are used. It's also character-based, allowing the model to create definitions for terms that aren't in the dictionary. ELMo embeddings provide deep contextualized word representations that model both the syntax & semantics and polysemy.

For our implementation purposes, we have used the pre-trained ELMo model available on the Tensorflow hub. However, note that unlike BERTweet or GloVe, this model wasn't trained on a tweets dataset. It was instead trained on the One Billion Word Benchmark dataset ([10]). After obtaining the embeddings, the subsequent model used is the same as that for BERTweet.

## 4.4 BERTweet Features based Classification

In this method, we first trained our BERTweet based model and stored the output of the last fully connected layer for each dataset (training, validation and test). We used these stored values as input features for each of the below models.

### 4.4.1 SVM

SVM is a supervised machine learning model for binary classification problems.

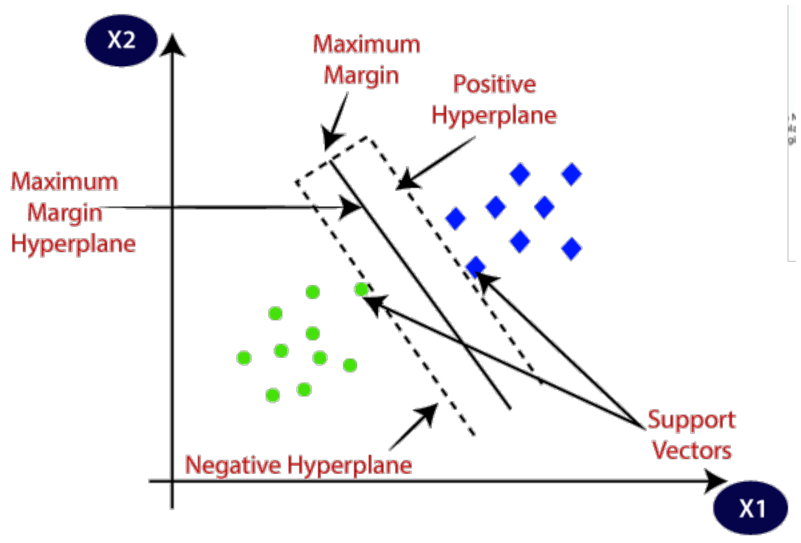

Figure 4.6: Working of an SVM

For linearly separable data, a good separation is formed by the hyper-plane having largest distance from data of both classes, Hence it constructs a hyper-plane by maximizing the margin [12]. But the data point may not be always separable , So it allows the point to deviate from the margin by introducing slack variables [13]. Mathematically the SVM optimisation can be stated as:

$$min_{\epsilon,w,b}\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\epsilon_i$$

$$\text{subject to } y^{(i)}(w^Tx^{(i)} + b) \geq 1 - \epsilon_i, i = 1, 2, \dots, m$$

$$\epsilon_i \geq 0, i = 1, 2, \dots, m$$

where $\epsilon_i$ are slack variables

### 4.4.2 Random Forest Classifier

Random forest [11], as the name suggests, is made up of a large number of individual decision trees that work together to form an ensemble. Random forest algorithm ensures that the trees are not too correlated by using following two methods

- **Bagging:** Each decision tree is formed on the data constructed by sampling from the training data with replacement
- **Random Feature:** Each decision tree is formed on random subset of original features

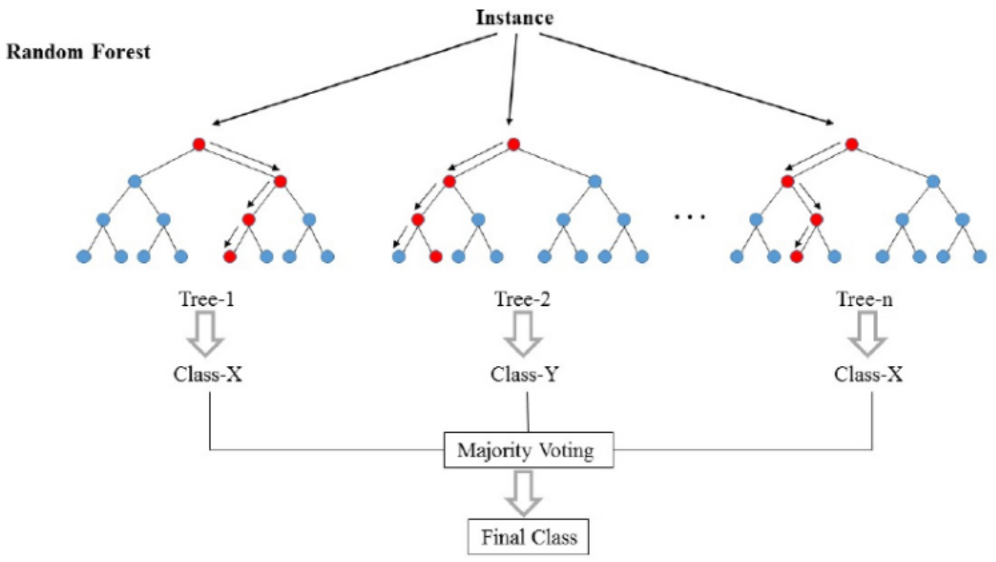

Figure 4.7: Random Forest

### 4.4.3 Gradient Boosting

Boosting is a weighted ensemble method in which each of the base algorithm is added sequentially and weights are updated to allow subsequent classifiers to pay more attention on training examples that were misclassified by previous classifier. Gradient boosting uses gradient descent to minimize the loss function. Many varaints of gradient boosting are formed by varying tree structure, feature engineering etc. We have used two variants CatBoost and LightGBM.

### XGBoost

Here, first XGBoost is described, and afterwards we describe CatBoost and LightGBM, along with reasons as to why they are a better choice than XGBoost.

XGBoost [18] stands for "Extreme Gradient Boosting", where the term "Gradient Boosting". It is is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In this work, maximum depth of the tree has been varied to find the best suitable hyper-parameter.

In XGBoost training, the specific objective at step $t$ is:

$$\sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t)$$

where $g_i$ and $h_i$ are defined as:

$$g_i = \partial_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)})$$
$$h_i = \partial^2_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)})$$

where $l$ is the given loss function and $\Omega$ is the regularization term. Thus, this becomes the optimization goal for the new tree. One important advantage of this definition is that the value of the objective function only depends on $g_i$ and $h_i$. This is how XGBoost supports custom loss functions. Every loss function can be optimized, including logistic regression and pairwise ranking, using exactly the same solver that takes $g_i$ and $h_i$ as input.

### CatBoost

CatBoost is a third-party library developed at Yandex that provides an efficient implementation of the gradient boosting algorithm [14]. The primary benefits of the CatBoost are:

- Ordered Boosting to overcome over fitting
- Allows to use non-numeric factors, instead of having to pre-process to turn it to numbers

- Use of symmetric trees for faster execution

**LightGBM**

LGBM, short for Light Gradient Boosted Machine, is a library developed at Microsoft that provides an efficient implementation of the gradient boosting algorithm [15]. LightGBM does not grow a tree level-wise (row by row) as most other implementations do. Instead it grows trees leaf-wise. It chooses the leaf it believes will yield the largest decrease in loss. The primary benefit of the LightGBM is such changes to the training algorithm that make the process dramatically faster, and in many cases, results in a more effective model.

## 4.5 BERTweet based Multi-Label Classifier

As discussed in 3.3, the labels in the dataset may have some dependence on each other. However, in the models described so far, all the labels have been assumed to be independent of each other (refer to Section 4.1.3). In this section, a Multi-Label based approach is described.

From the dataset description in Chapter 3, it can be inferred that:

1. When Q1 is "no", all of Q2, Q3, Q4 and Q5 are Nan, while Q6 and Q7 can attain values "yes" and "no".
2. When Q1 is "yes", all of Q2 to Q7 can attain values "yes" and "no".

Thus, the different questions can be combined to create 68 classes as:

1. When Q1 is "no", there are 4 ($2 * 2$) classes, one for each combination of Q6 and Q7. For this work, they have been taken from 0 to 3 (both inclusive).
2. When Q1 is "no", there are 64 ($2^6$) classes, one for each combination of Q2 to Q7. For this work, they have been taken from 4 to 67 (both inclusive).

After obtaining these labels, a model similar to described in Section 4.1.3 can be used. The key difference between the models will be that the output of the neural network will be a vector of size 68 (the number of classes), instead of 2 (number of classes when each label in considered independent of each other).

## 4.6 Ensemble

Ensemble modeling is a machine learning approach to combine multiple models in the prediction process. It tries to solve the problem of high variance. Intuitively we can comprehend that if we combine prediction of different models, classification score could improve. The combination can be implemented by aggregating the output of the models for regression problem or by taking max vote of the predictions for classification problem.

We created two ensemble models with BERTweet. Among the different models we obtained by fine-tuning BERTweet, we chose the best 3 and best 5 models. Then, we had a majority voting between the models, and chose the dominant answer for each label.

# Chapter 5

# Performance Evaluation

## 5.1 Evaluation Scheme

The given dataset has 869 tweets in the train dataset. We randomly split the dataset for training and validation purposes, with the splits having 695 and 174 tweets respectively ($80 - 20$ split). For testing, we are given a dev dataset with 53 tweets.

We have used F1-score as the main evaluation scheme. Apart from Q2 to Q5, we have assumed the labels to be independent of each other (because Q2 to Q5 only need to be checked when Q1 is "yes"). Thus, we first train a model for Q1 and obtain the predictions on the dev/test dataset. Then, we pick the tweets for which Q1 is "yes", and assign Q2 to Q5 to be NaN for the rest of the tweets. Subsequently, we have treated all the models for all the questions to be independent of each other. Due to this, it may be possible that while some model performs extremely well on one label, its performance may not be that good for some other label(s). Thus, we can have different models for different labels. So, we calculate label-wise F1-score to compare different models, and choose the best one.

## 5.2 Models

This section describes the results for all the models listed in chapter 4.

### 5.2.1 BERTweet

As was expected, in all our experiments, models based on BERTweet outperform all the other models that we described in chapter 4. Detailed results (F1-score) for all the labels (along with results for all the different models) can be found in Table 5.1.

### 5.2.2 GloVe

Although the dataset used in training for Glove Twitter embeddings is bigger than the one over which BERTweet was trained (2B vs 850M tweets), the GloVe vectors are "fixed", and unlike BERTweet, no Transfer Learning is involved for GloVe. As a result, GloVe performs much worse compared to BERTweet for most of the labels. The closest performance obtained is in Q3, when the GloVe based model was simply predicting all 1s (for Q3, the number of 1s is $> 90\%$ in the dataset (excluding NaNs)). Figure 5.1 shows the difference between results of BERTweet and GloVe based models.

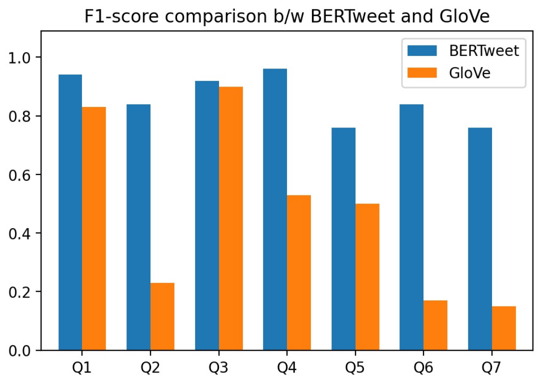

Figure 5.1: F1-score comparison between BERTweet and GloVe based models

### 5.2.3 ELMo

Since the ELMo model wasn't pre-trained on a Twitter dataset, it did not perform as well as BERTweet. However, since it was possible to use transfer learning here to fine-tune the weights of ELMo, it was mostly performing better than GloVe. On average, the difference between the F1-scores of BERTweet and GloVe is 0.39, while for ELMo it is 0.28. Further, ELMo performs better than GloVe for four labels, namely, Q2, Q5, Q6 and Q7. Even on the labels when the F1-score of ELMo is lesser than that of GloVe (Q1, Q3 and Q4), the difference between their scores is low (average difference of 0.06), but that is not true for the labels when ELMo beats GloVe (average difference of 0.2475). 5.2 shows the difference between results of BERTweet and ELMo based models & GloVe and ELMo based models.

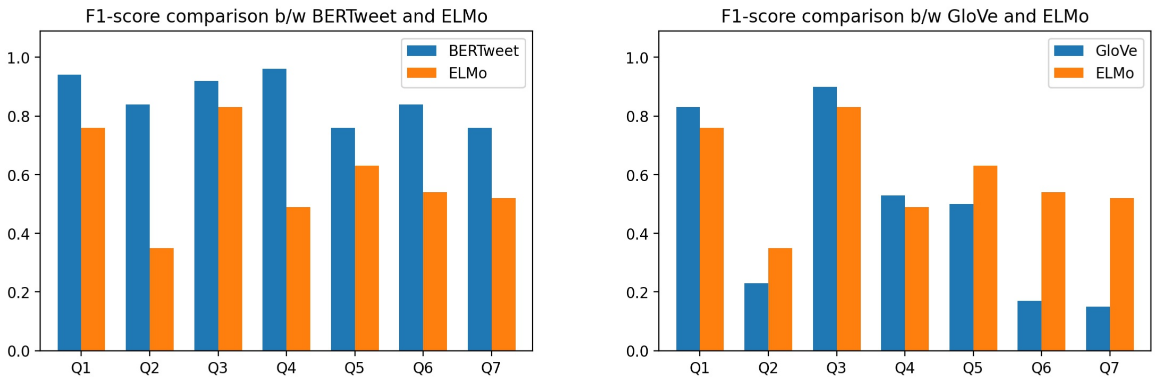

Figure 5.2: F1-score comparison between BERTweet and ELMo based models & GloVe and ELMo based models

### 5.2.4 BERTweet Features based Classification

#### SVM

Since this method takes last fully connected layer output of the BERTweet based model as input features, it performed better than GloVe and ELMo for almost all questions (except from ELMO for the label Q6). Still, the BERTweet based model outperforms this model for all labels.

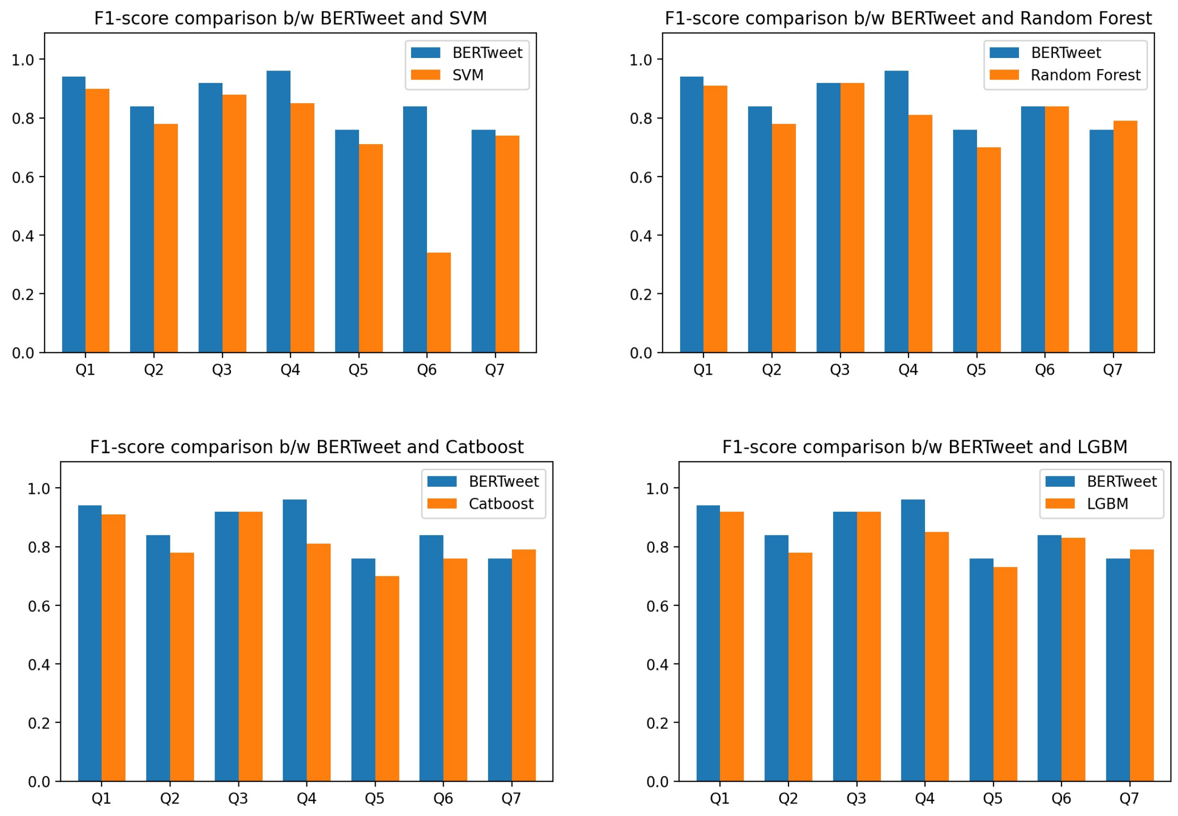

Figure 5.3: F1-score comparison between BERTweet and BERTweet Features based Classification models: (a) SVM (b) Random Forest (c) Catboost Gradient Boosting (d) LGBM

#### Random Forest & Gradient Boosting

For labels other than Q7, all the three models perform worse than BERTweet. However, unlike SVM, all three of these models (Random Forest, CatBoost and LGBM) beat GloVe and ELMo based models for all labels. This can be reasoned in the same way like previously done: all three of these models take last fully connected layer output of the BERTweet based model as input features, and BERTweet outperforms both GloVe and ELMo.

For Q7, however, all of these models give an F1-score of 0.79, which is better than the one obtained using BERTweet based models (0.76). The exact classification reports for these models are as follows:

```
Random Forest Classifier:
              precision    recall  f1-score   support

           0       0.97      0.73      0.84        45
           1       0.37      0.88      0.52         8
    accuracy                           0.75        53
   macro avg       0.67      0.80      0.68        53
weighted avg       0.88      0.75      0.79        53

CatBoost Classifier:
              precision    recall  f1-score   support

           0       0.94      0.76      0.84        45
           1       0.35      0.75      0.48         8
    accuracy                           0.75        53
   macro avg       0.65      0.75      0.66        53
weighted avg       0.86      0.75      0.79        53

LGBM Classifier:
              precision    recall  f1-score   support

           0       0.97      0.73      0.84        45
           1       0.37      0.88      0.52         8
    accuracy                           0.75        53
   macro avg       0.67      0.80      0.68        53
weighted avg       0.88      0.75      0.79        53
```

From the above classification report, note that all three models have the same accuracy (0.75). Table 5.1 gives the F1-scores of these models for all labels. From there, it can be seen that the average F1-scores for these models are:

- Random Forest: 0.821428571
- **CatBoost:** 0.831428571
- LGBM: 0.81

Since CatBoost has the highest average F1-score (and all the models have the same score for Q7), CatBoost based models should give the best performance on unseen data for Q7.

Figure 5.3 gives F1-score comparison between BERTweet and the four BERTweet Features based Classification models described above.

### 5.2.5 BERTweet based Multi-Label Classifier

As explained in Section 4.5, for Multi-Label classification, we work with 68 labels. Further, as mentioned in section 3.1, there are some tweets in the dataset which have inconsistency. In order to counter that, for Multi-Label classification, we drop those tweets. This leads to the following:

- For training dataset, the number of tweets decrease from 695 to 632.
- For validation dataset, the number of tweets decrease from 174 to 154.
- For dev dataset, the number of tweets decrease from 53 to 50.

With these changes in the dataset, the results may not be directly comparable with the other models. However, if these changes are not done, the number of labels would be 738 ($3^2$ when Q1 is "no" + $3^6$ when Q1 is "yes" ). After these changes in the datasets, Figure 5.4 shows the distribution of tweets for different labels.

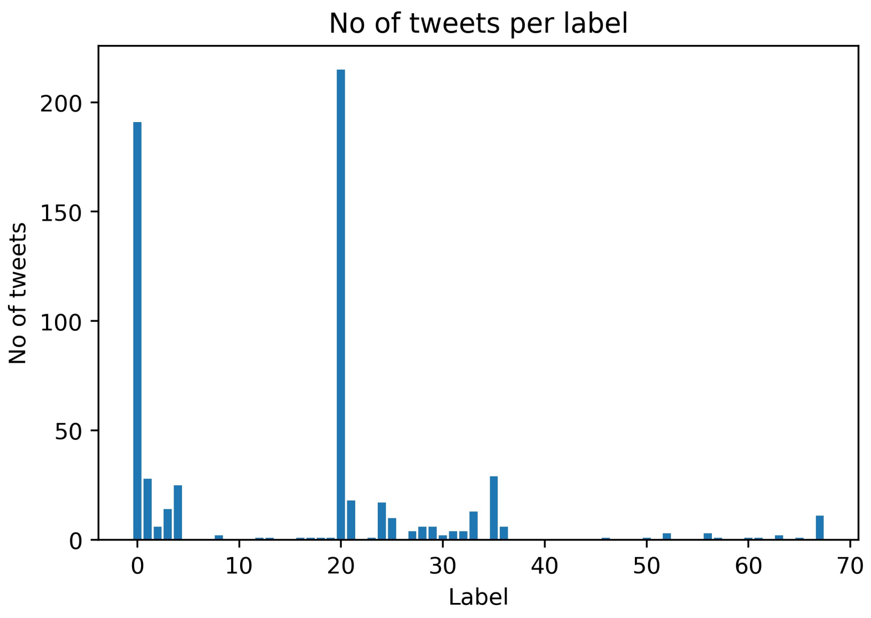

Figure 5.4: Histogram of the number of tweets per label

As can be seen from the above figure, the two maximum values occur for label 20 and 0. Label 20 corresponds to ("yes", "no", "yes", "no", "no", "no", "no"), while label 0 corresponds to ("no", "NaN", "NaN", "NaN", "NaN", "no", "no"). Overall:

- Only 36 labels have non-zero tweets corresponding to them.
- Only 15 labels have > 5 tweets corresponding to them.
- Only 10 labels have > 10 tweets corresponding to them.

Thus, the distribution of tweets of already very skewed, and it may become much worse if the "filtering" of tweets described above is not done.

For labels Q1 to Q5, BERTweet performs better than Multi-Label models. But, for Q6 and Q7, Multi-Label models performs better than BERTweet. For these two labels, Multi-Label models achieve F1-scores of 0.85 and 0.77 respectively, which are slightly better than the corresponding scores for BERTweet based models (0.84 and 0.76respectively). For Q7, the best result obtained so far is for CatBoost based models (0.79), and that still performs better than Multi-label models (0.77).

Thus, the new best model for Q6 is Multi-Label based model, with an F1-score of 0.85. Figure 5.5 gives comparison between the case when we assume all the questions to be independent and when we assume there can be some dependency between questions & between CatBoost based models and the Multi-Label models.

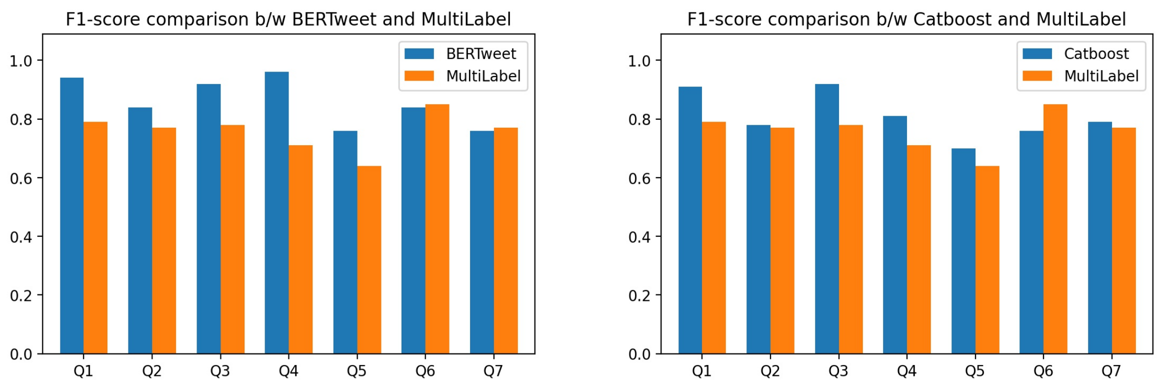

Figure 5.5: F1-score comparison between (a) BERTweet based models and Multi-Label BERTweet based models (b) CatBoost based models and Multi-Label BERTweet based models

### 5.2.6 Ensemble Models

Here, we have tried 3-BERTweet Ensemble (3BE) and 5-BERTweet Ensemble (5BE) models.

- For all the labels, 3BE performs atleast as good as 5BE.
- For all the labels, BERTwet performs atleast as good as 3BE.
- BERTweet is better than 3BE, which is better than 5BE, for all labels other than Q2.
- For Q2, all the three models have the same F1-score: 0.84.

Thus, the Ensemble Models fail to perform better than a single BERTweet model. This implies that for labels Q6 and Q7, they fail to perform better than the best models so far: Multi-Label and CatBoost based models respectively. Figure 5.6 provides comparisons between F1-scores of BERTweet, 3-BERTweet Ensemble and 5-BERTweet Ensemble models.

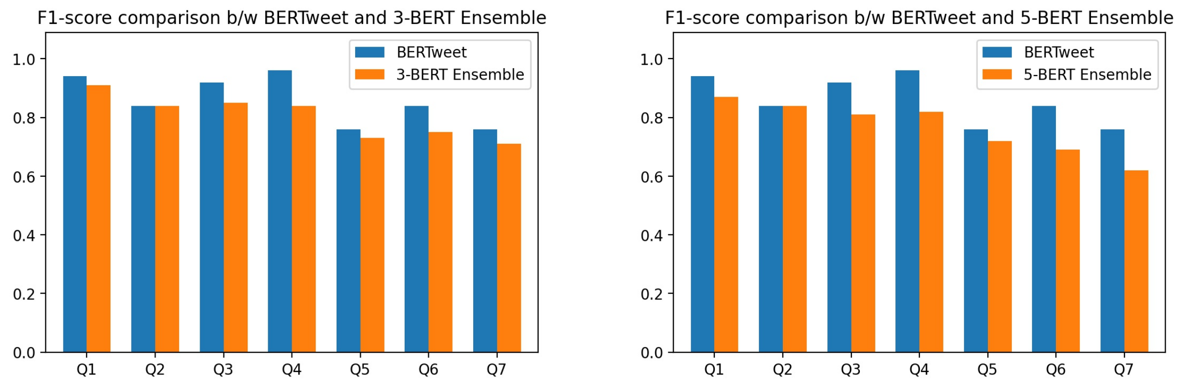

Figure 5.6: F1-score comparison between BERTweet and 3-BERTweet Ensemble based models & BERTweet and 5-BERTweet Ensemble based models

### 5.2.7 Best Model

The below table gives the F1-scores of different models.

| Models | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 |
|---|---|---|---|---|---|---|---|
| BERTweet | **0.94** | **0.84** | **0.92** | **0.96** | **0.76** | 0.84 | 0.76 |
| GloVe | 0.83 | 0.23 | 0.90 | 0.53 | 0.50 | 0.17 | 0.15 |
| ELMo | 0.76 | 0.35 | 0.83 | 0.49 | 0.63 | 0.54 | 0.52 |
| SVM | 0.90 | 0.78 | 0.88 | 0.85 | 0.71 | 0.34 | 0.74 |
| Random Forest | 0.91 | 0.78 | 0.92 | 0.81 | 0.70 | 0.84 | **0.79** |
| CatBoost | 0.92 | 0.78 | 0.92 | 0.85 | 0.73 | 0.83 | **0.79** |
| LGBM | 0.91 | 0.78 | 0.92 | 0.81 | 0.70 | 0.76 | **0.79** |
| Multi-Label | 0.79 | 0.77 | 0.78 | 0.71 | 0.64 | **0.85** | 0.77 |
| 3-BERT Ensemble | 0.91 | 0.84 | 0.85 | 0.84 | 0.73 | 0.75 | 0.71 |
| 5-BERT Ensemble | 0.87 | 0.84 | 0.81 | 0.82 | 0.72 | 0.69 | 0.62 |

Table 5.1: Comparison of F1-score of different models

Note that:

- The highest F1-score these models obtain is for Q4 (0.96), while the lowest F1-score is for Q5 (0.76).
- For labels Q1 to Q5, BERTweet based models perform the best.
- For Q6, Multi-Label BERTweet based model marginally beats the BERTweet based model (0.85 *vs* 0.84), coming out as the best model for label Q6.
- For Q7, all three of Random Forest, CatBoost and LGBM based models have the highest F1-score of 0.79. However, only one of these can be used, and Section 5.2.4 explains why CatBoost based model is a better choice among the three.

Figure 5.7 shows the comparison between the F1-scores of the above three models.

For the different labels, the following values of hyperparameters give the best results:

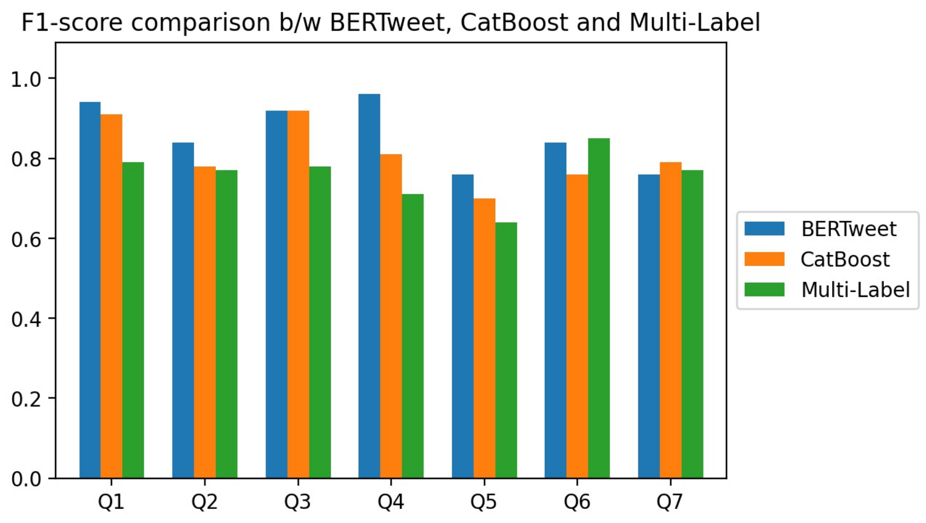

Figure 5.7: F1-score comparison between the top 3 models: BERTweet, CatBoost and Multi-Label BERTweet

- **Q1:**
  - Model: BERTweet Base
  - Parameters: AdaFactor optimizer, Learning rate $= 5 * 10^{-5}$
- **Q2:**
  - Model: BERTweet Cased
  - Parameters: Adam optimizer, Learning rate $= 1 * 10^{-5}$
- **Q3:**
  - Model: BERTweet Base
  - Parameters: Adam optimizer, Learning rate $= 1 * 10^{-5}$
- **Q4:**
  - Model: BERTweet Uncased
  - Parameters: AdaFactor optimizer, Learning rate $= 1 * 10^{-4}$
- **Q5:**
  - Model: BERTweet Uncased
  - Parameters: Adam optimizer, Learning rate $= 2 * 10^{-5}$
- **Q6:**
  - Model: BERTweet Features + CatBoost Gradient Boosting
  - Parameters: BERTweet - Adam optimizer, Learning rate $= 1 * 10^{-5}$; CatBoost - 1000 Iterations
- **Q7:**
  - Model: Multi-Label BERTweet Base
  - Parameters: AdaFactor optimizer, Learning rate $= 1 * 10^{-5}$

# Chapter 6

# Error Analysis

For Q1, only three tweets in the dev data (452, 490, 492: all having a verifiable claim) are predicted wrong by our model. Similarly, for Q2, three examples (491, 498, 500), which also have a positive label (denoting that the tweet appears to contain false information), have been predicted wrong while for Q4, four examples (461, 462, 484, 485), all having negative labels (denoting that the claim made in the tweet cannot be harmful to the society), are predicted wrong by our model. Some of these tweets (as described above) can be found in Table 6.1. Rest of the labels do not have such pattern.

| Labels | Example 1 | Example 2 |
|---|---|---|
| Q1 | (452) Instead of prioritizing regular Americans who need tests for #coronavirus, or paid sick leave because they live paycheck to paycheck, @realDonaldTrump wants to bail out oil billionaires. Thank goodness the House of Representatives, and not @POTUS, has the Power of the Purse. URL" | (490) We love this handwashing dance from Vietnamese dancer, Quang ng. Washing your hands with soap and water is one of the first steps to protect yourself from #coronavirus. |
| Q2 | (491) Just like all the other fake stuff they do, the COVID-19 over-hype will backfire on the Democrats. The economy will come roaring backs with China's grip on trade weakened and Trump's high approval on handling the virus will only help. | (498) But, but...Trump didn't prepare for the coronavirus...his admin still doesn't have a clue...we are just not ready to combat a pandemic...Trump ignored the HHS, CDC? #FakeNews WATCH ?? #coronavirus #RepMarkGreen thank you! URL" |
| Q4 | (461) The Italian COVID-19 outbreak has just been explosive... look at the numbers &amp; timeframe. Time is not a luxury we have! Feb 18: 3 cases Feb 21: 20 cases Feb 24: 231 cases Feb 27: 655 cases Mar 1: 1,694 cases Mar 4: 3,089 cases Mar 7: 5,883 | (462) A Youtuber who recently made a racist remark regarding BTS by relating them to Corona virus will now be making a video about them where he roasts the band and our fandom I request ARMYs to pls block him and report his channel, Ducky Bhai on YouTube URL |

Table 6.1: Example tweets (from dev data) on which the BERTweet model fails. For each tweet the preceding number in parenthesis denotes the tweet number in the database

# Chapter 7

# Conclusion and Future Work

Through this project, we have done a comprehensive analysis of different NLP classification, and tested their robustness with respect to the NLP4IF-2021-COVID19-task dataset. Overall, ten different models were implemented (described in Chapter 4), out of which nine posed the problem as binary classification problem, while one used a Multi-Label approach (Section 4.5).

For five of the seven questions asked in the dataset, BERTweet outperforms the rest of the models. But, for two of the questions (Q6 and Q7) the Multi-Label BERTweet and CatBoost model based on features extracted from BERTweet perform better than naive BERTweet (although note that both of these models still use BERTweet at their core, just in a different manner).

However, this work assumes that we only use the dataset given. But, there are some questions in the dataset (for instance, Q2 states: "Does the tweet contain any false information") for which performance can be improved if a database of claims or a Claim Verification Model, such as FEVER [16] is used.

In this work, we have considered only English tweets. This work can be extended for other popular languages, for which enough tweets are available, such as Hindi, Mandarin, Arabic, etc. as well. Although unlike English tweets, pre-trained BERTweet is not available for these languages, using either a BERT model or creating a version of BERTweet for these languages (probably a better choice considering the task is specific to tweets) can always be done. Kar et. al. [19] have recently worked to expand fake tweet detection approach to multiple Indic languages, resorting to mBERT based model, which is fine tuned over datasets created in Hindi and Bengali. Similar approach can be used to extend our work for other languages (and questions) as well.

# Bibliography

[1] Kris Hartley and Minh Khuong, "Fighting fake news in the COVID-19 era: policy insights from an equilibrium model", 2020.

[2] Jeffrey Pennington, Richard Socher and Christopher D. Manning, "GloVe: Global Vectors for Word Representation", 2014.

[3] Pablo Gamallo and Marcos Garcia, "A NaiveBayes Strategy for Sentiment Analysis on English Tweets", 2014.

[4] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer, "Deep contextualized word representations", 2018.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018.

[6] Shaden Shaar, Firoj Alam, Giovanni Da San Martino, Alex Nikolov, Wajdi Zaghouani, Preslav Nakov and Anna Feldman, "Proceedings of the Fourth Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda", 2021.

[7] Dat Quoc Nguyen, Thanh Vu and Anh Tuan Nguyen, "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations", 2020.

[8] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek and Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov, "Unsupervised cross-lingual representation learning at scale", 2019.

[9] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", 2019.

[10] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn and Tony Robinson, "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling", 2014.

[11] Leo Breiman, "Random Forest", 2001

[12] Vladimir Vapnik, "Estimation of Dependences Based on Empirical Data", 1982.

[13] Corinna Cortes and Vladimir Vapnik, "Support-Vector Networks", 1995.

[14] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush and Andrey Gulin, "CatBoost: unbiased boosting with categorical features", 2017.

[15] Guolin Ke1, Qi Meng2, Thomas Finley3, Taifeng Wang1, Wei Chen1, Weidong Ma1, Qiwei Ye1 and Tie-Yan Liu1, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", 2017.

[16] James Thorne, Andreas Vlachos, Christos Christodoulopoulos and Arpit Mittal, "The Fact Extraction and VERification (FEVER) Shared Task", 2018

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need", 2017

[18] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", 2016.

[19] Debanjana Kar, Mohit Bhardwaj, Suranjana Samanta, Amar Prakash Azad, "No Rumours Please! A Multi-Indic-Lingual Approach for COVID Fake-Tweet Detection", 2020.