Automated Headline and Sentiment Generator

A Final Report

Submitted in partial fulfillment of requirements for the

9th Inter IIT Tech Meet

By

Team 9

1 SubTask1 - Domain Classification

1.1 Approaches

For this subtask, we use a simple fuzzy string matching algorithm with a hand-crafted bag of words. We leverage the following observation we made about the dataset - all tweets and articles tagged as relevant to mobile tech contain certain words such as *smartphone*, *phone*, *android* and their Hindi equivalents. We hence check the input string to see if it contains any of these terms, using fuzzy string matching to account for spelling errors. We use this approach for its fast processing, and extensibility to multiple languages (through transliteration of our bag of words). A list of words we used along with their occurrences in the training and test dataset are detailed in table 1.

1.2 Results

We find that this simple algorithm gives us good results, achieving an accuracy of 99.9% on the tweets and 91.1% on the articles. *Time taken to run the experiment for Subtask1 is approximately 30 seconds on 4 core Intel-i5 Processor*. We also note that the misclassified articles are often those which are ambiguous in their content, for example which have user comments, or which might be talking about ancillary technologies such as apps. The complete numbers can be found in table 2.

Word	**Number of datapoints containing the word**			
	Relevant Tweets	**Irrelevant Tweets**	**Relevant Articles**	**Irrelevant Articles**
Smartphone	506	0	388	21
Smartphone(Devnagri)	351	0	320	18
android	42	1	289	11
Phone(Devnagri)	0	0	52	0
camera	178	2	298	3
camera(Devnagri)	22	0	205	8

Table 1: Word Frequencies of our handcrafted bag of words

Dataset	**F1-Score (class 0)**	**F1-Score (class 1)**	**Accuracy**
Tweets	1.00	1.00	100%
Articles	0.94	0.82	91%

Table 2: Experimental Results with our approach

2 SubTask2 - Brand Identification and Sentiment Generator

2.1 Approaches

2.1.1 Company Name Identification

In this task we focused on leveraging the essentially finite set of companies. We already know from problem description that we expect to find only real-world mobile tech and mobile accessory companies in the tweets and the articles. So it makes sense to construct a similar robust set of companies for which our model will look for in the documents. With this in mind we had the following choices of possible approaches -

1. A neural model doing a multi-class classification on the tokens of the document to the set of name of companies or the token being not a company name. This approach would need the complete retraining of the model

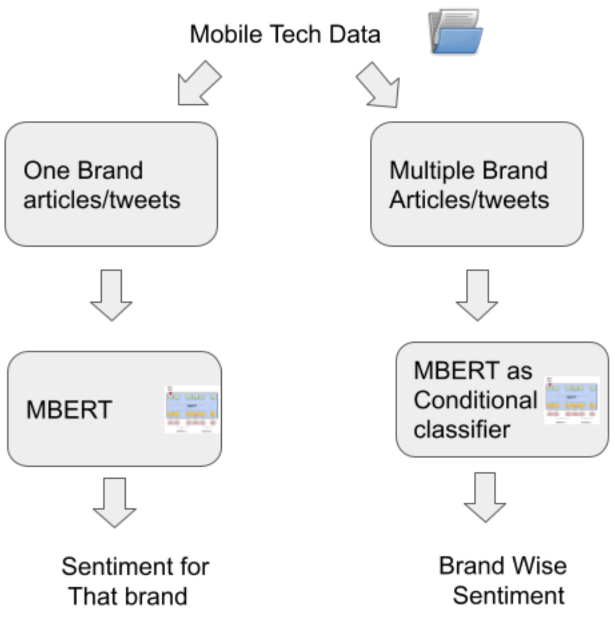

Mobile Tech Data
One Brand articles/tweets
Multiple Brand Articles/tweets
MBERT
MBERT as Conditional classifier
Sentiment for That brand
Brand Wise Sentiment

Figure 1: Architecture For Brand-wise sentiment analysis(for sub-task 2)

2. Adding a NER model to the pipeline and classifying the extracted entities to company names. While we expect such a model to work well, a neural model in the pipeline for such a simple task will increase runtime not to mention the the accuracy miss of the NER model itself

3. Thus, as our final approach, we soft-compare each and every token with a predefined list of companies and if the match score is beyond a threshold (a hyper parameter), we rule on that company being mentioned in the text

2.1.2 Identify sentiments against a brand

Upon brand identification we categorize a tweet/article into one of two categories

1. Having a single brand present in the tweet/article

2. Having multiple brands present in a given tweet/article

If a given tweet/article falls into the first category then we pass it through a sentiment analysis model to identify the associated sentiment value. The model we use for our sentiment analysis task is a multilingual-BERT model pre fine-tuned on MachineHack Product Sentiment Classification dataset which is a sentiment annotated twitter dataset with 6352 training data points. We also translate the tweets into Hindi to emulate the multi-linguality in the provided dataset. We later fine-tune this model using single brand tweets from the BRIDGEi2i twitter dataset.

For the tweets/articles with multiple brands we train a conditional classifier, This conditional classifier is also a multilingual-BERT model pre fine-tuned on SEMEVAL-16 laptop domain aspect level annotated data with the aspect level sentiment analysis objective, this helps the model learn isolation of granular sentiments, we then fine-tune this model using the BRIDGEi2i twitter dataset we annotated with company names and the associated company's sentiment. This m-BERT model acts as a conditional classifier because we give the brand name as the second input to the model;

Example:
Input to the model:
[CLS]*Realme mobile phones offer the best value for money!*[SEP]*Realme.*
Expected output:
Sentiment: Positive

IN CASE OF ARTICLES: We divide the articles into smaller chunks and only then feed them to one of the two models depending on the number of brands present. We divide the article into smaller pieces by identifying a brand name and selecting two sentences from above and two sentences from below along with the sentence containing the brand name. We use this window size of two because we have observed the presence of sentiment words in this window. For the overall brand sentiment of the article we average the sentiment scores over all the article chunks after computing sentiment scores for each of the article chunk. .

2.2 Results:

For both the parts in this subtask, there was no available labeled data. The team tried to label both the articles and the tweets for training/testing purposes. The recall of the the company identifier for articles and tweets was 93.5% and 94.5% respectively. *Time taken to run the experiment for Subtask2 is approximately 45 minutes on 2.3Ghz Quad-Core Intel i7 processor.* The data (especially articles) being huge, it was not possible to list out all companies in decent amount of time and a rough scan was carried out. Thus machine precision comes out to be 47% and 75% for articles and tweets. However, manual inspection indicates the precision to be 90% plus and errors only there in labelling not the model.

3 SubTask3 - Headline Generation / Summarisation

3.1 Approaches

3.1.1 Dataset Analysis and Preprocessing

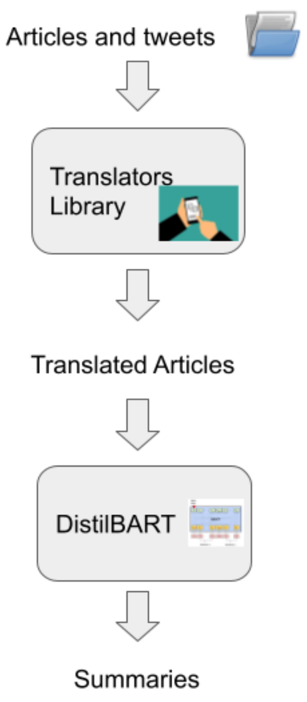
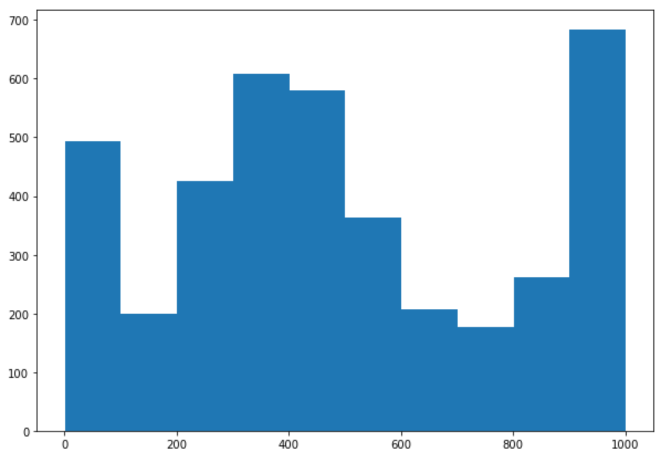


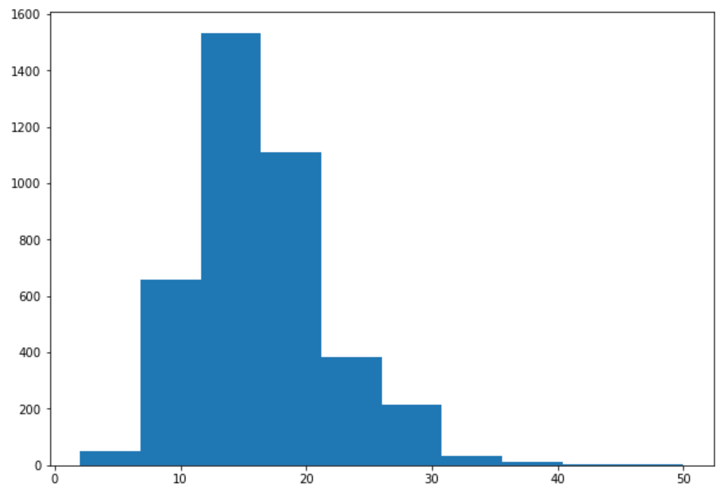
Figure 2: Architecture For Text Summerization(for sub-task 3)

We begin with a thorough examination of the dataset. The dataset has a lot of variability in the sense that we could find all 4 kinds of translated/transliterated articles between the English and Hindi Language i.e. Hindi and English written in both devanagari as well as roman scripts. Quite naturally, this implies that the problem would be more tractable if we get it all in one setting (roman). This would also vastly increase the scalability and robustness of the solution as then we can use this for articles in any language rather than hinging on just english-hindi ones. Moreover, this would open the doors to the plethora of excellent pretrained models available for the english language.
We clean the dataset and translate it using a widely available free python API translator. This uses

translation interfaces of Google, Yandex, Microsoft(Bing), Baidu, Alibaba etc to output decent english translations of the input article.



(a) Article Length Distribution. Note that articles above length 1000 are treated as of length 1000.



(b) Headline Length Distribution

3.1.2 The XSUM task

We note 2 more observations about the dataset.

- The dataset is small for a straight deep learning solution. Deep learning is known to work well only in a large data resource setting. Otherwise, the use of small data sets can lead to overfitting, a well known achilles heel of large models.

- The dataset is the case of "extreme" summarisation. We have very long articles and need relatively short headlines/summaries.

Dataset	#Article	#Headline/Summary	Compression Ratio
Bridgei2i Article Dataset	603	16	37
XSUM	431	23	19

Table 3: Dataset Statistics

Keeping the above 2 observations in mind, we plan an intermediate "pretraining" stage for our model. We augment the training with XSUM. The Extreme Summarization (XSUM) dataset is a dataset for evaluation of abstractive single-document summarization systems. The goal is to create a short, one-sentence new summary answering the question “What is the article about?” . We find that it is a perfect fit as it has a very close target length as our dataset and also encourages extreme summarisation.

3.1.3 Models and Training

We plan to use some standard pretrained models and finetune them on the XSUM task (if not already done). We start with distilBART, a distilled (lighter) version of BART, denoising autoencoder for pretraining sequence-to-sequence models. Further we move to the original BART. Different models like PEGASUS, GPT2 etc remain to be tried as future work if compute capabilites allow.
For finetuning on the given dataset, we use all the articles at the intial stages and not just the domain ones. It has been a common observation among the summarisation community that data usually trumps the domain when finetuning pretrained models. We expect the additional non-mobile themed articles to teach the model the "task" while the mobile articles the "domain". Moreover, we will carry out the training with just mobile themed articles to confirm this hunch.
We divide the articles into train and validation splits. For every tenth article we add it to the val dataset. Hence, the training set has 3600 pairs while the validation one has 400. We use the 400 to compare performance across models. *Time taken to run the experiment for Subtask3 is 45 seconds per article, including all the pre-processing time.* Results can be found in table 4.

Model	R1	R2	RL
DistilBART-12-6	52.37	39.23	49.30
BART-large	49.29	35.83	46.30

Table 4: Tried Models and Results on the validation dataset (size 400 articles)

4 Innovation

1. Since the dataset was very limited, we used MachineHack Product Sentiment Classification Dataset for the Sub-Task 2 to fine-tune the m-BERT model.

2. Due to limited size of the dataset, we have used X-SUM dataset and SemEval-16 Laptop Domain Dataset to fine-tune the m-BERT model.

3. To check how our model will perform on Sub-Tasks 2 and 3 (for previously unseen tweets/articles), we needed unique test data points which we selected using cosine similarity score lower than 0.3.

5 Scalability

1. Our system is capable of handling inputs in 104 languages.

2. We do not need new training data for adaptability of new languages in our system.

3. Our system is not limited by the size of article due to chunking technique used in Sub-Task 2.