FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Sentiment Analysis of Tweets and their Potential Replies

Seminar Deep Learning (WS19/20)

S. Tayebi Arasteh, M. Monajem, S. Sitaula

February, 2020

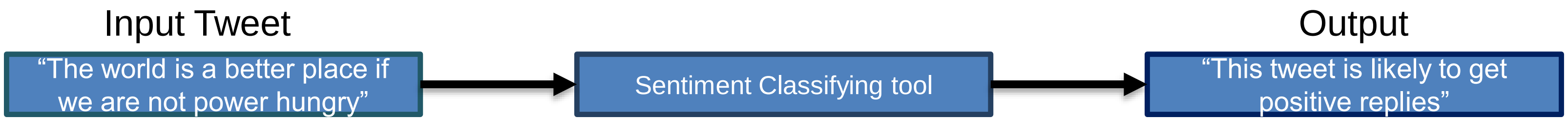
What is sentiment Analysis?

It is a machine learning technique that automatically analyses and detects the sentiment of text.

Project Overview

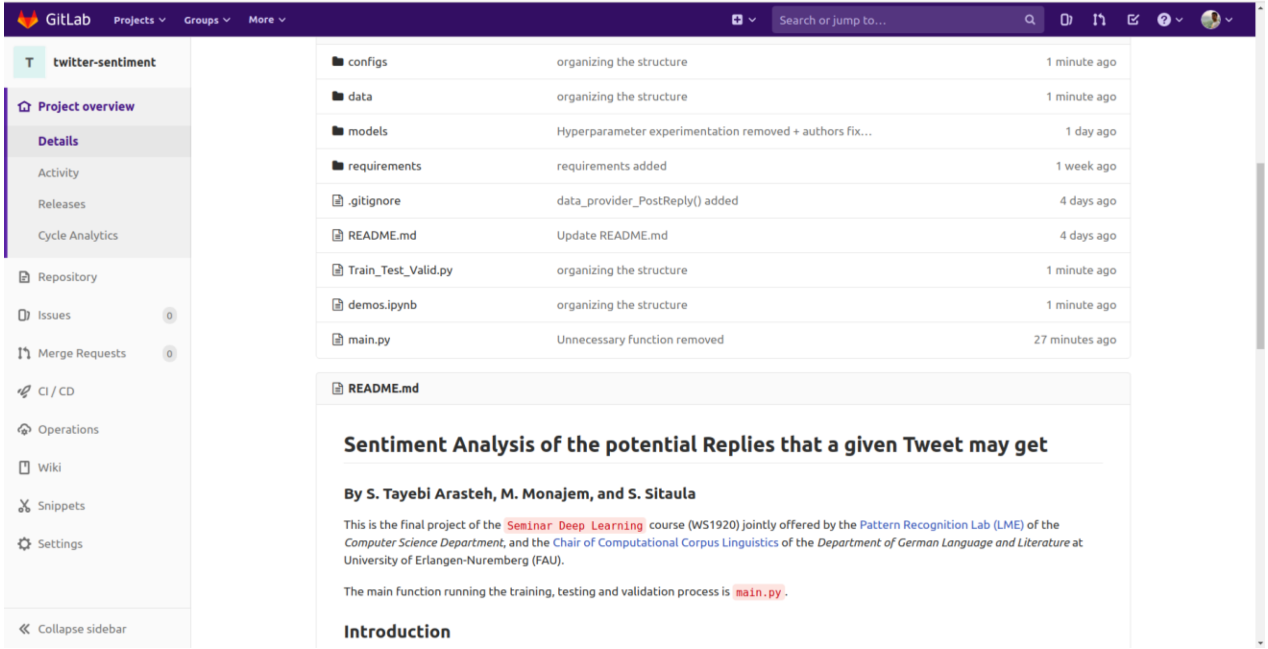
Sentiment Analysis and Post-Replies

- The aim of this project is to build a sentiment classifying tool which, given a tweet predicts how likely it gets positive, negative or neutral replies. In other words, it is a tool to predict the sentiment of replies of a tweet is most likely to get.
- Here we do Message-level sentiment analysis, i.e. we classify the sentiment of the whole given message (tweet or reply).



Project Overview

- The full project is available on FAU Computer Science Department’s GitLab under this link.



1. Overview
2. **Approach**
3. Results
4. Interactive Sentiment Analysis
5. Future Work

Approach to the challange

This project can be divided into two consecutive parts to reach our goal.

1. Creating a Sentiment Classifying tool for tweets

2. Sentiment Predictor for Tweet-Replies

Project in Nutshell

1. Train a Sentiment Analyzer
2. Predict the sentiments of replies to a tweet
3. Use this prediction to label a tweet to create dataset
4. Use the dataset to train a reply-predicting model

	Replies	sentiments
Tweet	Reply_01	Positive
	Reply_02	Neutral
	………….	………….
	Reply_500	Negative

	Sentiments of replies	label
Tweet	Positive	**Positive** (argmax value of sentiments of replies)
	Positive	
	Positive	
	Neutral	
	Negative	
	Neutral	
	Positive	

Part 1:

Sentiment Analysis of the labeled tweets

Part 1: Overview

- Developed in *Python 3.7* using *PyTorch 1.3.1* (including *TorchText 0.5.0*).
- **Supervised** deep learning method.
- Message-level Sentiment Analysis of tweets, based on the Subtask B of Task 10 of the SemEval 2015 challenge.
- **Preliminary goal**: To beat the state-of-the-art of the corresponding task of the SemEval.
- **Final goal**: To tackle the unsupervised nature of the part 2 of the project as supervised.

Part 1: Dataset

Background of the Datasets

- *SemEval* (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems; it evolved from the Senseval word sense evaluation series. (Wikipedia)

- The datasets contains of annotated tweets with a variety of topics which are politics, social issues, products, movies, events etc.

Part 1: Dataset

Training data

- Development dataset – SemEval 2013 (same for SemEval 2013-15)
- Training dataset – SemEval 2013 (same for SemEval 2013-15)

Part 1: Dataset

Training data

- Development dataset – SemEval 2013 (same for SemEval 2013-15)
- Training dataset – SemEval 2013 (same for SemEval 2013-15)

Test data

- Test gold dataset SemEval 2014
- Test gold dataset SemEval 2015

Part 1: Dataset

Preprocessing

- The maximum vocabulary size of 25000
  - Everything else will be ragarded as the unkown token UNK_IDX.
  - And one extra padding token PAD_IDX makes it 25002 in total.
  - Only on the training set and not the validation set.
  - Words in the vocabulary initialized with Gaussian distribution.

Part 1: Dataset

Preprocessing

- The maximum vocabulary size of 25000
  - Everything else will be ragarded as the unkown token UNK_IDX.
  - And one extra padding token PAD_IDX makes it 25002 in total.
  - Only on the training set and not the validation set.
  - Words in the vocabulary initialized with Gaussian distribution.

- Tokenizer using SpaCy with *Pack-Padded-Sequence*.

Part 1: Dataset

Preprocessing

- The maximum vocabulary size of 25000
    - Everything else will be ragarded as the unkown token UNK_IDX.
    - And one extra padding token PAD_IDX makes it 25002 in total.
    - Only on the training set and not the validation set.
    - Words in the vocabulary initialized with Gaussian distribution.

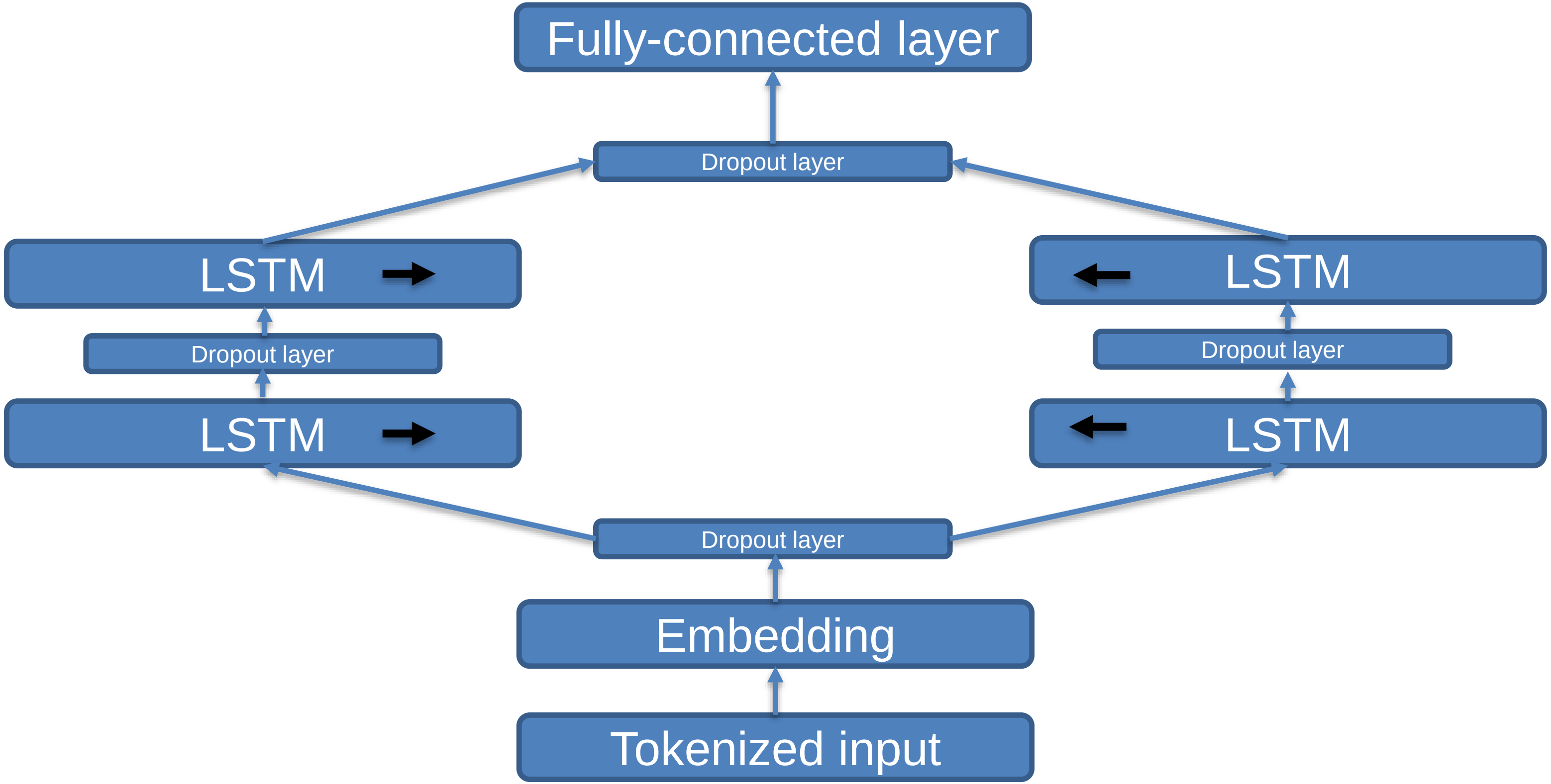
- Tokenizer using SpaCy with *Pack-Padded-Sequence*.
- 16,146 labeled training samples in total.
- 11,377 labeled test samples in total.

Part 1: Model Architecture

Bi-directional Long-Short Term Memory Units (BiLSTM)

- Embedding layer
  - Using the pre-trained ***glove.6B.100d*** (100-dimensional **GloVe** model on 6 billion data).
- 2 bi-directional layers
- Dropout layer
- Fully connected layer

Part 1: Model Architecture



Part 1: Training Parameters

- 20% of the data as validation.

Part 1: Training Parameters

- 20% of the data as validation.

- Loss functions: Cross Entropy loss
- Optimizer: ADAM, with a learning rate of 5e-5 with weight decay of 1e-5
- 4,811,883 trainable parameters

Part 1: Training Parameters

- 20% of the data as validation.

- Loss functions: Cross Entropy loss
- Optimizer: ADAM, with a learning rate of 5e-5 with weight decay of 1e-5
- 4,811,883 trainable parameters

- Hidden and cell dimension of the LSTM: 256
- Embedding dimmension as mentioned before: 100

Part 1: Training Parameters

- 20% of the data as validation.

- Loss functions: Cross Entropy loss
- Optimizer: ADAM, with a learning rate of 5e-5 with weight decay of 1e-5
- 4,811,883 trainable parameters

- Hidden and cell dimension of the LSTM: 256
- Embedding dimmension as mentioned before: 100

- Trained on Nvidia GeForce 940MX (not so powerful GPU).
- Training duration: 57 minutes and 08 seconds with batch size of 32 for 60 epochs.

Part 1: Testing Results

- Accuracy of 72.17% on the test data.

Part 1: Testing Results

- Accuracy of 72.17% on the test data.

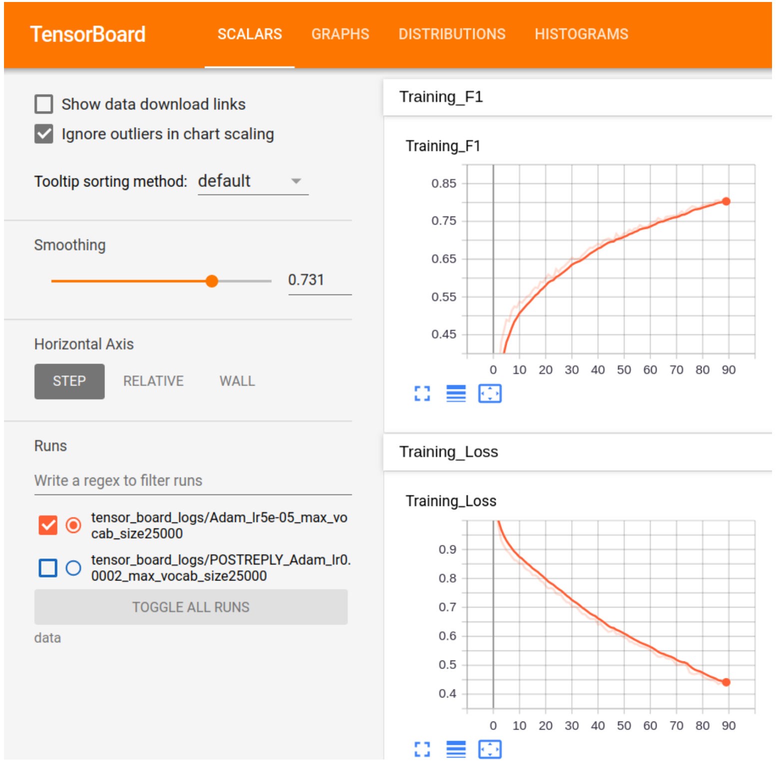
- But our main evaluation metric here is F1 score
  - F1 score is the official mettric of the SemEval 2015, Task 10.

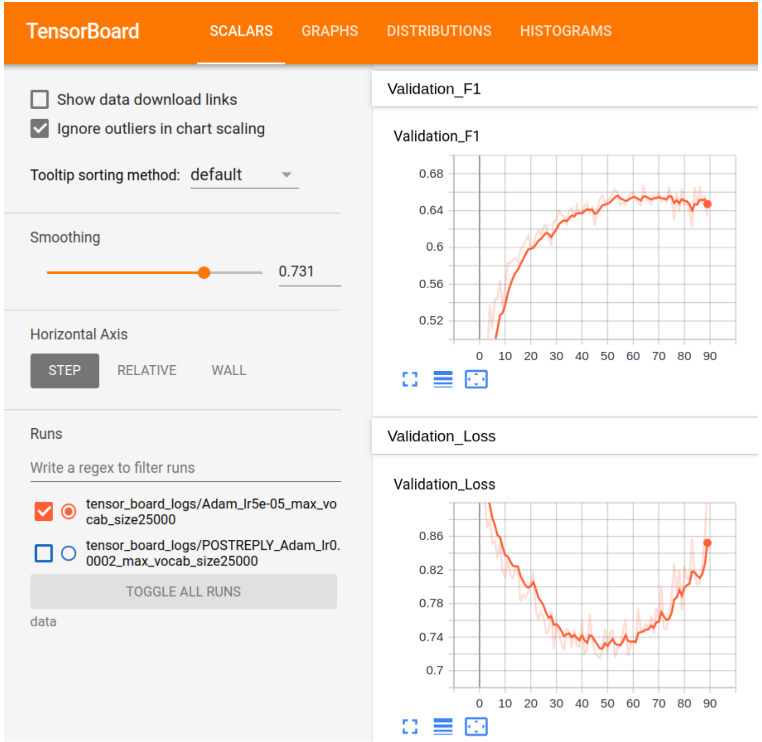
- (macro) **F1 score** of **0.697** on the test data,

Part 1: Testing Results

- Accuracy of 72.17% on the test data.
- But our main evaluation metric here is F1 score
    - F1 score is the official mettric of the SemEval 2015, Task 10.
- (macro) **F1 score** of **0.697** on the test data,
    - **The sate-of-the-art (0.685) is beaten!**

Part 1: Learning Curves (extracted from the TensorBoard)





Part 2:

Sentiment Analysis of the Unlabeled Tweet-Replies

Part 2: Overview

- Developed in *Python 3.7* using *PyTorch 1.3.1* (including *TorchText 0.5.0*).
- **Originally Unsupervised** problem.
- **Idea**: To leverage the model in the part 1, to tackle the unsupervised nature of the problem.

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.
2. If a tweet has multiple replies, regard each reply as a separate data and repeat the tweet.

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.
2. If a tweet has multiple replies, regard each reply as a separate data and repeat the tweet.
3. First ignore the tweets and predict the sentiment of each reply using the model from the part 1.

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.
2. If a tweet has multiple replies, regard each reply as a separate data and repeat the tweet.
3. First ignore the tweets and predict the sentiment of each reply using the model from the part 1.
4. For the replies corresponding to the same tweet, choose the label which has the maximum occurance and assign it as the tweet‘s label and then ignore the replies.

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.
2. If a tweet has multiple replies, regard each reply as a separate data and repeat the tweet.
3. First ignore the tweets and predict the sentiment of each reply using the model from the part 1.
4. For the replies corresponding to the same tweet, choose the label which has the maximum occurance and assign it as the tweet‘s label and then ignore the replies.
5. Now we have a training set of some tweets with their labels, SUPERVISED!

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.
2. If a tweet has multiple replies, regard each reply as a separate data and repeat the tweet.
3. First ignore the tweets and predict the sentiment of each reply using the model from the part 1.
4. For the replies corresponding to the same tweet, choose the label which has the maximum occurance and assign it as the tweet‘s label and then ignore the replies.
5. Now we have a training set of some tweets with their labels, SUPERVISED!
6. Train another model with this data.

Part 2: Strategy

1. Extract tweets from Twitter with their corresponding replies.
2. If a tweet has multiple replies, regard each reply as a separate data and repeat the tweet.
3. First ignore the tweets and predict the sentiment of each reply using the model from the part 1.
4. For the replies corresponding to the same tweet, choose the label which has the maximum occurance and assign it as the tweet‘s label and then ignore the replies.
5. Now we have a training set of some tweets with their labels, SUPERVISED!
6. Train another model with this data.
7. Now the final model is ready. Given only tweets as the input, this model predicts the sentiment of the potential reply that tweet is likely to get!

Part 2: Dataset

GetOldTweets3

- It is a command line tool and library to extract tweet without the limitation of Rate limiting of the standard API.
- It is powerful and fast but it only downloads tweets.
- By modifying it, we can extract also all the replies of each tweets.
- Because of using *api.search* it becomes slow.

Part 2: Dataset

GetOldTweets3

- It is a command line tool and library to extract tweet without the limitation of Rate limiting of the standard API.
- It is powerful and fast but it only downloads tweets.
- By modifying it, we can extract also all the replies of each tweets.
- Because of using *api.search* it becomes slow.

The data collection

- It is downloading a list of words which is fix and not random.
- Up to now it contains 172,174 replies with 5,832 tweets .
- In the replies the name of tweet user is deleted(e.g. @mark23).
- The data after the preprocessing part and set the label as maximum number of occurrence use for training second part.

Part 2: Dataset

GetOldTweets3

- It is a command line tool and library to extract tweet without the limitation of Rate limiting of the standard API.
- It is powerful and fast but it only downloads tweets.
- By modifying it, we can extract also all the replies of each tweets.
- Because of using *api.search* it becomes slow.

The data collection

- It is downloading a list of words which is fix and not random.
- Up to now it contains 172,174 replies with 5,832 tweets .
- In the replies the name of tweet user is deleted(e.g. @mark23).
- The data after the preprocessing part and set the label as maximum number of occurrence use for training second part.

Preprocessing

The same as the part 1.

Part 2: Model Architecture

Part 2: Model Architecture

- For now, the exact same architecture as the part 1 is used!

Part 2: Training Parameters

- 20% of the data as validation.

- Loss functions: Cross Entropy loss
- Optimizer: ADAM, with a fixed learning rate of 8e-4
- 4,576,283 trainable parameters

- Hidden and cell dimension of the LSTM: 256
- Embedding dimmension as mentioned before: 100

- Trained on Nvidia GeForce 940MX (not so powerful GPU).
- Training duration: 09 minutes and 45 seconds with batch size of 16 for 18 epochs.

Part 2: Testing Results

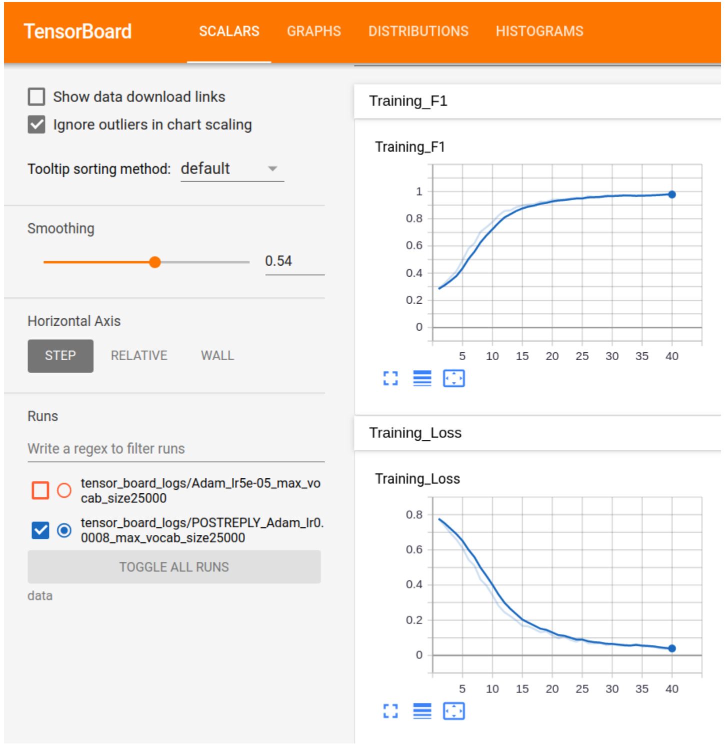
- Accuracy of 69.09% on the validation data.

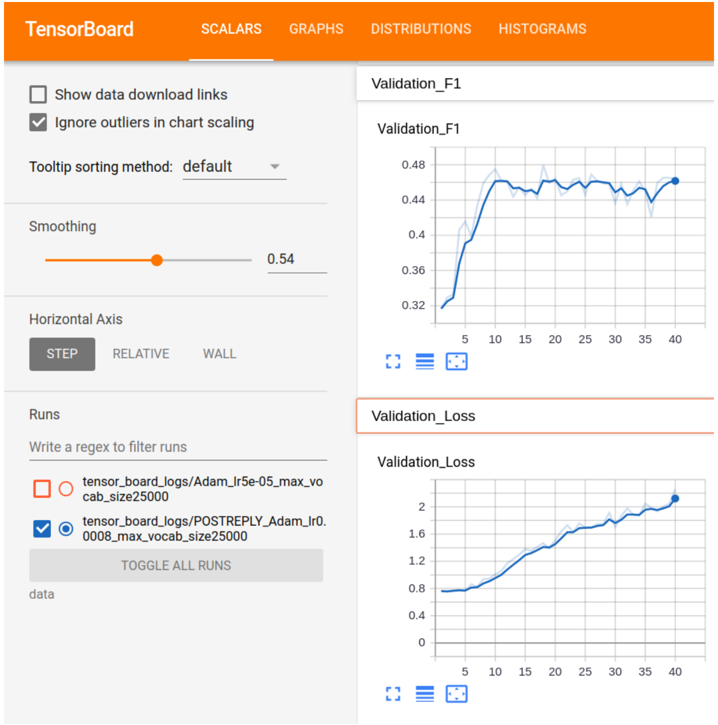
- (macro) **F1 score** of **0.480** on the validation data,

Part 2: Testing Results

- Accuracy of 69.09% on the validation data.
- (macro) **F1 score** of **0.480** on the validation data,
    - **We need way more data!!!**

Part 2: Learning Curves (extracted from the TensorBoard)





Results so far

- Most of the predicted replies are neutral as expected.

Results so far

- Most of the predicted replies are neutral as expected.
- A data set of 69K samples (as an example) for the part 2, had only 2.7K unique tweets. We need more data!

Results so far

- Most of the predicted replies are neutral as expected.
- A data set of 69K samples (as an example) for the part 2, had only 2.7K unique tweets. We need more data!
- NLP problems are harder to deal with, than Computer Vision :D

Interactive Sentiment Analysis

- Here, we will show you interactive demos of each part.
- Click here for the Jupyter notebook file!

1. Overview
2. Approach
3. Results
4. Interactive Sentiment Analysis

5. Future Work

Future Work

- Using the ensemble model of BiLSTM and CNN as the architecture of the part 1.

Future Work

- Using the ensemble model of BiLSTM and CNN as the architecture of the part 1.
- Removing the unnecessary symbols and characters from the vocabulary.

Future Work

- Using the ensemble model of BiLSTM and CNN as the architecture of the part 1.
- Removing the unnecessary symbols and characters from the vocabulary.
- A new architecture for the part 2 can be explored.

Future Work

- Using the ensemble model of BiLSTM and CNN as the architecture of the part 1.
- Removing the unnecessary symbols and characters from the vocabulary.
- A new architecture for the part 2 can be explored.
- A dataset of more than a million tweets with their replies should be used for the part 2!

Future Work

- Using the ensemble model of BiLSTM and CNN as the architecture of the part 1.
- Removing the unnecessary symbols and characters from the vocabulary.
- A new architecture for the part 2 can be explored.
- A dataset of more than a million tweets with their replies should be used for the part 2!
- Researching for a valid test set for the part 2 in case of publication.

Thank you for listening!