Inter-IIT Bridgei2i

Team Number 14
Submission Code : H2_B2I_14
March 21, 2021

1 Introduction

The given problem statement has been segregated into 3 sub problems, namely:

- Automated identification of mobile-tech themed articles and tweets from the given dataset.
- Entity (mobile brand) based sentiment analysis.
- Summarization and Headline Generation for the given text.

The following diagram depicts the workflow adopted by our team for the aforementioned tasks:

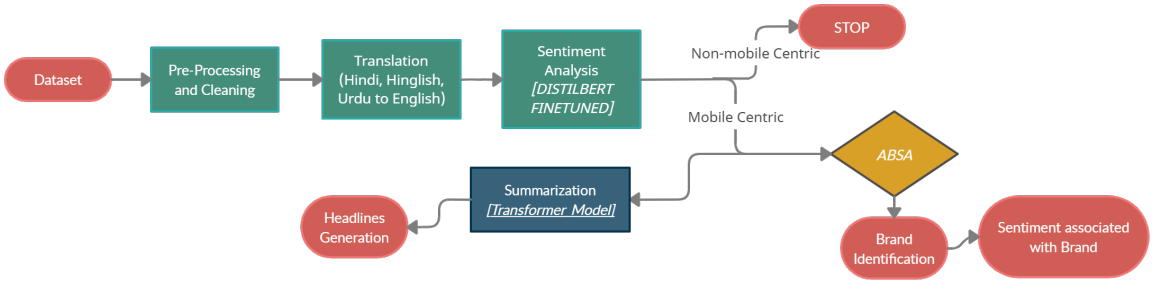

Figure 1: Flow Diagram

2 Dataset Characteristics

2.1 Preprocessing

The dataset provided had 4000 tweets and 4000 articles. Some articles and most of the tweets were repetitive, hence on removing duplicate examples, the corpus was reduced to 2400 tweets and 3567 articles. Another interesting characteristic of the dataset was the class imbalance. The number of mobile centric articles were just 873 out of 3567 (24.47%) and the mobile centric tweets were a paltry 569 out of the total 2400 (23.71%). This concern was alleviated by assigning class specific weights during the model fine-tuning.

Also, as the given dataset was possibly created via web scraping, it contained a lot of noisy information, like the text fields of menus of the web page, HTML tags, URLs, irrelevant special characters, and so on. This issue was mitigated to a great extent by aggressive preprocessing.

In case of tweets, the data is made bereft of irrelevant elements using the *preprocessor* library, and tweets with less than 5 characters are removed to augment the robustness of the dataset. On the other hand, *Beautiful Soup* library is used to eliminate the presence of HTML tags and attributes in articles. The first line of every article was then checked for the presence of menu tags and other irrelevant objects, and was removed from the article if the said objects were found. This was followed by the removal of empty articles in the corpus. A total of 2347 tweets and 3481 articles were finally obtained. The preprocessing code is quite versatile and has been deftly designed to enhance the scalability of the model.

2.2 Dealing with Multiple Languages

The dataset provided contains languages Hindi, English, Hinglish and Urdu. We use Google Translate API to automatically identify the language and translate it to English language. Google Translate API can detect and translate

more than 100 languages. It is particularly capable of translating various Indian regional languages such as Bengali, Punjabi, Tamil and Telugu. In this way, our end-to-end model is scalable to various different languages. This ensures that the disparity between the model and the dataset language is weeded out, while retaining the accuracy and the performance of the model

3 Task 1: Automated Theme Identification

3.1 Idea

The task requires binary classification of the theme of articles and tweets into mobile technology and all others. The next two tasks are contingent on accurate theme classification in this step, as summarization and entity-based sentiment analysis is required only for 'mobile_tech' themed articles and articles/tweets respectively.

3.2 Model

We propose using a transformer based architecture for our task, owing to the fact that they currently produce state-of-the-art results on a number of downstream tasks. For our purpose, we employ the DistilBERT base uncased model, fine tuned for classification on the given dataset. This model is preferred over native BERT as it contains about half the total number of parameters of BERT base and hence, is computationally efficient (about 60% faster) and provides results having similar accuracy (more than 95% of BERT's performance). The hyperparameters such as the batch size (16) and the learning rate (3e-5) are experimentally obtained for optimal fine tuning, with Adam optimizer and appropriate callbacks and loss functions to prevent overfitting. Furthermore, to tackle the class imbalance, the $class\_weights$ parameter was used.

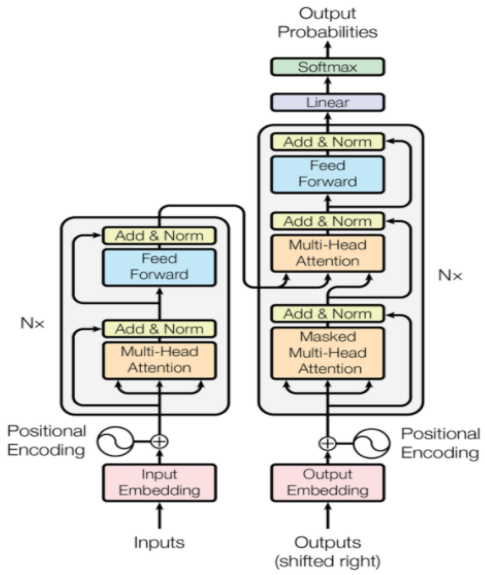

Figure 2: Transformer Architecture
(Attention is all you need, Vaswani et al.)

The dataset was transformed into encodings using the DistilBERT tokenizer and split into training and validation sets in the ratio 80:20. The scores obtained on the model after training on the preprocessed dataset are shown in Table 1.

The training time translated to around 2.20 minutes/epoch with a Tesla T4 GPU on Google Colaboratory, for a total of 6 epochs before Early Stopping callback interrupted the training.

Metric	Value
Training Accuracy	99.55%
Validation Accuracy	96.57%
F1 score	0.9725
Recall	0.9941
Precision	0.9519

Table 1: Theme Classification Results

4 Task 2: Entity Based Sentiment Analysis

4.1 Idea

The task requires identifying all the brands present in mobile-technology themed articles and tweets along with the corresponding sentiment associated with each entry of these brands at the tweet/article level.

4.2 Aspect Based Sentiment Classification Model (ABSC)

Aspect Based Sentiment Analysis (ABSA) is a state-of-the-art text analysis method that first categorizes data by aspect and then identifies the sentiment associated with each of those aspects. It investigates multiple sentiments involved with different objects present in a single text. For instance, if a book review makes a comparison between two novels, ABSA gives the opinions of the author pertaining to each novel after understanding the concealed sentiment for both the novels.

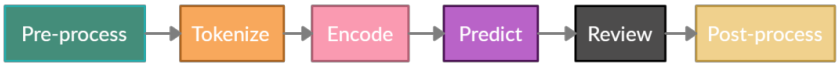

Figure 3: *aspect-based-sentiment-analysis* Library Workflow

Figure 4: An example to illustrate ASBC

In Task 2, we have used an Aspect Based Sentiment Model based on Ada-BERT. The classification is performed in two steps. Firstly, self-supervised fine-tuning of pre-trained weights of the language model is done on domain-specific corpus. Subsequently, the fine-tuned model is applied on the ABSC end-task in a supervised way. The BERT-Ada model here is fine-tuned on restaurant and laptop corpora to achieve top performance metrics. For the above architecture and pipeline, we utilize the *aspect-based-sentiment-analysis* package to implement the same.

ABSA is different from the traditional approaches for sentiment analysis. Instead of directly processing the model's outputs, it has a review process in between that has an independent component called 'professor'. The 'professor' reviews the model's hidden states and outputs to identify and correct pessimistic predictions. Because the professor considers information from auxiliary models to make the final decision, we have greater control over the decision-making process. Hence, the model's behavior can be freely customized.

ABSC is used to classify the sentiment of a given entity-target into either of the three classes - positive, negative or neutral. The input to the model is a tokenized sequence (compatible with BERT Architecture), where the position of the embeddings is strictly preserved. A fully-connected layer with 3 output neurons on top of the last

hidden representation, ensued by the softmax activation function is used to predict a probability distribution over the above mentioned three classes.

ABSC is also remarkable with respect to the fact that the model is highly scalable. ABSC has the capability to sift through large corpuses and extract information at a granular level. It can analyse massive information meticulously, thereby saving time and delivering powerful results.

5 Task 3: Automated Headline Generation

5.1 Idea

The task requires generation of headlines for mobile technology themed articles. It is essentially conceived as a text summarization task, wherein the character limit is to be kept to a minimum threshold . The generated headlines then need to be compared with the actual headlines to generate embedding based similarity score, BLEU score and ROUGE metrics. Specifically, abstractive text summarization techniques were used in this part to avoid copying of the data used in summarization. This task again employs the use of transformer based architecture in order to take into consideration the long range dependencies of the main text, and owing to the superior performance in Conditional Generations tasks compared to Single Encoder-Decoder based models.

5.2 Models Used

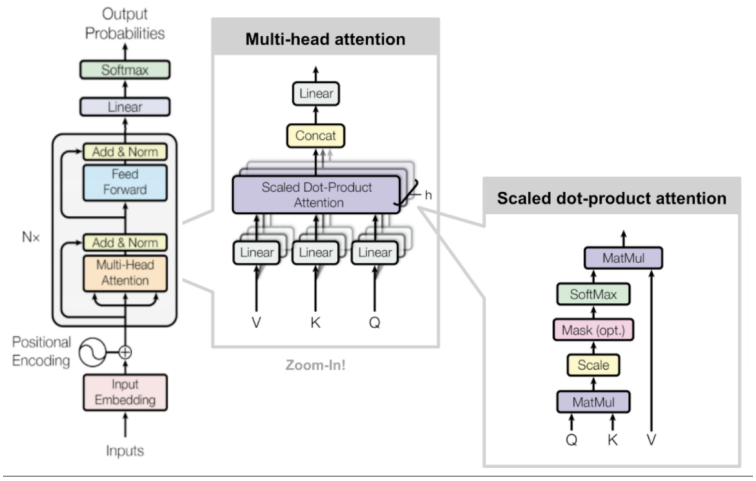

Figure 5: Attention mechanism in Transformer Architecture
(Attention is all you need, Vaswani et al.)

5.2.1 T5

The T5 model is a traditional encoder-decoder Transformer with few tweaks, such as: using a different position embedding scheme, removing the Layer Norm bias, and placing the layer normalization outside the residual path. The encoder and decoder consist of 12 blocks. Each block consists of self-attention, optional encoder-decoder attention, and a feed-forward network.

5.2.2 PEGASUS

The basic concept used behind this model is an encoder-decoder, in which both the types of pre-training methods : Gap Sentences Generation (GSG) and Masked Language Model (MLM) , are applied simultaneously.In GSG, a sequence-to-sequence self-supervised model is designed in the absence of abstractive summaries, generating summary-like text from an input document. Now, while using MLM, the transformer decoder shares all parameters with the encoder when it is fine-tuned on downstream tasks. 15% tokens are selected in input text, and maximum tokens are

Model	Similarity	BLEU Score	ROGUE F-Score	ROGUE Precision	ROGUE Recall
Pegasus	0.725	0.335	0.334	0.517	0.259
Pegasus with BART	0.704	0.334	0.319	0.476	0.251
T5	0.701	0.396	0.317	0.341	0.315
T5 with BART	0.690	0.337	0.298	0.366	0.269

Table 2: Results for Headline Generation (100 random examples)

replaced by masked tokens. MLM is applied along with GSG to pre-train the transformer autoencoder. The BART autoencoder is used for denoising the seq-2-seq model while pre-training.

5.2.3 BART

The superiority of BART lies in the fact that it has flexibility in introduction of noise and establishing arbitrary transformations on the original text according to the project requirements. Therefore unlike conventional encoders, BART allows us to decide the noise function instead of sticking to a particular noise function. Since BART also comprises a decoder which is auto regressive, fine tuning it can lead to positive results in the text generation fields like summarization and headline generation.

5.3 Fine Tuning

We fine tuned both Pegasus and T5 model on the task of abstractive summarization by using the article corpora after thorough cleaning and preprocessing, as done in Task 1. We used the PyTorch framework to fine tune both the models. We kept the Encoder length (equivalent to the input Length) to 512 characters and the Decoder length(equivalent to the Output Length) to 64 characters. The reason for keeping the Encoder length to such a small value was due to the resource limits in the Colab. The decoder length was kept small owing to the fact that we require to generate headlines for the articles, which have a short length. Both the models were trained for 10 epochs with appropriate hyperparameters, using Adam as the optimizer. The fine tuning was done with a 90:10 split of the entire dataset. Both the models were then saved as PyTorch models for future inference.

The fine-tuning time for Pegasus translated to about 17 minutes/epoch, while T5 had a rate of 11 minutes/epoch on the Tesla T4 GPU.

5.4 Inference

We employed and tested four combinations of summarizer models for obtaining inference on a random sample of 100 data articles. The combinations are as listed below:

- T5 fine-tuned
- Pegasus fine-tuned
- T5 fine-tuned cascaded with BART summarizer
- Pegasus fine-tuned cascaded with BART summarizer

The BART based Summarizer was used as-is, without any fine tuning, which was pretrained on the CNN/Daily Mail news dataset. The reason why we chose not to fine tune is due to the fact that we merely used BART summarizer to further denoise the outputs and return a more coherent language. The metrics obtained for the aforementioned combinations are listed in Table 2.

6 Additional Remarks

For an effective real world application for processing news articles, we propose using the Longformer architecture, owing to the fact it supports a length of 4096 characters. However, strict runtimes and limited resource availability on Colab prevented us from fine tuning it for the downstream tasks required in this project.