IDS703 Final Project Report

Anna Dai, Satvik Kishore, Moritz Wilksch

December 12th, 2021

1 Introduction

In this project, we work on tweet classification as a Natural Language Processing problem, more specifically, as a document classification problem. Twitter is a microblogging service where users post publicly visible "Tweets", which are essentially texts with less than 280 characters. These Tweets may also contain other media objects which are discarded for the purposes of this project. These Tweets most often serve as discussion pieces as part of larger conversations. They are relevant to any number of topics under discussion. These "topics" are also often explicitly highlighted by the user using a "hashtag", i.e. text with '#' followed by the topic name, or a commonly used shorthand for it. For the purpose of our project, we treat these hashtags as "topics" for our document classification model, where each Tweet is an instance of a document.

2 Data

2.1 Collection

We manually select seven popular topics (hashtags) for classification: crypto, tesla, championsleague, formula1, thanksgiving, holidays, and covid19. These topics were intentionally selected so that some topics have some degree of overlap between them (e.g. holidays and thanksgiving), others are easier to differentiate (e.g. crypto vs formula1), and is an independent topic that is often mentioned with the others (i.e. covid19). We leverage the python library twint (twintproject 2021) to scrape approximately 10,000 Tweets, or documents, for each of these seven topics.

2.2 Pre-Processing

The raw scraped Tweets are quite messy, as they are cluttered with links to webpages, images, mentions of other Twitter users (i.e. @usernames) and other hashtags, so we spend some time to carefully pre-process the data before proceeding with modeling. As a first step, we tokenize each document into words. The id of each unique token is stored in a encoder object. [TODO: elimination of common words?]. Conversion of emojis into tokens (each appearance of an emoji is a token. Multiple emojis strung together are treated as different tokens in sequence). We also removed punctuation marks. Following these steps, we split our data into three parts using a 60:20:20 split to form a training dataset, a validation dataset, and a testing dataset.

The collected corpus of almost 70,000 tweets is randomly split into a training, validation, and test set in a 60/20/20 ratio. We will train models on the train set, find optimal hyperparameters using the validation set, and report all performance metrics in the *Results* section on the test set, which is not used for any other purpose than evaluating each model once.

3 Modeling

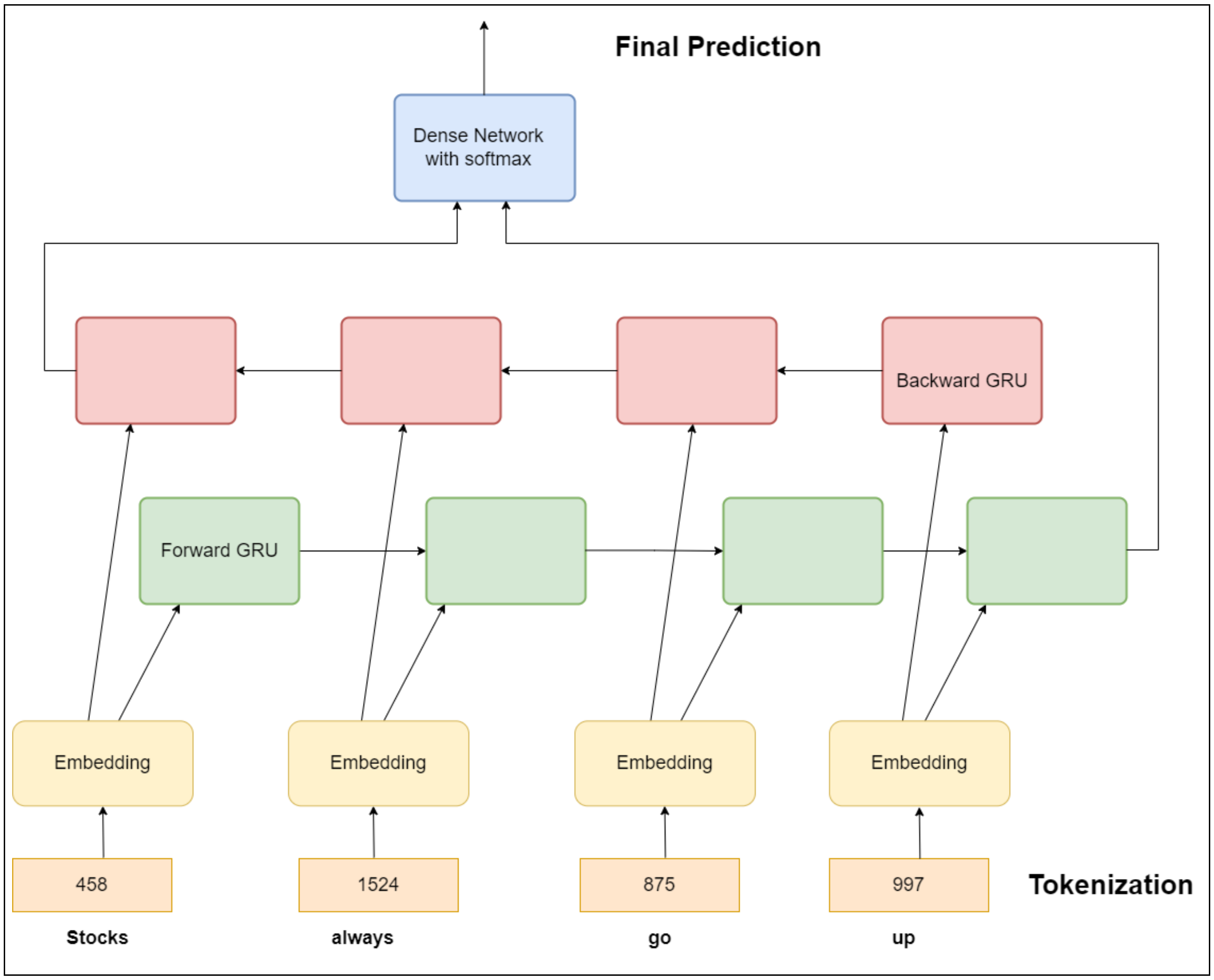
3.1 Generative Model

We use a Latent Dirichlet Allocation (LDA) model as a generative model to learn from the corpus we have collected. This type of model does not require any hyperparamter tuning, and thus is trained using a combination of the training and validation dataset. We used the LDA implementation from scikit-learn (Pedregosa et al. 2011). It is fit to the corpus to find seven separate topics, as this is the number of actual topics in the training set. Subsequently, for each inferred topic, we manually inspect the top 50 words that are associated with it to assign it a name. This manual step is necessary, as the order of topics is not preserved. In fact, the LDA model does not even guarantee to find the same topics that were collected in the original data set. This shortcoming will be discussed in the *Results* and *Conclusion* sections. To use the LDA for document classification, we let it infer the topic distribution of each document in the test set and use the $argmax(\cdot)$ function to assign each document the topic that is most prevalent according to the LDA. Finally, the LDA is used to generate synthetic data. For each topic, we sample 10,000 artificial documents, the length of which is sampled from the empirical distribution of tweet lengths each time. Similar to the actual data set, the synthetic data set is also split into a train, validation, and test set.

3.2 Discriminative Model

We implemented a discriminative model as a neural network model to be able to classify input documents into their respective cateogries. These documents can arise from the synthetic data generated by the LDA or the real data scraped from twitter. The model architecture we used was a bidirectional Gated Recurrent Unit (GRU) at its core. This model is able to accept inputs of varying length and outputs a vector of length 7 for each input document. This is the vector of predicted probabilities for each of the seven topics. These probabilities are generated by the softmax layer at its end of the model. This model was created and trained using Tensorflow Abadi et al. 2015

3.2.1 Model Architecture



The model contains three essential subparts: the embedding, the bidirectional GRU, and the final densely connected layer. Each layer had one or more hyperparameter that were subject to tuning. These hyperparameters are denoted by $HP(i)$. The model is fed the document in the form of tokens identified from the encoder. There were "X" number of unique tokens. The embedding layer is a neural net that converts each token into a vector of length $HP(0)$. These embedded vectors are then received as input by the two GRU layers. GRU are a form of recurrent neural networks that use gates to better pass long term memory but have smaller complexity than LSTM. GRU also do not differentiate between output vector and hidden vector. The size of these hidden vectors was $HP(1)$. They are faster and easier to train than LSTMs while being more expressive than simple RNNs. The final hidden vectors of both GRU (forward and backward) are then concatenated and fed into a dropout layer with dropout rate $HP(2)$ followed by a densely connected network of two layers seperated by another dropout layer with rate $HP(2)$., These dense layers were of size $HP(3)$ and 7, and had activation layers of ReLU and Softmax respectively. - Adam optimizer - Learning Rate found using the Learning Rate range test (= 10 ** -2.5)(Smith 2018)

3.2.2 Training Process

1) hyperparameter tuning on synth data 2) benchmark on synth and real 3) continue training on real data 4) benchmark on synth and real 5) Completely new model: Hyperparametertuning only on real data 6) benchmark on synth and real

4 Results

4.1 Benchmarking on Synthetic Data

	precision	recall
thanksgiving	0.916	0.969
formula1	0.986	0.897
covid19	0.960	0.966
championsleague	0.957	0.965
crypto	0.945	0.901
tesla	0.943	0.985
holidays	0.924	0.954

	precision	recall
thanksgiving	0.499	0.401
formula1	0.033	0.037
covid19	0.183	0.454
championsleague	0.484	0.503
crypto	0.717	0.317
tesla	0.326	0.495
holidays	0.099	0.150

Table 1 & 2: Benchmark results of neural net (trained on synthetic data only) on synthetic data (left) and real data (right)

4.2 Benchmarking on Real Data

	precision	recall
thanksgiving	0.386	0.524
formula1	0.009	0.012
covid19	0.797	0.518
championsleague	0.857	0.542
crypto	0.543	0.751
tesla	0.838	0.567
holidays	0.032	0.164

	precision	recall
thanksgiving	0.814	0.921
formula1	0.742	0.889
covid19	0.811	0.798
championsleague	0.878	0.673
crypto	0.764	0.782
tesla	0.798	0.709
holidays	0.824	0.974

Table 3 & 4: Benchmark results of neural net (trained on synthetic data and real data) on synthetic data (left) and real data (right)

	precision	recall
thanksgiving	0.395	0.482
formula1	0.052	0.046
covid19	0.519	0.350
championsleague	0.616	0.708
crypto	0.566	0.553
tesla	0.636	0.546
holidays	0.048	0.092

	precision	recall
thanksgiving	0.885	0.822
formula1	0.825	0.810
covid19	0.865	0.769
championsleague	0.796	0.844
crypto	0.862	0.710
tesla	0.680	0.855
holidays	0.838	0.979

Table 5 & 6: Benchmark results of neural net (trained on real data only) on synthetic data (left) and real data (right)

	precision	recall
thanksgiving	0.946	0.979
formula1	0.986	0.977
covid19	0.986	0.986
championsleague	0.986	0.969
crypto	0.986	0.987
tesla	0.986	0.976
holidays	0.967	0.969

	precision	recall
thanksgiving	0.150	0.107
formula1	0.558	0.562
covid19	0.034	0.022
championsleague	0.339	0.443
crypto	0.333	0.304
tesla	0.412	0.319
holidays	0.387	0.670

Table 7 & 8: Benchmark results of LDA model on synthetic data (left) and real data (right)

4.3 Learned Word Embeddings

To verify that the neural network model has learned meaningful word embeddings, we feed it 7 manually chosen words (one for each topic, but never the name of the topic itself) and find the ten closest word in the embedding space using cosine similarity. We find that the word embeddings capture an impressive amount of meaning: The name of Formula1 driver Lewis Hamilton is closest to other driver's names as well as "formula" and "race". "Barcelona" maps to other sports clubs that played in the championsleague. "Vaccine" is close to health- and covid-related words while "Christmas" is similar to texttttdecember and "festive". For "Elon" and "BTC" the closest words are coherent and reffer to Tesla and Cryptocurrencies. Only "turkey" is not clearly close to thanksgiving, presumably as the topic "thanksgiving" has been hard to identify throughout the experiments. TODO: - Do the accuracies support this view?

hamilton	barcelona	vaccine	christmas	turkey	elon	btc
max	bayernbara	immunity	december		musk	cryptos
ocon		vaccines	nicholas	nick	fsd	coins
lewis	zenit	deaths	festive	cotton	teslas	eth
vsc	barca	tests	visit	flex	ev	ssfeed
championship	bayern	measures	snowman	gratitude	supercharger	tether
abu	ucl	boris	pack	nov	binance	io
fia	liverpool	booster	wreath	rosemary	cointrade	opportunity
formula	milan	vaccinated	rainbow	chronicles	giga	dump
race	matchday	vaccination		skyrocket	autopilot	analyzing

Table 9: Top 10 closest words in embedding space for seven examples.

5 Discussion and Conclusion

References

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Smith, Leslie N (2018). "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay". In: *arXiv preprint arXiv:1803.09820*.

twintproject (2021). *TWINT - Twitter Intelligence Tool*. URL: `https://github.com/twintproject/twint`.