**Team 9: Automated Headline and Sentiment Generator**

1. **SubTask 1- Domain Classification**
   1. **Approach:** Our approach for this task involves matching words in the given text with a hand crafted bag of words to determine if the tweet/article is relevant to mobile technology. We use fuzzy string matching to account for spelling errors.
   2. **Steps to run:** Run the script rule\_based.py with python3, make sure that the data is in the folder with the path ‘evaluation\_data.csv’. The output will be generated at the path 'eval\_preds\_mob\_tech.csv'.
   3. **Time Taken:**  For inference on the given file -

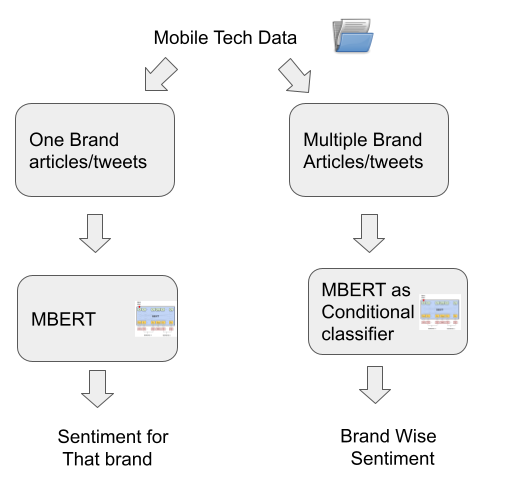
real 0m26.388s

user 0m10.121s

sys 0m1.024s

Done on a 4 core Intel(R) Core(TM) i5CPU @ 2.50GHz machine

1. **SubTask 2 - Brand Identification and Sentiment Generator**
   1. **Approach:** 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. 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.

* 1. **Steps to run:** For the company extraction part, the experiments can be run directly using the bash script - extract\_companies.sh in the Task 2 folder. An example run has been shown where evaluation happens on the evaluation dataset released. All the data files have also been added to the submission folder. Commented bash commands can be uncommented to run similar experiments on tweets and articles. The output of the evaluation is dumped in a JSON file in the following format -

[

{

"doc\_id": ~~~~,

"raw\_text": ~~~~,

"text": ~~~~,

"companies": <list of gold companies if there in data, otherwise empty>,

"company\_extractions": {

<company name>: [

[

<company instance>,

[

<sentences with context containing that instance>

]

]

]

}

},

.

.

.

]For reference see data/evaluation\_data.jsons.

* 1. **Time Taken:** Time taken to run this for evaluation\_data -

real 4m12.096s

user 4m9.866s

sys 0m0.978s

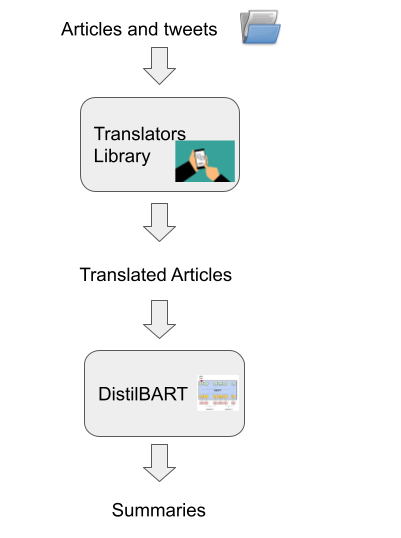
Done one 2.3 GHz Quad-Core Intel Core i7 machine

This Json file is then used to make predictions, this is done by passing the tokenized paragraphs(into sentences) with the brand names to the fine tuned m-BERT model(trained as described in preliminary submission doc.) which then generates one out of three possible outcomes; we combine the list of all predictions made for a given brand [ex:[1,0,1,1](negative for this brand if the brand is mentioned 4 times], we then take the brand sentiment to be negative or positive depending on the number of number of occurrences of the positive or negative class(whichever appears more is taken to be the overall sentiment); If neither positive nor negative sentence is present then we take the brand sentiment to be neutral.

Trained on kaggle using gpu: the code is in the file Task2/subtask-2\_sentiment\_prediction.ipynb

Estimated training time: (40 min)

Prediction time: (90 sec)

1. **Subtask3- Headline Generation/ Summarisation**
   1. **Approach-** We start with distilBART, a distilled (lighter) version of BART, denoising autoencoders for pretraining sequence-to-sequence models. For fine tuning on the given dataset, we use all the articles at the initial stages and not just the domain ones. 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 finetune it for 5000 steps. 
   2. **Steps to Reproduce-** For preprocessing:

run translate.sh with the “evaluation\_data.csv” in the same path Will generate a test.source file

All relevant preprocessed data available at: <https://drive.google.com/drive/folders/1IIWsGj8aLcSr8NLev4rAooOxpkIrhJQ7?usp=sharing>

For model training and evaluation:

Train\_and\_eval.ipynb run on colab. Note that paths need to be changed and a transformer library installed before running. A small change (at line 118 replace self.sharded\_dpp by false in seq2seq\_trainer.py) after installation.

The trained model is available at <https://drive.google.com/drive/folders/13IurxI9KMti9QFtEw57XsPxZVEwBhUo3?usp=sharing>

For training as well as evaluation, you will need to run different parts of the code in the notebook.

* 1. **Time to run:** Preprocessing takes around 30-40 seconds per article. Inference using the model takes 1 second per article. Total Training Time:Around 2.5 hours for 5000 steps on Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz