University of Rhode Island

Department of Computer Science and Statistics

Machine Learning Course Project

Sentiment Analysis of Coronavirus Tweets

Group Members:

Masoud faramarzi

Maryam Kafi Kang

Lily Sisouvong

Advisor:

Marco Alvarez

Fall 2021

Table of Contents

ABSTRACT

Recent Covid-19 pandemic caused various reactions from people regarding how to treat it. Some people were suspicious of the vaccination effectiveness, while others supported it. These disputes not only could be observed in society, but also in social media platforms such as twitter. The objective of this project was to train a model, able to judge if a tweet has a negative, neutral or positive sentiment regarding the Covid-10 subject. For this purpose fastai as a high level framework in deep learning was applied to construct the desired Natural Language Processing (NLP) pipeline. This framework self contains the state of art algorithms for text preprocessing, language modeling and text classifying. As a result, it was a functional tool to produce a relatively accurate, covid-19 sentiment classifier, with minimum lines of coding. Preprocessed data were used to make a language model and the obtained language model was used as an encoder to train the classifier. Hyper parameters were optimized and accuracies of best combinations of them were compared. The best classifier with highest accuracy was used to conduct some analysis on the covid tweets and results were visualized at the end.

Keywords: Natural Language Processing; fastai; Transfer Learning; Language Model; Classifier

1. INTRODUCTION

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus. This disease has caused over sixty-five million infections and a half million deaths all over the world. While extensive studies have been conducted to examine various types of influence of COVID-19 on public health, including the studies concerning the infected cases number and the fatality, investigations of the impact of the pandemic on people's emotion are relatively limited (Ridhwan et al., 2021). Many countries have announced "state of emergency" and they encouraged people to stay at home instead of having face to face communication. Therefore, online communication such as social media have become active in communication. One of the most discussed topics has become CoronaVirus on the internet. After the creation of different types of vaccines like Pfizer/BioNTech, Sinopharm, Sinovac, Moderna, Oxford/AstraZeneca, Covexin, Sputnik V for Coronavirus, people share their comments about them in social media. Since people easily share their feelings and comments in social media, It is interesting to conduct text mining of public information in social media to extract useful information. One of the applications of this useful information is that they can help companies or

businesses to predict the level of product acceptance and devise strategies in order to increase product quality. Machine learning and NLP plays an important role in sentimental analysis of tweets data. Sentimental analysis of text can be done using Classification tasks which classify sentiment of text into some groups. In this paper, sentiment analysis method was applied on COVID-19 vaccines Twitter data. 'Complete Tweet Sentiment Extraction Data' and 'All Covid-19 Vaccines Tweets' datasets were used in this study.

2. RELATED WORKS

Many papers have worked on sentimental analysis of tweets data. For example, Villavicencio et al. (2021) conducted a study in the Philippines on sentiment analysis of COVID-19 vaccination-related tweets using Naïve Bayes classifier and found 81.77% accuracy. The study considered 11,974 manually labeled tweets for the classifier to test the accuracy. In another study, Khan et al. (2020) performed sentiment scoring of 50,000 COVID-19- related tweets using Naïve Bayes classifier and found 19% positive and 70% negative tweets. Deep learning-based classifiers were used by Kaur et al. (2021) to classify 600 COVID-19- related tweets into different sentiment labels. Also Rahman et al. (2022) performed a sentiment analysis of COVID-19-related Twitter data using ensemble machine learning algorithms.

3. METHODS

3.1. Framework

In this study fastai will be used as a high level framework which works on top of PyTorch, while PyTorch works on top of Python. Pytorch is a different kind of deep learning library (dynamic, rather than static), which has been adopted by many (if not most) of the researchers that we most respect, and in a recent Kaggle competition was used by nearly all of the top 10 finishers (fastai, 2021).

3.2. About fastai

fastai is a deep learning library which provides practitioners with high-level components that can quickly and easily provide state-of-the-art results in standard deep learning domains, and provides researchers with low-level components that can be mixed and matched to build new approaches. The power of fastai shines in reducing the amount of code we require for training, and more easily takes advantage of modern best practices (fastai, 2021).

3.3. Three Type of Modeling Methods

Scenario 1: Model from scratch

Scenario 2: Transfer learning

Scenario 3: Transfer learning with new vocab addition. This methodology is the sweet spot between using transfer learning and making a model from scratch (Substack, 2021).

3.4. Transfer Learning

We will benefit from transfer learning techniques to achieve a high performance language model and classifier without training neural networks from scratch. There are two mains transfer learning methods could be applied to a NLP project:

- OpenAI's GPTs, BERTs, ELMos
- Universal Language Model FIne-Tuning (ULMFit)

In this project the latter one has been chosen, since it is the method used by fastai (as the platform used in this project) and also being state of art (Howard et al., 2018).

3.5. ULMFit

The core idea is two-fold — using generative pre-trained Language Model + task-specific fine-tuning was first explored in ULMFiT (Howard & Ruder, 2018), directly motivated by the success of using ImageNet pre-training for computer vision tasks. The base model is AWD-LSTM. ULMFit follows three steps to achieve good transfer learning results on downstream language classification tasks:

1. General Language Model pre-training: on Wikipedia text.
2. Target task Language Model fine-tuning: ULMFiT proposed two training techniques for stabilizing the fine-tuning process.
3. Target task classifier fine-tuning: The pretrained LM is augmented with two standard feed-forward layers and a softmax normalization at the end to predict a target label distribution(Medium, 2021).

Since the introduction of ULMFiT, Transfer Learning became very popular in NLP and yet Google (BERT, Transformer-XL, XLNet), Facebook (RoBERTa, XLM) and even OpenAI (GPT, GPT-2) begin to pre-train their own model on very large corpora. This time, instead of using the AWD-LSTM neural network, they all used a more powerful architecture based on the Transformer **(**Vaswani et al., 2017**).**

3.6. Discriminative Learning Rates and Gradual Unfreezing

In the training of both language model and classifier we have used *discriminative learning rates* and *gradual unfreezing*. It was recommended by fastai as a method that leads to better results for this type of model. Below *discriminative learning rates* and *gradual unfreezing are* explained briefly (GitHub-fastai, 2021).

A discriminative learning rate is when you train a neural net with different learning rates for different layers. The main application for these is when you are fine-tuning a pre-trained model for your own classifier, like this case (aidancoco, 2021).

Gradual unfreezing is that all layers except the last layer are frozen in the first epoch and only the last layer is fine-tuned. In the next epoch, the last frozen layer will be unfrozen and fine-tuning all unfrozen layers. More and more layers will be unfrozen in the coming epoch (towardsdatascience, 2021).

3.7. Dataset

In our project, we have two datasets, the first dataset is a dataset of vaccine tweets (Kaggle, 2021) which contains 212,982 rows and 19 columns, shown in Fig 1. We downloaded vaccine tweets dataset from a kaggle dataset and it contained recent tweets about the COVID-19 vaccines used in entire world on large scale, including: Pfizer/BioNTech, Sinopharm, Sinovac, Moderna, Oxford/AstraZeneca, Covexin, Sputnik V. Relevant features in vaccine tweets dataset include: id, user_name, user_location, user_description, user_created, user_followers, user_friends, user_favourites, user_verified, date, text, hashtags, source, retweets, favorites, is_retweet, sentiment, and orig_text.

```
vax_tweets.head()
```

	id	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	text	hashtags
0	1340539111971516416	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning diverse sources 24/7/365. RT's, Follows and 'Likes' will fuel me 🤷‍♀️	2009-04-08 17:52:46	405	1692	3247	False	2020-12-20 06:06:44	Same folks said daikon paste could treat a cytokine storm #PfizerBioNTech https://t.co/xeHhIMg1kF	['PfizerBioNTech']
1	1338158543359250433	Albert Fong	San Francisco, CA	Marketing dude, tech geek, heavy metal & '80s music junkie. Fascinated by meteorology and all things in the cloud. Opinions are my own.	2009-06-21 15:27:30	834	666	178	False	2020-12-13 16:27:13	While the world has been on the wrong side of history this year, hopefully, the biggest vaccination effort we've ev… https://t.co/dICHnrZjNm	NaN
2	1337858199140118533	🏳️‍🌈🇨🇦🇺🇦	Your Bed	heal, hydra 🏴‍☠️ ✡	2020-06-25 23:30:28	10	88	155	False	2020-12-12 20:33:45	#coronavirus #SputnikV #AstraZeneca #PfizerBioNTech #Moderna #Covid_19 Russian vaccine is created to last 2-4 years … https://t.co/eYlCKBbr8P	['coronavirus', 'SputnikV', 'AstraZeneca', 'PfizerBioNTech', 'Moderna', 'Covid_19']
3	1337855739918835717	Charles Adler	Vancouver, BC - Canada	Hosting "CharlesAdlerTonight" Global News Radio Network. Weeknights 7 Pacific-10 Eastern - Email comments/ideas to charles@charlesadlertonight.ca	2008-09-10 11:28:53	49165	3933	21853	True	2020-12-12 20:23:59	Facts are immutable, Senator, even when you're not ethically sturdy enough to acknowledge them. (1) You were born i… https://t.co/jqgV19kcN4	NaN
4	1337854064604966912	Citizen News Channel	NaN	Citizen News Channel bringing you an alternative news source from citizen journalists that haven't sold out. Real news & real views	2020-04-23 17:58:42	152	580	1473	False	2020-12-12 20:17:19	Explain to me again why we need a vaccine @BorisJohnson @MattHancock #whereareallthesickpeople #PfizerBioNTech… https://t.co/KxbSRoBEHq	['whereareallthesickpeople', 'PfizerBioNTech']

Fig 1. Vaccination Tweets features

Text column of the vaccine tweets dataset was used to train a Twitter language model, but since this project's goal was sentiment analysis, it was needed to find another dataset that also contains sentiment labels to train our classifier. The second dataset was "Complete Tweet Sentiment Extraction Data" (Max, 2021) which contains 40,000 tweets labelled as either

negative, neutral or positive sentiment (Fig 2). Therefore text columns were used from both datasets as input for the language model. However, for the classification model it was needed to remove all rows with missing sentiment. Relevant features in tweets dataset include : textID, sentiment, author, Old_text, aux_id, new_seniment, selected text.

```
[ ] tweets.head()
```

	textID	sentiment	author	text	old_text	aux_id	new_sentiment	selected_text
0	1956967341	empty	xoshayzers	i know i was listenin to bad habit earlier and i started freakin at his part =[	@tiffanylue i know i was listenin to bad habit earlier and i started freakin at his part =[	p1000000000	NaN	NaN
1	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin on your call...	Layin n bed with a headache ughhhh...waitin on your call...	c811396dc2	negative	headache
2	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...	Funeral ceremony...gloomy friday...	9063631ab1	negative	gloomy
3	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!	wants to hang out with friends SOON!	2a815f151d	positive	wants to hang out with friends SOON!
4	1956968416	neutral	xkilljoyx	We want to trade with someone who has Houston tickets, but no one will.	@dannycastillo We want to trade with someone who has Houston tickets, but no one will.	82565a56d3	neutral	We want to trade with someone who has Houston tickets, but no one will.

Fig 2. General Tweets features

3.8 . Preprocessing

Sentences can be different lengths, and documents can be very long. So, how can we predict the next word of a sentence using a neural network?

First all of the documents in the dataset should be concatenated into one big long string and splited into words, resulting in a very long list of words (or "tokens"). The independent variable will be the sequence of words starting with the first word in the resulting long list and ending with the second to last, and the dependent variable will be the sequence of words starting with the second word and ending with the last word. The vocab will consist of a mix of common words that are already in the vocabulary of our pretrained model and new words specific to the corpus (cinematographic terms or actors names, for instance). The embedding matrix will be built accordingly: for words that are in the vocabulary of our pretrained model, the corresponding row in the embedding matrix will be taken from the pretrained model; but for new words such a word does not exists, so the only option is initialize the corresponding row with a random vector.

```
def de_emojify(inputString):
    return inputString.encode('ascii', 'ignore').decode('ascii')

def tweet_proc(df, text_col='text'):#removing handles, urls, hashtags
#always the name of "text_col" should be "text"

    df['orig_text'] = df[text_col]#making a new col named "orig_text"

    # Remove twitter handles
    df[text_col] = df[text_col].apply(lambda x:re.sub('@[^\s]+','',x))

    # Remove URLs
    df[text_col] = df[text_col].apply(lambda x:re.sub(r"http\S+", "", x))

    # Remove emojis
    df[text_col] = df[text_col].apply(de_emojify)

    # Remove hashtags
    df[text_col] = df[text_col].apply(lambda x:re.sub(r'\B#\S+','',x))

    return df[df[text_col]!='']#removing rows with blank tweets
```

Fig 3. Preprocessing Function

3.9. Language Modeling

Each of the steps necessary to create a language model are:

- **Tokenization**:: Convert the text into a list of words (or characters, or substrings, depending on the granularity of your model)
- **Numericalization**:: Make a list of all of the unique words that appear (the vocab), and convert each word into a number, by looking up its index in the vocab
- **Language model data loader creation**:: fastai provides an LMDataLoader class which automatically handles creating a dependent variable that is offset from the independent variable by one token. It also handles some important details, such as how to shuffle the training data in such a way that the dependent and independent variables maintain their structure as required
- **Language model creation**:: We need a special kind of model that does something we haven't seen before: handles input lists which could be arbitrarily big or small. There are a number of ways to do this; in this project we will be using a *recurrent neural network* (RNN) (fastai_NLP, 2021).

3.10. Tokenization

There are three main approaches for tokenization:

- Word-based:: Split a sentence on spaces, as well as applying language-specific rules to try to separate parts of meaning even when there are no spaces (such as turning "don't" into "do n't"). Generally, punctuation marks are also split into separate tokens.
- Subword based:: Split words into smaller parts, based on the most commonly occurring substrings. For instance, "occasion" might be tokenized as "occasion."
- Character-based:: Split a sentence into its individual characters (fastai_NLP, 2021).

The default English word tokenizer for fastai uses a library called *spaCy, as shown in fig 4*. It has a sophisticated rules engine with special rules for URLs, individual special English words, and much more. Rather than directly using SpacyTokenizer, however, WordTokenizer could be used, since that will always point to fastai's current default word tokenizer (which may not necessarily be spaCy, depending when this paper is being read).

Fig 4. Making world tokenizer and applying to a sample sentence

fastai then adds some additional functionality to the tokenization process with the Tokenizer class (Fig 5).

Fig 5. Applying fastai tokenizer to the tokenizer object for extra functionality

After this step, there will be some tokens that start with the characters "xx", which is not a common word prefix in English. These are *special tokens*. For example, the first item in the list, xxbos, is a special token that indicates the start of a new text ("BOS" is a standard NLP

acronym that means "beginning of stream"). By recognizing this start token, the model will be able to learn it needs to "forget" what was said previously and focus on upcoming words

3.11. Numericalization with fastai

Numericalization is the process of mapping tokens to integers. The steps are basically identical to those necessary to create a Category variable, such as the dependent variable of digits in MNIST:

1. Make a list of all possible levels of that categorical variable (the vocab).
2. Replace each level with its index in the vocab.

3.11. Putting Texts into Batches for a Language Model

Text cannot be simply resized to a desired length. Also, our language model is supposed to read text in order, so that it can efficiently predict what the next word is. This means that each new batch should begin precisely where the previous one left off.

3.12. Training a Text Classifier

There are two steps to training a state-of-the-art text classifier using transfer learning: first a language model pre-trained on Wikipedia should be fine-tuned to the corpus of COVID tweets, and then the obtained language model could be used to train a classifier.

4.EXPERIMENTS

4.1. Modeling Environment

Google collab was the platform used in this project to train the model. It provides several benefits in an educational team work on deep learning projects, including free 12GB NVIDIA Tesla K80 GPU to run up to 12 hours continuously. Also the notebooks on the google coab could be shared and synchronized between team members and users are able to update relevant repositories on the github directly from the colab environment (medium_collab).

4.2. Data Cleaning

Data was preprocessed by cleaning and numerizing vocabs. In the cleaning process the following were removed from the original text:

- Handles
- Urls
- Hashtags
- Twitter handles
- Emojis
- Rows with blank tweets

A function was written for the purpose of data cleaning and rendered on the text column of both “tweets“ and “vax_tweets“ dataframes.

4.3. Fine-Tuning the Language Model

To make a language model, customized with our application,the default language model which was pretrained on wikipedia, got fine-tuned on the two concatenated twitter datasets, one was a general tweets dataset (40,000 samples) and the other one was vaccination dataset (212,982 samples). The former dataset included sentiment labels for each sample, but the latter on didnt have it. However for the purpose of training language model sentiment labels were not needed and therefore could be dropped. On the other hand, later in the process of training classifiers only a dataset including sentiment labels will be used.

To input text data to the fastai language model we need to save that in a data loader object, after being preprocessed and in proper format. Tokenization and numericalization were performed automatically by fastai as input data converted into a DataLoader object. Data loader object later will be used in the next step, as an input to a fastai Learner object.

```
dls_lm = TextDataLoaders.from_df(df_lm, text_col='text', is_lm=True, valid_pct=0.1)
```

Fig 6. Saving Text Data Into A Data Loader Object

Below, the data loader object (dls_lm) was fed into the language_model_learner as the input. Also AWD_LSTM was used as a hyperparameter that represents the type of model. The other

hyper parameters was dropout multiplier which is a global multiplier applied to control all dropouts.

```
learn = language_model_learner(dls_lm, AWD_LSTM, drop_mult=0.3, metrics=[accuracy, P
erplexity()]).to_fp16()
```

Fig 7. Making A Language Model Learner Object

4.3.1. Learning Rate

To find the optimum learning rate in fine-tuning of the language model, fastai provides the following functions which was applied to the learn object. Learning rate finder plots lr vs loss relationship for a Learner. The idea is to reduce the amount of guesswork on picking a good starting learning rate. We can see the point where the gradient is the steepest with a red dot on the graph. We can use that point as a first guess for an LR (Fastai learning rate, 2021).

```
learn.lr_find()
```

Fig 8. Learning Rate Finder Method Applied to Learner Object

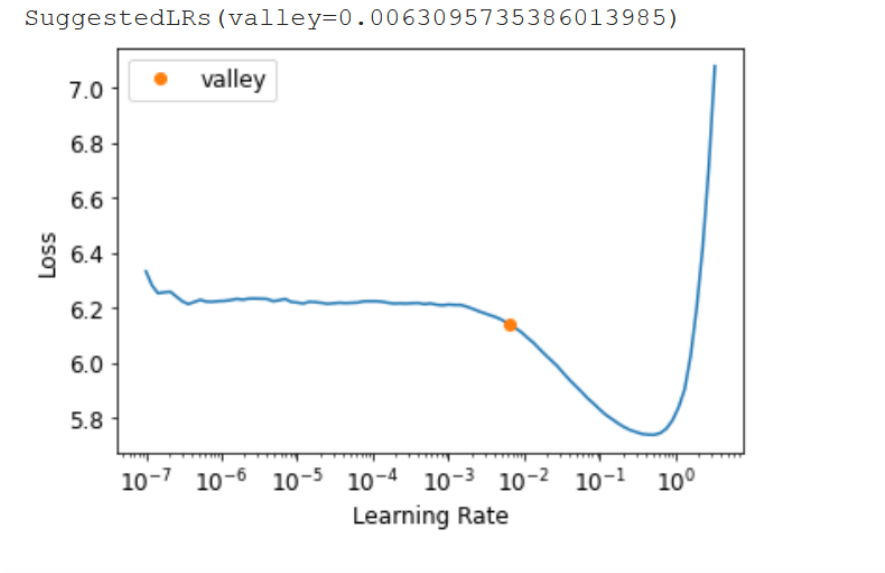

Fig 9. Proposed Optimal Learning Rate And Corresponding Diagram

After trying several values for learning rate (started from suggested one) the lr = 0.3 was selected as the optimum value for training the last layer. The proposed learning rates for the case of training all layers always had much smaller values.

4.3.2. Training Algorithm

For changing the config of the model which is AWD-LSTM, it was tried to change n_hid(size of middle layers) and n_layer(number of layers) instead of just using the predefined config for the language model. 200, 500, 1000 as the number for n_hid were tried for the language model but it resulted a size mismatch errors for the language model. Also we tried 2, 4, 5 as the number for n_layer for the language model but it gave us size mismatch errors. Therefore, we used the default config of AWD-LSTM for the language model.

4.3.3. Hyper-parameter Optimization

For the purpose of hyperparameter optimization, a small part of the whole dataset (40,000 out of 212,983 samples) were used. Using this partial dataset, it was practically possible to try different combinations of hyper parameters and find a rough estimation of their optimum. In the next step, we trained the model on the whole dataset trying a few hyperparameters around the estimated values obtained from the previous step (partial dataset). For example the partial dataset determined optimum dropout multiplier as 0.3, therefore we started trying a couple of different values around this start point for the full dataset. Finally it was found that the optimum dropout multiplier is 0.15 for the full dataset. Models were trained for 10 epochs due to limited permitted google colab GPU accessible time and speed.

Table 1. Hyperparameter Selection For Language Model

Hyper Parameters			Accuracy
Learning Rate (lr)	**Dropout Multiplier (drop_mult)**	**Epochs**	
0.1	0.30	10	0.440
0.0025	0.30	10	0.416
0.03	0.30	10	0.445
0.03	**0.15**	10	**0.456**
0.03	0.50	10	0.463

4.3.4 . Creating the Classifier DataLoaders

For the classifier model we use a data block object to get the data loader, since the data block is more customizable than the data loader. A DataLoaders (note the plural), is a thin class that automatically generates multiple DataLoader (singular) objects based on the rules specified in our DataBlock (Muttoni, 2021).

```
dls_clas = DataBlock(
    blocks = (TextBlock.from_df('text', seq_len=dls_lm.seq_len, vocab=dls_lm.vocab),
CategoryBlock),
    get_x=ColReader('text'),
    get_y=ColReader('sentiment'),
    splitter=RandomSplitter()
).dataloaders(df_clas, bs=64)
```

Fig 10. Creating Classifier Data Loaders Using Data Block

In the above code block, dataloaders(df_clas, bs=64) Builds the DataLoaders using the DataBlock template we just defined, the df_clas DataFrame and a batch size of 64.

As we needed a learner object for language modeling, here we need another learn object for classifier as well and we can make that as below.

```
learn = text_classifier_learner(dls_clas, AWD_LSTM, drop_mult=0.5, metrics=accurac
y).to_fp16()
```

Fig 11. Creating Classifier Model Data Loaders Using Data Block

Finally, we want to load the encoder from the language model we trained earlier, so our classifier uses pre-trained weights.

4.4. Fine-Tuning The Classifier Model

For changing the config of the classifier model which is AWD-LSTM, we tried to change n_hid(size of middle layers) and n_layer(number of layers) instead of just using the predefined config for the classifier model. We tried 200, 500, 1000 as the number for n_hid for the language model but it gave us size mismatch errors for the classifier model. Also we tried 2, 4, 5 as the number for n_layer for the classifier model but it gave us size mismatch errors. Therefore, we used the default config of AWD-LSTM for the classifier model.

Table 2. Hyperparameter Selection For Classifier Model

Hyper Parameters			Accuracy
Learning Rate (lr)	**Dropout Multiplier (drop_mult)**	**Epochs**	
0.006	0.2	10	73.7
0.006	0.5	10	74.89
0.006	0.6	10	74.72
0.006	0.4	10	75.58
0.006	**0.3**	10	76.1
0.01	0.3	10	75.03
0.001	0.3	10	75
0.03	0.3	10	**76.2**

```
learn = learn.load_encoder('finetuned_lm')
```

Fig 12. Creating Classifier Model Learner Object

4.5. Probabilistic Sentiment Classification of unseen tweets

Final classifier was tested on some unseen sentences. As could be seen below, The first sentence, which was a positive one, was predicted correctly with 73% accuracy and second one, which was a negative one, was predicted correctly with 85% accuracy.

First sentence : While the world has been on the wrong side of history this year, hopefully, the biggest vaccination effort we've have

```
learn.predict(" While the world has been on the wrong side of history this year, hopefully, the biggest vaccination effort we've have")
```

Fig 13. Predicting Probabilities of Sentiments on a Sentence

Output:

('positive', tensor(2), tensor([0.0472, 0.2188, 0.7341]))

It means that the first sentence is negative with the probability of 4%

It means that the first sentence is neutral with the probability of 21%

It means that the first sentence is positive with the probability of 73%

Final sentiment

Therefore, because of the fact that the probability of being positive is much higher than the probability of being neutral and negative, the classifier model predicts the sentiment of the first sentence as positive.

Second sentence : I have just been in a bad mood for 40 years!

```
learn.predict("I have just been in a bad mood for 40 years!")
```

Fig 14. Predicting Probabilities of Sentiments on a Sentence

Output :

('negative', tensor(0), tensor([0.8558, 0.1184, 0.0258]))

It means that the second sentence is negative with the probability of 85 %

It means that the second sentence is neutral with the probability of 11 %

It means that the second sentence is positive with the probability of 2 %

Final sentiment

Therefore, because of the fact that the probability of being negative is much higher than the probability of being neutral and positive, the classifier model predicts the sentiment of the second sentence as negative .

4.6. Visual Analysis of Vaccination tweets

Also, we test the final classifier model on the vaccination tweets dataset to predict the sentiment of sentences and as we can see in this diagram, sentiment of most of the sentences are neutral and some sentences have positive and negative sentiments.

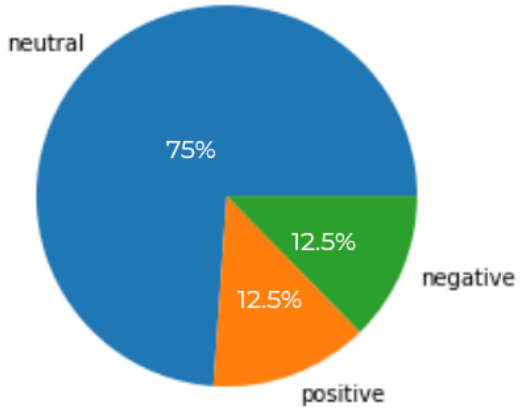

Fig 15. Sentiment Distribution of Vaccination Tweets

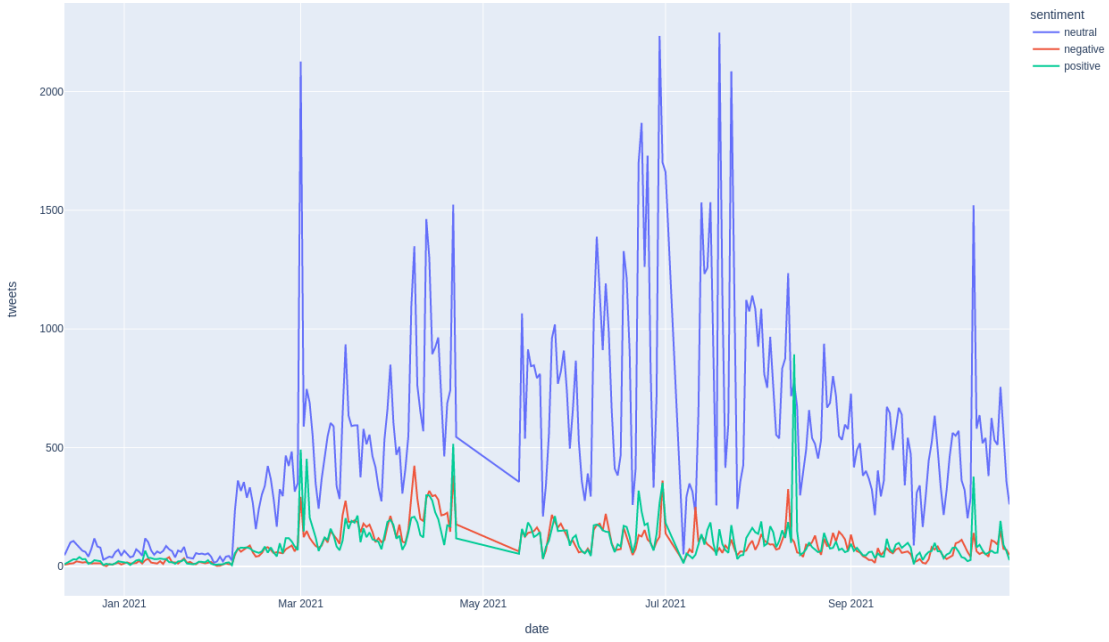

Fig 16. Timeline showing sentiments of Tweets About Covid-19 Vaccines

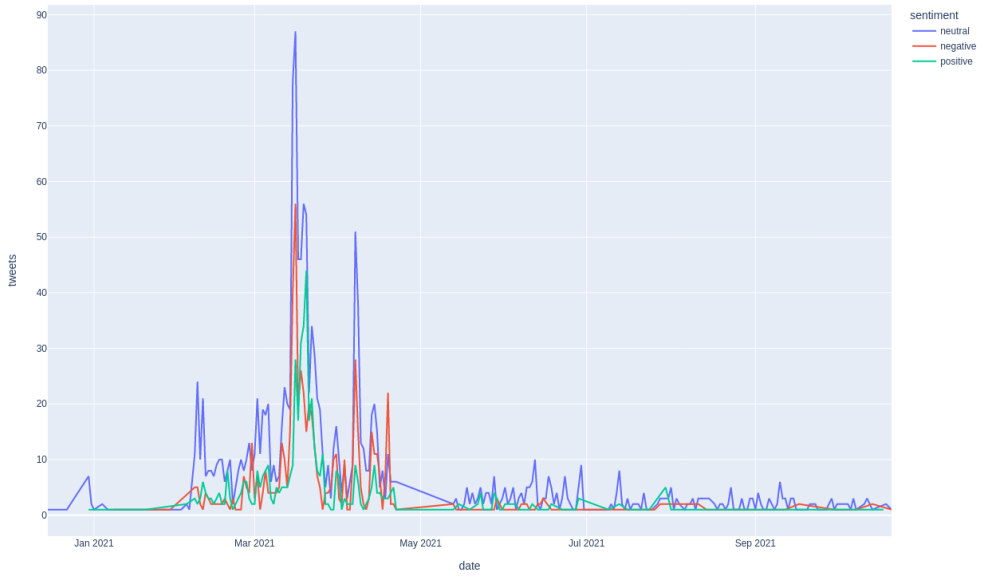

Fig 17. Timeline showing sentiments of Tweets About The Oxford/AstraZeneca Vaccine

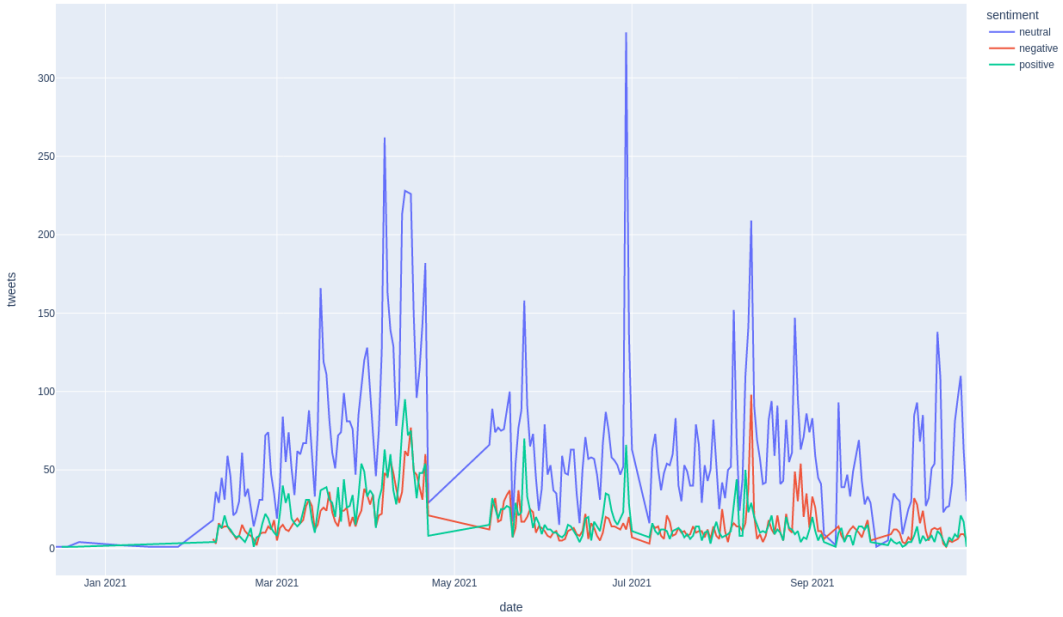

Fig 18. Timeline showing sentiments of Tweets About The Moderna Vaccine

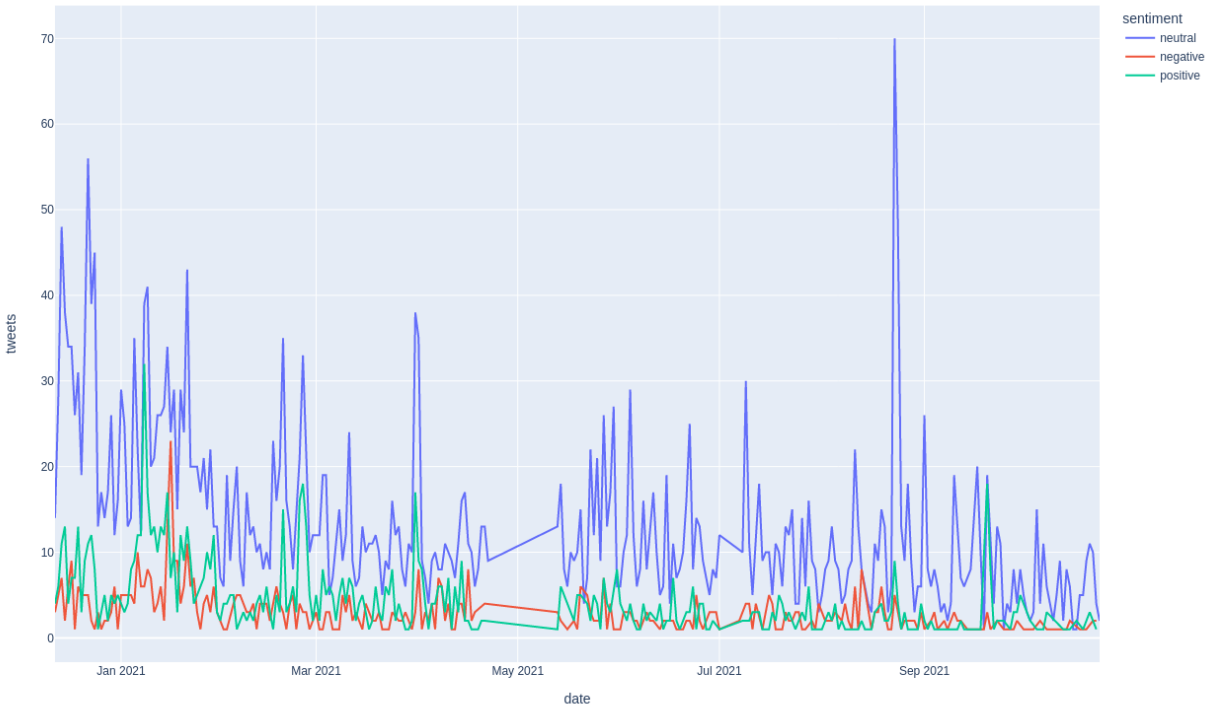

Fig 19. Timeline showing sentiments of Tweets About The Fizer/BioNTechVaccine

V. CONCLUSIONS

A tweet sentiment analyzer was modeled using a deep neural network based NLP method. fastai was applied as a deep learning framework in this project. Dropout Multiplier, learning rate and selected model were tweaked as the hyper parameters of the language and classifier models. Results showed that the highest accuracies for language model and classifier could be achieved when hyper parameter are selected as below:

Table 3. Optimum Hyperprameters of Language and Classifier Models

Function	Learning Rate	Dropout Multiplier	Model Algorithm
Language model	0.03	0.15	AWD-LSTM
Classifier	0.03	0.3	AWD-LSTM

Regarding the accuracy values, 76.2 % was the highest accuracy that could be achieved for predicting sentiment of tweets validation dataset. This accuracy could depend on the analyzing text as well. The more preprocessing a raw text needs and complicated language structures, the lower the achieved accuracy could be.

Finally inference was conducted on the vaccination tweet dataset to predict their sentiment labels. Using the predicted sentiments and other provided data such as tweet location and date, informative diagrams were drawn.

6. REFERENCES

aidancoco, retrieved Dec 19 2021 from:
https://aidancoco.medium.com/optimizing-the-learning-rate-of-your-neural-networks-32e38addd8a3

fastai, retrieved Dec 19 2021 from:
https://docs.fast.ai/

Fastai learning rate, retrieved Dec 19 2021 from:
https://fastai1.fast.ai/callbacks.lr_finder.html

fastai_NLP, retrieved Dec 19 2021 from:
https://github.com/fastai/fastbook/blob/master/10_nlp.ipynb

GitHub,fastai, retrieved Dec 19 2021 from:
https://github.com/fastai/fastbook/blob/master/10_nlp.ipynb

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Kaggle, retrieved Dec 19 2021 from:
https://www.kaggle.com/gpreda/all-covid19-vaccines-tweets/code?datasetId=1158238&sortBy=voteCount

Kaur, H., Ahsaan, S. U., Alankar, B., & Chang, V. (2021). A proposed sentiment analysis deep learning algorithm for analyzing covid-19 tweets. In Information Systems Frontiers (pp. 1–13)

Khan, R., Shrivastava, P., Kapoor, A., Tiwari, A., & Mittal, A. (2020). Social media analysis with AI: Sentiment analysis techniques for the analysis of twitter covid-19 data. Journal of Critical Review, 7(9), 2761–2774.

Max, retrieved Dec 19 2021 from:

https://www.kaggle.com/maxjon/complete-tweet-sentiment-extraction-data

Medium, retrieved Dec 19 2021 from: https://medium.com/analytics-vidhya/evolution-of-nlp-part-3-transfer-learning-using-ulmfit-267d0a73421e

Medium_colab, retrieved Dec 19 2021 from: https://medium.com/dataman-in-ai/start-using-google-colab-free-gpu-7968acb7ef92

Muttoni, Medium_colab, retrieved Dec 19 2021 from: https://muttoni.github.io/blog/machine-learning/fastai/2020/12/26/datablocks-vs-dataloaders.html

Rahman, M., & Islam, M. N. (2022). Exploring the performance of ensemble machine learning classifiers for sentiment analysis of covid-19 tweets. In *Sentimental Analysis and Deep Learning* (pp. 383-396). Springer, Singapore.

towardsdatascience, retrieved Dec 19 2021 from: https://towardsdatascience.com/multi-task-learning-in-language-model-for-text-classification-c3acc1fedd89

Substack, retrieved Dec 19 2021 from: https://pakodas.substack.com/p/3-ways-to-make-new-language-models-f3642e3a4816

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

Villavicencio, C., Macrohon, J. J., Inbaraj, X. A., Jeng, J. H., & Hsieh, J. G. (2021). Twitter sentiment analysis towards covid-19 vaccines in the philippines using naïve bayes. Information, 12(5), 204.

Nabi, J. (2019, February 6). *Machine learning-text classification, language modelling using fast.ai*.from:

https://towardsdatascience.com/machine-learning-text-classification-language-modelling-using-fast-ai-b1b334f2872d

Rao, P. (2021, December 7). *Transfer Learning in NLP for Tweet Stance Classification*. Medium. https://towardsdatascience.com/transfer-learning-in-nlp-for-tweet-stance-classification-8ab014da8dde

Ridhwan, K. M., & Hargreaves, C. A. (2021). Leveraging Twitter Data to Understand Public Sentiment for the COVID-19 Outbreak in Singapore. International Journal of Information Management Data Insights, 100021.

Howard, & Gugger. (2020, February 13). *fastai—A Layered API for Deep Learning*.Fast.Ai. https://www.fast.ai/2020/02/13/fastai-A-Layered-API-for-Deep-Learning/#learner