Statistical Methods in AI

Team 59 - Mission Learning

Gokulraj R - 2020102042
Chinmay Deshpande - 2020102069
Sushil Kumar Yalla - 2020102071

Project Topic - A Deep Learning Model Based on BERT and Sentence Transformer for Semantic Keyphrase Extraction on Big Social Data

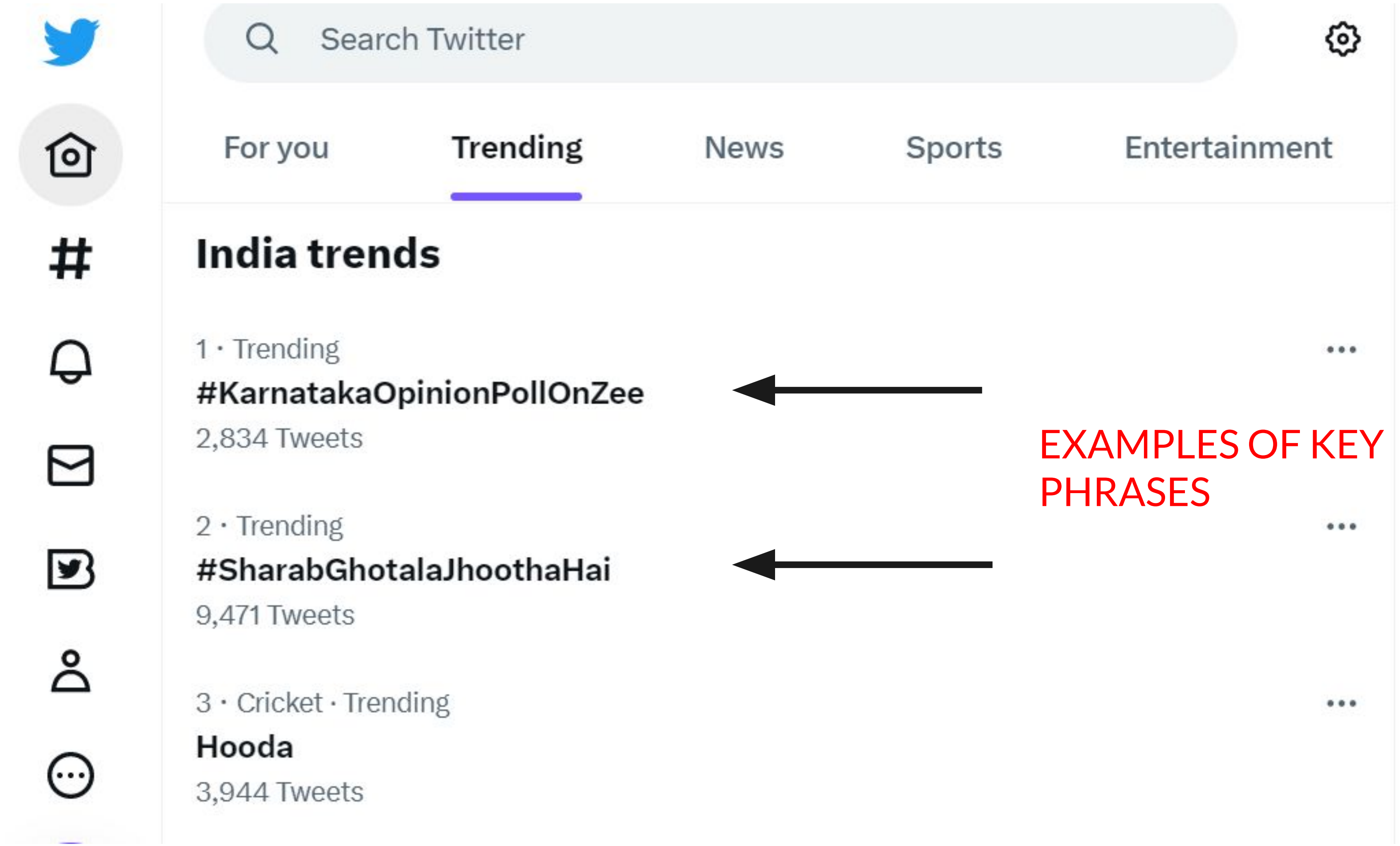
Introduction

‘A Deep Learning Model Based on BERT and Sentence Transformer for Semantic Keyphrase Extraction on Big Social Data was a paper published’ by **R. DEVIKA , SUBRAMANIYASWAMY VAIRAVASUNDARAM , C. SAKTHI JAY MAHENTHAR , VIJAYAKUMAR VARADARAJAN , AND KETAN KOTECHA**

In the evolution of the internet with social platforms like , twitter there are ‘for you’ and trending pages with certain phrases to attract the user to engage in the discourse of the respective tweet.

These hottopics or words of interest are commonly found using methods like Keyphrase extraction which is a commonly used tool for various different social data analysis methods, such as sentiment analysis, text classification and more.

This paper proposes a method to extract these keyphrases from tweets on certain datasets using sentence transformers which use the **Bidirectional Encoder Representation Transformers (BERT)** deep learning model.



India trends

1 · Trending
#KarnatakaOpinionPollOnZee
2,834 Tweets

2 · Trending
#SharabGhotalaJhoothaHai
9,471 Tweets

3 · Cricket · Trending
Hooda
3,944 Tweets

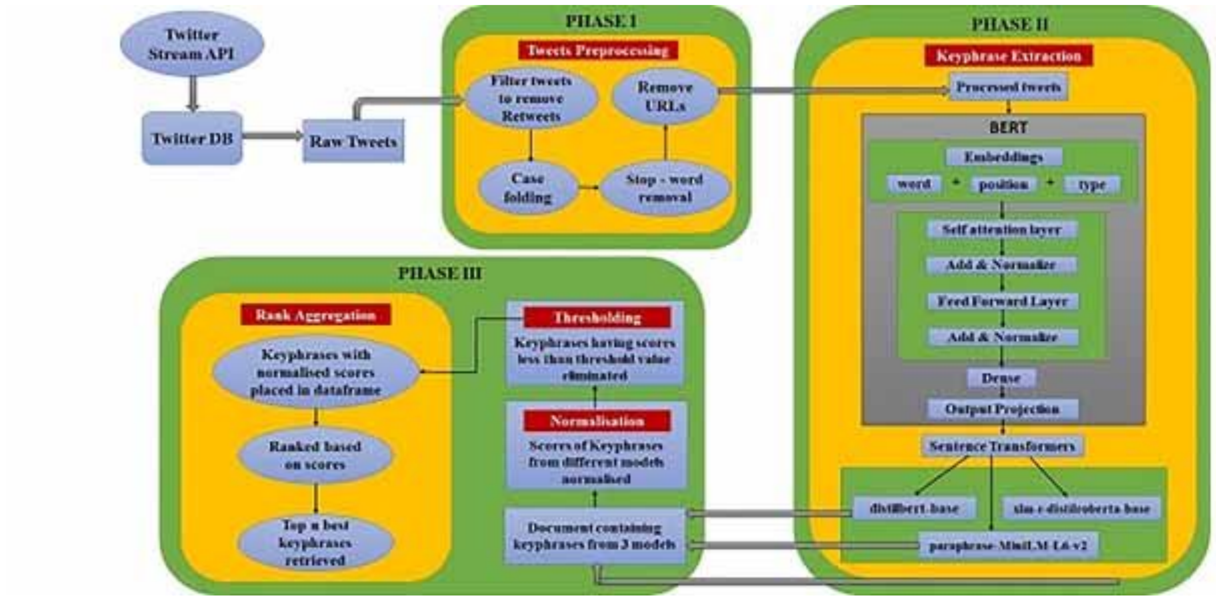
EXAMPLES OF KEY PHRASES

Workflow

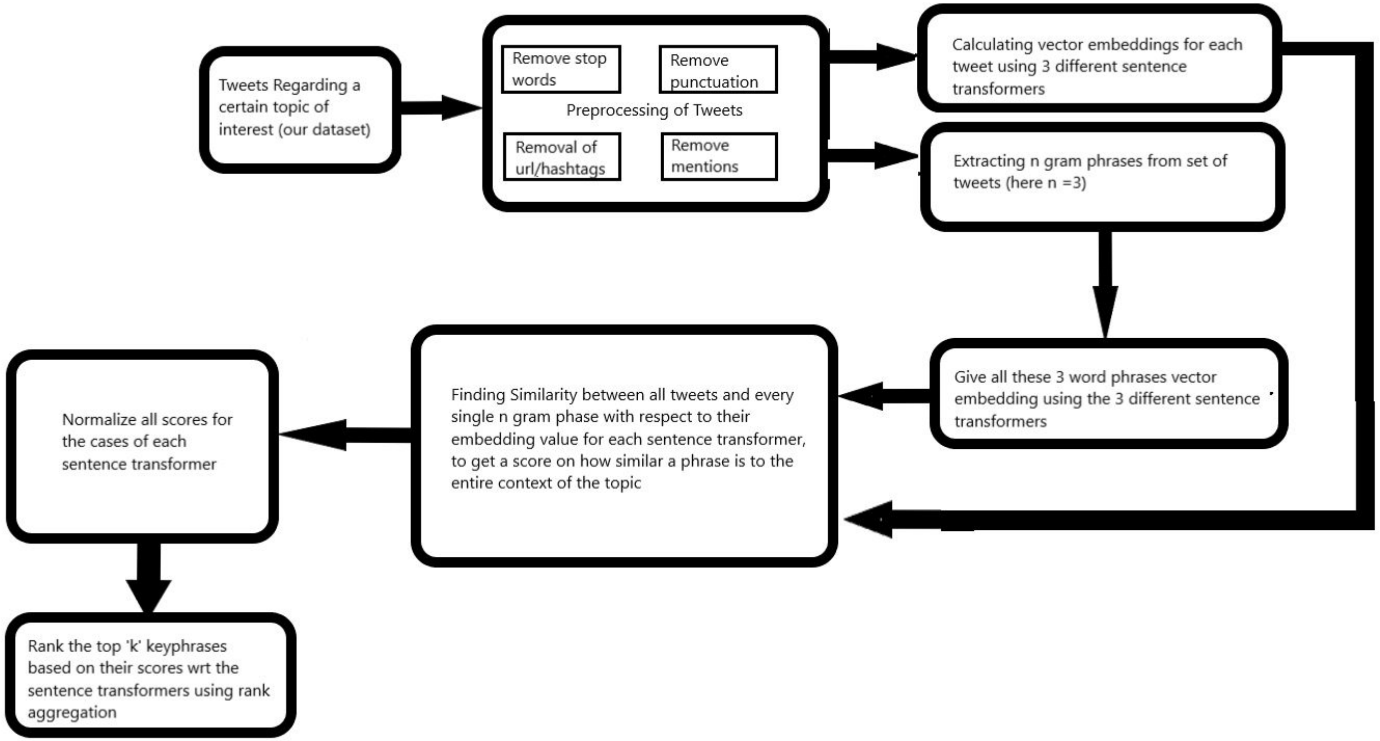
The project as a whole consists of four parts:

1. Preprocessing the tweets
2. Extracting the embeddings
3. Calculating similarities
4. Scoring the keyphrases and ranking them

Proposed Model in the paper



Workflow Diagram



Tweets Regarding a certain topic of interest (our dataset)
Remove stop words
Remove punctuation
Preprocessing of Tweets
Removal of url/hashtags
Remove mentions
Calculating vector embeddings for each tweet using 3 different sentence transformers
Extracting n gram phrases from set of tweets (here n =3)
Give all these 3 word phrases vector embedding using the 3 different sentence transformers
Finding Similarity between all tweets and every single n gram phase with respect to their embedding value for each sentence transformer, to get a score on how similar a phrase is to the entire context of the topic
Normalize all scores for the cases of each sentence transformer
Rank the top 'k' keyphrases based on their scores wrt the sentence transformers using rank aggregation

Datasets Used

- National Education Policy Tweets
- Covid-19 Tweets
- Cyber Crime Tweets
- Tokyo Olympics 2021 Tweets

You can find the datasets used in this project here.

Part - 1 : Preprocessing

Preprocessing

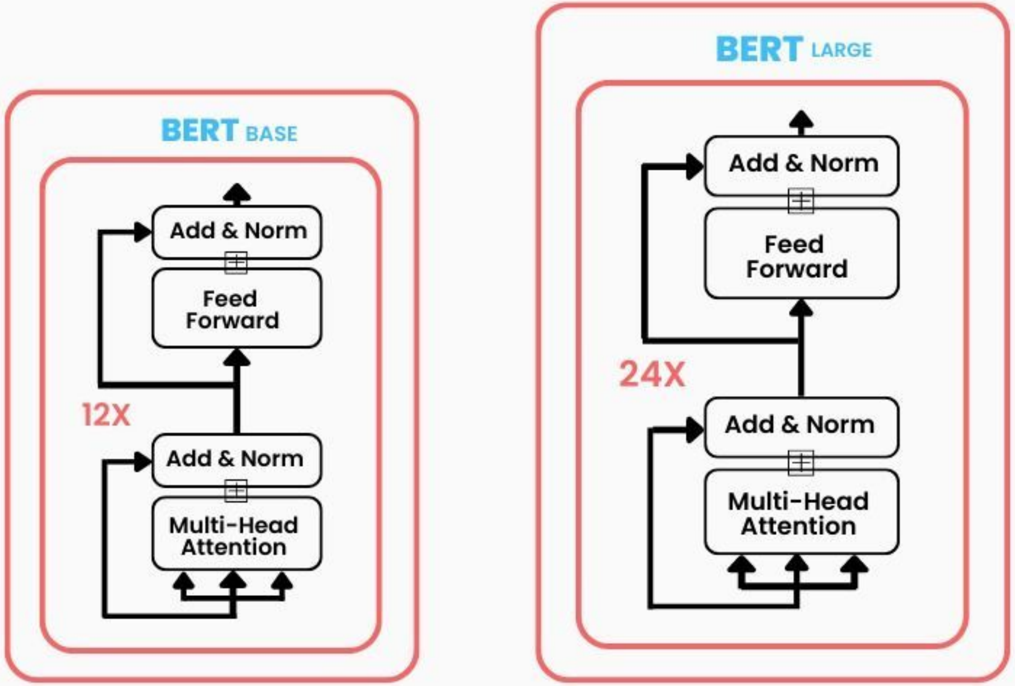
- We obtained our datasets premade from Kaggle, but they can just as easily be scraped using either the Twitter API and Tweepy or the snscrape library
- For preprocessing, we removed mentions of other users, URLs, punctuations and the hash character. We also removed the stopwords using the stopwords dictionary from nltk.
- Then we put them into a dataframe, so that we can quickly access them in further parts.
- In order to remove URLs and mentions, we used the re.sub function.
- We used

Part 2 - Embedding Extraction

Embeddings - What and Why

- Embeddings are a method of representing words as vectors. Many models such as BERT, are capable of converting words into vectors on the basis of the vocabulary that they are trained with, as well as the semantic features and contexts associated with the word itself.
- Now we need these vectors so that we can process words and sentences on a computer using matrix methods and/or transforms.
- Sentence embeddings are similar to word embeddings and share the exact same core idea and purpose, the only difference being that the vectors so formed represent sentences, or collections of words rather than a single word.
- We can apply vector properties on these vectors as well, which is what we do when assigning similarity scores

BERT Size & Architecture



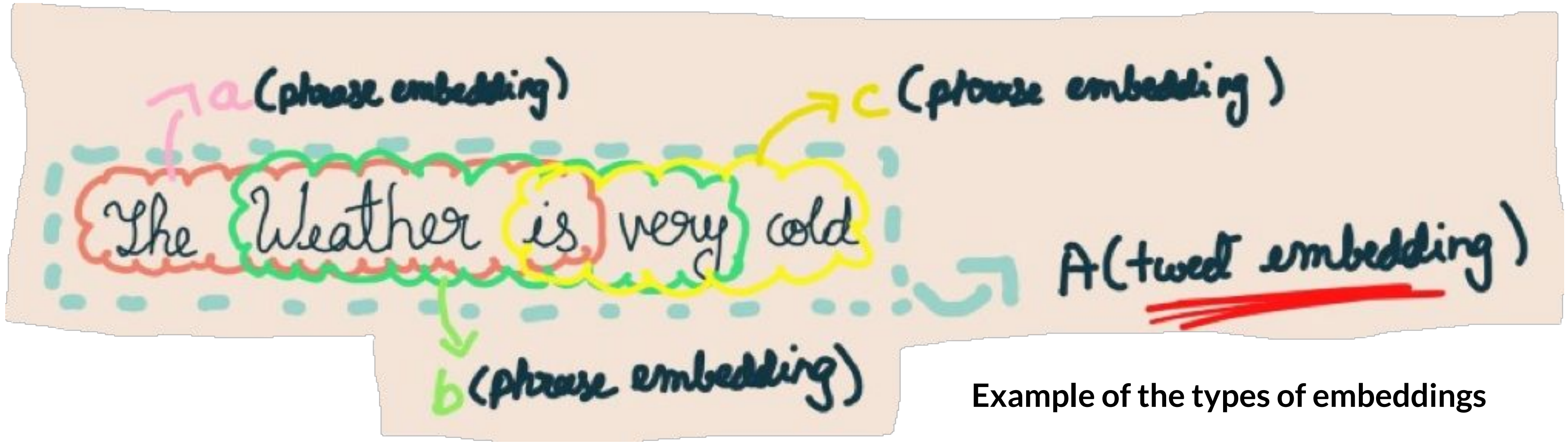
110M Parameters

340M Parameters

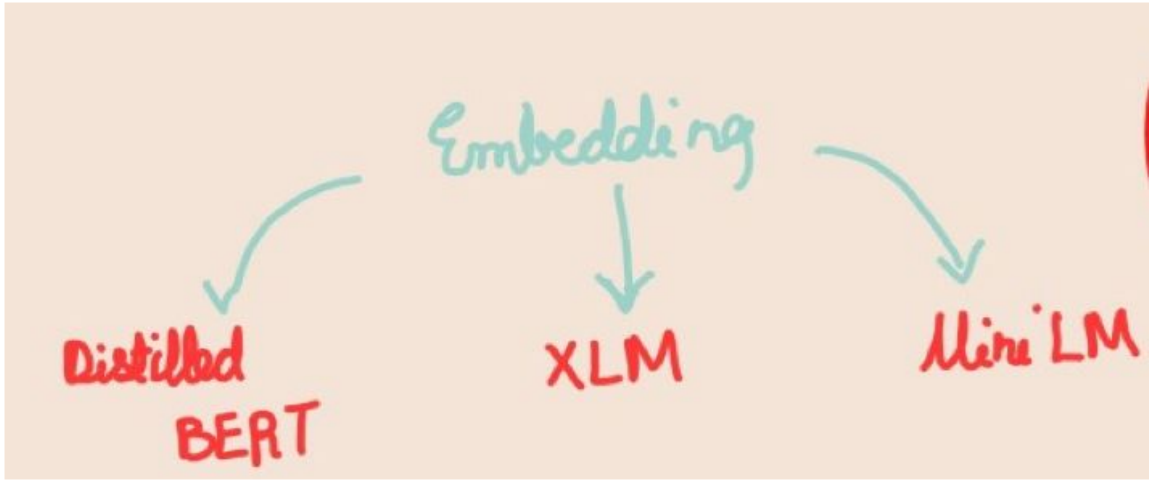
Turing

Embeddings Used

- For this project, we used sentence embeddings for two types of cases. Firstly, we extracted the sentence embeddings for every tweet in the dataset.
- After that, we used N-Gram splitting to split each tweet into phrases. We calculated the sentence embeddings for these individual phrases as well.
- For obtaining these embeddings, we used three sentence transforms - Distilled BERT, XLM and MiniLM which are all based on **BERT architecture**
- Sentence embeddings for the tweets and N-gram phrases were calculated separately for each of these transformers.



Example of the types of embeddings



We obtain embeddings for both the phrases for tweets using these 3 sentence transformers

Part 3 - Similarity Calculation

Similarity Calculation

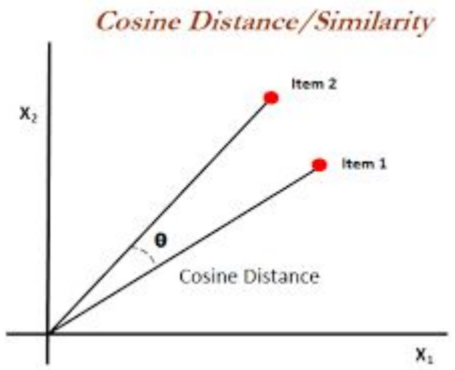
- Now to find a key phrase , so we need to a find a phrase which is very similar to the overall context of the dataset .

- We already have the sentence and phrase vectors from the previous step. We can use a cosine similarity function to see the similarity between each phrases with respect to all tweets and calculate the mean which gives us the phrase similarity with respect to the data set and give us some scores.

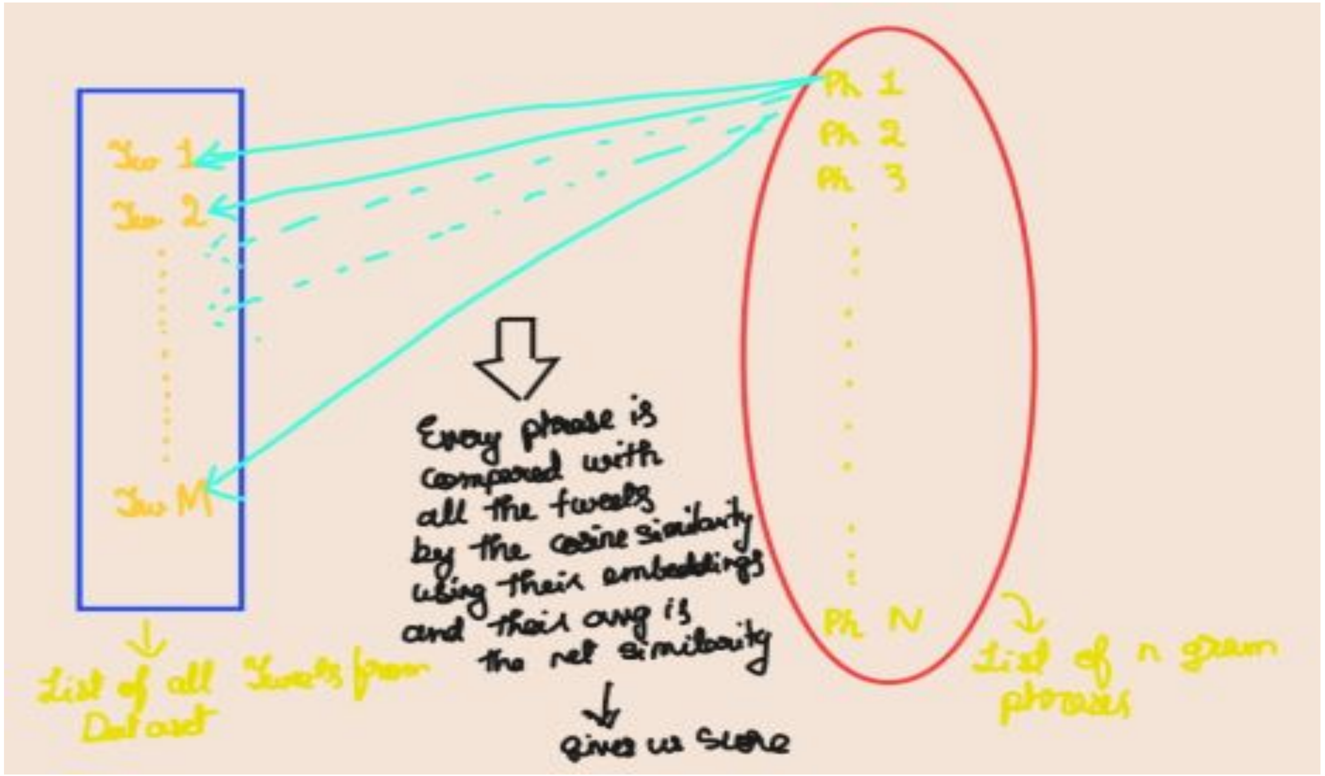
Similarity Calculation - How

- We get different scores with respect to which of the 3 sentence transformers we applied .

- Now we make a list of all these scores , wrt the three transformers and save them with their corresponding phrases before taking them to the next stage



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$


Tw 1
Tw 2
Tw M
List of all Tweets from Dataset
Ph 1
Ph 2
Ph 3
Ph N
List of n gram phrases
Every phrase is compared with all the tweets by the cosine similarity using their embeddings and their avg is the net similarity
gives us score

Brief explanation of how the scoring system is for each phrase

Part 4 - Scoring and Ranking

Thresholding

- We apply thresholding , here to take care of time constraints and to remove completely irrelevant phrases.
- After we get the cosine similarities we apply normalization to get the scores between 0 and 1
- **Upon trial and error we chose a threshold value of 0.3**
- Here we implement our technique to remove these irrelevant phrases.
- So we checked scores wrt each sentence transformer and made sure the if the phrase score is less than the threshold for any of the transformers we remove them from the set of n gram phrases

Scoring & Ranking

- Now with irrelevant phrases removed , we calculate the net score of each phrase for all the three transformers by using a average which uses the below following formula
- **Net score = ((XLM score)+(Distil-bert score)+(Mini LM score)) / 3**
- Then based on the scores of these above weighted average we rank them accordingly and take then top n ranks as our desired keyphrase outputs

Comparison with KeyBERT

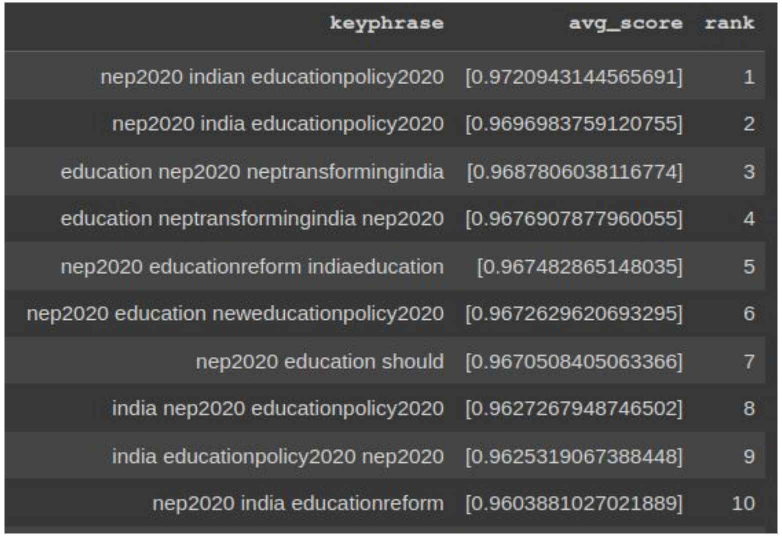
Comparison with KeyBERT

We used the KeyBERT tool to extract the ranked keyphrases and used them for comparison. There was a sizable mismatch between our results and the keyphrases extracted from KeyBERT. This is likely due to the fact that the models in our approach and KeyBERT were different, and that they also may have used different comparison metrics and ranking methods.

Since there was not any mention of the ground truth they have used to test the model in the paper, we compared the result with models like KeyBERT.

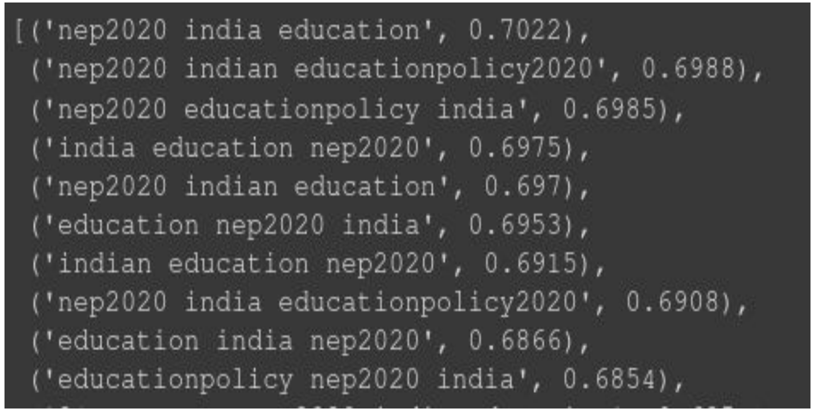
On NEP tweets dataset

Keyphrases extracted using our model:



	keyphrase	avg_score	rank
	nep2020 indian educationpolicy2020	[0.9720943144565691]	1
	nep2020 india educationpolicy2020	[0.9696983759120755]	2
	education nep2020 neptransformingindia	[0.9687806038116774]	3
	education neptransformingindia nep2020	[0.9676907877960055]	4
	nep2020 educationreform indiaeducation	[0.967482865148035]	5
	nep2020 education neweducationpolicy2020	[0.9672629620693295]	6
	nep2020 education should	[0.9670508405063366]	7
	india nep2020 educationpolicy2020	[0.9627267948746502]	8
	india educationpolicy2020 nep2020	[0.9625319067388448]	9
	nep2020 india educationreform	[0.9603881027021889]	10

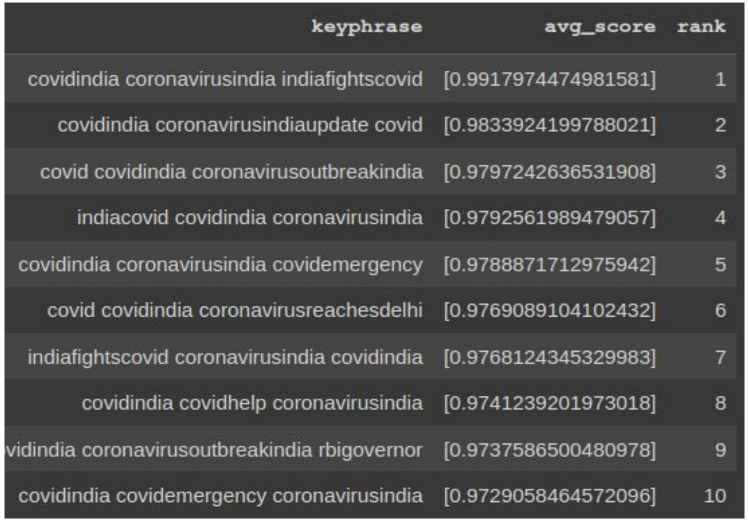
Keyphrases extracted using KeyBERT:



```
[('nep2020 india education', 0.7022),
 ('nep2020 indian educationpolicy2020', 0.6988),
 ('nep2020 educationpolicy india', 0.6985),
 ('india education nep2020', 0.6975),
 ('nep2020 indian education', 0.697),
 ('education nep2020 india', 0.6953),
 ('indian education nep2020', 0.6915),
 ('nep2020 india educationpolicy2020', 0.6908),
 ('education india nep2020', 0.6866),
 ('educationpolicy nep2020 india', 0.6854),
```

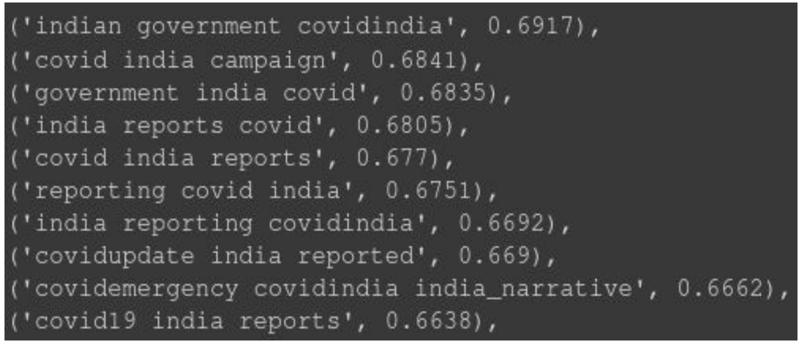
On Covid-19 tweets dataset

Keyphrases extracted using our model:



keyphrase	avg_score	rank
covidindia coronavirusindia indiafightscovid	[0.9917974474981581]	1
covidindia coronavirusindiaupdate covid	[0.983392419978 8021]	2
covid covidindia coronavirusoutbreakindia	[0.9797242636531908]	3
indiacovid covidindia coronavirusindia	[0.9792561989479057]	4
covidindia coronavirusindia covidemergency	[0.9788871712975942]	5
covid covidindia coronavirusreachesdelhi	[0.9769089104102432]	6
indiafightscovid coronavirusindia covidindia	[0.9768124345329983]	7
covidindia covidhelp coronavirusindia	[0.9741239201973018]	8
vidindia coronavirusoutbreakindia rbigovernor	[0.9737586500480978]	9
covidindia covidemergency coronavirusindia	[0.9729058464572096]	10

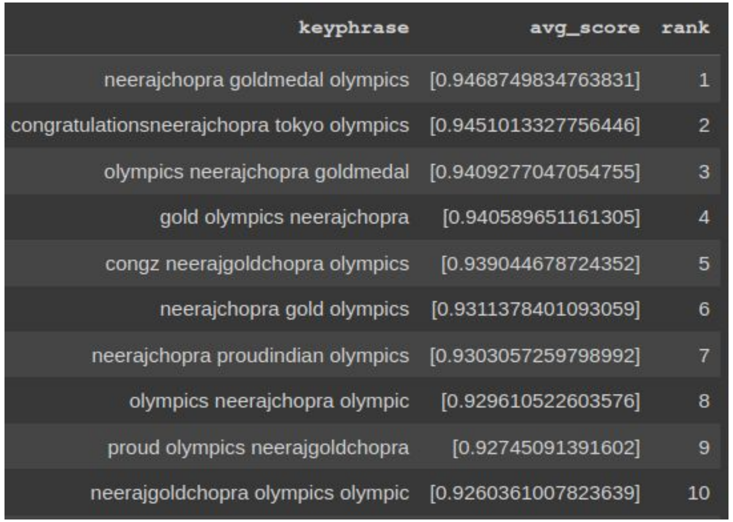
Keyphrases extracted using KeyBERT:



```
('indian government covidindia', 0.6917),
('covid india campaign', 0.6841),
('government india covid', 0.6835),
('india reports covid', 0.6805),
('covid india reports', 0.677),
('reporting covid india', 0.6751),
('india reporting covidindia', 0.6692),
('covidupdate india reported', 0.669),
('covidemergency covidindia india_narrative', 0.6662),
('covid19 india reports', 0.6638),
```

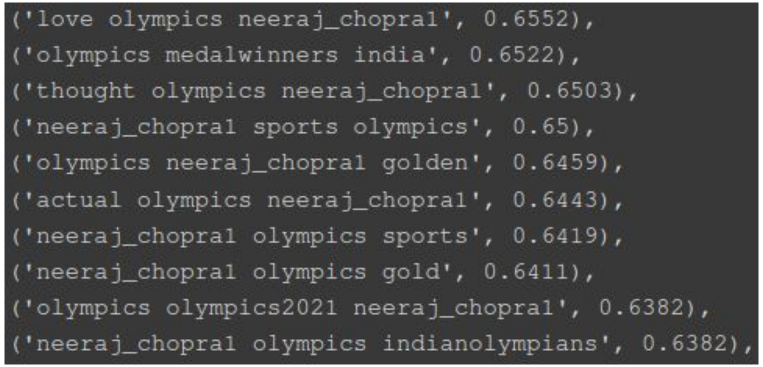
On Tokyo Olympics 2021 tweets dataset

Keyphrases extracted using our model:



keyphrase	avg_score	rank
neerajchopra goldmedal olympics	[0.9468749834763831]	1
congratulationsneerajchopra tokyo olympics	[0.9451013327756446]	2
olympics neerajchopra goldmedal	[0.9409277047054755]	3
gold olympics neerajchopra	[0.940589651161305]	4
congz neerajgoldchopra olympics	[0.939044678724352]	5
neerajchopra gold olympics	[0.9311378401093059]	6
neerajchopra proudindian olympics	[0.9303057259798992]	7
olympics neerajchopra olympic	[0.929610522603576]	8
proud olympics neerajgoldchopra	[0.92745091391602]	9
neerajgoldchopra olympics olympic	[0.9260361007823639]	10

Keyphrases extracted using KeyBERT:



```
('love olympics neeraj_chopra1', 0.6552),
('olympics medalwinners india', 0.6522),
('thought olympics neeraj_chopra1', 0.6503),
('neeraj_chopra1 sports olympics', 0.65),
('olympics neeraj_chopra1 golden', 0.6459),
('actual olympics neeraj_chopra1', 0.6443),
('neeraj_chopra1 olympics sports', 0.6419),
('neeraj_chopra1 olympics gold', 0.6411),
('olympics olympics2021 neeraj_chopra1', 0.6382),
('neeraj_chopra1 olympics indianolympians', 0.6382),
```

Problems Faced and Contributions

Problems Faced

- The paper was very vague and very poorly worded. There were almost no clear details on the process followed behind the paper. Whatever details were provided were inconsistent, with the chosen algorithm describing something and the flow diagram describing something else
- Because of this, we had to make a lot of guesses about what may have been going on

Contributions

Chinmay - Presentation and preprocessing

Gokulraj - Embedding extraction and similarity calculation, ranking

Sushil - Scoring, flow diagram, ranking

THANK YOU!!