Real-Time Twitter Emotion Classification by Topic

Christian Norrie
Filotas Theodosiou

Modelling Public Opinion

This project was inspired through posing the question of ‘is it possible to see how groups of people feel about a topic based on what they are saying online?’

Every day, Twitter users publish a staggering amount of publically available data - around 500 million tweets.

Our goal was to see if this data can be used to model public opinion.

For example, is it possible to query a system with a topic (such as ‘covid-19’) and see if Twitter users are feeling fearful, sad, or angry about this topic?

Modelling Public Opinion

Can this system be extended further to model public opinion by geographic location and date?

For example, given the topic of ‘Donald Trump’, it might seem that Americans are angry about this, while Canadians might be sad about this.

Could it be said that ‘64% of Americans in Arizona are angry about Donald Trump, 20% in DC are happy, and 16% in Minnesota are fearful? Determining some public opinion statistics such as these were the ultimately goal.

Related Research

Sentiment Analysis is the domain of modelling public opinion based on 'positive' and 'negative' sentiment.

For example, if a company wants to evaluate if their marketing is effective, they can query social media and run data through their system.

Their system might suggest that 'users feel 26% negative about this product', which can then inform their further marketing.

Emotion detection gives a deeper understanding, as for example negativity can be either anger, fear or sadness, or positivity can be either happiness, relief, or excitement.

Defining the Problem

To begin, we have to quantify what exactly we are trying to do.

We want to determine how people feel, in terms of emotion, about any topic being discussed on Twitter.

Four distinct classes of emotions were defined.[1]

Happiness	Sadness	Fear	Anger

[1] Shaver, Philip (1987) Emotion Knowledge: Further Exploration of a Prototype Approach. Journal of Personality and Social Psychology

Getting Data

Due to Twitter restrictions, there were not any relevant or suitably large datasets. As a result, we queried our own from the Twitter developer API.

For a two-week window, we queried tweets that met certain criteria and automatically assigned them one of the four previously defined classes.

The classification by emotion was done automatically based on hashtags - if a user ended a tweet with ‘#angry’, it was classified as angry.

Getting the Data

For every emotion class, a list of synonyms and emotional words that describe each particular class was created.

If a tweet contained one of such words as a hashtag, we kept it.

Some examples of emotional words:

- Anger -> #mad, #hate , #rage , #annoying,
- Fear -> #terror , #fright, #nervous, #worried, #worrying
- Happy -> #joy , #exciting, #thriled, #glad , #delight
- Sadness -> #Sad , #depressed , #heartbroken, #dissapointing

Evaluating the Dataset

As some tweets might contain a hashtag, but not clearly indicate their emotion class, a set of heuristics was use to only keep relevant tweets [2]

- Keep only tweets with the emotional hashtag at the end of the tweet
- Remove tweets with less than 3 words and over 5 hashtags
- Remove non-English tweets, retweets and replies.

Despite the length of time we queried data for, only around 10000 tweets per class were collected.

Class	**Anger**	**Fear**	**Happiness**	**Sadness**
Number of Tweets	**11181**	**9536**	**14296**	**13802**

[2] Wang, W., Chen, L., Thirunarayan, K., Sheth, A., (2012). Harnessing Twitter "Big Data" for Automatic Emotion Identification. Retrieved March 6, 2020, from https://www.researchgate.net/publication/258762213_Harnessing_Twitter_'Big_Data'_for_Automatic_Emotion_Identification

Model Selection

Before talking about how the data should be cleaned, it is important to figure out what we are going to do with the data.

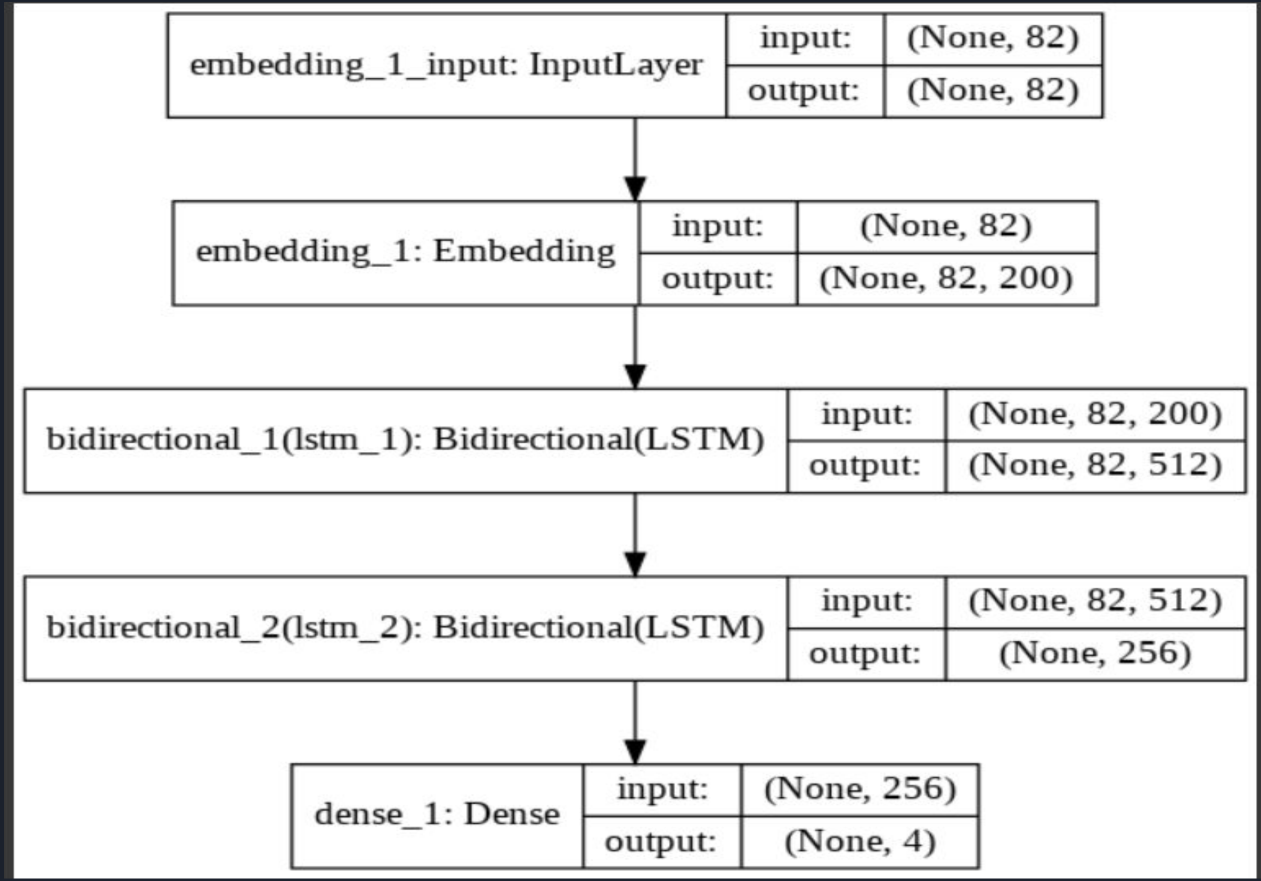
Our approach was to train machine learning models on this dataset which could then classify future datasets of tweets.

There are a few different approaches for NLP problems. Our choice was to train LSTM (Long Short-Term Memory) and bidirectional LSTM models.

For our word Embedding layer we utilized the pre-trained GloVe as it was trained on millions of twitter datapoints.

Model Selection

Input is transformed into word embeddings, then run through LSTM layer(s). Each output node corresponds to the probability that tweet belongs to that emotional class. The highest probability class is then selected as the classification.



Data Cleaning

Data needed to be cleaned extensively before training our models.

- Lowercase words and correct their spelling,remove most pancuations
- Change all Urls to <Url> and Usernames to <User>
- Remove emotional hashtag and segment merged words
- Fix elongated words and convert emojis and smileys into words
- Converted contractions to their complete form(ie. I cant-> I can not)

Below is an example of a tweet that results from this data cleaning pipeline:

```
“the sun is shining and the skys are blue . its going to
be a good day . sun_with_face sun grinning_face good day
its friday”
```

Results

The dataset was split into train/test/validation, and 8 models were trained on it.

We used similar hyperparameters for every model and fine-tuned the best-performing model for integration with streaming and visualization.

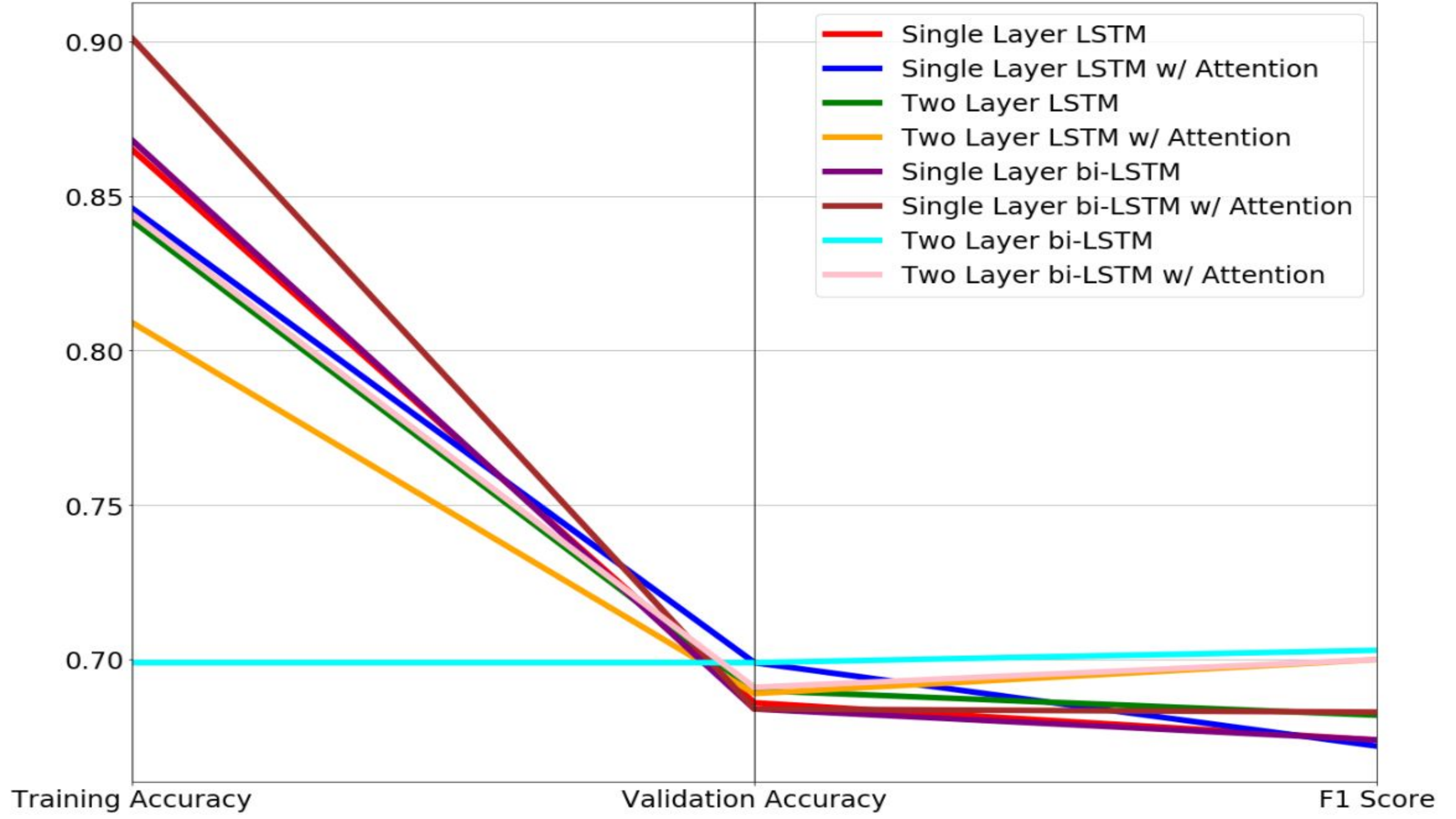
We would have liked to train more or to modify their parameters for each model, but each model took nearly a day to train locally

Model	LSTM Layer Type	Number of LSTM Layers	Number of Nodes Per Layer	Optimizer	Loss Function	Modifications
Single layer LSTM	Single LSTM	1	256	adam	Categorical Cross Entropy	
Single layer LSTM w/ attention	Single LSTM	1	256	adam	Categorical Cross Entropy	Attention Mechanism
Stacked LSTM	Single LSTM	2	1. 256 2. 128	adam	Categorical Cross Entropy	
Stacked LSTM w/ attention	Single LSTM	2	1. 256 2. 128	adam	Categorical Cross Entropy	Attention Mechanism
Single layer bi-LSTM	Bidirectional LSTM	1	256 (each way)	adam	Categorical Cross Entropy	
Single layer bi-LSTM w/ attention	Bidirectional LSTM	1	256 (each way)	adam	Categorical Cross Entropy	Attention Mechanism
Stacked bi-LSTM	Bidirectional LSTM	2	1. 256 2. 128 (each way)	adam	Categorical Cross Entropy	
Stacked bi-LSTM w/ attention	Bidirectional LSTM	2	1. 256 2. 128 (each way)	adam	Categorical Cross Entropy	Attention mechanism

Results

Accuracy and F1 were the evaluation methods utilized. Both training set and validation set accuracy were recorded to gauge overfitting.

Most models had a training set accuracy between 0.8 and 0.9, and a validation accuracy of 0.68. Our best model (two layer bidirectional LSTM) had a training and validation accuracy of 0.699, which, while a nice coincidence that the numbers are identical, also implies that the model did not overfit and had a ‘reasonable’ accuracy score. It also had the highest F1 Score of 0.700.

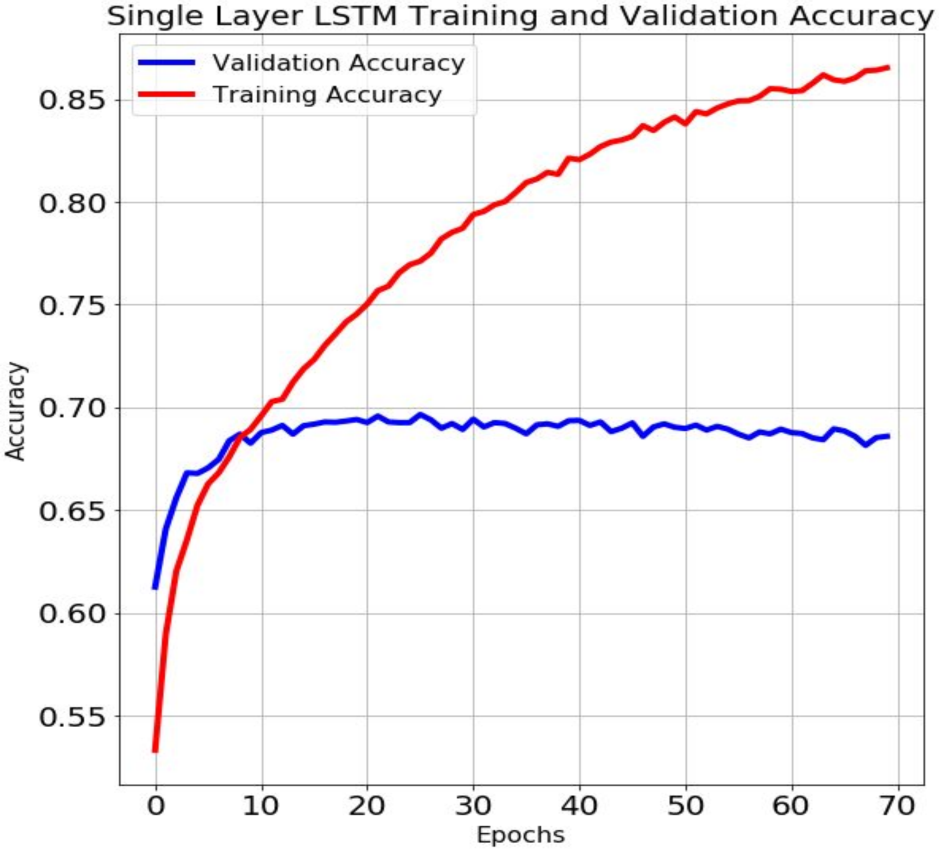

0.90
0.85
0.80
0.75
0.70
Training Accuracy
Validation Accuracy
F1 Score
Single Layer LSTM
Single Layer LSTM w/ Attention
Two Layer LSTM
Two Layer LSTM w/ Attention
Single Layer bi-LSTM
Single Layer bi-LSTM w/ Attention
Two Layer bi-LSTM
Two Layer bi-LSTM w/ Attention

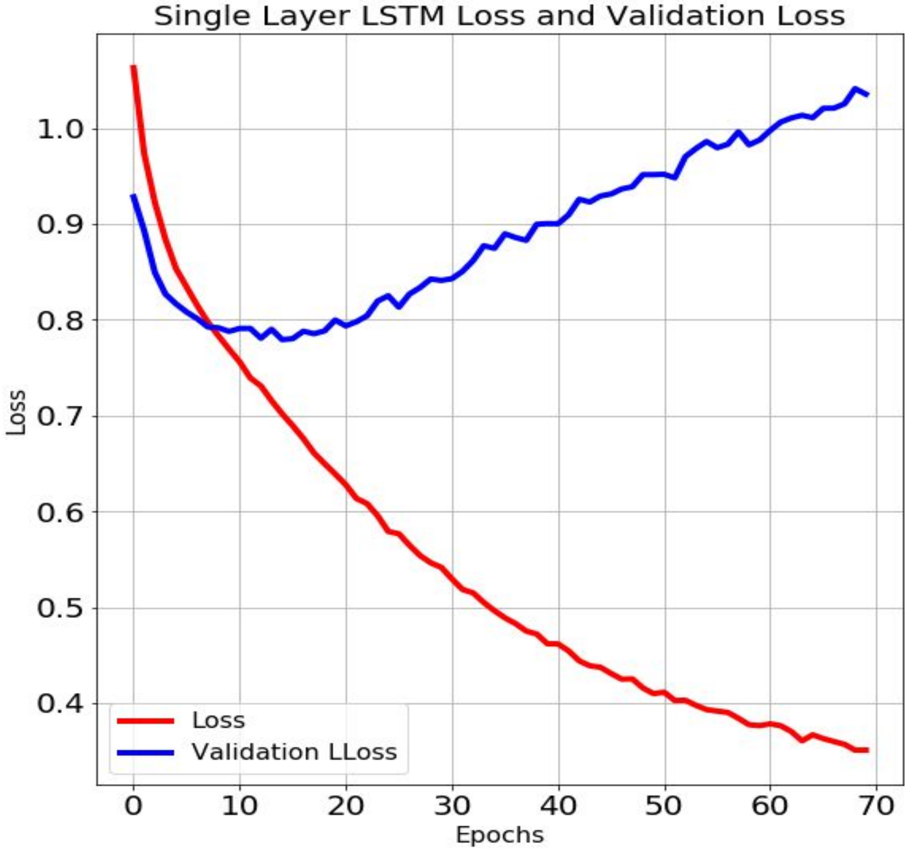
Results

Confusion matrices were also recorded.

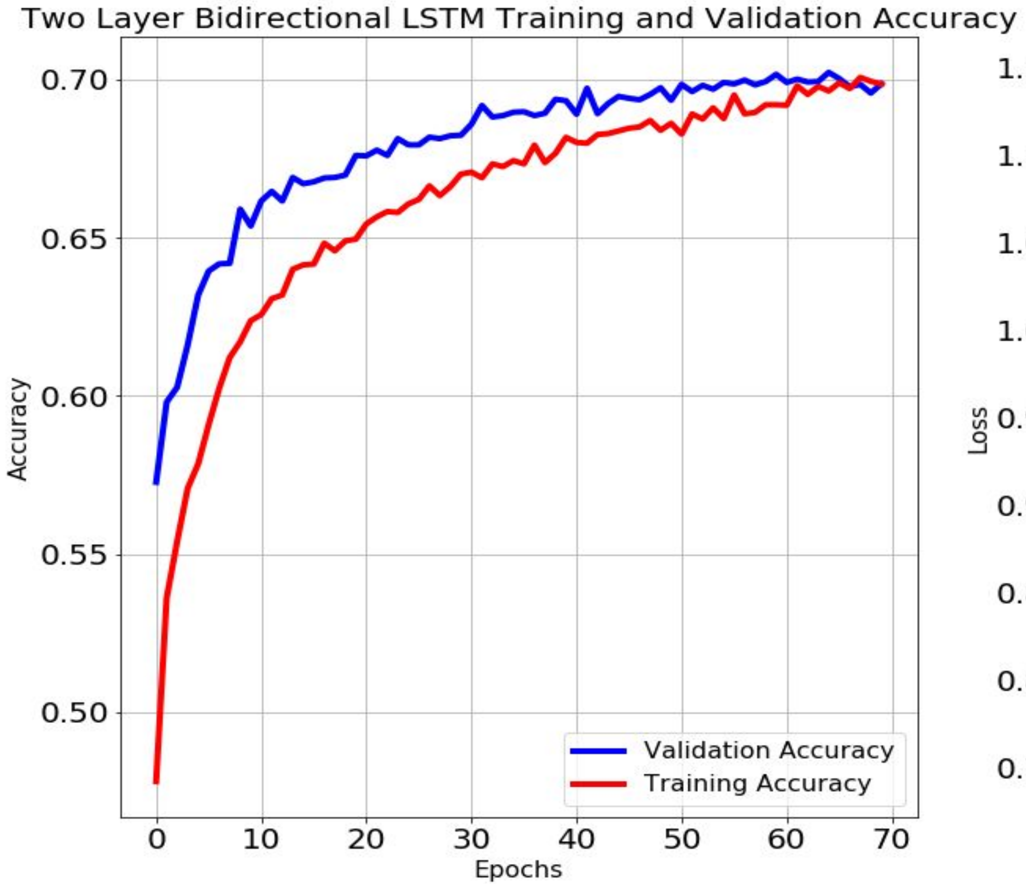
Stacked bi-LSTM				
	Anger	**Fear**	**Happy**	**Sad**
Anger	**0.60**	0.09	0.06	0.25
Fear	0.12	**0.60**	0.07	0.20
Happy	0.03	0.04	**0.85**	0.09
Sad	0.14	0.08	0.08	**0.70**

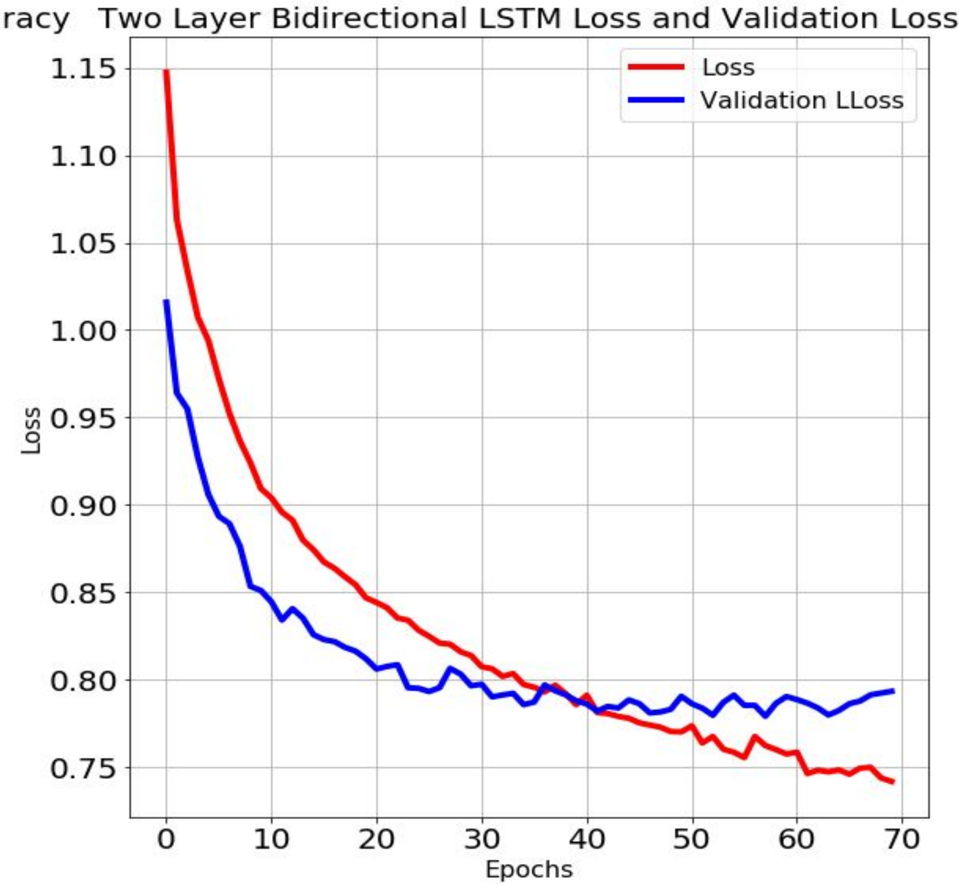
Single Layer LSTM (worst model)





Two Layer bi-LSTM (best model)

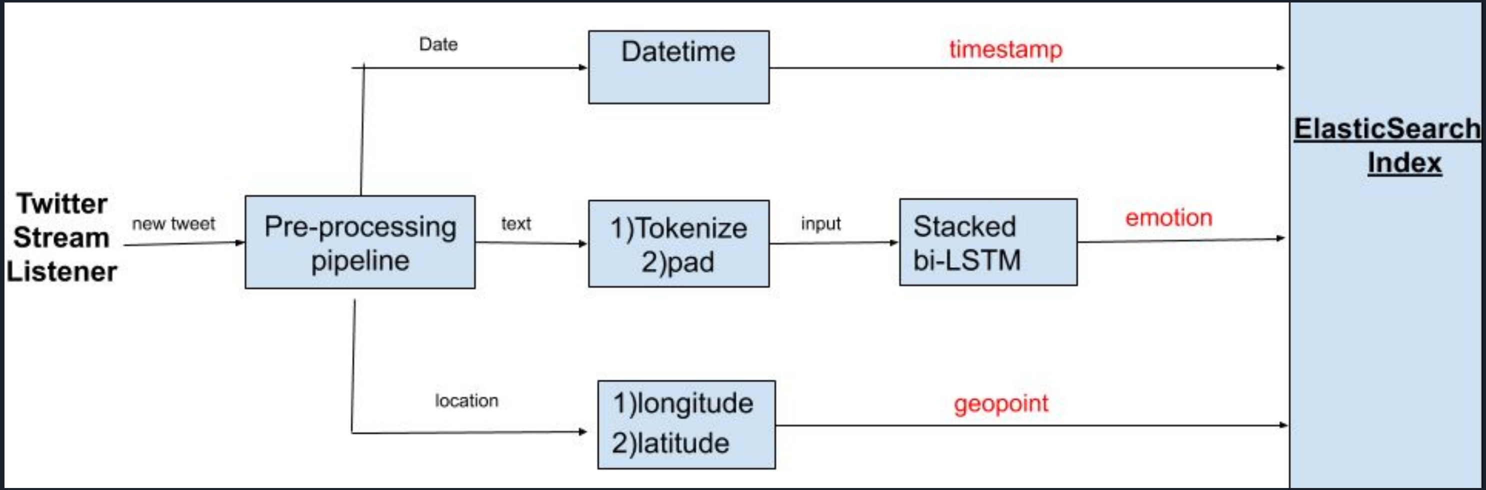




Streaming

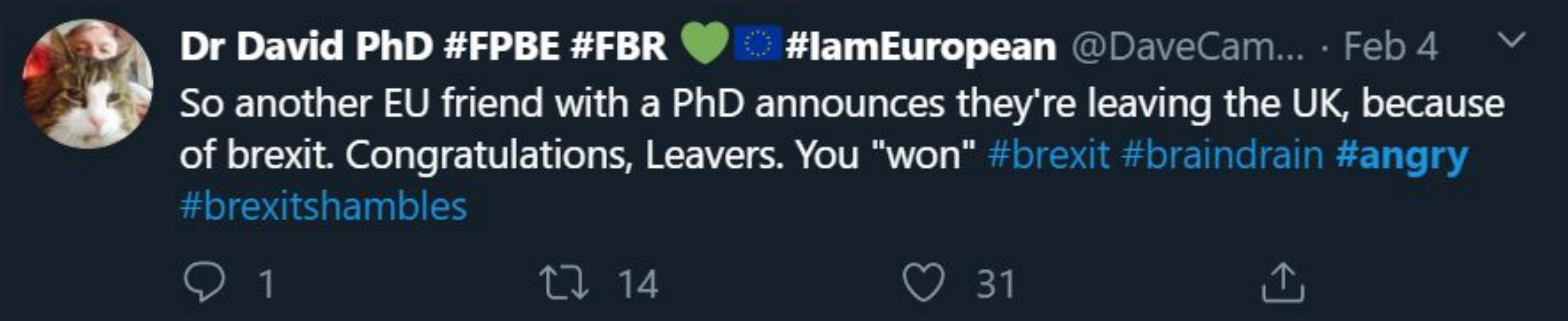
With this in mind, we were able to take our highest performing model and as a proof-of-concept, made some tweaks to it to integrate it with a streaming pipeline and save results into an ElasticSearch Index.

The following pipeline was used to query tweets containing a given keyword.



Classification of Emotion

With all this in mind, our model could ideally take tweets such as below...



....the model then classifies these tweets as belonging to one of the four classes.

Analysis by Topic

When integrated with the streaming pipeline, the model can then...

1. Query tweets based on a hashtag (such as ‘#covid19’, ‘#trump’, ‘#NBA’, or ‘#NHL’, any topic being discussed on Twitter)
2. Query geolocation and datetime data from these tweets, then aggregate statistics about these topics.

A Dashboard on Kibana was created where results could be visualized and analyzed even further

Analysis by Topic

Travis Fong @TravisFong · Mar 3

$5 million dollars saved to close 20 parks, and privatize 164 parks.

$100 million given away ("loans") to belligerent oil companies after a $4.7 billion tax give away.

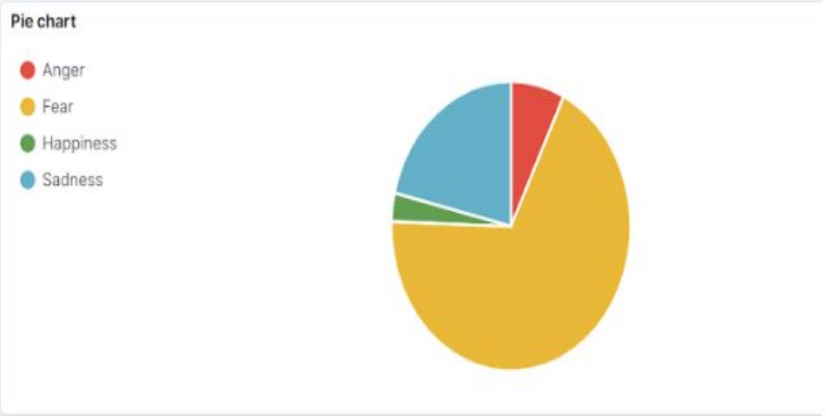
Who on earth thinks this government is fiscally conservative?

THEY ARE LOOTING ALBERTA

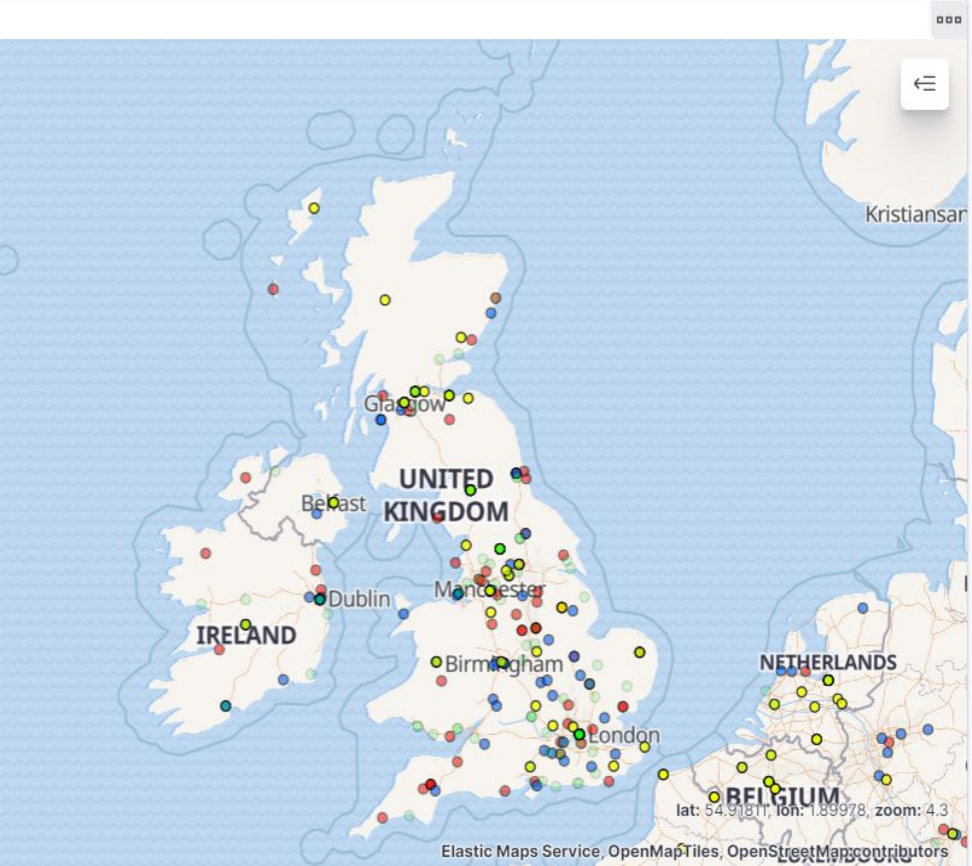
#AbLeg

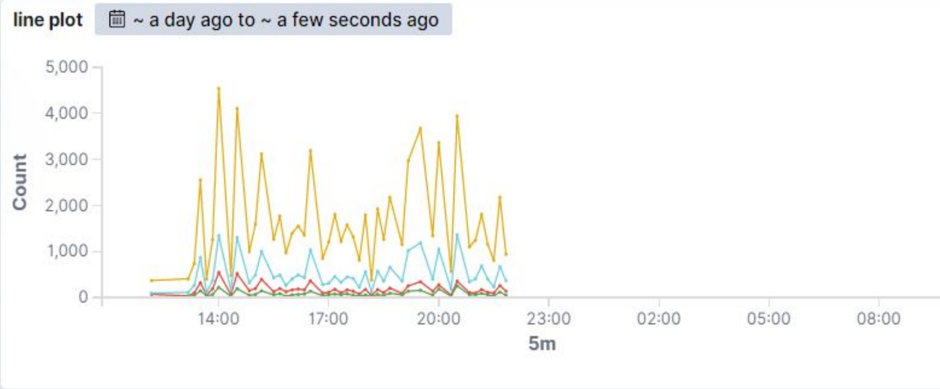
> **Janis Irwin** @JanisIrwin · Mar 3
>
> The UCP will close 20 parks and privatize 164 parks. This saves very little money, but very much puts at risk our natural heritage. How can @jkenney find $30 million for a laughable war room, but he can't find $5 million to protect our parks? globalnews.ca/news/6623820/a... #ableg

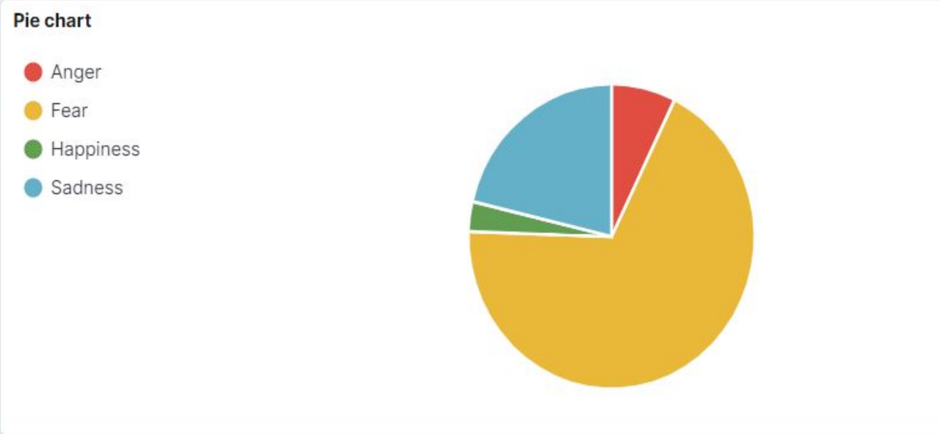
Streaming Results on #covid19



Streaming Results







Discussion

Results were much better than expected, however, there were still problems.

1. Manually analyzing the training dataset indicated some problems with automatic classification, there was confusion between the ‘negative’ emotion classes.
2. Due to the above problem, the model was able to classify ‘happy’ tweets with greater accuracy than the other three categories.
3. There was ambiguity when a tweet could realistically belong to more than one category (ie: both angry and fearful).
4. There was extremely limited data availability and very tight time constraints. This project would intuitively scale much better to millions of data points with less hardware limitations.

Discussion

5. Different choices could have been made in terms of data cleaning. More research could be done in this area to determine the best way to do this.

6. There are many more architectures and models beyond LSTMs and bi-LSTMs that can be used for this.

7. K-fold validation really should have been utilized, but again, there were hardware constraints that made this impossible.