Personal Tweet Classification

Sharvil V. Shah
shahsharvil96@knights.ucf.edu

Vinit V. Bhosale
vinit.bhosale@knights.ucf.edu

Rohit P. Marathe
rpm360@knights.ucf.edu

Yesaswini Valavala
yasaswinivalavala@knights.ucf.edu

1. MOTIVATION & PROBLEM STATEMENT

Classification is an important task to understand the data to come up with a conclusion. With this information, various problems in the field of machine learning, Natural language could be solved. We are trying to classify whether a tweet is personal or non-personal. A personal tweet is something that involves a user in a certain way, while a non-personal tweet is something like shared information/post, news post, repost, and media. Many people across the internet created by posting, commenting, or sharing the information.

We are processing this information to get the involvement of the person in the shared content. This classification information is useful for a content-creating platform and advertisement companies as they can come up with a quantitative understanding of all users that are contributing to fresh content creation.

With a new user added to internet, they start sharing new content each day and individuals expressing their options, feeling, thoughts, and perspectives towards people, product and services. The internet provides rich information and data. Advertisement companies invest millions to target specific user, currently, most of the firms have gone digital in the previous decade.

Various users across platforms like Twitter and Reddit are targeted for advertisement. They need to reach the targeted audience, the solution to this is focusing on users who tweet about their personal sentiments, opinions or reviews etc.

2. RELATED WORK

The work done in [7] mainly discusses about the pattern extraction using bootstrapping process. On using this method we can extract opinionated expressions from the labelled unannotated data and using bootstrapping it yields to high precision.

The data retrieved from the above methods are further classified into personal and news (non-personal) tweets in [4] and it also aims at further classification of negative and non-negative sentiment classifier which is not being implemented in our project.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

3. PROJECT DESCRIPTION

3.1 Data Collection

While doing the data collection we have considered the twitter data while there few other good content creation platforms of Facebook, LinkedIn. We are making use of the twitter Rest API to collect the data from twitter website. The advantage of using a Rest API is that it provides real-time data as well. We can configure the data collection process depending upon the requirement, we set the range of data we want to collect from twitter. While working on this problem we have collected the dataset of about 10000 tweets. We randomly picked 2000 tweets to be annotated out of that.

3.2 Data Pre-Processing

On collecting the data, we first clean it by removing the unnecessary text such as URLs, mention that starts with '@', hashtags '#' and non ASCII characters. To perform this preprocessing we make use of regex library in python. Rest of the pre-processing is dependent of the models selected. The pipeline is shown in Figure 1

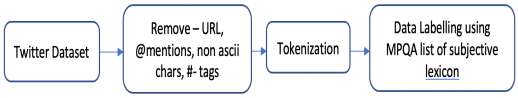

Figure 1: Visualization of data collection and data pre-processing.

3.3 Bidirectional Encoder Representations from Transformers

It is widely known that Deep models require large datasets to learn good representations. However, large labeled datasets might not be available to small teams with less resources. To alleviate the problem an unsupervised or semi-supervised learning algorithms are used sometimes. Self-supervised is one such unsupervised learning where the model learns on a really large dataset but the labels are generated by some inherit property of the data itself. For example, some models are trained to generate language. In this task we don't need

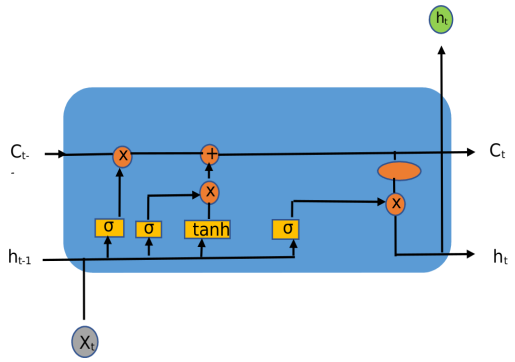

Figure 2: Architecture of LSTM

to have the labels annotated by humans, instead the next word in the sentence becomes the label. BERT[1] leverages this idea of language generation to pre-train models

BERT is a bidirectional transformer based model. Its architecture constitutes of an encoder and decoder module. Where the encoder will bring the text to a semantic representation and the decoder will generate the same text from the representation. This model is trained on an extremely large dataset to generate text and predict which sentence in the pair is second. Once it learns a good representation we can train a linear classifier to transfer that to a specific task(text classification in our case). In our work we use https://huggingface.co/transformers/ for the PyTorch implementation.

This implementation provides the entire pipeline of BERT including the text preprocessing. First the input text is tokenized, then they add token of [CLS] at the beginning of each sequence. This token is used because bert provides ability to process sequence level and token level processing in single model. So this special token is used in the input to get the sequence level classification output from the encoder. Another special token [SEP] is appended at the end of the sequence to separate 2 sequence for next sentence prediction task. The text is then lemmatized which means that words like "I'm" will be converted to "I am". Sequences are also padded with [PAD] token to keep all of them of same size. This entire preprocessing is done using encode_plus function of huggingface implementation.

3.3.1 Long Short Term Memory Networks:

Long short term memory (LSTM)[3] is a RNN model which mainly aims at solving the vanishing gradient problem. The gradient will become smaller during back propagation such that it will not be in stage to receive the signal from farthest ones, so the weights will be updated only based on the signals obtained from the nearer ones such that it will fail in capturing long term dependencies. LSTM uses additional cell state and hidden state to capture and store the information. It also helps in retrieval of information when needed. Cell state in LSTM is used for storing of information where the information can be stored and also used in long period of time. This information plays key role in deciding which information needs to written or erases or read.

The architecture of LSTM is described in Figure 2. LSTM uses following gates to decide which information needs to written or erases or read.

1. Forget gate: It decides which information from the previous state needs to be kept in the cell or which information from the previous state needs to be for which it uses a filter to do so.
   $f^t = \sigma(W_f h^{t-1} + U_f x^t + b_f)$
2. Input gate: It decides which part of information obtained from the new cell is written into cell state.
   $i^t = \sigma(W_i h^{t-1} + U_i x^t + b_i)$
3. Output gate: It decides which cell part can is outputted to hidden state.
   $o^t = \sigma(W_o h^{t-1} + U_o x^t + b_o)$

We need hidden state and cell state in LSTM which can be computed based on given input sequence:
$\tilde{c} = tanh(W_c h^{t-1} + U_c x^t + b_c)$

Cell state: It stores the current information details that are obtained after removing content details from previous time state. The information is stored for long period of time. The equation for computing current cell state is:
$C_t = \sigma(f_t \circ C_{t-1} + i_t \circ \tilde{C}_t)$

Hidden state: It helps in storing the output that is obtained from the previous state. So it acts as a memory for the network. The equation for computing the hidden state is:
$h^t = o^t \circ tanh(c^t)$

For every time step t, W is input weight B is bias, U will be updated weights, h^t values are updated based on previous time step, h^{t-1} will be the hidden layer of the previous time step at RNN. The σ is the sigmoid function used as a non-linearity and to weight different gates.

3.4 Convolution Neural Networks for Text Classification:

As per the current trend of the Convolution Neural Network(ConvNet), it is making a significant impact in the computer vision and speech recognition sector. More and more researchers are moving towards a convolution neural network for text classification solutions. As per the 2014 research, we can use unique embedding words as input for ConvNet and make sentence classification. In this process, the knowledge of syntactic or semantic of language is not required [2]. This method is called a word-based Convolution neural network. There is another study for character-level convolution neural network. Where character-level N-grams will be used with a linear classifier, and that will be used as input for the convolution neural network[5]. Similarly, we will be using character-level CNN for evaluating the tag for the tweet. As we discussed above, ConvNet does not need to understand the syntactic or semantic structure of the language.

The pipeline is described in Figure 3 and works as follow:

1. **Basic Tweet Level Pre-processing**: - Here we will remove unwanted information such as, URL, hastags, user-handle names, rt-fav, etc. from tweet and convert uppercase letters to smaller case.

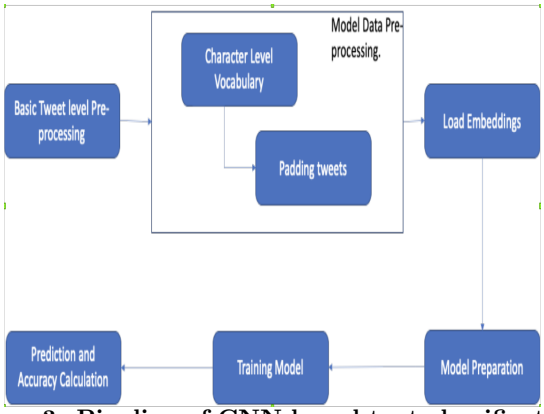

Figure 3: Pipeline of CNN based text classification.

2. **Model-Based Data Pre-processing**: - Here, the process will be divided into two halves. In the first half, we will extract character vocabulary, and in second, we will pad the tweet with 0. As there is a tweet limit of 160 for an individual tweet, and 280 for group tweet. Thus we are padding the tweet of max length 300.

3. **Embedding Stage**: - As in previous stage we have extracted all the characters it has used for a tweet. From that we will get a unknown tag; we will mark it as 'UNK'. After this we will use one-hot vector to represent the characters. We consider English smaller case letters, digits, special characters and new line character.

4. **Model Design**: - Here we are using 6-convolution layers and 3 – fully connected layers and 1 max-pooling. It has two main key modules, first, temporal convolution module, which calculates 1D convolution, and second, temporal max-pooling.

3.5 Naive Bayes Classifier:

The model we'll be using is the Naive Bayes classifier, the main idea behind the Bayes classifier is what is the probability that event X would happen given Y. To understand the equation, we use the naming terminology in the theorem. Here the Hypothesis to predict the class of tweet and evidence is the word(W) appearing the tweets (Personal and Non-personal). The problem is formulated as below: Let's say the text is made of words w_1, $w_2 ... w_n$ and the classes are C_0 and C_1

$p(C_0|w_1, w_2, ...w_n) = \frac{p(w_1,w_2,...w_n|C_0)p(C_0)}{p(w_1,w_2,...w_n)}$

As the denominator is constant we can ignore it for comparison across all classes. The numerator is the joint probability $p(text, PersonalTweet)$.

$p(w_1, w_2, ..., w_n, C_0) = p(w_1|w_2....w_n, C_0)p(w_2|w_3....w_n, C_0)$
$...p(w_{n-1}|w_n, C_0)p(w_n|C_0)p(C_0)$

The naive assumption is that all the words or features here are independent of each other(mutual independence). Thus the equation becomes

$$p(w_1, w_2, ..., w_n, C_0) = p(C_0) \prod_{i=1}^{n} p(w_i|C_0)$$

To feed the tweets into the model we have to remove few unwanted words from the tweet in the preprocessing we are removing the stop words, links, unwanted characters, repetition of words. We are making using sklearn library[6] to clean the data and as we are also making use of count vectorization as it essential for to provide frequency of word for naïve Bayes working. Once the count vectorization we split the data into train and test data using 80% and 20% splits respectively and train the model on the training data. Finally, we predict the class of test data which are the input tweets. The pipeline is shown in 4.

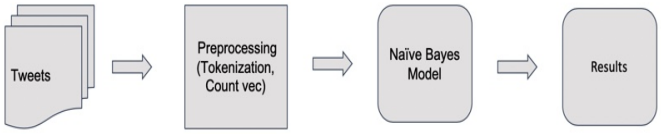

Figure 4: Pipeline of Naive Bayes Classifier

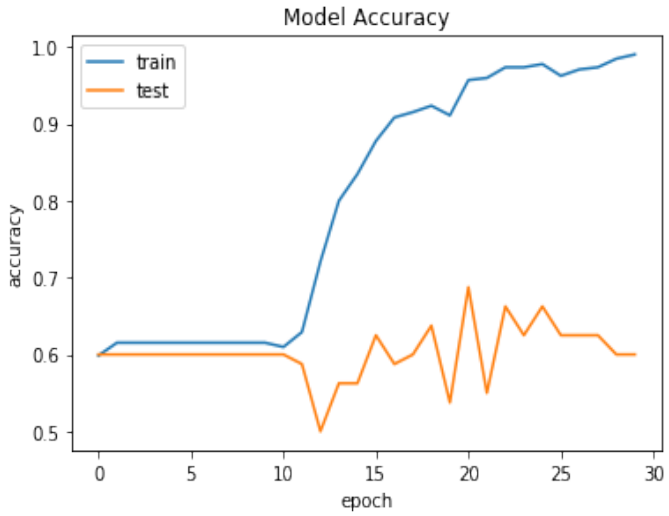

Figure 5: Character Level CNN - Model Accuracy

4. EVALUATION AND RESULTS

As this task involves text classification, we report the accuracy, precision, recall and F-1 score of each model. The results of each model are shown in Table 1. The accuracy and loss plot for CNN at each epoch is shown in Figure 5 6 respectively. We can see from the figure that after 10 epochs the CNN starts overfitting and the loss overshoots.

5. CONCLUSION:

As shown in the results except CNN based model all models are performing similarly. This might be due to the fact that CNN is being trained from scratch and has not used sufficient labeled data to get to a good representation. These results can further be improved by labeling more data. Cur-

Method	Accuracy	Precision	Recall	F-1
BERT[1]	73%	68%	71%	70%
Naive Bayes	65.73%	71.42%	70.75%	71.09%
LSTM[3]	68.50%	73.07%	68.46%	70.69%
CNN[5]	57.50%	63.93%	65.54%	64.73%

Table 1: Results of Personal Tweet classification

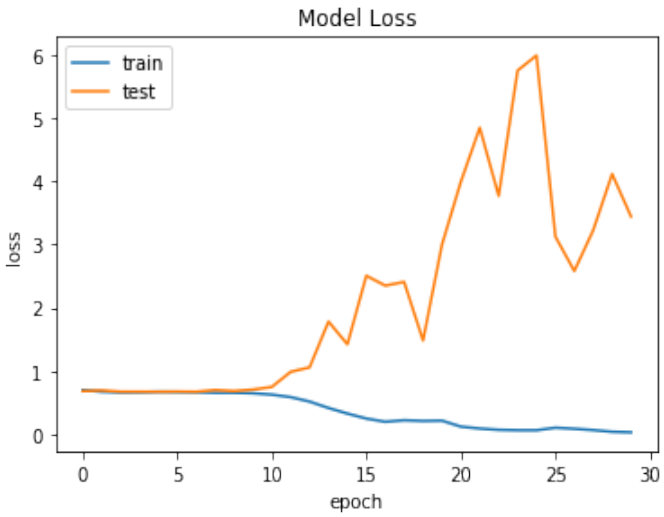

Figure 6: Character Level CNN - Model Loss

	Sharvil	Rohit	Vinit	Yesaswini
Topic selection	25%	25%	25%	25%
Related work	20%	20%	30%	30%
Project proposal	30%	20%	30%	20%
Data labeling	25%	25%	20%	30%
Model creating	27%	27%	27%	16%
Presentation and Report	25%	25%	25%	25%

Table 2: Roles of Collaborators

rently only 800 labeled tweets were used to train all models.

6. CODE:

The code is available at https://github.com/sharvil10/NLP_submission.

7. ROLES OF COLLABORATORS

The work distribution that we have done is divided into phases starting from the topic selection, related work, project proposal to presentation and Report. Table 2 is the complete contribution of each individual in the whole process.

8. REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Cícero dos Santos and Maíra Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[4] Xiang Ji, Soon Chun, and James Geller. *Knowledge-Based Tweet Classification for Disease Sentiment Monitoring*, volume 639, pages 425–454. 03 2016.

[5] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16:1047–1067, 12 2007.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 105–112, 2003.