

# Identifying Fake News

Project Option I

By: Jamie Mandolini



# 01

## Pre-Processing

# Data Cleaning and Formatting

## Cleaning Methods

- Change uppercase to lowercase
- Remove punctuation, digits, and stopwords
- Tokenize and lemmatize

## Formatting

- Create corpus and convert tokens to numerical representation using TFIDF
- Create new dataframe with features:
  - Cosine similarity
  - Polarity for each title
  - Subjectivity for each title

\*Note: EDA showed a very uneven dataset



02

Model  
Training

# Model Training: Choosing a Classifier

Classifiers used:

- Unweighted and balanced Random Forest
- Gaussian Naive Bayes
- Logistic Regression (one vs rest model)
- Unweighted, balanced, and weighted Multinomial Logistic Regression



# 03

# ANALYSIS

# Performance Metrics

Multinomial Logistic Regression Performance

	precision	recall	f1-score	support
agreed	0.66	0.46	0.54	22176
disagreed	0.00	0.00	0.00	2012
unrelated	0.78	0.91	0.84	52745
accuracy			0.76	76933
macro avg	0.48	0.46	0.46	76933
weighted avg	0.72	0.76	0.73	76933

Balanced Multinomial Logistic Regression Performance

	precision	recall	f1-score	support
agreed	0.61	0.55	0.58	22176
disagreed	0.03	0.22	0.06	2012
unrelated	0.86	0.73	0.79	52745
accuracy			0.66	76933
macro avg	0.50	0.50	0.48	76933
weighted avg	0.77	0.66	0.71	76933



**Thanks!**

# How to get retweeted? Factor analysis on Twitter

---



**Chenjie Li**

**Killian Trolès**

# Problem



- What factors have significant influence on the retweet number ?
- Correlations among factors
- What are some of the most popular domains getting mentioned in Twitter in a certain period of time?

# Dataset and features extraction

---

- Twitter archive ("Spritzer" version, the most light and shallow of Twitter grabs)
  - 36M tweets in 2020-11
- Processed raw JSON files to a Database
- Aggregated content and extracted features (using SQL)
  - Factor Analysis (FA) features
    - hashtags\_count, urls\_count, mentions\_count, followers\_count, followees\_count, days, statuses\_count, favorites\_count, retweet\_count
  - Retweet rate features
    - hashtags, urls, mentions

# Retweet rate

---

$$\text{retweet\_rate} = \frac{\text{nb\_retweets\_with\_urls}}{\text{nb\_tweets\_with\_urls}} * \frac{\text{nb\_tweets}}{\text{nb\_retweets}}$$

Number of tweets: 36.3 Million

Number of retweets only: 17.8 Million (49%)

Domain X has 1 Million retweets

Domain X has 2 Million tweets



$$\text{retweet\_rate}(X) = (2/1) \times (36.3/17.8)$$

# Tops: Most used domains and retweets

---

Domain name	In tweets	In retweets	Retweet rate
twitter.com	15330194	9230686	1.22
youtu.be	344365	234083	1.38
bit.ly	298271	175639	1.19
instagram.com	124577	48046	0.78
flwrs.com	75979	107	0
onlyfans.com	67399	57924	1.74
twitch.tv	49293	17208	0.70
youtube.com	47524	20984	0.89
open.spotify.com	36578	16606	0.92
dlvr.it	28623	8873	0.630164

Table 2: 10 most used URLs in tweets

This is actually a twitter application that keep track of who follows and unfollows you on twitter

DON'T ASK US WHY THIS HAS THE HIGHEST RETWEET RATE!

# Tops: Most used hashtags and retweets

---

Hashtag	In tweets	In retweets	Retweet rate
AMAs	340379	195786	1.169276
MAMAVOTE	330389	261446	1.608624
2020MAMA	283494	232449	1.666794
BTS	242347	159710	1.339654
NCT	111602	92876	1.691725
TDYAwards	107885	24473	0.461131
FridayLivestream	94139	5357	0.115678
방탄소년단	90001	83466	1.885213
BTS_BE	80637	76286	1.923130
NCT_RESONANCE	78940	69401	1.787174

Table 3: 10 most used hashtags in tweets

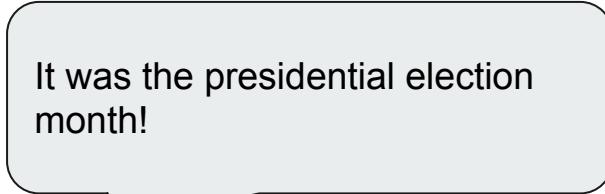
Some Korean band related hashtag

Another Korean band BTS related

# Tops: Most mentioned users

---

Username	Number of Mentions
BTS_twt	856704
realDonaldTrump	485581
MnetMAMA	448097
965TDY	189455
GOT7Official	137502
JoeBiden	131340
BLACKPINK	117122
weareoneEXO	114625
ShopeeID	85052
treasuremembers	84996



It was the presidential election month!

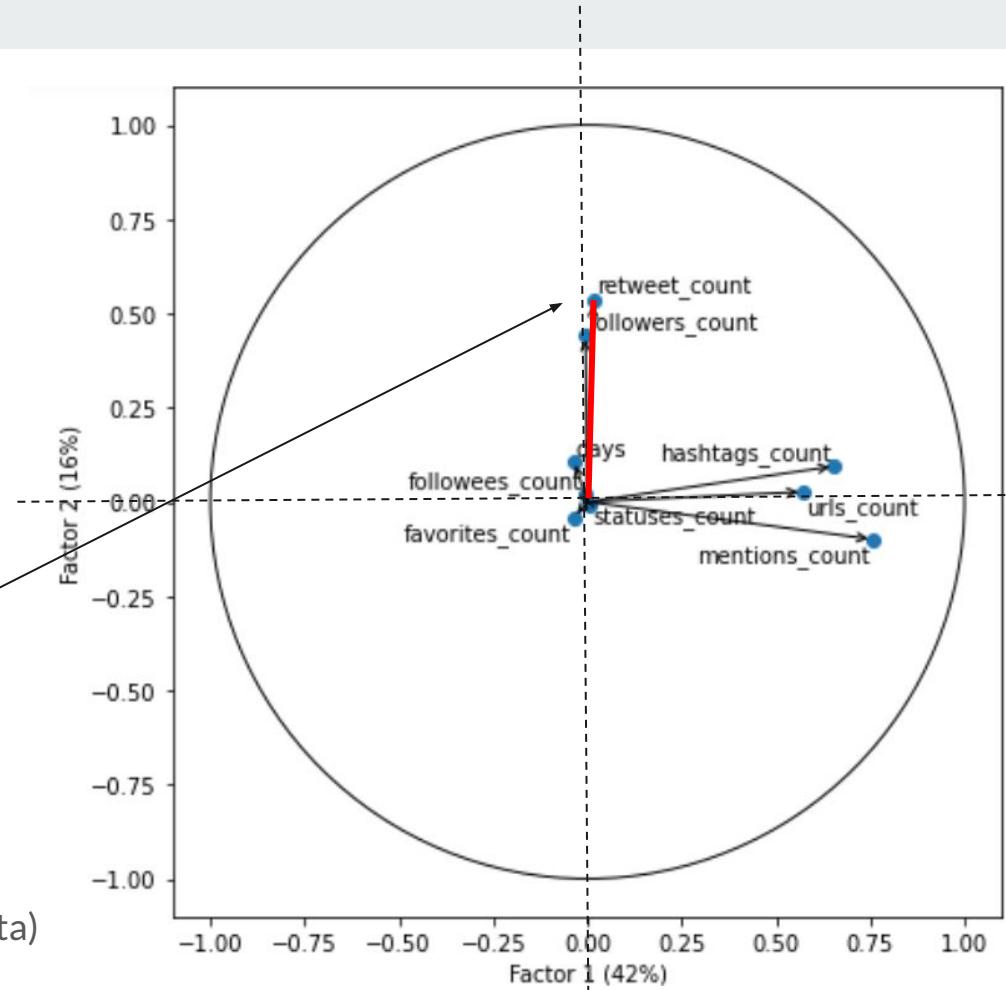
# Factor analysis

Find possible correlations between features. It determines new components (latent variables)

Quadrant of correlated factors with retweet count

Positively Correlated factors:

- Followers count
- Hashtag count
- The “age” of the account (*days* column in data)



# CS 579: Fake News Classification

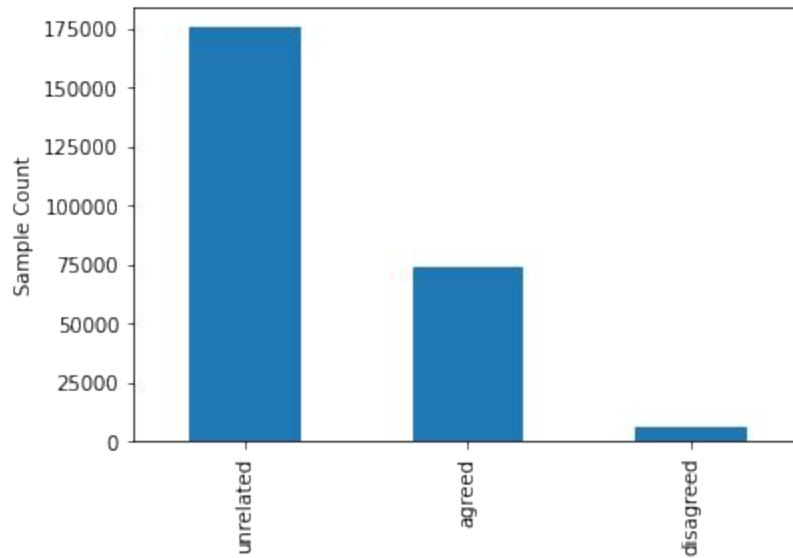
Sam Golden & Theo Guidroz

# Task

Given 2 news titles A and B, predict if:

- B talks about the same fake news as A
- B refutes the fake news in A
- B is unrelated to A

# Dataset



# Data Augmentation

Pick one sentence

Translate to French

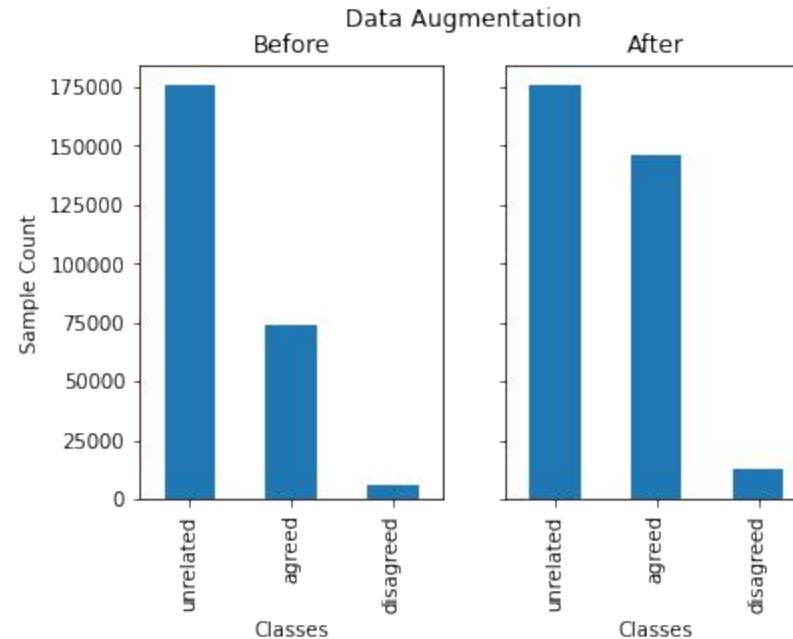
Translate back to English

Is beer belly because of beer?

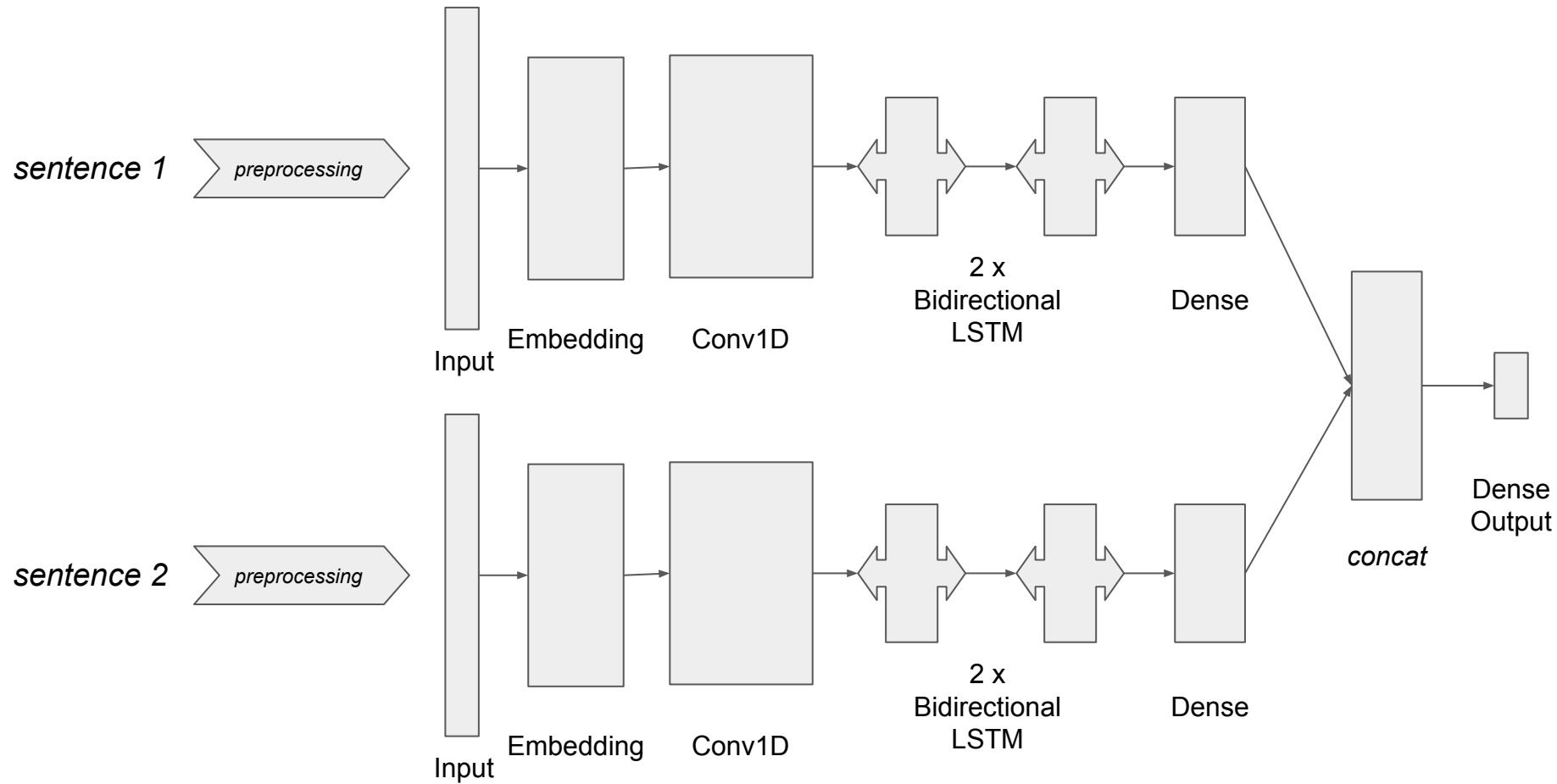
Le ventre de bière est-il causé par la  
bière?

Is beer belly caused by beer?

# Data Augmentation Results



# Model Architecture



# Results

Model Architecture	Validation Accuracy
Bidirectional LSTM	69.23 %
Bidirectional LSTM + pre-trained Embedding Layer	69.74 %
Bidirectional LSTM + post-concat fully connected section	71.16 %
Bidirectional LSTM + CNN	74.14 %

Unrelated:

The great coat brother Zhu Zhu Wen, in the mandarin love song to sing the song is really the lanca- talent is very sweet!

Lin xinsheng after the birth of "hard milking," Huo jianhua is not seen, "forced marriage" is real?

Agreed:

Game of Thrones has finally confirmed that season 8 will return in 2019

HBO "Game of Thrones," the final season of season 8 will be back in 2019!

Disagreed:

liu mingwei has lost his examination certificate

Liu Mingwei's test card has been lost? It's a rumor. Don't be fooled again!

# Supplementing Missing Visions via Dialog for Scene Graph Generations

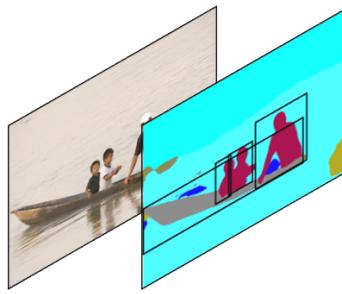
Ye Zhu (A20479446)  
Bin Duan (A20479442)  
April 2022

Paper and Code:  
<https://github.com/L-YeZhu/SI-Dial>

# Background & Motivation

## 1. Scene Graph Generation

Task: generate a graphical representation of the scene within the given image.



## 1. SGG with missing visual information, supplemented by natural language dialog:

- a. Multimodal learning
- b. Novel setting
- c. Practical scenarios

Original image

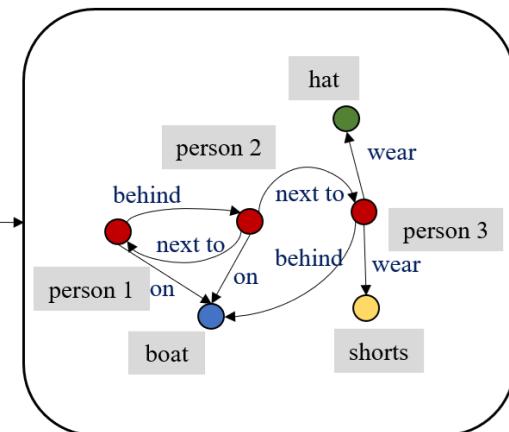
Image input w/  
missing vison

Q1: How many persons are there ?  
A1: Three.

Q2: What is the man wearing ?  
A2: A hat.  
.....

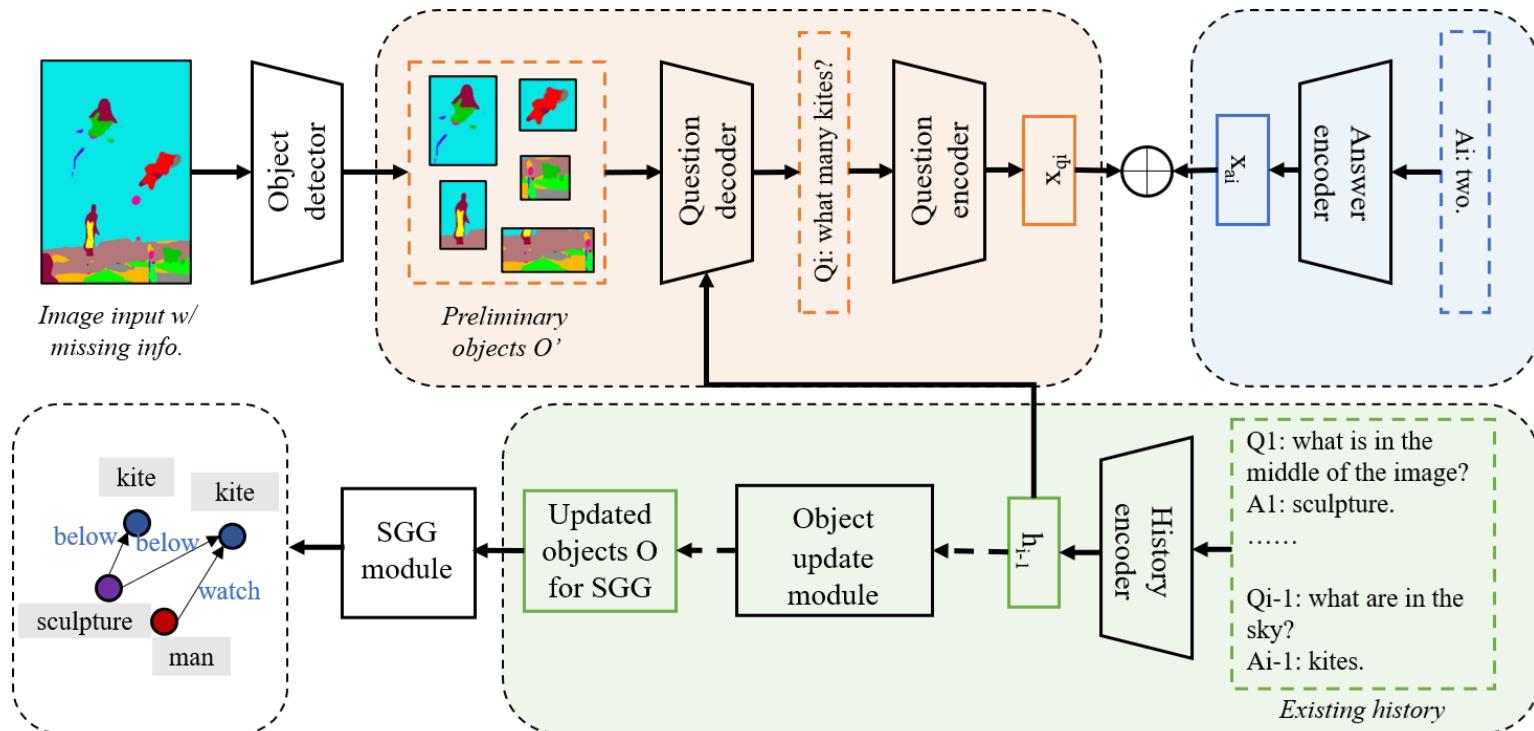
Q10: Where are they?  
A10: On a boat.

Dialog interaction  
as supplementary



Generated  
scene graph

# Methodology

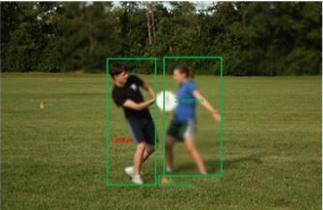
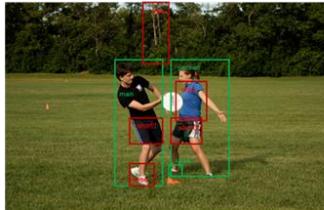


# Results and Evaluations

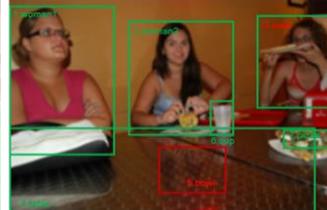
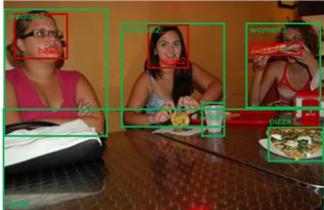
Vision Input	Model	Predicate Classification			Scene Graph Classification			Scene Graph Detection		
		mR@20	mR@50	mR@100	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100
Original VG	IMP <sup>†</sup>	-	9.8	10.5	-	5.8	6.0	-	3.8	4.8
	FREQ. <sup>†</sup>	8.3	13.0	16.0	5.1	7.2	8.5	4.5	6.1	7.1
	MOTIF <sup>†</sup>	11.5	14.6	15.8	6.5	8.0	8.5	4.1	5.5	6.8
	VCTREE <sup>†</sup>	11.7	14.9	16.1	6.2	7.5	7.9	4.2	5.7	6.9
Object Blur	MOTIF	11.23	14.36	15.71	6.20	7.51	7.90	4.13	5.48	6.82
	VCTREE	11.48	14.61	15.88	5.97	7.46	7.85	4.24	5.66	6.93
	MOTIF+Random QA	11.52	14.78	16.08	5.82	7.34	7.86	4.64	6.18	7.24
	VCTREE+Random QA	11.55	14.83	16.17	5.51	7.03	7.48	4.71	6.23	7.43
	MOTIF+SI-Dial	13.31	16.74	18.09	6.96	8.51	9.00	5.77	7.76	9.12
	VCTREE+SI-Dial	13.40	16.88	18.26	6.67	8.12	8.59	5.88	7.92	9.28
Image Blur	MOTIF	11.69	14.31	15.64	6.24	7.56	7.90	3.88	5.21	6.37
	VCTREE	11.72	14.38	15.78	6.03	7.39	7.83	3.82	5.18	6.33
	MOTIF+Random QA	11.57	13.93	15.14	6.75	8.21	8.69	4.10	5.39	6.26
	VCTREE+Random QA	11.70	14.26	15.53	6.68	8.03	8.55	4.07	5.34	6.25
	MOTIF+SI-Dial	12.90	16.26	17.91	8.41	10.33	11.00	5.05	6.96	8.23
	VCTREE+SI-Dial	13.62	17.18	18.49	7.93	10.02	10.86	5.24	7.08	8.11
Semantic Masked	MOTIF	11.61	14.28	15.57	4.45	5.41	5.68	2.80	3.89	4.76
	VCTREE	11.68	14.32	15.59	4.40	5.38	5.69	2.80	3.87	4.68
	MOTIF+Random QA	12.00	15.32	16.67	5.83	7.14	7.65	2.79	3.86	4.68
	VCTREE+Random QA	12.28	15.69	17.04	5.66	7.01	7.28	2.92	4.01	4.85
	MOTIF+SI-Dial	12.79	16.26	17.58	6.44	7.85	8.33	3.03	4.21	4.92
	VCTREE+SI-Dial	12.73	16.35	17.63	6.21	7.68	8.05	3.15	4.28	5.00

Table 1: Quantitative evaluations for the SGG with missing visions. The results are reported on mean Recall@K [Chen *et al.*, 2019b; Tang *et al.*, 2019]. Note that the models with <sup>†</sup> are implemented and reported using the previous version of Faster R-CNN as the object detector, while we re-implemented the models with the up-to-date ones.

# Results and Evaluations



1. What is this picture of ? A tennis player.
2. When was the woman playing a game in the backyard? Sunday morning.
3. What is the color of the shoe? Black.



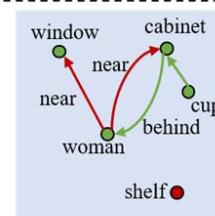
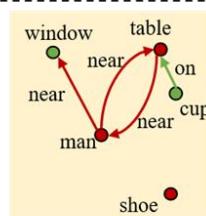
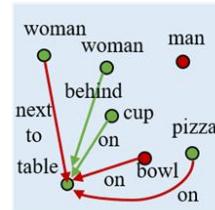
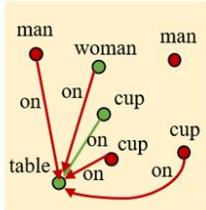
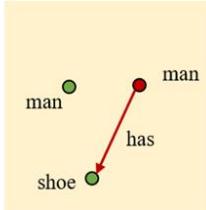
1. What is the woman doing? Using a computer.
2. What is the woman holding? Kite.
3. How many people? Two.
4. What color is the table? Brown.
5. What is on the table? Food.



1. What color is the window? Black.
2. What is on the table? Glass.
3. What is on the table? Glass.
4. Who is this? A woman.
5. Who is in the kitchen? The woman.
6. What is the person holding? Cup.
7. What is on the table? Food.
8. What color is the table? Brown.
9. Where is the cup? On the table.
10. What is the person holding? Cup.
11. What is on the table? Glass.

(a) Original images

(b) Images with missing visions



(c) Dialog interactions

(d) Baselines

(e) SI-Dial

# Modeling Data with Explainable Graph Neural Network

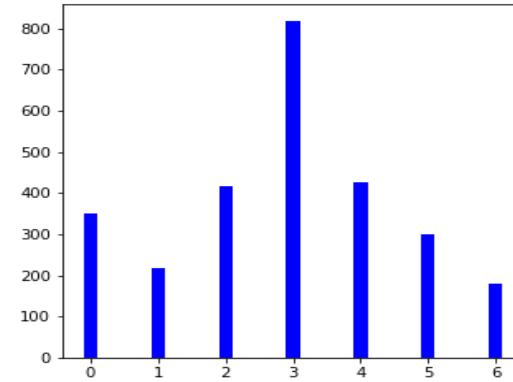
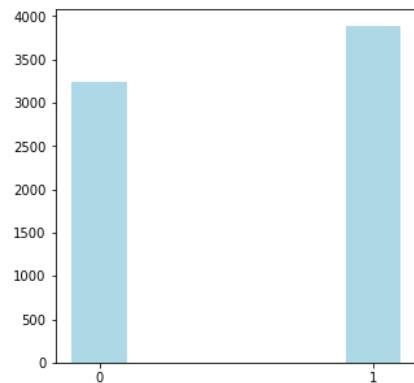
Shubham Ganesh Modi (A20492276)  
Oleksandr Shashkov (A20229995)

# Problem Statement

- The field on Artificial Intelligence has witnessed exponential advancements since the introduction of Deep Neural Networks.
- The working of Deep Neural Models and their inner structure however lacks transparency and creates a black box which provides little understanding of how the predictions are done and which part of the information was critical for the decision
- This drawback has led to research on many explainable Machine learning models and one such branch is Graph Neural Network (GNN).
- In this project we reproduce a published GNN explanation model which allows us to understand and visualize the logic behind predictions.

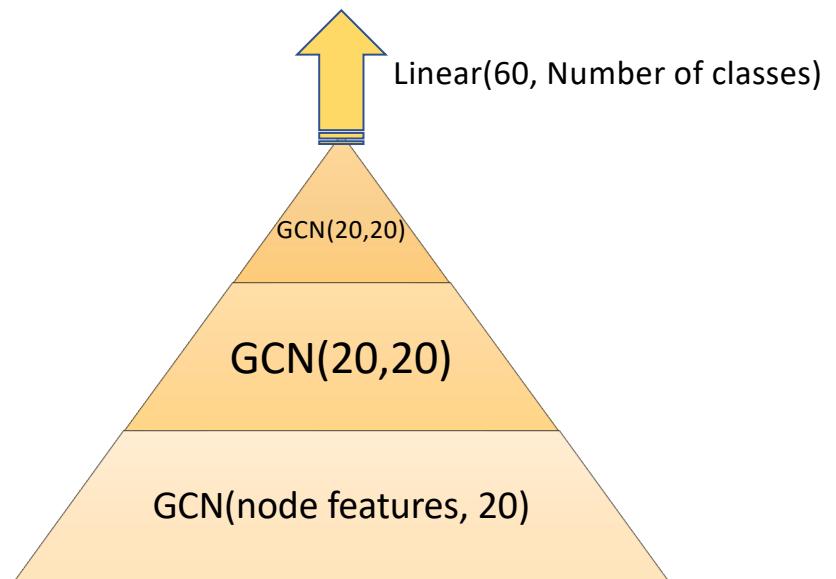
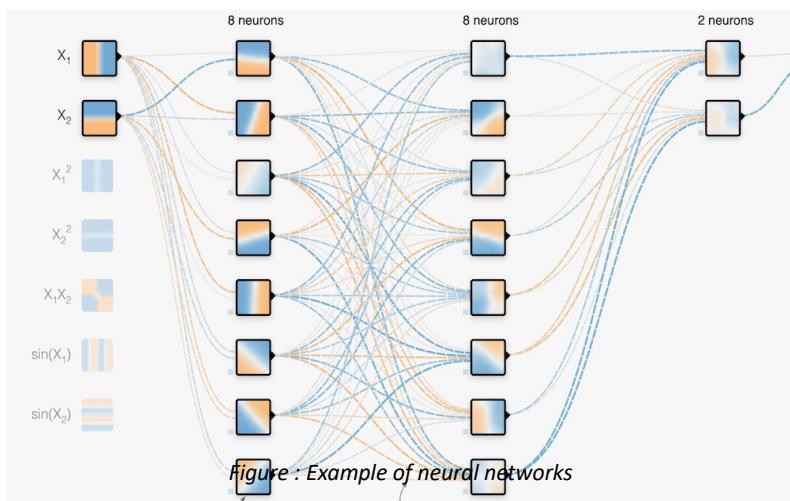
# Project Datasets

- **Twitch Dataset Explanation**
- Twitch dataset is a graph dataset consisting of 7126 nodes, where each node consists of 128 node features and the dataset has 2 classes.
- Each node represents a Twitch Streamer and edges are followership between them.
- The node features represent embeddings of games played by the Twitch users.
- The task is to predict whether a user stream make use of explicit language in their content.
- **Cora Dataset Explanation**
- Cora dataset is a graph dataset consisting of 2,708 nodes, where each node consists of 1433 node features and the dataset has 7 classes.
- Each node represents a document and edges represent citation links.
- The node features represent embeddings of words in a document.
- The task is to classify the documents in one of the 7 available categories such as Neural Networks, Reinforcement Learning, Probabilistic Methods, Theory, Rule Learning, Genetic Algorithms, Case Based.



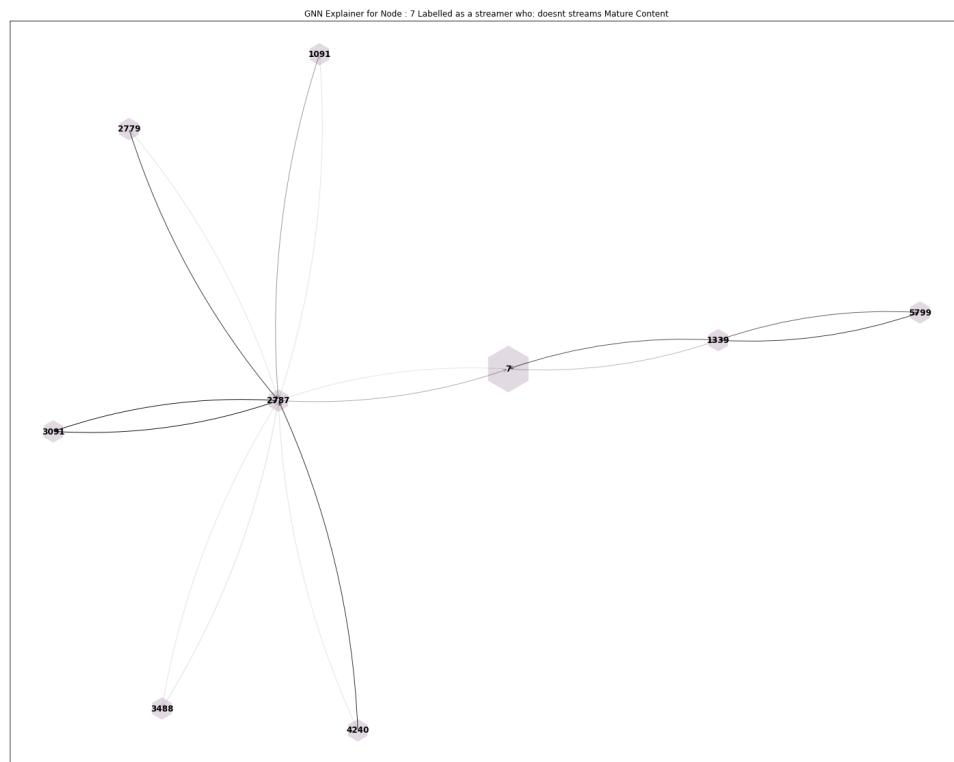
# Graph Neural Network Model

- The GNN model makes use of 3 GCN Convolution layers and a Linear layer.
- The model uses ReLU activation function.
- The model also makes use of L2 normalization for edge weights.



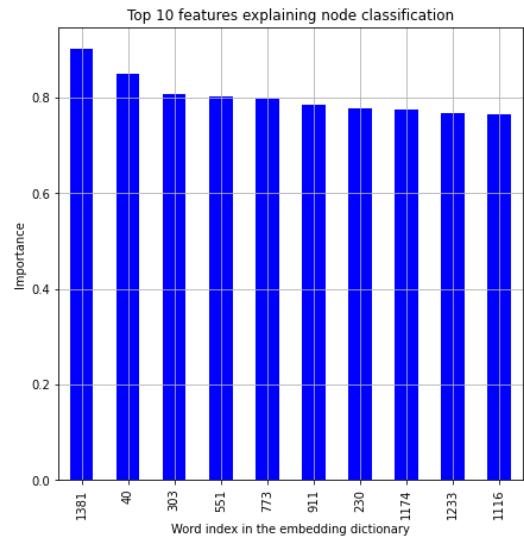
# Graph Neural Network Explainer

- After training the GNN model and initializing the GNN explainer using the best model we can select a node and explain it.
- The model aggregates information obtained from the target node's neighbors and understands which nodes contain important information which is passed on to the target node.
- This piece of information is important in determining the explanations for the target node to be classified with its correct label.

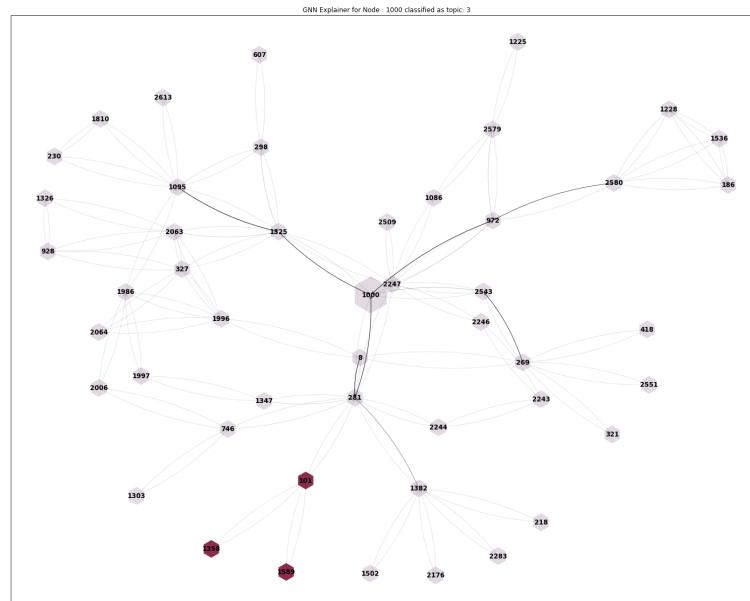


# GNN Explanation for Node in Cora dataset

## - Edges



- The subgraph is an explanation for why publication with index 1000 was classified as topic 3
- The edges highlighted in bold are the most influential connections explaining the prediction



# Online social network analysis

• • •

Project 2 : semantic comparison

- 180 000 labeled samples in training data
- 64 000 unlabeled samples we need to label
- 3 different labels : agreed, unrelated, disagreed

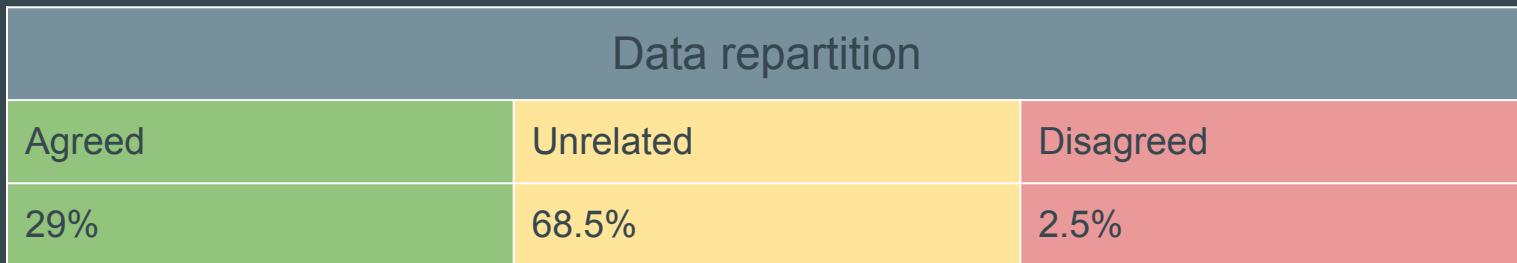
# Data

# Preprocessing

# Model & results

# Conclusion

Sample form					
row_id : int	title_1_id : int	title_2_id : int	title1_en : str	title2_en : str	label : str



- Preprocessing in 3 steps
  - Removing non alphanumeric symbols
  - Lowercasing the string
  - Removing any stop words in the string

# Data

# Preprocessing

# Model & results

# Conclusion

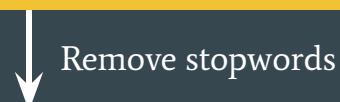
“This is string test 123 \$# stopwords : between yourself but again”



“This is string test 123 stopwords between yourself but again”



“this is string test 123 stopwords between yourself but again”



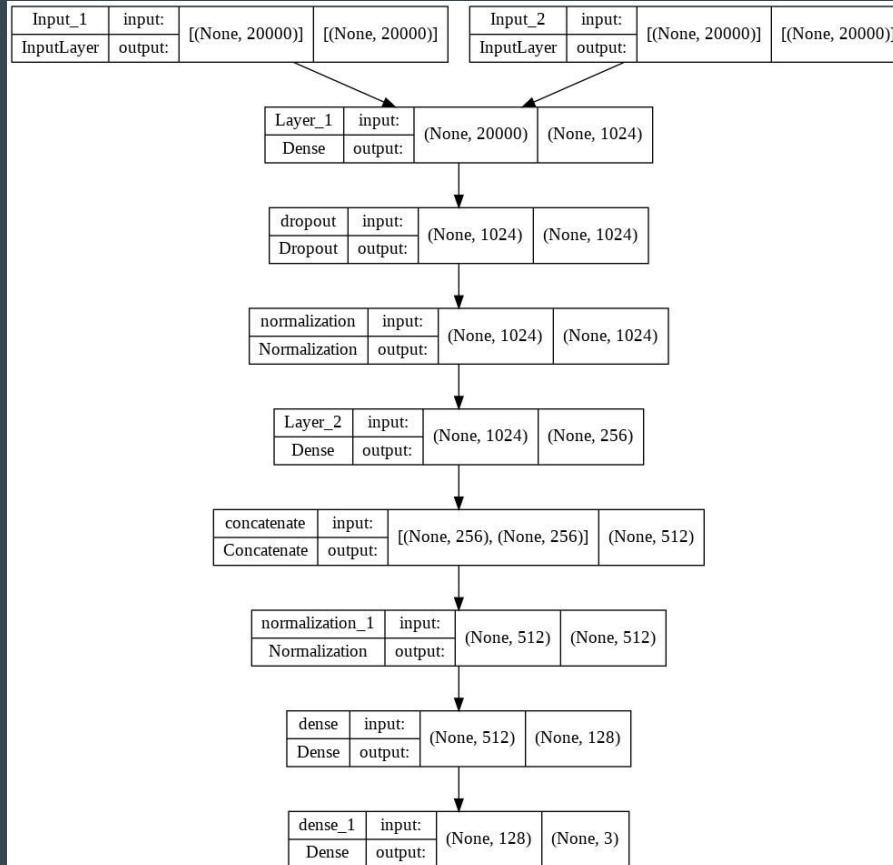
“string test 123 stopwords”

# Data

# Preprocessing

# Model & results

# Conclusion

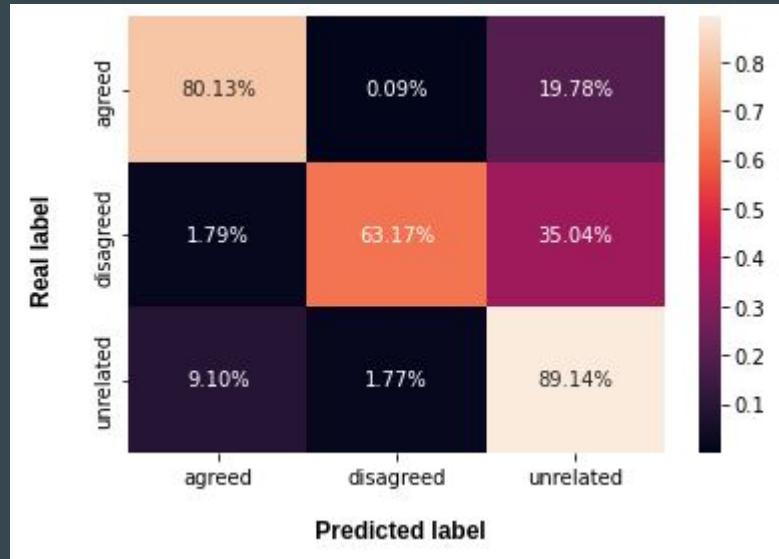


# Data

# Preprocessing

# Model & results

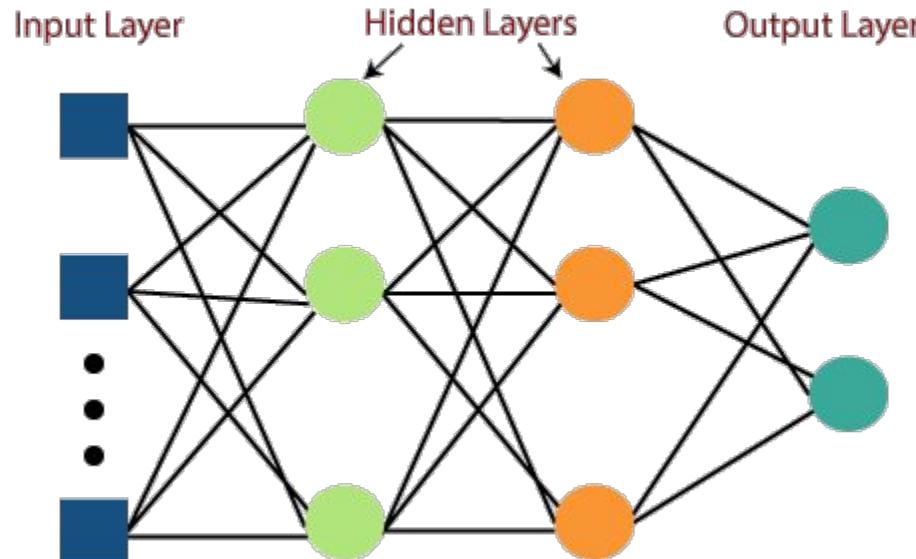
# Conclusion



	Precision	Recall	F1-score	support
agreed	80%	78%	79%	14902
disagreed	63%	52%	57%	1348
unrelated	89%	91%	90%	35039
accuracy			86%	51289
macro avg	77%	74%	75%	51289
weighted avg	86%	86%	86%	51289

Table 2: Classification report

## Basic MLP architecture

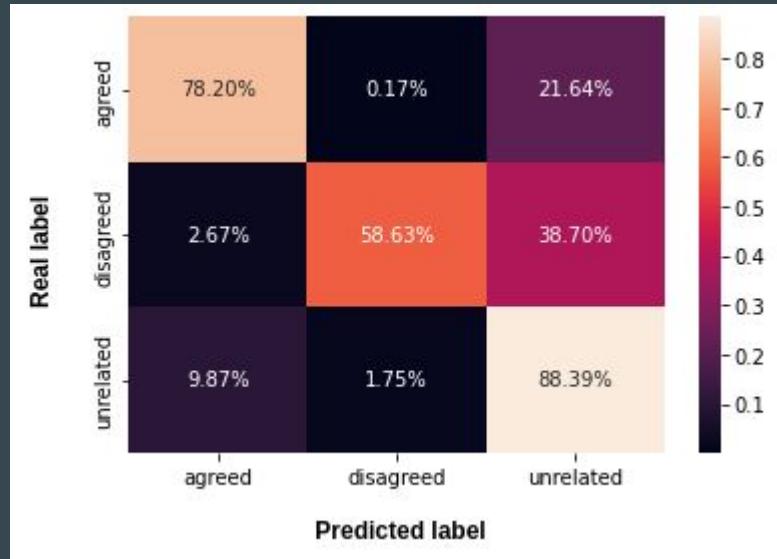


# Data

# Preprocessing

# Model & results

# Conclusion



	Precision	Recall	F1-score	support
agreed	78%	76%	77%	14902
disagreed	59%	52%	55%	1348
unrelated	88%	90%	89%	35039
accuracy			85%	51289
macro avg	75%	73%	74%	51289
weighted avg	85%	85%	85%	51289

Table 3: Classification report

Data

Preprocessing

Model & results

Conclusion

## Fake news detection: Binary classification?

(by merging “unrelated” and “agreed” label)

# Online Social Networks CS 579

Rodrigo López and Luis Castañeda

# Detection of Fake News based on how it has been propagated

# Index

1. Problem Description
2. State of Art
3. Proposed solution
4. Results

# Problem Description

- Emerging Digital Media
- Clickbait, retweets and likes
- Fake News

# State of the art

## Objective

Input



Output

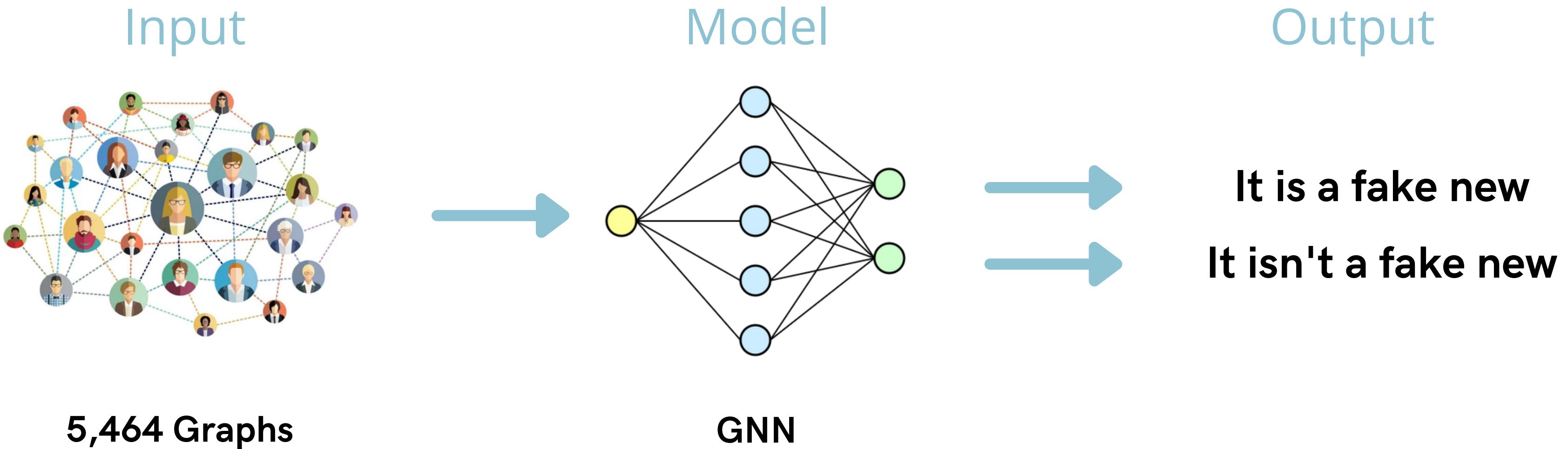
It is a fake new



It isn't a fake new

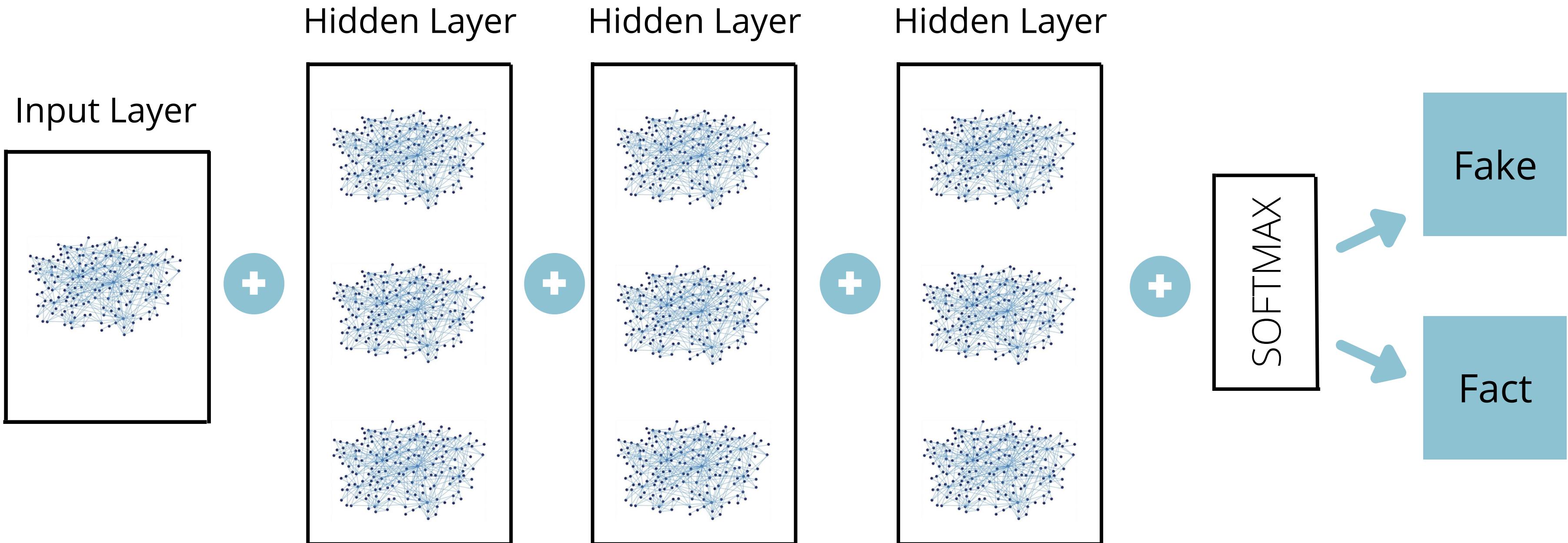
# State of the art

## Process





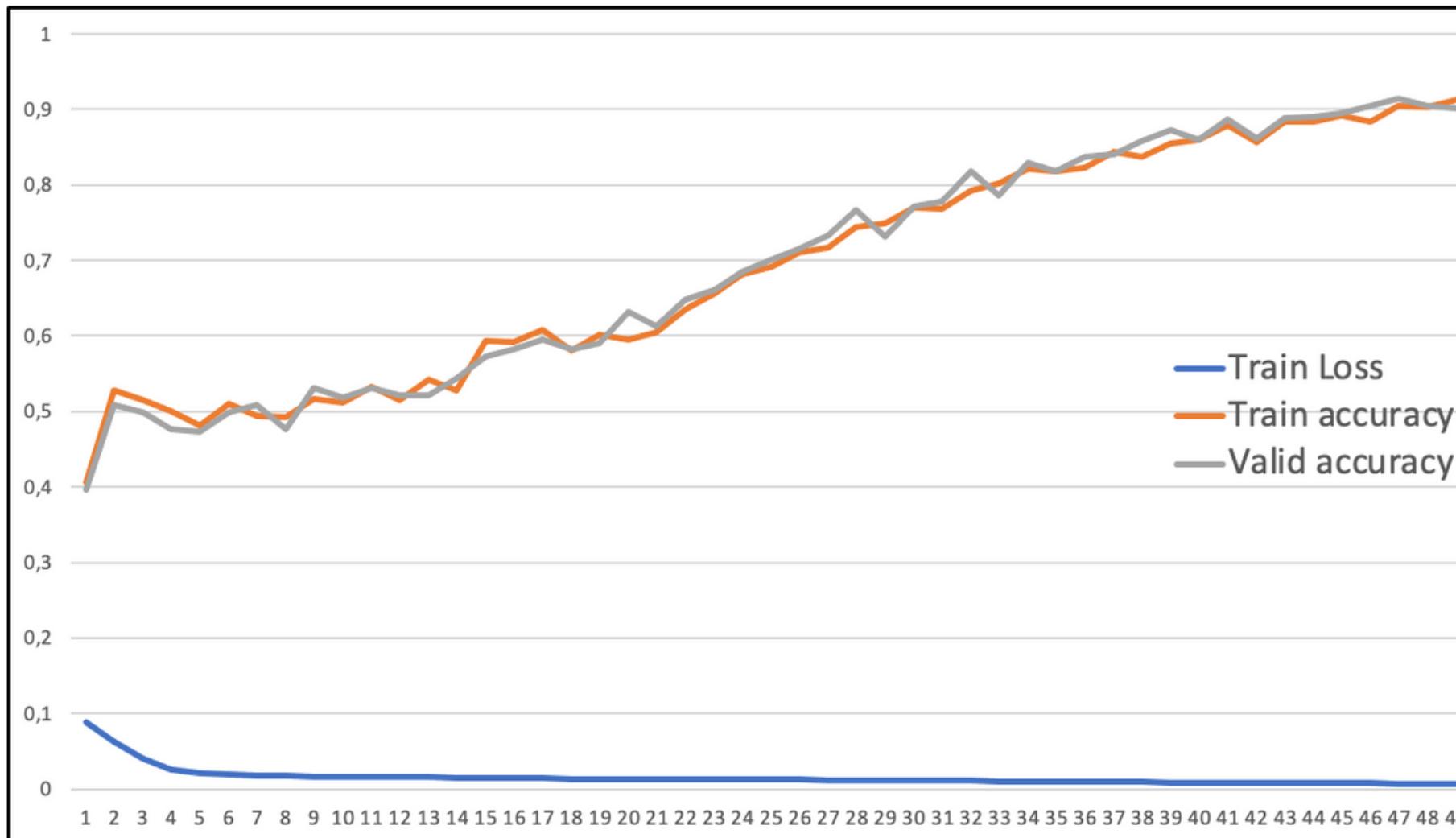
# Graph Neural Network



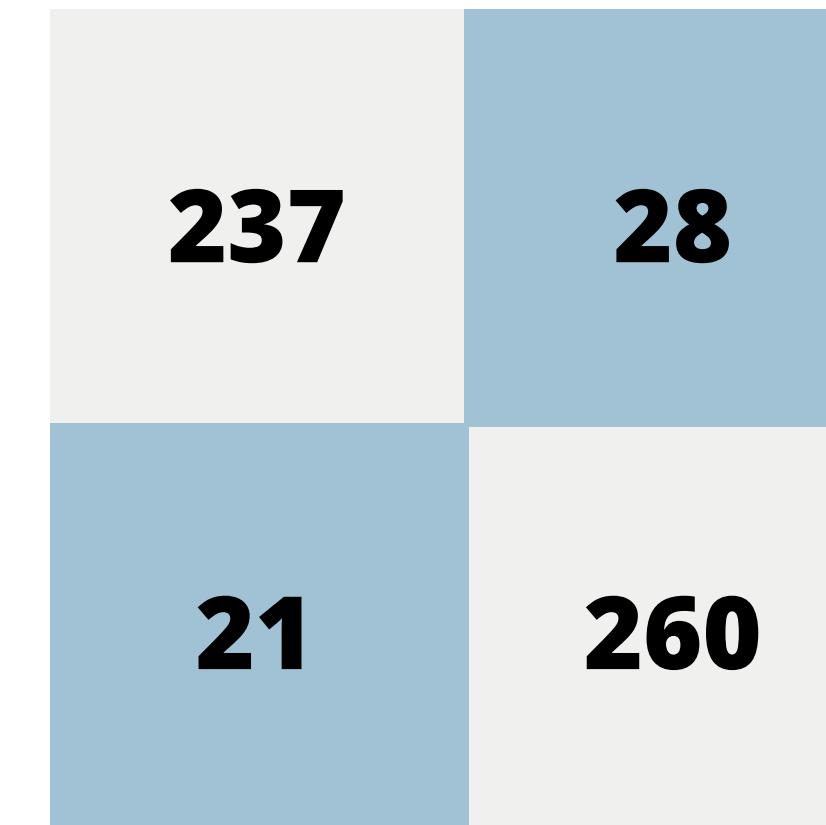


# Results

## Train



## Test



Accuracy = 91.03 %

Precision = 89.43 %

Recall = 91.86 %



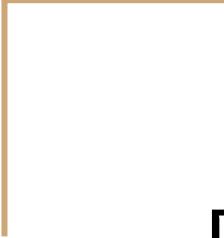
ILLINOIS INSTITUTE  
OF TECHNOLOGY

# Thank you so much!

Rodrigo López and Luis Castañeda

Do you have  
questions





# Fake News Classification

Kemen Goicoechea  
Brandon Bennett



# Dataset

- Big and heavily unbalanced dataset:

Class	# of data	% of the whole dataset
Unrelated	175598	68.47 %
Agreed	74238	28.95 %
Disagreed	6606	2.58 %

- Due to computation power:  
→ use a smaller balanced set for our tests

# Preprocessing

title1_en
There are two new old-age insurance benefits f...
"If you do not come to Shenzhen, sooner or lat...
"If you do not come to Shenzhen, sooner or lat...
"If you do not come to Shenzhen, sooner or lat...
"If you do not come to Shenzhen, sooner or lat...
...
egypt 's presidential election failed to win m...
egypt 's presidential election failed to win m...
egypt 's presidential election failed to win m...
egypt 's presidential election failed to win m...
Will the United States wage war on Iraq without...

title1\_en column before the preprocessing

- Remove Punctuation
- Lower case
- Remove Stop Words

title1_en
two new oldag insur benefit old peopl rural ar...
come shenzhen sooner later son also come less ...
come shenzhen sooner later son also come less ...
come shenzhen sooner later son also come less ...
come shenzhen sooner later son also come less ...
...
egypt presidenti elect fail win million vote e...
egypt presidenti elect fail win million vote e...
egypt presidenti elect fail win million vote e...
egypt presidenti elect fail win million vote e...
unit state wage war iraq without destruct sadd...

title1\_en column after the preprocessing

# Preprocessing

- TF-IDF Vectorization

$$\text{tf} - \text{idf} = \text{tf} \times \text{idf}$$

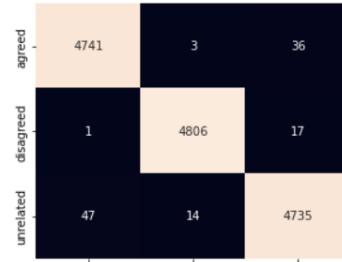
$$\text{idf}(t) = \log \frac{n + 1}{\text{df}(d, t) + 1} + 1$$

- Concatenation of the 2 titles in one Dataframe  
→ More than 10,000 features on the smaller dataset
- Test / Train split for on the balanced dataset

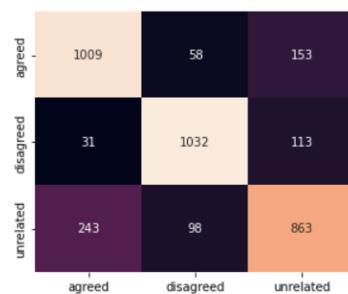
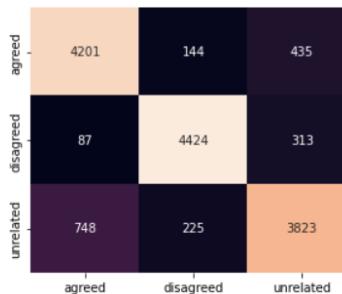
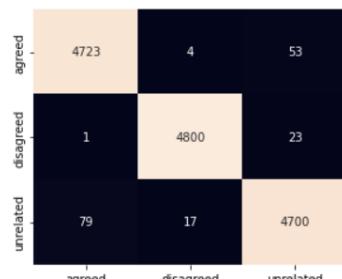
# Models (on balanced dataset)

- SVM
  - 99.18% training accuracy
  - 89.17% testing accuracy

Training Confusion Matrices

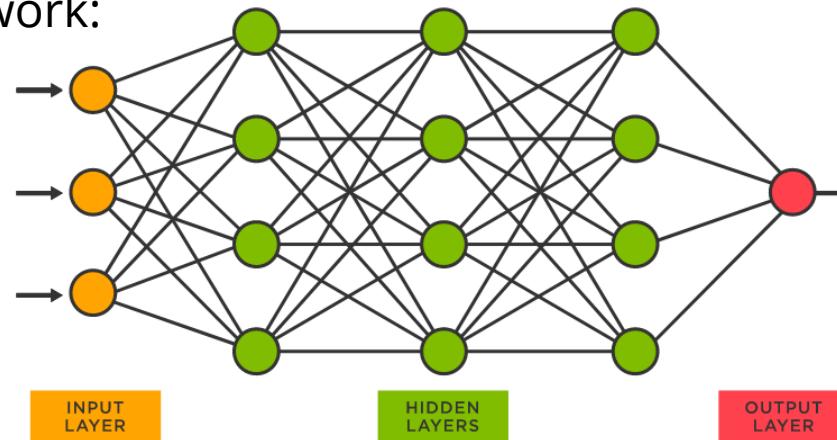


Testing Confusion Matrices



# Conclusion

- SVM is the leading model in with our preprocessed data
- Random Forest is also really good and much faster to train than SVM  
→ Models with a lot of variance seem to perform better
- Hence, future work:



# Amazon Fake Review Detection Model

Reported by: Meng Tang, Fan Guo



# a Project Introduction

With the boom in social media information sharing, people's opinion about a product or a business has become an important metric that influences other people's decisions.

**Effectively identifying fake reviews improves the credibility of websites that provide online reviewing platforms.**

**In this project, We aim to extract appropriate features from the selected sets of amazon reviews and train classification models from such datasets.**

Our Project:

- Considered the upvotes and user-written reviews
- Evaluate and improve features and predictive model



# a Methodology

- Extract features from the Amazon Reviews network for classification, we defined two key features : **“goodness” and “fairness”** . Define ground truth.
- Run different classification models on the Amazon dataset.



Amazon reviews dataset **497577 lines**

	reviewerID	asin	overall	verified	vote	reviewText	summary
0	7248	55223	5	1	0	This game is a b... but when you do ...	
1	7984	55223	4	0	0	I played it a wh... But in spite of ...	
2	14024	55223	3	1	0	ok game.	Three Stars
3	15411	55223	2	1	0	found the game a...	Two Stars
4	25672	55223	5	1	0	great game, I lo...	love this game

# a Datasets Preprocessing



## User Reviews Scoring

We first removed stop words and punctuations. Then we score the user reviews based on the TF-IDF values of each word in their sentences.



## Values Normalization

The values are normalized into range of [-1, 1]. Normalization is based on the highest value.



## Feature

### Extractions

good the review is, which consists of the vote score for the user review.

**Fairness:** the difference between the current user rating and the average product rating.

$$\text{Fairness}(e_i) = \text{UserRating}(e_i) - \frac{1}{(m-1)} * (\sum_{j=1}^{j=m} \text{UserRating}(e_j) - \text{UserRating}(e_i)) + 1$$

**Ground Truth Definition**  $\text{Trust}(e_i) = \frac{(\text{Verified}(e_i) + \text{Vote}(e_i) + \text{TF-IDF}(e_i))}{3}$

# a Machine learning – model selection, pros, and cons

**K-Nearest Neighbors (K-NN):** K-NN is learned by simply storing the training instances.

**Support Vector Machine (SVM):** SVM is a non-probabilistic binary classifier that depends on a small number of selected support vectors.

**Ensemble BaggingClassifier and AdaBoostClassifier:** Ensemble algorithms construct multiple different classifiers separately and the decisions of these classifiers are then combined as a new instance to do the final class prediction.

Model Name	Pros	Cons
K-NN	no parameters to tune	require large storage space; high requirement on feature selection
SVM	its complexity is independent of the number of applied features	require large training datasets
Ensemble	can reduce the generalization error; robust, accurate and precise predictions	introduce lots of extra computation

Table 1. Comparisons of Selected Machine Learning Models.

# a Machine Learning Results

In general, there is not a huge difference in accuracy between different models. But we can see that the Ensemble AdaBoostClassifier gives the highest precision and F1 score, with a relatively short training time.

We think our best model would be the **Ensemble AdaBoostClassifier**.

name	accuracy	precision	recall	f1_score	training_time
K-NN	0.754	0.791	0.859	0.708	39.24
SVM	0.734	0.733	0.944	0.631	1711.24
Ensemble BaggingClassifier	0.755	0.785	0.873	0.705	389.64
Ensemble AdaBoostClassifier	0.757	0.793	0.862	0.711	58.98

Table 2. Machine Learning Results.

# Thanks

Reported by:

Meng Tang([mtang11@hawk.iit.edu](mailto:mtang11@hawk.iit.edu))

Fan Guo([fguo10@hawk.iit.edu](mailto:fguo10@hawk.iit.edu))

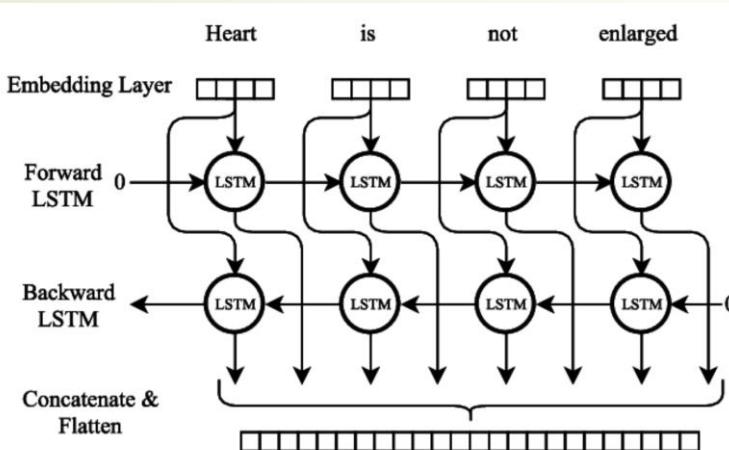
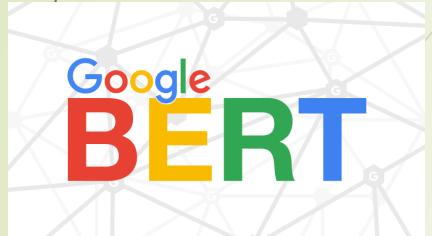


# Fake News Classification

Team: Ismail Elomari Alaoui - **A20497221**  
Reda Chaguer - **A20497223**

Under the supervision of:  
Prof. Kai Shu

# Model Architecture



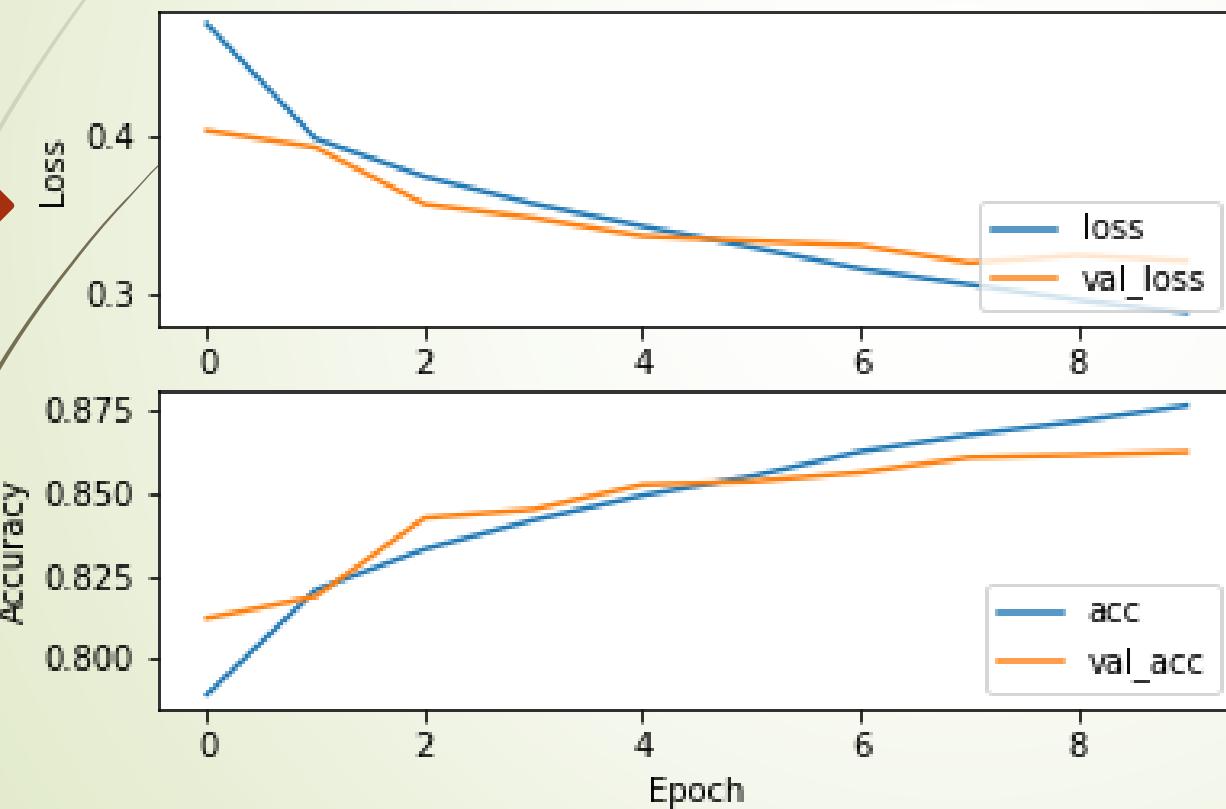
## Training Parameters:

- **Epochs** : 10
- **Batch size** : 512
- **Optimizer** : Adam
- **Learning rate** : 1e-4
- **Bert Weights** : Frozen
- **Max length size** : 128
- **Loss** : Categorical Crossentropy

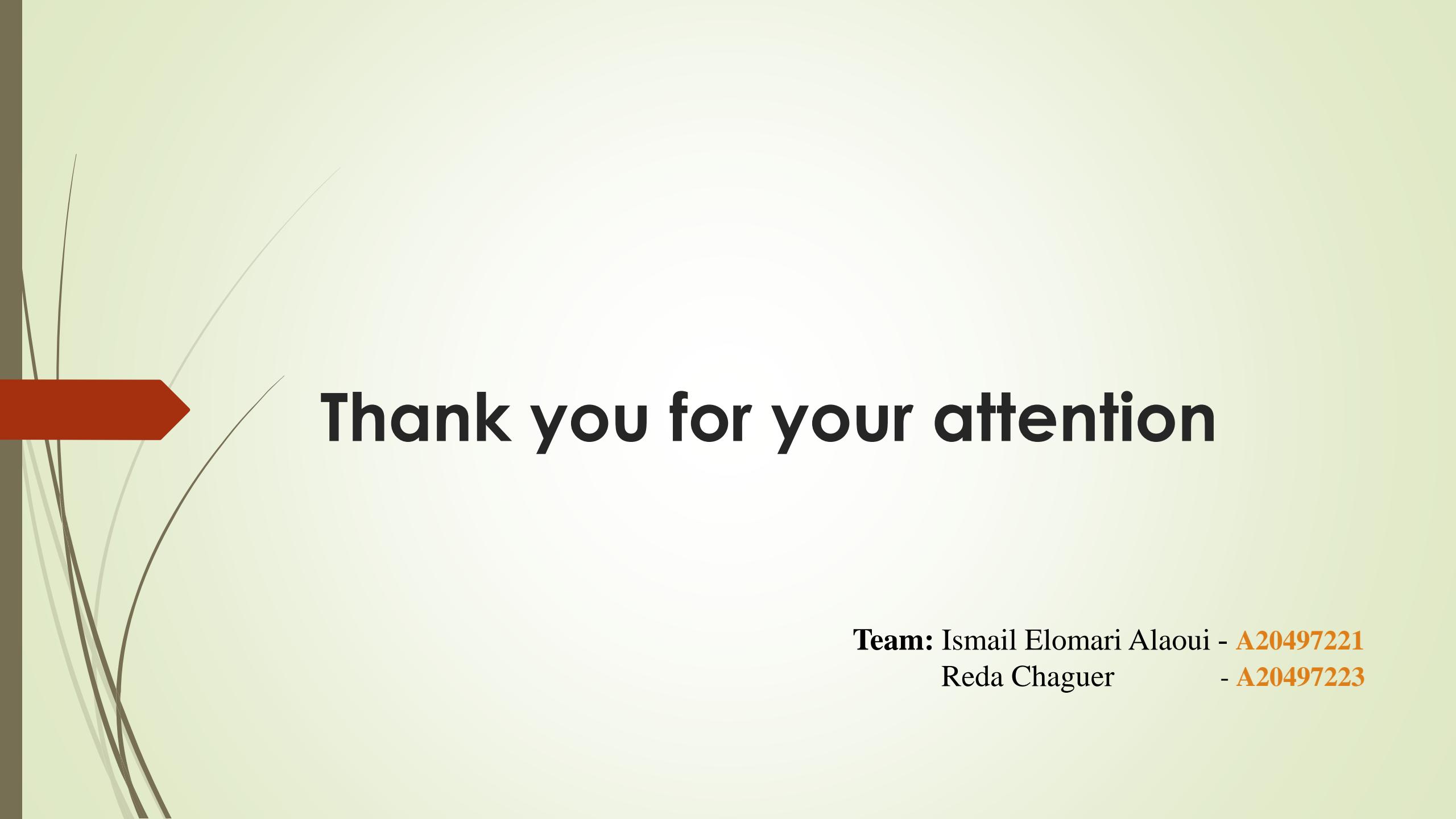
Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[None, 128]	0	[]
attention_masks (InputLayer)	[None, 128]	0	[]
token_type_ids (InputLayer)	[None, 128]	0	[]
tf_bert_model_2 (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttention(last_hidden_state=(None, 128, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	109482240	['input_ids[0][0]', 'attention_masks[0][0]', 'token_type_ids[0][0]']
bidirectional_1 (Bidirectional)	(None, 128, 128)	426496	['tf_bert_model_2[0][0]']
global_average_pooling1d_1 (GlobalAveragePooling1D)	(None, 128)	0	['bidirectional_1[0][0]']
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 128)	0	['bidirectional_1[0][0]']
concatenate_1 (Concatenate)	(None, 256)	0	['global_average_pooling1d_1[0][0]', 'global_max_pooling1d_1[0][0]']
dropout_112 (Dropout)	(None, 256)	0	['concatenate_1[0][0]']
dense_1 (Dense)	(None, 3)	771	['dropout_112[0][0]']
<hr/>			
Total params: 109,909,507			
Trainable params: 427,267			
Non-trainable params: 109,482,240			

# Training and Results

```
Epoch 10/10
400/400 [=====] - 1151s 3s/step - loss: 0.2878 - acc: 0.8767 - val_loss: 0.3209 - val_acc: 0.8626
```



Metrics \ Set	Training	Validation
Accuracy	87.67%	86.26%
Loss	0.2878	0.3209



# **Thank you for your attention**

**Team:** Ismail Elomari Alaoui - [A20497221](#)  
Reda Chaguer - [A20497223](#)

# Fake News Classification

By: Kajol Tanesh Shah (A20496724)  
Tinh Cao Co (A20476426)

# Introduction

Fake news/misinformation is nothing new, but it has made its prominent entrance through the 2016 US Presidential election, where it became an important tool to deliver false or even misleading political stories on social media. Even recently, the COVID pandemic has brought an upheaval of disinformation targeting public fear of the virus, its transmission rate, vaccinations, and post-infection recovery.

Despite existing in many forms, we will put our focus on the online social media platform as it is one of the most powerful tools that we use to communicate right now.

# Problem Statement

We define our problem as the ability to classify incoming news B as fake news or not by comparing it with a known fake news A on the same topic. Three labels are used for comparing the two articles: agreed, disagreed, and unrelated.

- Agreed: B talks about the same fake news as A
- Disagreed: B refutes the fake news in A
- Unrelated: B is unrelated to A

With the ability to determine whether an incoming news article are fake new or not, we hope to minimize the spread of fake news and rebuild the trust for the online community.

## Proposed Solution

To classify the news article B into one of the three categories which are agreed, disagreed and unrelated, we created a machine learning model and trained it using NLP techniques and machine learning algorithms.

Below are the steps that were used to create the model:-

- Data preprocessing using NLP techniques
- Bag of words and TF-IDF transformer
- Data augmentation
- Model training using machine learning algorithms
- Model testing
- Model evaluation using different evaluation metrics

# Data Preprocessing

In this step, we applied some NLP preprocessing techniques to both the train and test datasets in order to make our datasets light. Using such techniques, we only kept the information which will be useful and has some meaning and removed the data which will not be important for prediction. Below are the techniques which were applied:-

- Converting strings to lowercase
- Removal of stopwords
- Removal of punctuation marks
- Lemmatization

# Bag of words and TF-IDF Transformer

- Bag of words creates a set of vectors which contains the count of the word occurrences in the text/document. For this, we have used CountVectorizer from scikit-learn
- TF-IDF means term frequency - inverse document frequency and is used to find how important a word is to a document/text in a corpus. We implemented this by using TfidfTransformer from scikit-learn

$$TF(t, d) = \frac{\text{number of times } t \text{ appears in } d}{\text{total number of terms in } d}$$

$$IDF(t) = \log \frac{N}{1 + df}$$

$$TF-IDF(t, d) = TF(t, d) * IDF(t)$$

# Approaches

We used two different approaches for training our model. As our training data was highly imbalanced, we were not sure whether our model will be able to make correct predictions as it will be trained on a majority of data which has label as ‘unrelated’. So, we used two approaches to solve this problem:-

- Using the original dataset to train our model
- Using data augmentation to balance all the three classes i.e. the % of each class label in the training dataset will be 33.33%

# Approach 1

In this approach, we used the original dataset to train our model using different machine learning algorithms and neural networks. For this, we made use of scikit-learn library in Python.

Below are the machine learning algorithms which we used:-

- Naive Bayes
- Stochastic Gradient Descent Classifier
- Logistic Regression
- Random Forest
- Linear Support Vector Machine Classifier

## Approach 1 - LSTM

We also used LSTM model which is a Recurrent Neural Network architecture. LSTM has feedback connection which is not present in other standard neural networks and so, LSTM can process entire sequences of data. We used Keras for creating LSTM model in Python.

LSTM unit consists of :-

- A cell
- Input gate
- Output gate
- Forget gate

The gates regulate the flow of information and the cell remembers the data for some time intervals.

LSTM is used in many applications like time series, speech recognition, handwriting recognition, etc.

## Approach 1- LSTM

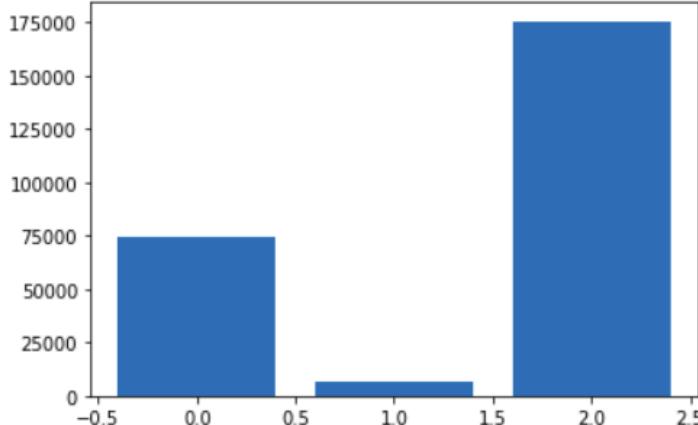
Text-preprocessing for LSTM: We use Keras' Tokenizer to convert the title string into a sequence of integers representing each word. The corpus is built upon our combined titles' vocabulary.

We choose to represent words as a sequence in an attempt to capture the word similarity that exists between title A and title B as well as the word dependency for its semantic meaning. In addition, the ability to process news that make reference to a long-time ago information also makes LSTM a prominent candidate. From this we hope to catch certain patterns that fake news follows to better classify them.

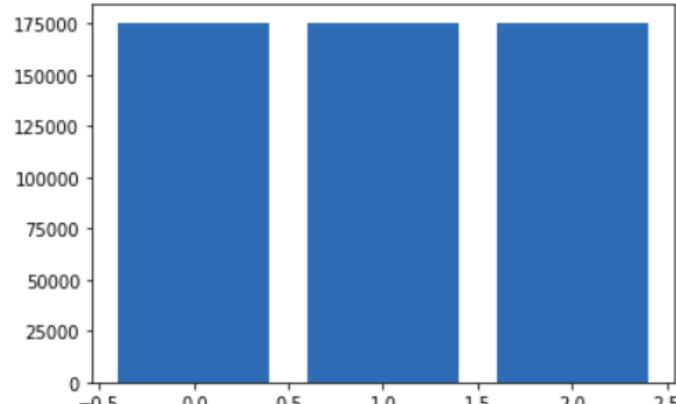
## Approach 2

In this approach, we used data augmentation to solve the class imbalance problem as approximately 70% of our training data had class label as ‘unrelated’ and the remaining data 30% included of both ‘agreed’ and ‘disagreed’ label. We used the Imblearn library for this.

Class=2, n=175598 (68.475%)  
Class=0, n=74238 (28.949%)  
Class=1, n=6606 (2.576%)



Class=2, n=175598 (33.333%)  
Class=0, n=175598 (33.333%)  
Class=1, n=175598 (33.333%)



## Approach 2

After doing data augmentation, we made use of the same machine learning algorithms to train our model which were used in the first approach. They are as follows:-

- Naive Bayes
- Stochastic Gradient Descent Classifier
- Logistic Regression
- Random Forest
- Linear Support Vector Machine Classifier

# Evaluation

We tested our model on the validation data in order to find the best model to predict the class labels of our test data. We also used k-fold cross validation.

The metrics which we used to evaluate our model are:-

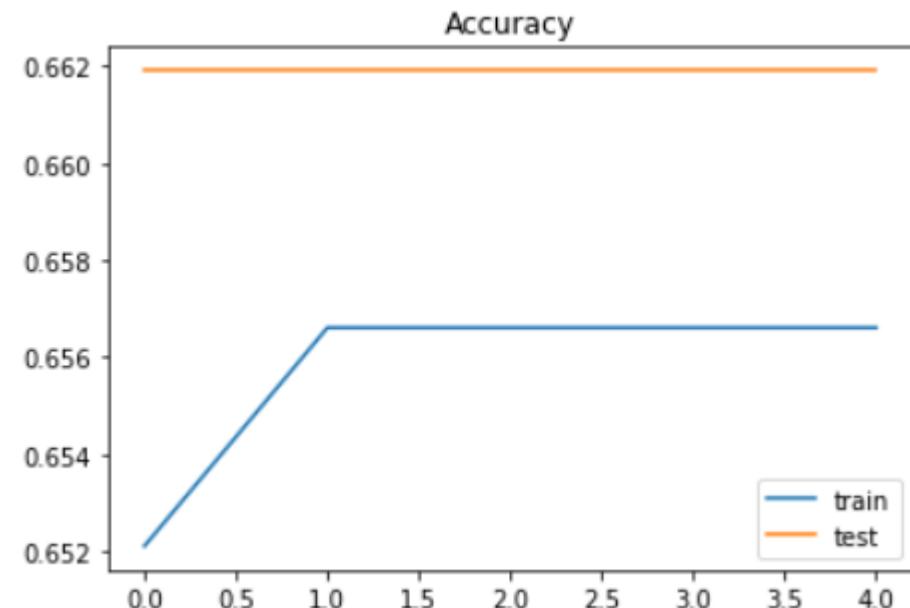
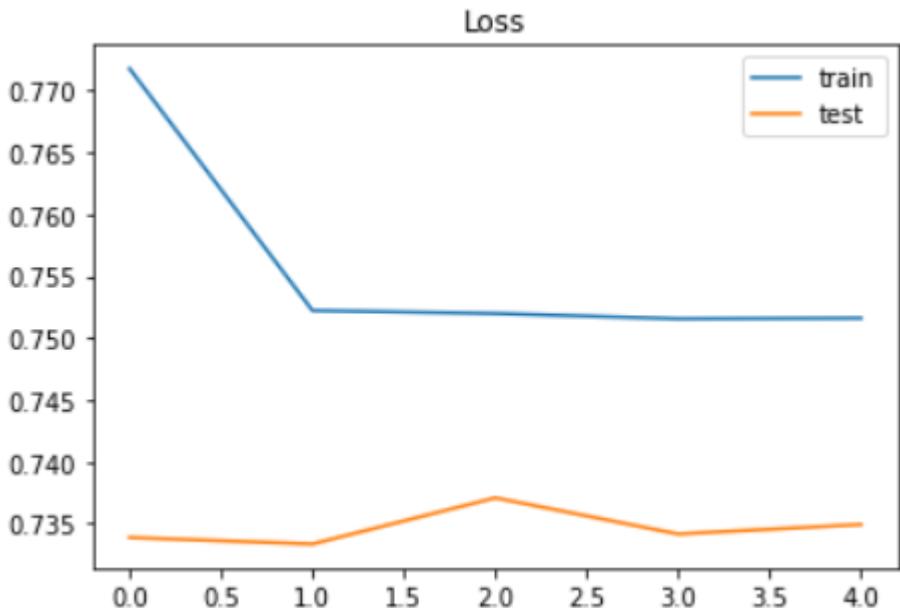
- Accuracy
- Precision
- Recall
- F-1 score

# Results - Approach 1

Method	Accuracy	Precision	Recall	F-1 score
Naive Bayes	71.58	0.73	0.11	0.19
		0.00	0.00	0.00
		0.72	0.98	0.83
SGD Classifier	69.74	0.00	0.00	0.00
		0.00	0.00	0.00
		0.70	1.00	0.82
Logistic Regression	73.02	0.57	0.44	0.49
		0.73	0.10	0.18
		0.77	0.87	0.82
Random Forest	69.74	0.00	0.00	0.00
		0.00	0.00	0.00
		0.70	1.00	0.82
Linear SVC	73.02	0.57	0.45	0.50
		0.64	0.12	0.20
		0.78	0.87	0.82

# LSTM - Result

Using 5 epochs and a batch size of 64, below are the results after the training.



# Results - Approach 2

Method	Accuracy	Precision	Recall	F-1 score
Naive Bayes	75.91	0.71	0.81	0.76
		0.89	0.81	0.85
		0.69	0.65	0.67
SGD Classifier	63.84	0.62	0.78	0.69
		0.65	0.94	0.76
		0.70	0.20	0.31
Logistic Regression	86.00	0.81	0.84	0.82
		0.91	0.97	0.94
		0.86	0.77	0.81
Random Forest	53.77	0.42	0.61	0.50
		0.78	0.54	0.64
		0.54	0.46	0.50
Linear SVC	86.64	0.81	0.85	0.83
		0.91	0.98	0.94
		0.87	0.77	0.82

# Conclusion

- Using different machine learning algorithms and LSTM model, we were able to create a model which can classify whether a given news B talks about the same fake news as A or not
- As we saw earlier in the results section, we achieved the best accuracy with Logistic Regression and Linear Support Vector Classifier

## Future Scope

On the technical side, we believe that our models can better classify fake news by employing dynamic sampling techniques introduced by Pouyanfar [8]. This puts heavier weights on rare but significant events where fake news share the same sentiments, and less on unrelated fake news.

Broadly speaking, we can see the future of fake news classifications being used in industry setting, such as detecting fraudulent activities on Instagram, to scammers on Facebook Groups/ Posts that leverage the loopholes of online platforms for personal interest. The ability to make decision and intervention in timely manner can prevent those ill-practices from harming the community.

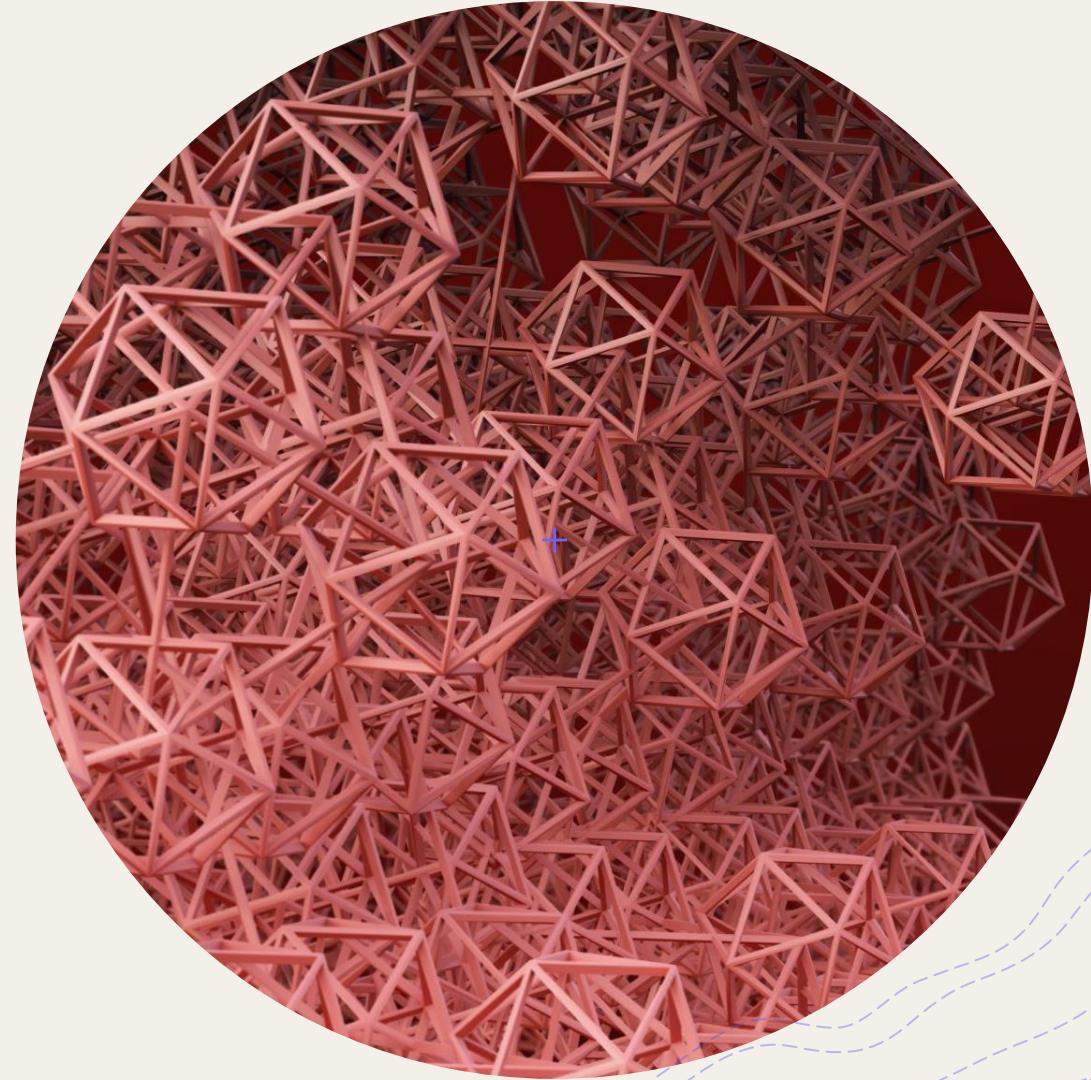
# References

- [1] H. Sak, A. Senior, F. Beaufays, “ Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling” in Interspeech 2014
- [2] Andrej, Karpathy: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- [3] <https://vitalflux.com/text-classification-bag-of-words-model-python-sklearn>
- [4] NLTK. Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc
- [5] Tensor Flow. LSTM. [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/LSTM](https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM)
- [6] Keras. Project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). <https://keras.io/>
- [7] [Scikit-learn: Machine Learning in Python](#), Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
- [8] Pouyanfar S, et al.. Dynamic sampling in convolutional neural networks for imbalanced data classification. In: 2018 IEEE conference on multimedia information processing and retrieval (MIPR). 2018. p. 112–7. <https://doi.org/10.1109/MIPR.2018.00027>.
- [9] Jeff Reback, et al. (2020). pandas-dev/pandas: Pandas 1.0.3 (v1.0.3). Zenodo. <https://doi.org/10.5281/zenodo.3715232>
- [10] [https://www.tensorflow.org/api\\_docs/python/tf/keras/preprocessing/text/Tokenizer](https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer)



# Fake News Classification

Neil Nemi Shah A20500775  
DhruvKumar KiranBhai Patel  
A20501734



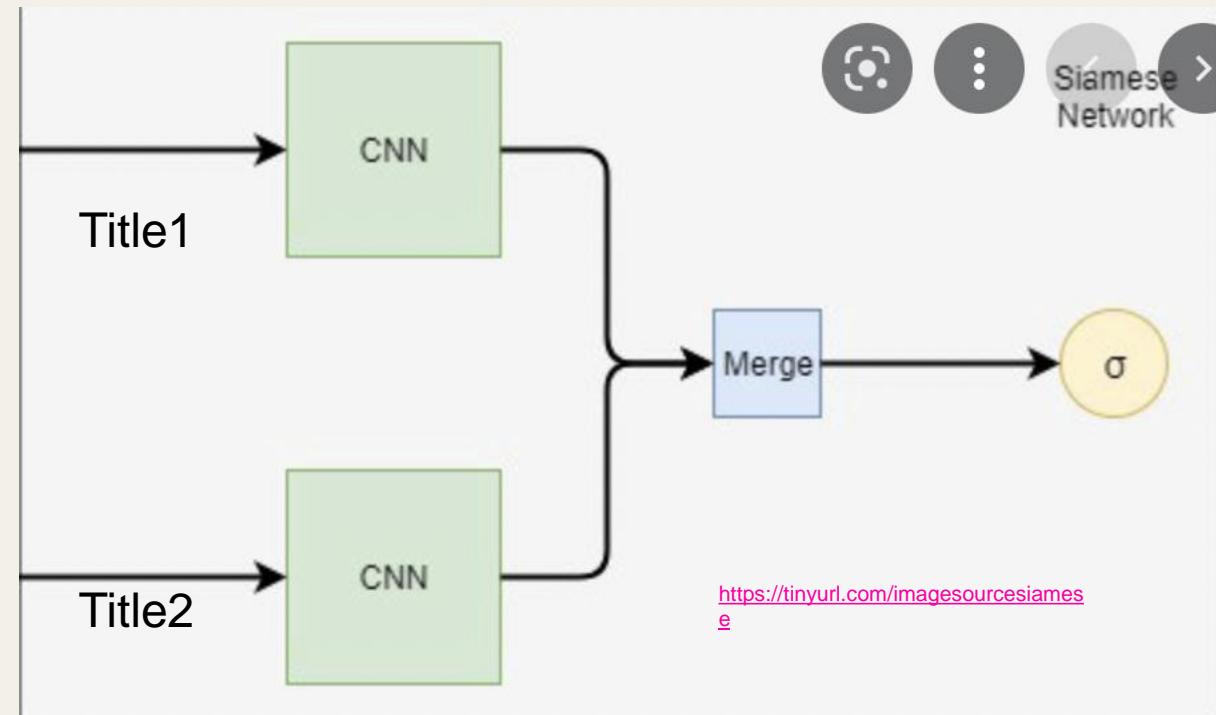
# Problem Statement

To effectively classify news into three categories: agreed,  
disagreed and unrelated

Need?

# Solution

- Our proposed solution calls for the use of the Siamese model in conjunction with GRU.
- We will construct the model and test its accuracy on training and validation data sets.



# Solution

- But before, we did some data pre-processing and took some liberties with tokenization and pre-padding.
- We also do one hot encoding.
- Following that, we take the test data set and forecast on it, as well as classify all of the news in the test data set.

```
▶ df.drop(['tid1', 'tid2', 'id'], axis=1, inplace=True) #dropping  
df.dropna(inplace = True)  
  
df['title1_en'] = df['title1_en'].apply(lambda x: x.lower())  
df['title2_en'] = df['title2_en'].apply(lambda x: x.lower())
```

# Training

- We take two titles as input.
- We. Use GRU twice concatenate and pass it through the dense layer with 3 classes.

↳ Model: "model"

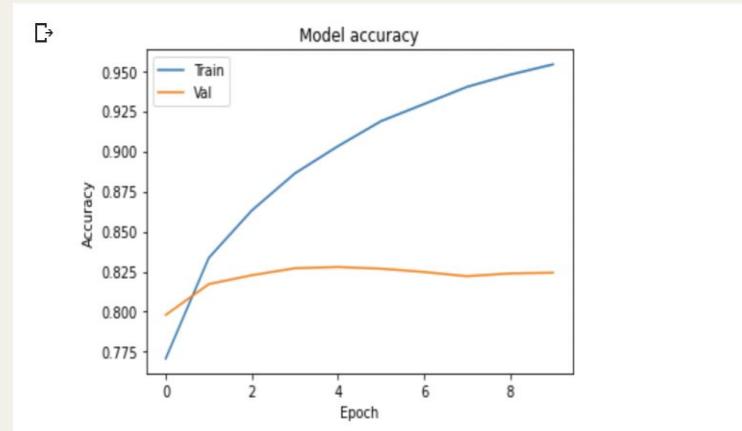
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 30)]	0	[]
input_2 (InputLayer)	[(None, 30)]	0	[]
embedding (Embedding)	(None, 30, 128)	1280000	['input_1[0][0]']
embedding_1 (Embedding)	(None, 30, 128)	1280000	['input_2[0][0]']
gru (GRU)	(None, 64)	37248	['embedding[0][0]']
gru_1 (GRU)	(None, 64)	37248	['embedding_1[0][0]']
concatenate (Concatenate)	(None, 128)	0	['gru[0][0]', 'gru_1[0][0]']
dense (Dense)	(None, 3)	387	['concatenate[0][0]']

Total params: 2,634,883

Trainable params: 2,634,883

Non-trainable params: 0

# Model Results on Validation(Test) DataSet



```
history = model.fit(x=[t1_df, t2_df], y=y_train, batch_size=64, epochs=10, validation_data=(x1_val, x2_val), y_val)

Epoch 1/10
3206/3206 [=====] - 88s 26ms/step - loss: 0.4910 - accuracy: 0.7709 - recall_2: 0.7657 - precision_2: 0.7728 - f1_score: 0.5783
Epoch 2/10
3206/3206 [=====] - 82s 26ms/step - loss: 0.3765 - accuracy: 0.8335 - recall_2: 0.8325 - precision_2: 0.8343 - f1_score: 0.7022
Epoch 3/10
3206/3206 [=====] - 82s 26ms/step - loss: 0.3166 - accuracy: 0.8631 - recall_2: 0.8624 - precision_2: 0.8636 - f1_score: 0.7636
Epoch 4/10
3206/3206 [=====] - 84s 26ms/step - loss: 0.2690 - accuracy: 0.8862 - recall_2: 0.8857 - precision_2: 0.8866 - f1_score: 0.8061
Epoch 5/10
3206/3206 [=====] - 83s 26ms/step - loss: 0.2301 - accuracy: 0.9031 - recall_2: 0.9028 - precision_2: 0.9034 - f1_score: 0.8373
Epoch 6/10
3206/3206 [=====] - 83s 26ms/step - loss: 0.1984 - accuracy: 0.9187 - recall_2: 0.9185 - precision_2: 0.9189 - f1_score: 0.8635
Epoch 7/10
3206/3206 [=====] - 82s 26ms/step - loss: 0.1725 - accuracy: 0.9294 - recall_2: 0.9292 - precision_2: 0.9295 - f1_score: 0.8837
Epoch 8/10
3206/3206 [=====] - 82s 26ms/step - loss: 0.1507 - accuracy: 0.9401 - recall_2: 0.9400 - precision_2: 0.9401 - f1_score: 0.9006
Epoch 9/10
3206/3206 [=====] - 82s 26ms/step - loss: 0.1322 - accuracy: 0.9477 - recall_2: 0.9476 - precision_2: 0.9477 - f1_score: 0.9118
Epoch 10/10
3206/3206 [=====] - 85s 26ms/step - loss: 0.1176 - accuracy: 0.9541 - recall_2: 0.9541 - precision_2: 0.9542 - f1_score: 0.9219
```

```
history = model.fit(x=[t1_df, t2_df], y=y_train, batch_size=64, epochs=10, validation_data=(x1_val, x2_val), y_val)

57 - precision_2: 0.7728 - f1_score: 0.5783 - val_loss: 0.4404 - val_accuracy: 0.7980 - val_recall_2: 0.7964 - val_precision_2: 0.7989 - val_f1_score: 0.6211
25 - precision_2: 0.8343 - f1_score: 0.7022 - val_loss: 0.4186 - val_accuracy: 0.8172 - val_recall_2: 0.8164 - val_precision_2: 0.8179 - val_f1_score: 0.6551
24 - precision_2: 0.8636 - f1_score: 0.7636 - val_loss: 0.4240 - val_accuracy: 0.8227 - val_recall_2: 0.8222 - val_precision_2: 0.8232 - val_f1_score: 0.6800
57 - precision_2: 0.8866 - f1_score: 0.8061 - val_loss: 0.4297 - val_accuracy: 0.8271 - val_recall_2: 0.8264 - val_precision_2: 0.8273 - val_f1_score: 0.6942
28 - precision_2: 0.9034 - f1_score: 0.8373 - val_loss: 0.4509 - val_accuracy: 0.8279 - val_recall_2: 0.8275 - val_precision_2: 0.8282 - val_f1_score: 0.7005
85 - precision_2: 0.9189 - f1_score: 0.8635 - val_loss: 0.4888 - val_accuracy: 0.8268 - val_recall_2: 0.8265 - val_precision_2: 0.8272 - val_f1_score: 0.7028
92 - precision_2: 0.9295 - f1_score: 0.8837 - val_loss: 0.5209 - val_accuracy: 0.8248 - val_recall_2: 0.8245 - val_precision_2: 0.8251 - val_f1_score: 0.7069
00 - precision_2: 0.9401 - f1_score: 0.9006 - val_loss: 0.5540 - val_accuracy: 0.8221 - val_recall_2: 0.8219 - val_precision_2: 0.8222 - val_f1_score: 0.7079
76 - precision_2: 0.9477 - f1_score: 0.9118 - val_loss: 0.6125 - val_accuracy: 0.8238 - val_recall_2: 0.8238 - val_precision_2: 0.8241 - val_f1_score: 0.7104
41 - precision_2: 0.9542 - f1_score: 0.9219 - val_loss: 0.6315 - val_accuracy: 0.8243 - val_recall_2: 0.8242 - val_precision_2: 0.8244 - val_f1_score: 0.7047
```

The accuracy of the training dataset is 0.9541

The recall of the training dataset is 0.9541

The precision of the training dataset is 0.9542

The f1 score of the training dataset is 0.9219

The validation(test partitioned from training) loss is 0.6315

The validation(test partitioned from training) accuracy 0.8243

The validation(test partitioned from training) recall is 0.8242

The validation(test partitioned from training) precision is 0.8244

The validation(test partitioned from training) f1 score is 0.7047

# Results on Submission.CSV

```
▶ test = pd.read_csv('test.csv')

▶ x1_test = tokenizer.texts_to_sequences(test['title1_en'])
x2_test = tokenizer.texts_to_sequences(test['title2_en'])

x1_test = tf.keras.preprocessing.sequence.pad_sequences(x1_test, maxlen=max_len)
x2_test = tf.keras.preprocessing.sequence.pad_sequences(x2_test, maxlen=max_len)

pred = model.predict([x1_test, x2_test])

[ ] test.drop(['tid1', 'tid2', 'title1_en', 'title2_en'], axis=1, inplace=True)

ilabels = {0: 'agreed', 1: 'disagreed', 2: 'unrelated'}

for i in range(len(pred)):
    test.loc[i, 'label'] = ilabels[np.argmax(pred[i])]

test.to_csv('submission.csv', index=False)
test.head()
```

Sheet Name	Sheet 1
Background	<input checked="" type="checkbox"/>
Duplicate Sheet	<input type="button" value=""/>
Delete Sheet	<input type="button" value=""/>
125%	Zoom
Add Category	
View	Insert Table Chart Text Shape Media Comment
Collaborate	
Format	Organise
Sheet	
+	Sheet 1
256449	unrelated
256450	unrelated
256451	unrelated
256452	agreed
256453	agreed
256454	agreed
256455	agreed
256456	unrelated
256457	unrelated
256458	agreed
256459	unrelated
256460	unrelated
256461	agreed
256462	unrelated
256463	disagreed
256464	unrelated
256465	unrelated
256466	agreed
256467	agreed
256468	unrelated
256469	agreed
256470	unrelated
256471	unrelated
256472	agreed
256473	agreed
256474	agreed
256475	unrelated

# Thank You

+



ILLINOIS INSTITUTE OF TECHNOLOGY

# Classifying Fake News

Sarvesh Shroff | Akshay Jain  
A20488681 | A20502846

# Outline

- ◀ Problem Statement
- ◀ Data Preprocessing
- ◀ Model
- ◀ Results
- ◀ Q & A

## Problem Statement

*Social media has become one of the major resources for people to obtain news and information*

*So in this project we would use classification model to classify if a news article is related to a fake news or not*

# Data Preprocessing

## Remove Stop Words

Create a list of all stop words and punctuations from the stopwords package of nltk.corpus for english language and remove them from the data

## Tokenize and Stem

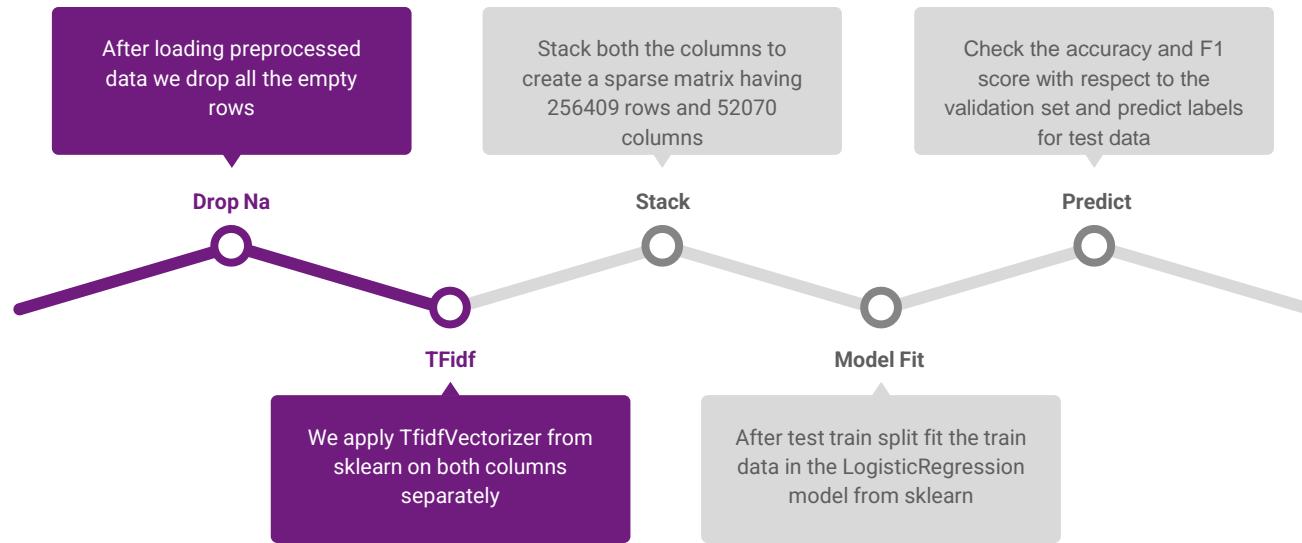
Using WhitespaceTokenizer and PorterStemmer from nltk library we tokenize and stem the data

## Lemmatize and Save

Using WordNetLemmatizer we lemmatize the data and later drop the title1\_id and title2\_id and store the remaining data in train\_preprocessed.csv and test\_preprocessed.csv

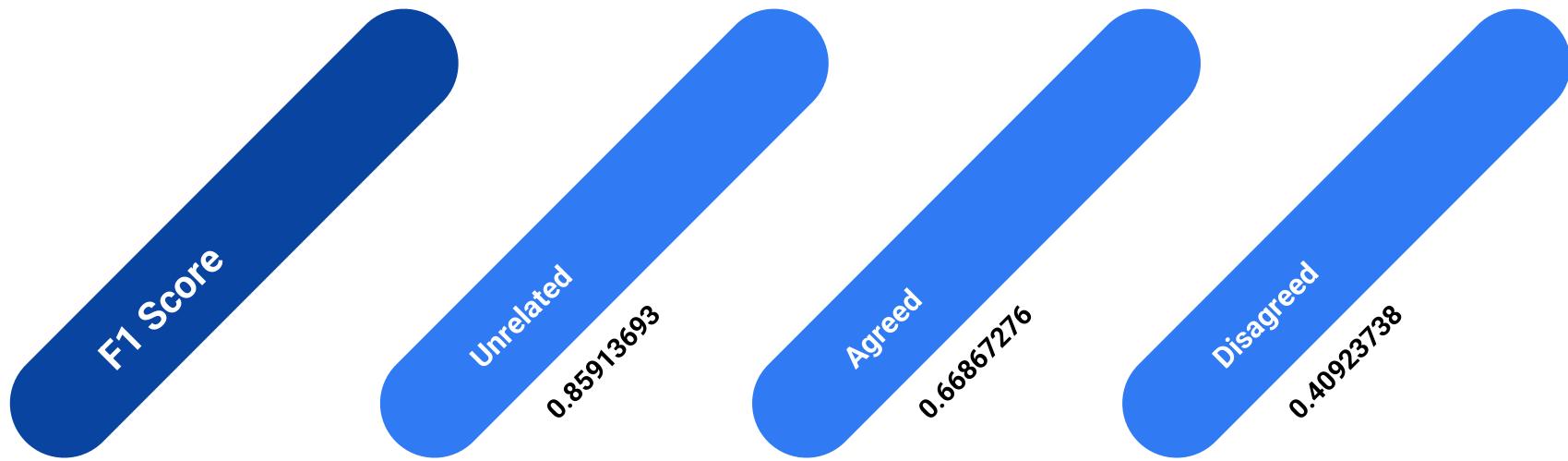


# Logistic Regression

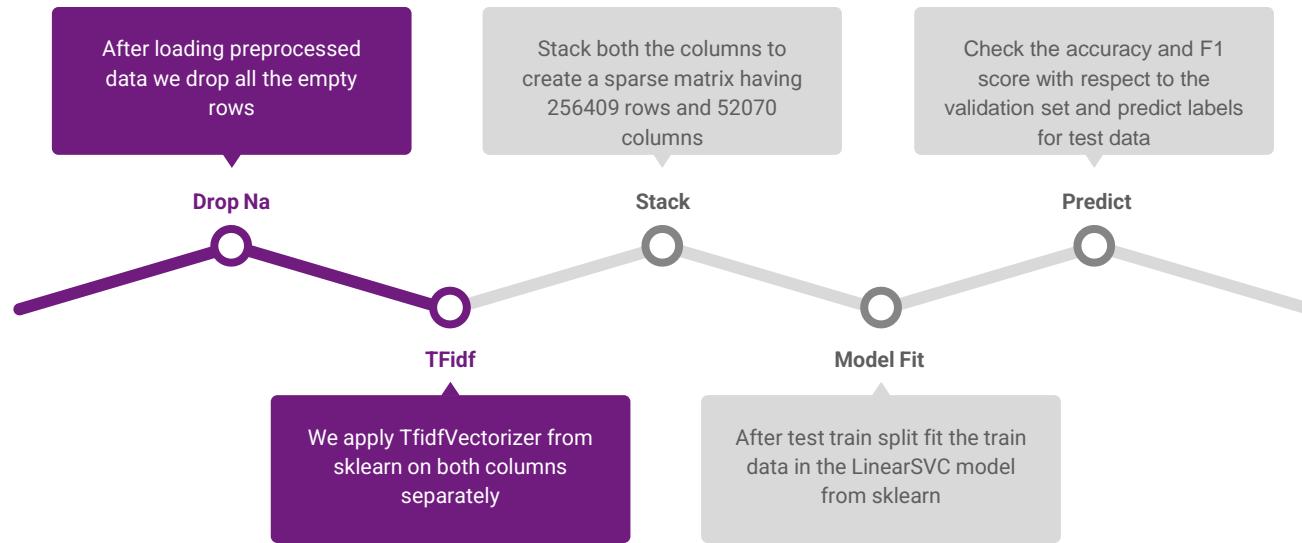


# Logistic Regression Result

Overall Accuracy was 0.7991

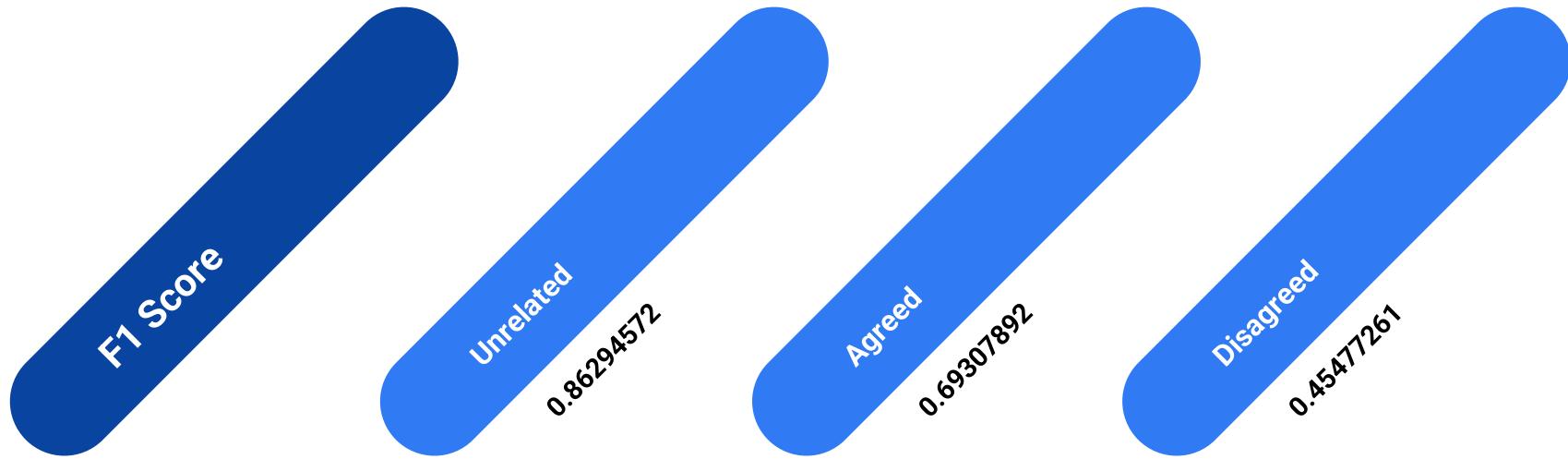


# LinearSVC

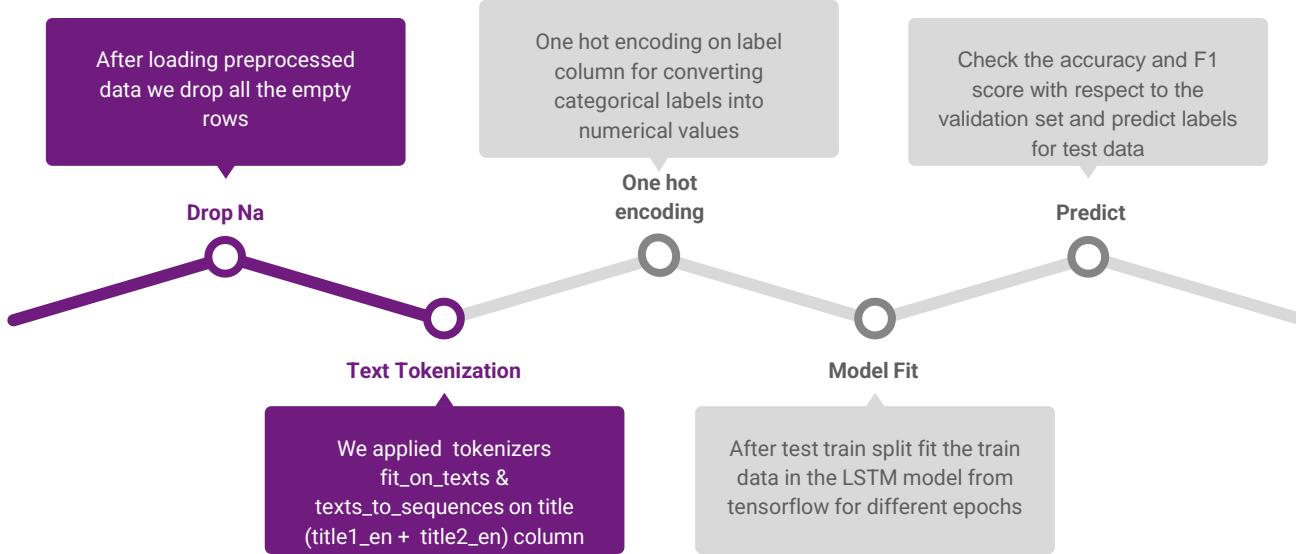


# LinearSVC Result

Overall Accuracy was 0.8072

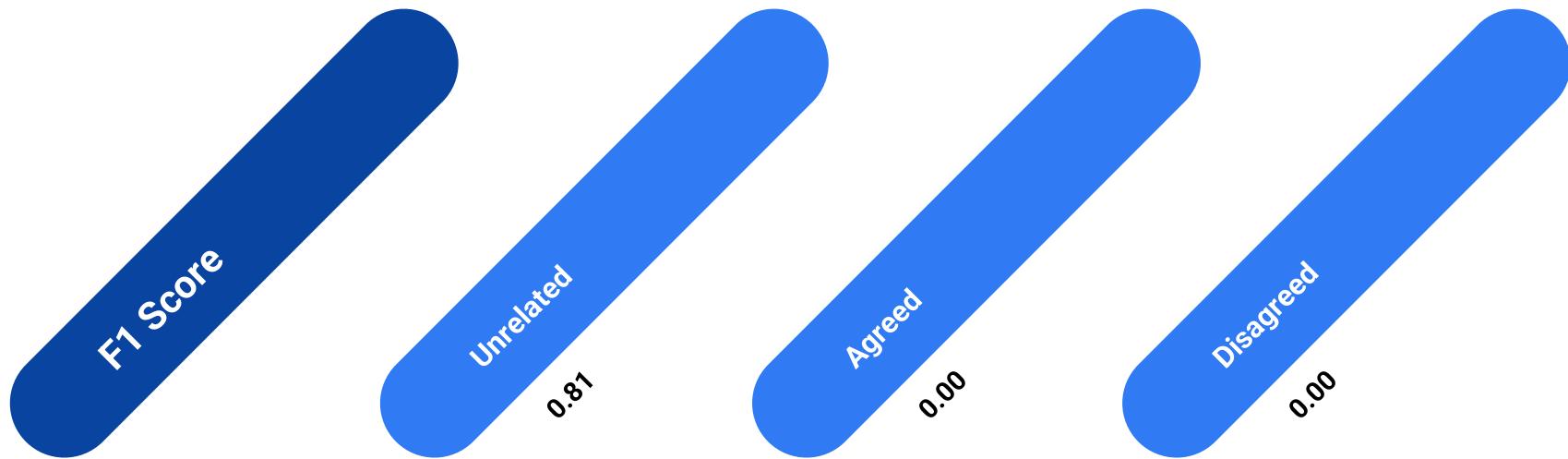


# LSTM



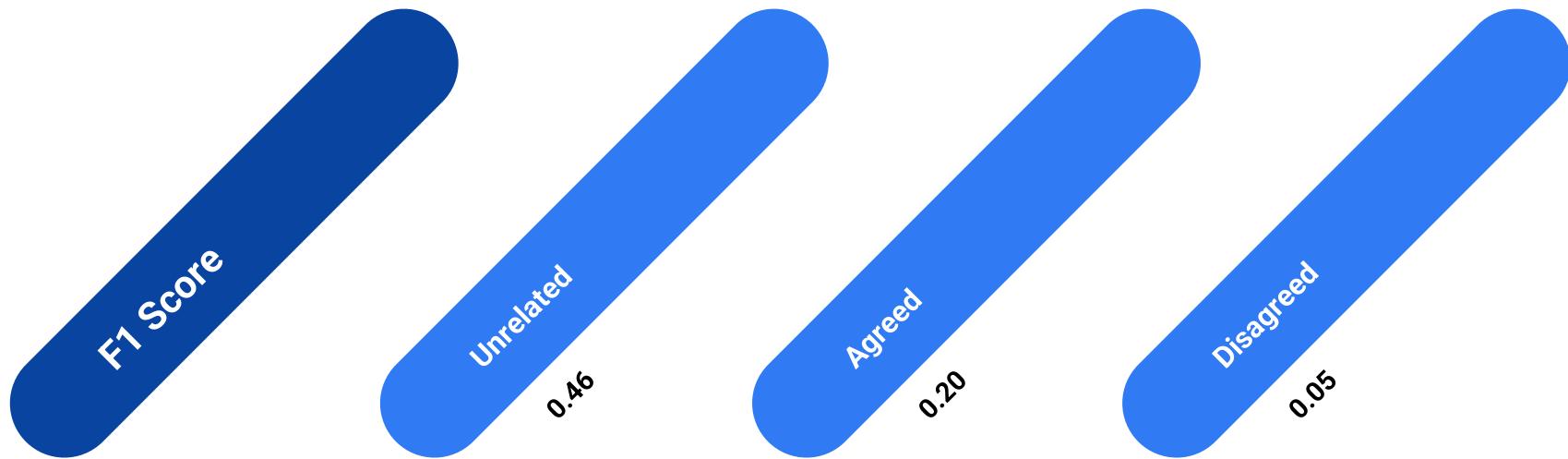
# LSTM Result without undersampling

Overall Accuracy was 0.68 (for epoch =1)



## LSTM Result with undersampling

Overall Accuracy was 0.63 (for epoch =5)



# Thank you

Any Questions

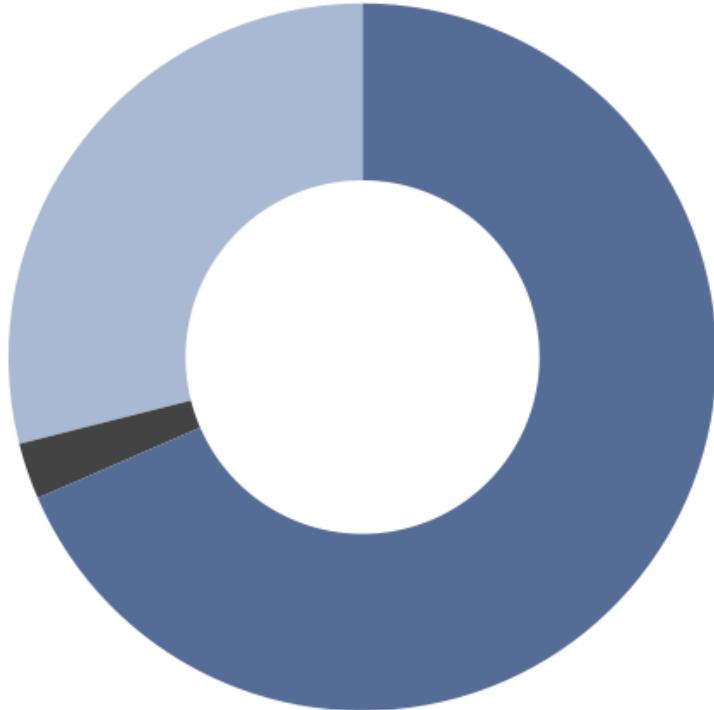


# THE DEFAULT PROJECT

Jose Ignacio Pozuelo and Mateo Fernandez

CS 579 Online Social Network Analysis

# DATA VISUALIZATION



AGREED

**29%**

UNRELATED

**68%**

DISAGREED

**3%**

# PREPROCESSING

Preparing the text data for the model using regular expressions, stop words, lemmatization, and vectorization.

```
stemmer = WordNetLemmatizer()
# Converting to Lowercase
document = line.lower()
# remove all links
document = re.sub(r'^ [^ ]*http[^ ]*', ' ', str(document))
# remove all links
document = re.sub(r'^ [^ ]*www[^ ]*', ' ', document)
# Remove all the special characters
document = re.sub(r'\W', ' ', document)
# remove all single characters
document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)
# Remove single characters from the start
document = re.sub(r'^[a-zA-Z]\s+', ' ', document)
# Substituting multiple spaces with single space
document = re.sub(r'\s+', ' ', document, flags=re.I)
# Lemmatization
document = document.split()
document = [stemmer.lemmatize(word) for word in document]
lst_stopwords = nltk.corpus.stopwords.words("english")
document = [word for word in document if word not in lst_stopwords]
document = ' '.join(document)
```

# VECTORIZATION



## COUNT VECTORIZER

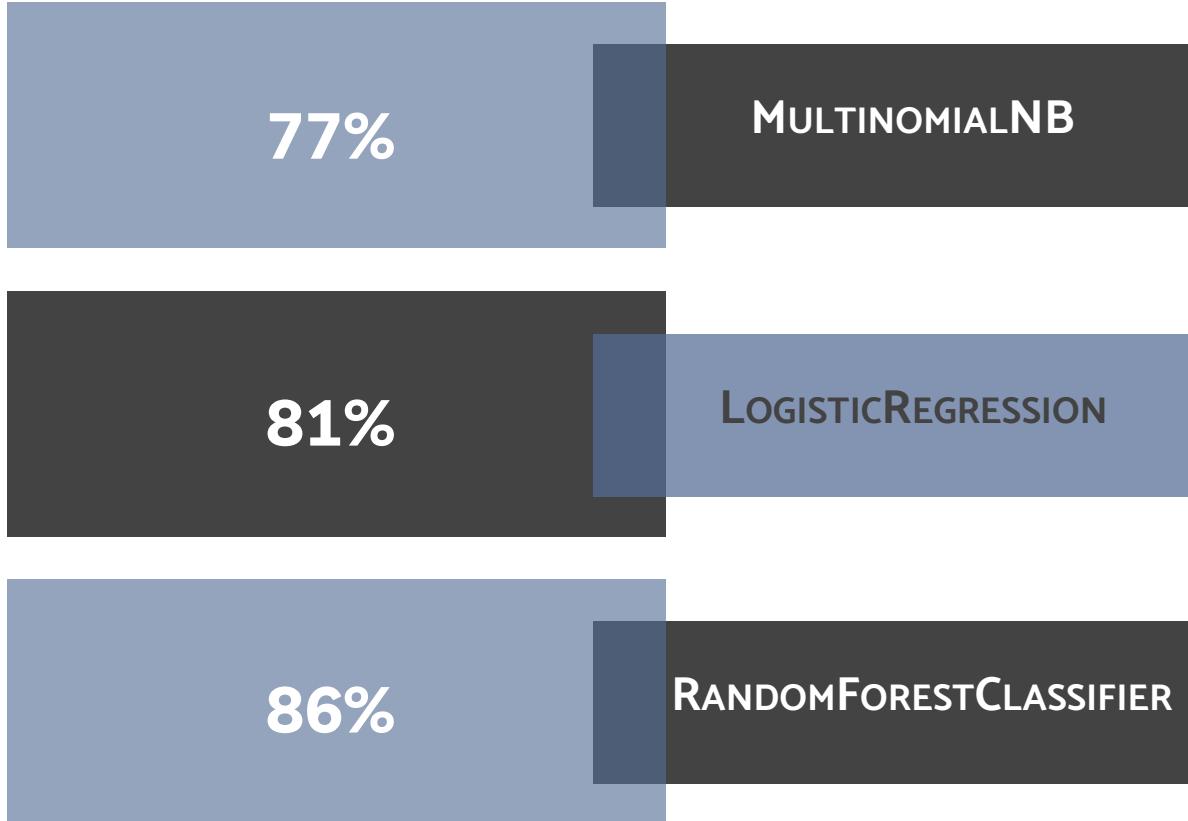
Count vectorization is used to convert a text into a vector based on the frequency, or count, of each word that appears throughout the entire text

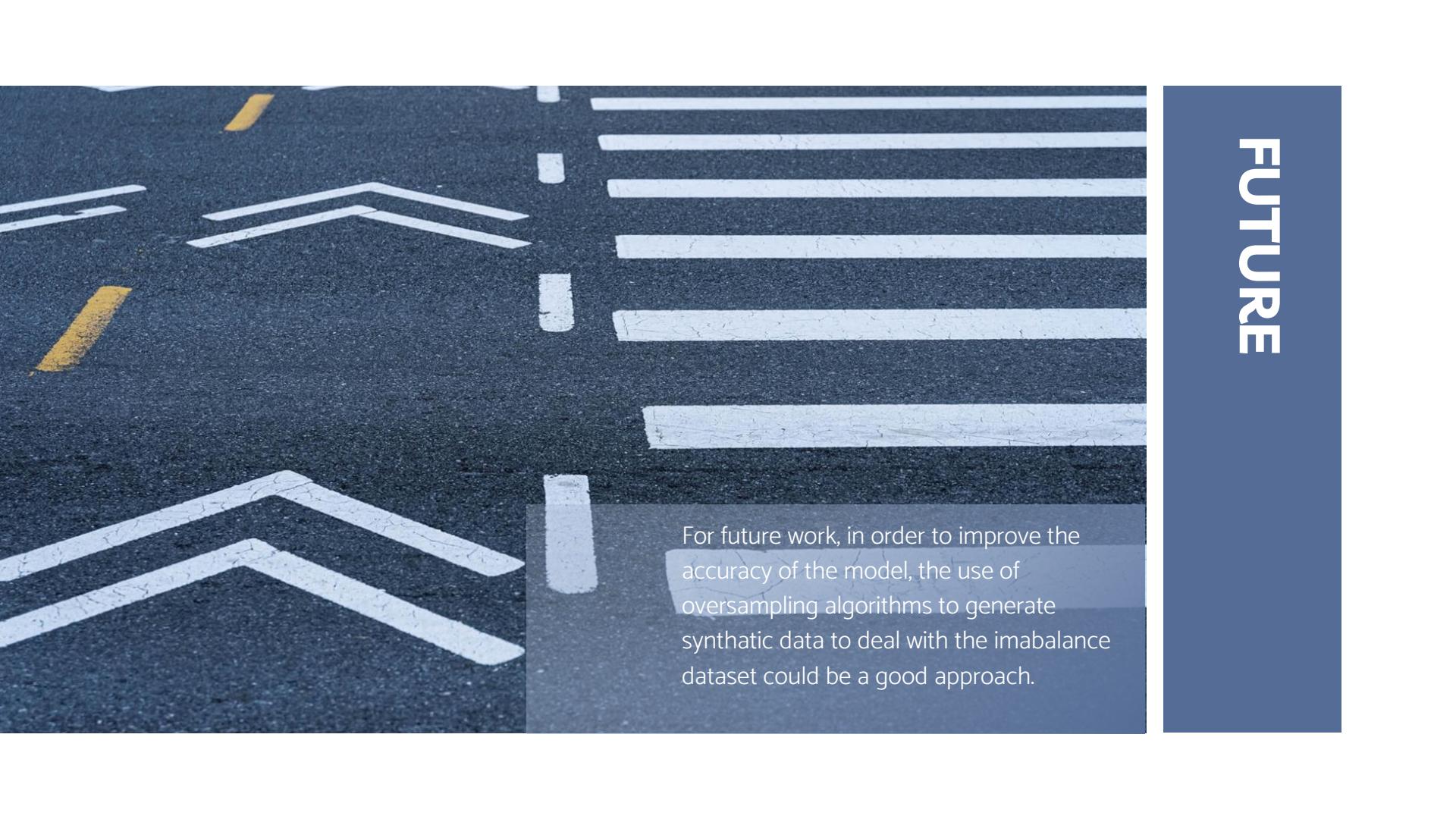


## TFIDF VECTORIZER

TFIDF vectorization is based on, a term frequency measuring the frequency of a word within a document, and an inverse document frequency measuring the rarity of a word across documents.

# DATA MODELING





# FUTURE

For future work, in order to improve the accuracy of the model, the use of oversampling algorithms to generate synthetic data to deal with the imbalanced dataset could be a good approach.

# THANKS

Does anyone have any questions?

jpozuelosaizdebustam@hawk.iit.edu  
mfdezlopezareal@hawk.iit.edu



# Fake News Classification

CS579 - Project II

Bharat Bandaru  
Himamshu Lakkaraju

# Objective

- Classify tweets into 3 categories (Agreed/Disagreed/Unrelated) based on two tweet titles available.

# Dataset stats

Train Data: Total rows: 256,442

Data distribution:

Class	No of records	Percentage of data
Unrelated	175598	68.475
Agreed	74238	28.949
Disagreed	6606	2.576

As we can see from the above distribution of data, the classes are highly unbalanced and hence any classification approach most likely would not be able to perform or generalize well.

# Methods to balance datasets

- Resample existing data.
- Generate new text similar to the existing text.
- subsample the train data and take similar amounts of data for each class.

# Methods explored

- Generate new text similar to the existing text(GPT-2)
- Use existing deep models to generate text:

There are several methods to generate text some of them include GPT2, BERT models etc. While these models are really good at generating grammatically correct sentences they did not always produce similar sentences as the given sentence (The method used is MASK words in sentence and obtain similar sentences)

- Translate to another language and translate it back to english:

The other approach to create similar sentences is to translate text to another language using a ML model and translate it back to english. This would change some words in the sentences but would most likely preserve the overall meaning of the sentence with additional words added to the vocabulary.

- Augment/replace words in text:

Replace word in text with synonyms.

# Methods used

- Translation of text to another language.

while this idea is good and it works technically, When we tried to use Helsinki models this took a long time for the whole dataset and hence this idea was discarded.

- nlpaug synonym function:

This function generates similar words and replaces them in the sentences. This is quick and easy to generate text with similar meaning without any issues faced above and is used to generate data for agreed and disagreed class records. the minimum words to be replaced are set to 2 and the maximum words that can be replaced is set to 4.

# Dataset stats after rebalancing.

Train Data: Total rows: 436,341

Data distribution:

Class	No of records	Percentage of data
Unrelated	175598	40.240
Agreed	148456	34.023
Disagreed	112287	25.734

# Logistic Regression

A simple logistic Regression model is built with title1\_en and title2\_en combined. These combined titles are vectorized using the TF-IDF vectorizer and passed to the logistic regression model using data from the both titles to fit with minimum document frequency of around 20.

# Logistic Regression

The train data is split into train and validation datasets (70%,30%)

Classification report for train data is:

	precision	recall	f1-score	support
0	0.85	0.87	0.86	122918
1	0.86	0.85	0.85	103919
2	0.96	0.93	0.95	78601
accuracy				0.88 305438
macro avg	0.89	0.88	0.89	305438
weighted avg	0.88	0.88	0.88	305438

Classification report for validation data is:

	precision	recall	f1-score	support
0	0.82	0.84	0.83	52680
1	0.82	0.82	0.82	44537
2	0.95	0.92	0.94	33686
accuracy			0.85	130903
macro avg	0.87	0.86	0.86	130903
weighted avg	0.85	0.85	0.85	130903

# Advantages and Issues with Logistic Regression

## Advantages:

- The model is very easy to build.
- The vectorizer fits the vocabulary of the training data so this would work well if the test data this is used on has similar patterns or vocabulary to the training dataset (which is not usually the case in the tweet/text data)
- The training time for the model is comparatively less than training a neural network.

## Issues:

- Although the precision, recall and accuracy of the model is very high, this model is not a generalized model that would work for new data reliably as it's trained on the vocabulary of the available titles in the training data
- This also doesn't accurately depict the relation between the words in the titles or identify patterns that can be generalized.
- If the test data isn't similar to the training data, this model might not perform as well as it did on the training and validation data

# Alternative models & Challenges

- There are several neural network architectures that work well for sequences like text like LTSM models or siamese nets that use LSTM layers or GRU layers to find similarities between the inputs or between inputs and a reference value.
- These nets usually capture the word patterns in the data far better than the logistic regression can.
- There are many existing models like BERT which can be used to classify the text with very high accuracies.

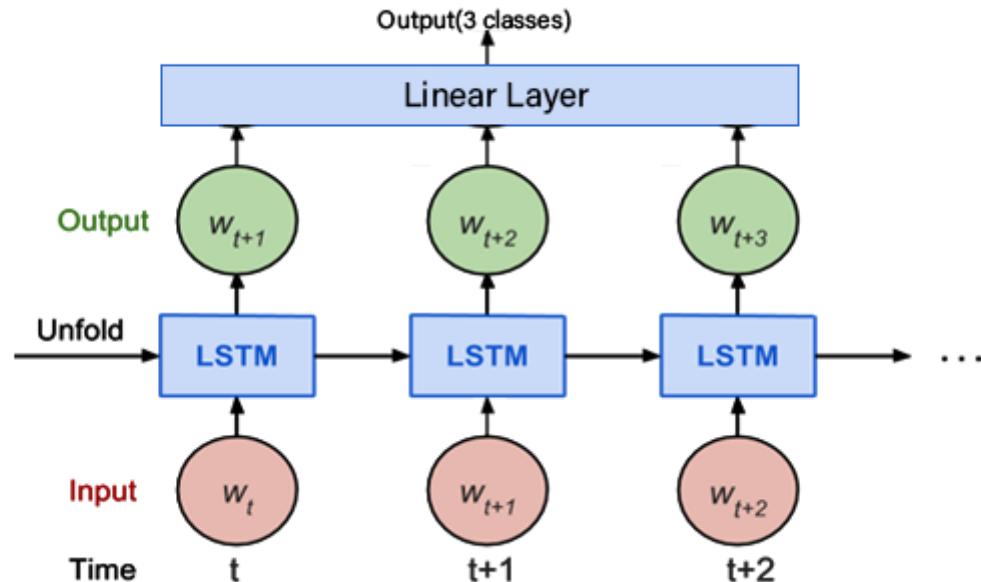
Challenges with neural networks:

- Neural networks tend to have overfitting/ underfitting issues
- Require large amounts of computing resources
- Hyper parameter optimization becomes difficult especially if the computational resources are limited.
- Have to train with multiple epochs so the training time will be longer.
- Imbalanced data would mean the network overfits to a class with large amount of data.

## Simple LSTM neural Network.

- The title 1 and tile 2 are combined and are encoded.
- This is passed to a LSTM network with vocabulary size 49491, embed\_dimension=256, lstm\_units,hidden\_dimension as 128 and 128.
- SGD optimizer with a learning rate of 0.01 and momentum of 0.9 is used.
- The loss function used is categorical cross entropy loss with weights calculated from sklearn class weights function to help the model generalize well.
- We could only use a batch size of upto 250 (due to cuda memory limitations)
- The loss goes into a plateau after an epoch and stays the same after. The accuracy is in the range of 34 to 50%.
- This might be primarily due to the batch size. With data this big the batch used heavily determines the loss and learning of the model.

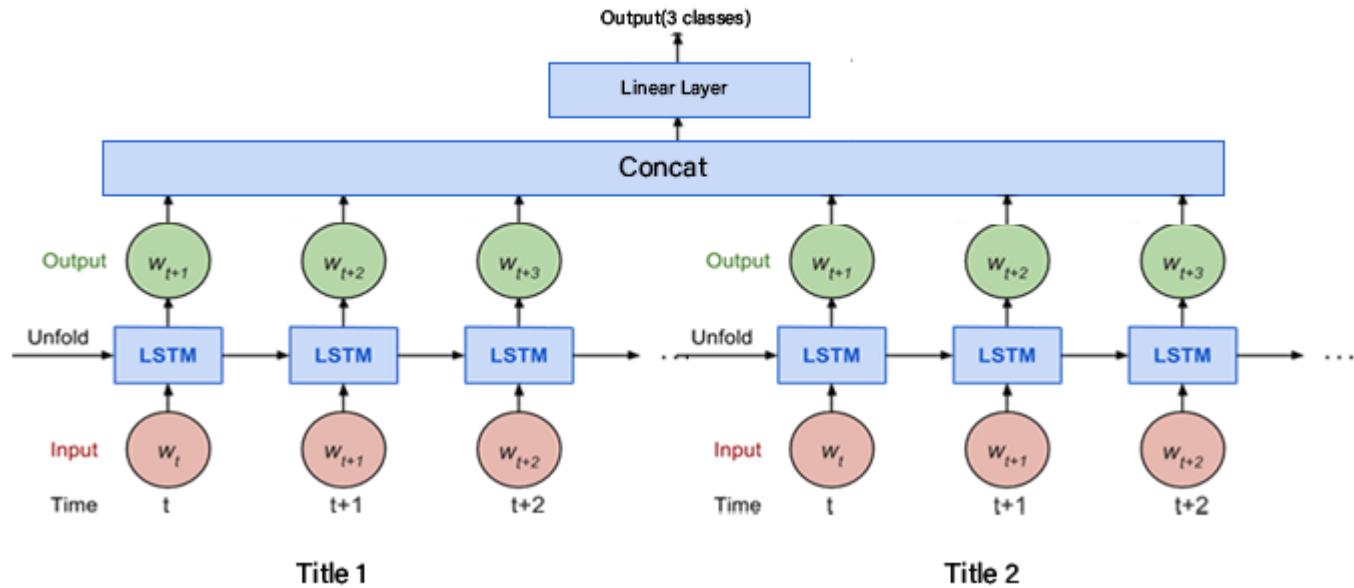
## Simple LSTM architecture:



## **Siamese network:**

- These neural networks are used to identify similarities or dissimilarities between two input.
- The architecture is similar to the Simple LSTM model we created previously, it takes two encoded titles separately as inputs and passed through the LSTM. the outputs for each titles are concatenated and sent to the output layer.

## Siamese network:



## Accuracy of LSTM and Siamese LSTM nets:

- The accuracy of our simple LSTM network is usually around 34%
- The accuracy of Siamese network improves accuracy a little. It goes up to 60%

## Future Work:

- Since these models are just trained for 1 epoch and small batch size, we'd like to run this for more epochs with larger batch size and see if that improves the accuracy of these networks.

# Thank You



# Fake news detection

Jayaramchandar Pazhanikumar and Clinton george

# Weak learners

Feature Engineering -> Sif embedding<sup>1</sup>, Word mover distance, sentiment

Models made -> Random forest with bagging from the imblearn library.

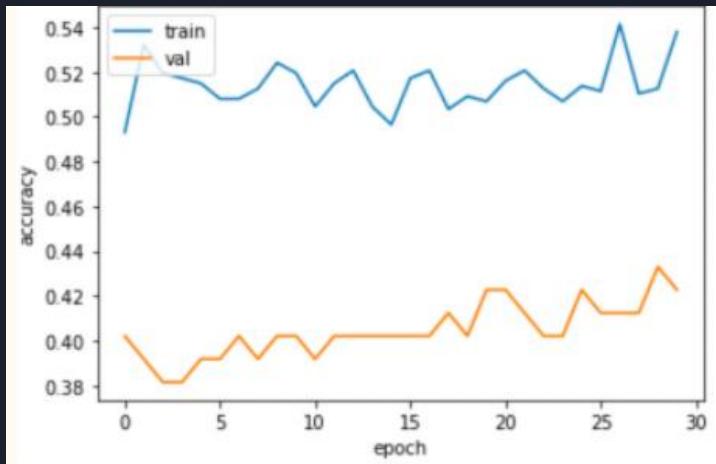
Classification report

	precision	recall	f1-score	support
-1	0.49	0.56	0.52	14843
1	0.05	0.28	0.08	1320
2	0.80	0.61	0.69	35104
accuracy			0.59	51267
macro avg	0.45	0.48	0.43	51267
weighted avg	0.69	0.59	0.63	51267

unrelated	175598
agreed	74238
disagreed	6606
Name: label, dtype: int64	

# Neural Networks (LSTM)

Voc\_size = 1000, sent\_length = 500, # of features=40



Layer (type)	Output Shape	Param #
<hr/>		
embedding_9 (Embedding)	(None, 500, 40)	40000
lstm_12 (LSTM)	(None, 100)	56400
dropout_12 (Dropout)	(None, 100)	0
flatten (Flatten)	(None, 100)	0
dense_9 (Dense)	(None, 3)	303
<hr/>		
Total params: 96,703		
Trainable params: 96,703		
Non-trainable params: 0		

# Social Media for Good: Cyberbullying on Instagram

Presented by Kayenat Patil

# Introduction

- Successful cyberbullying detection requires three things:
  - 1) Analyzing Language
  - 2) Identification of roles
  - 3) Successful Detection

But there is so much more to this!

# Basic approach

```
positive_word = [ 'awesome', 'interesting','fabulous','lovely','outstanding', 'fantastic', 'terrific', 'good', 'nice', 'great', ':)' ]  
negative_word = ['dislike', 'horrible','gross','bad','worse', 'terrible','useless', 'hate', 'meaningless','waste',':( ' ]  
neutral_word = [ 'movie','think','game','program','project','the','sound','was','is','actors','did','know','words','not' ]
```

Enter Sentence: I think this movie is gross, who comes up with such meaningless shit

~~~~~

Sentiment score:

~~~~~

Negetive: 0.46153846153846156

Positive: 0.15384615384615385

~~~~~

Sentiment Detected: Negetive

~~~~~

Enter Sentence: Its outstanding how not great a movie can be

~~~~~

Sentiment score:

~~~~~

Positive: 0.777777777777778

Negetive: 0.1111111111111111

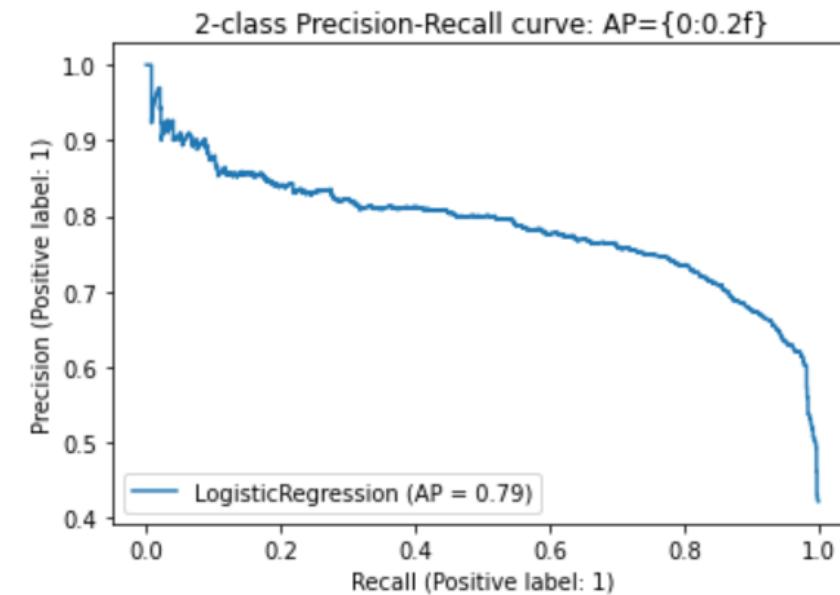
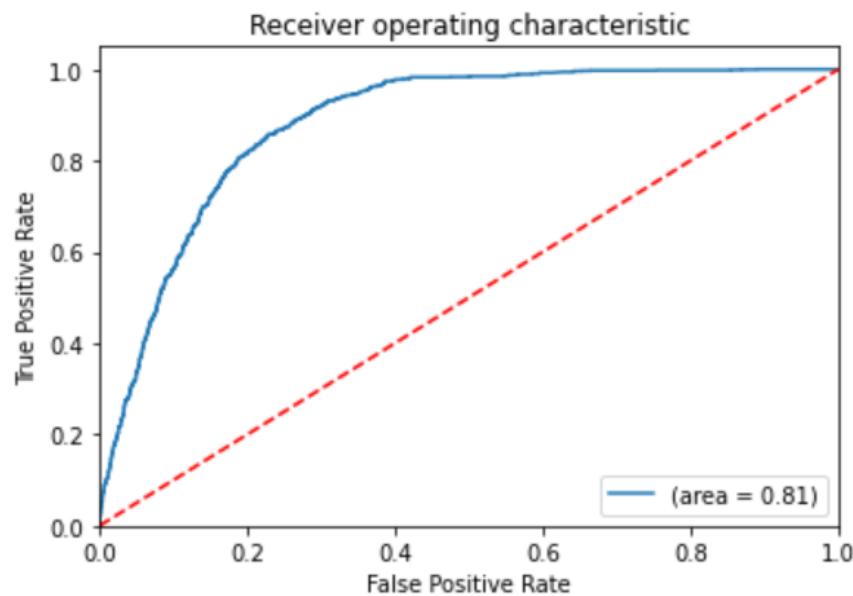
~~~~~

Sentiment Detected: Positive

~~~~~

# Supervised learning

- Logistic regression



Accuracy: 0.8055486128467882

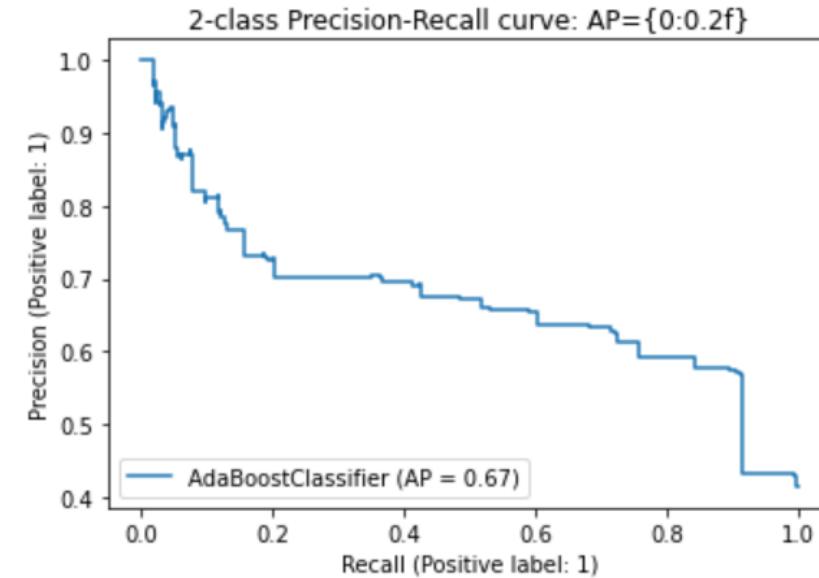
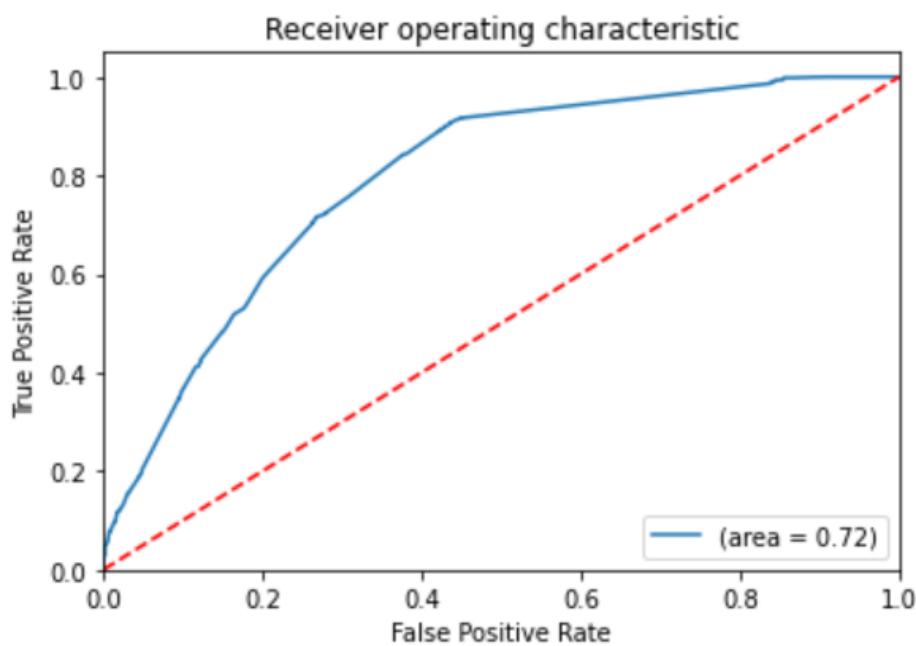
Confusion Matrix:

```
[[1918 511]
 [ 267 1305]]
```

	precision	recall	f1-score	support
0	0.88	0.79	0.83	2429
1	0.72	0.83	0.77	1572
accuracy			0.81	4001
macro avg	0.80	0.81	0.80	4001
weighted avg	0.82	0.81	0.81	4001

# Supervised learning

- Adaboost classifier



Accuracy: 0.7218195451137216

Confusion Matrix:

```
[[1755 674]
 [ 439 1133]]
```

	precision	recall	f1-score	support
0	0.80	0.72	0.76	2429
1	0.63	0.72	0.67	1572
accuracy			0.72	4001
macro avg	0.71	0.72	0.71	4001
weighted avg	0.73	0.72	0.72	4001

# Discrepancies and Bias

hate	Score: 533.8157298036014
fuck	Score: 503.76150769255435
damn	Score: 482.3875012051478
suck	Score: 407.37790877127185
ass	Score: 337.54089621427744
that	Score: 311.6250930420745
lol	Score: 298.0085779872157
im	Score: 296.0216055277791
like	Score: 287.8183474868775
you	Score: 284.7850587424088
it	Score: 254.75722294501585
get	Score: 253.19747902607998
what	Score: 221.43673623523864
know	Score: 211.53595900888456
would	Score: 202.5073882820925
bitch	Score: 193.08800391463464
ye	Score: 182.22364463196365
love	Score: 181.49014270754344

# Mitigating Bias using AIF360

AIF360 Algorithm: Adversarial debiasing

It is an in-processing technique that learns a classifier to maximize prediction accuracy and simultaneously reduce an adversary's ability to determine the protected attribute from the predictions

### Plain model - without debiasing - classification metrics

```
Test set: Classification accuracy = 0.804955
Test set: Balanced classification accuracy = 0.666400
Test set: Disparate impact = 0.000000
Test set: Equal opportunity difference = -0.470687
Test set: Average odds difference = -0.291055
Test set: Theil_index = 0.175113
```

### Model - with debiasing - classification metrics

```
Test set: Classification accuracy = 0.792056
Test set: Balanced classification accuracy = 0.672481
Test set: Disparate impact = 0.553746
Test set: Equal opportunity difference = -0.090716
Test set: Average odds difference = -0.053841
Test set: Theil_index = 0.170358
```

# Project 2: Fake News Classification

---

Adam Miller and Oladapo Ogunnaike

# Preprocessing

---

- Set all text in the headlines to lowercase
- Lemmatized each headline
- Deleted all stop words
- Did an 80-20 train-test split

# Model

---

- Took the cosine similarity of each headline pair
- Used that to train a decision tree classifier
  - Depth of 2
  - Purpose is to determine relatedness of articles
- Cut all unrelated articles to create “related” table
- Created TF-IDF with it to train a Passive-Aggressive Classifier
- Applied the model to the test data to get a “related” and “agreed/disagreed” column
- Used this info to get final labels

# Results

---

- Overall accuracy of 77% on average
- F1-Scores:
  - Agreed: 65%
  - Disagreed: 34%
  - Unrelated: 84%
- Many pairs were labeled unrelated because of the classifier
- PAC was good at determining agreement



# Fake News Classification

Charles Korban, Luke McDaneld



# Data Processing

- Undersampling
  - Large label imbalance
  - 68% unrelated, 28% agree, 2% disagree
- Train test split
  - 80-20 split in training sample to create validation set
- Tf-idf Vectorization
  - Form corpus using all headlines in training set
  - Fit training, validation, and test set on using vectorizer
  - Concatenate article A and article B vectors to form feature vector



# Model

- Multi-Layer Perceptron Classifier
  - Sklearn implementation
- Feed Forward Neural Network
  - Input layer
  - 3 hidden layers, 150 nodes. Relu activation
  - Output layer, Softmax to generate probabilities
  - Adam optimizer

# Validation Results

Validation Set Performance				
	precision	recall	f1-score	support
agreed	0.89	0.90	0.89	1300
disagreed	0.88	0.88	0.88	1341
unrelated	0.87	0.85	0.86	1322
accuracy			0.88	3963
macro avg	0.88	0.88	0.88	3963
weighted avg	0.88	0.88	0.88	3963

---

# Evidence-enhanced Detection of Disinformation in Crypto News

CS 597: Online Social Network Analysis

Spring 2022

Final project by:

Priyam Dinesh Shah ([pshah129@hawk.iit.edu](mailto:pshah129@hawk.iit.edu)),

William Edward Bodell III ([wbodell@hawk.iit.edu](mailto:wbodell@hawk.iit.edu))

## Problem statement:

Crypto has a fake news problem.  
Given the rate at which new  
information is disseminated through  
so called “Crypto Twitter”, how can an  
average user filter out the fake news?

---

# Motivation

---

[MARKETS](#)[BUSINESS](#)[INVESTING](#)[TECH](#)[POLITICS](#)[CNBC TV](#)[INVESTING CLUB](#)[PRO](#)

## **Walmart says crypto payments announcement is fake. Litecoin tumbles after spike**

PUBLISHED MON, SEP 13 2021 9:54 AM EDT | UPDATED TUE, SEP 14 2021 7:03 AM EDT

# Motivation

---



MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV INVESTING CLUB PRO

## Walmart says crypto payment announcement is fake. Litecoin tumbles after spike

PUBLISHED MON, SEP 13 2021 9:54 AM EDT | UPDATED TUE, SEP 14 2021 7:03 AM EDT

2022

CoinDesk

Layer 2 Newsletters

-1.67%

Ethereum \$2,054.86 -0.61%

XRP \$0.719640 -1.31%

Terra \$93.50 +3.11%

Solana \$100.89 +

Crypto Prices →

### Markets

#### Market Wrap: Bitcoin Whipsaws on Fake News, Investors Flow Into Solana Funds

Bitcoin traded in a choppy range on a fake press release about a Walmart partnership with Litecoin.

By Damanick Dantes · Sep 13, 2021 at 3:34 p.m. CDT · Updated Sep 13, 2021 at 3:44 p.m. CDT



# Motivation



MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV INVESTING CLUB PRO

## Walmart says crypto payment announcement is fake. Litecoin tumbles after spike

FEATURES

### Fake News, Websites And Companies: Inside The Dark World Of Crypto Ponzi Schemes



Suprita Anupam

Inc42 Staff

10 Jan'22 • 8 min read

SHARE STORY



- Beyond WhatsApp and Telegram, the international racket behind crypto Ponzi schemes also uses fake websites and posts fake interviews to attract more eyeballs
- The unknown operators posing as bitcoin investment analysts always insist on opening a Binance account at the first go

2022

CoinDesk

Layer 2 Newsletters

-1.67%

Ethereum \$2,054.86 -0.61%

XRP \$0.719640 -1.31%

Terra \$93.50 +3.11%

Solana \$100.89 +

Crypto Prices →

### Markets

#### Market Wrap: Bitcoin Whipsaws on Fake News, Investors Flow Into Solana Funds

Bitcoin traded in a choppy range on a fake press release about a Walmart partnership with Litecoin.

By Damanick Dantes · Sep 13, 2021 at 3:34 p.m. CDT · Updated Sep 13, 2021 at 3:44 p.m. CDT



# Motivation



MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV INVESTING CLUB PRO

## Walmart says crypto payment announcement is fake. [REDACTED] tumbles after spike

FEATURES

### Fake News, Websites And Companies: Inside The Dark World Of Crypto Ponzi Schemes



Suprita Anupam

Inc42 Staff

10 Jan'22 • 8 min read

SHARE STORY



- Beyond WhatsApp and Telegram, the international racket behind crypto Ponzi schemes also uses fake websites and posts fake interviews to attract more eyeballs
- The unknown operators posing as bitcoin investment analysts always insist on opening a Binance account at the first go

2022

CoinDesk

Layer 2 Newsletters

-1.67%

Ethereum \$2,054.86 -0.61%

XRP \$0.710640 -1.31%

Terra \$93.50 +3.11%

Solana \$100.89 -

Crypto Prices →

### Markets

#### Market Wrap: Bitcoin Whipsaws on Fake News, Investors Flow Into Solana Funds

Bitcoin traded in a choppy range on a fake press release about a Walmart partnership with Litecoin.

By Damanick Dantes · ① Sep 13, 2021 at 3:34 p.m. CDT · Updated Sep 13, 2021 at 3:44 p.m. CDT

Last Updated: 12th December, 2021 10:51 IST

#### Fake News Spread Via PM Modi's Hacked Twitter Account Underscores India's Crypto Concerns

The Centre had stated that it has taken no decision on placing a ban on cryptocurrency advertisements but still it expressed that it is a 'risky market'.

Written By Digital Desk



IMAGE: UNSPLASH



coindesk

# Motivation

≡ MARKETS BUSINESS

## Walma annou tumble

### FEATURES

## Fake News, Websites A Inside The Dark World Schemes



Suprita Anupam  
Inc42 Staff

10 Jan'22 • 8 min read

- Beyond WhatsApp and Telegram, the international地下组织 also uses fake websites and posts fake interviews to spread rumors.
- The unknown operators posing as bitcoin investors have Binance account at the first go.



Mr. Whale  
@CryptoWhale

Be careful of FAKE #Bitcoin news!

Many big crypto pages have been spreading false rumors that Alibaba, Google, Apple, Walmart, Facebook, and countless other big companies have bought into \$BTC

Don't Trust, Verify!

11:11 PM · Apr 30, 2021 · Twitter Web App

60 Retweets 2 Quote Tweets 274 Likes

CoinDesk

Layer 2 Newsletters

ket

## Wrap: Bitcoin Whipsaws on Fake News, Investors Flow Into Solana Funds

in traded in a choppy range on a fake press release about a Walmart partnership with

nick Dantes · ① Sep 13, 2021 at 3:34 p.m. CDT · Updated Sep 13, 2021 at 3:44 p.m. CDT

10:51 IST

## pread Via PM Modi's Hacked Account Underscores India's Crypto

it has taken no decision on placing a ban on cryptocurrency expressed that it is a 'risky market'.



# Motivation

---

1. The crypto market space is **1.78T** and forecasted to grow even higher.

# Motivation

---

1. The crypto market space is **1.78T** and forecasted to grow even higher.
2. The space is highly dynamic and market prices are very responsive to a **constant flood** of newsbites (**often misleading or fake**) spread mainly on social networks like Twitter.

# Motivation

---

1. The crypto market space is **1.78T** and forecasted to grow even higher.
2. The space is highly dynamic and market prices are very responsive to a **constant flood** of newsbites (**often misleading or fake**) spread mainly on social networks like Twitter.
3. Especially as crypto has begun to achieve mainstream adoption, scammers have relied heavily on tactics such as **embedding malicious links** in fake news articles to lure in unsuspecting and inexperienced users as victims.

# Motivation

---

1. The crypto market space is **1.78T** and forecasted to grow even higher.
2. The space is highly dynamic and market prices are very responsive to a **constant flood** of newsbites (**often misleading or fake**) spread mainly on social networks like Twitter.
3. Especially as crypto has begun to achieve mainstream adoption, scammers have relied heavily on tactics such as **embedding malicious links** in fake news articles to lure in unsuspecting and inexperienced users as victims.
4. Our research aims to help users assess the validity of crypto news articles based on evidence sourced from online knowledge bases, with cosine similarity scores between **0.0 and 1.0 on a grayscale**.

---

# Methodology

# Methodology

---

Our proposed methodology as we began our research was as follows:

1. Take input from the user some crypto news (i.e. a Tweet or a full article) he/she wants to verify.

# Methodology

---

Our proposed methodology as we began our research was as follows:

1. Take input from the user some crypto news (i.e. a Tweet or a full article) he/she wants to verify.
2. We apply basic filters to check the article has enough word count and is from the crypto space.

# Methodology

---

Our proposed methodology as we began our research was as follows:

1. Take input from the user some crypto news (i.e. a Tweet or a full article) he/she wants to verify.
2. We apply basic filters to check the article has enough word count and is from the crypto space.
3. If the input passes these filters, we process the input to create a bag of words.

# Methodology

---

Our proposed methodology as we began our research was as follows:

1. Take input from the user some crypto news (i.e. a Tweet or a full article) he/she wants to verify.
2. We apply basic filters to check the article has enough word count and is from the crypto space.
3. If the input passes these filters, we process the input to create a bag of words.
4. We then separately query our KBs (i.e. Google, Twitter, EtherScan) with the prominent key words, retrieving the top results to analyze and compare with the input article using TF-IDF.

# Methodology

---

Our proposed methodology as we began our research was as follows:

1. Take input from the user some crypto news (i.e. a Tweet or a full article) he/she wants to verify.
2. We apply basic filters to check the article has enough word count and is from the crypto space.
3. If the input passes these filters, we process the input to create a bag of words.
4. We then separately query our KBs (i.e. Google, Twitter, EtherScan) with the prominent key words, retrieving the top results to analyze and compare with the input article using TF-IDF.
5. With these we calculate a cosine similarity score from 0.0 to 1.0 for each query result.

# Methodology

---

Our proposed methodology as we began our research was as follows:

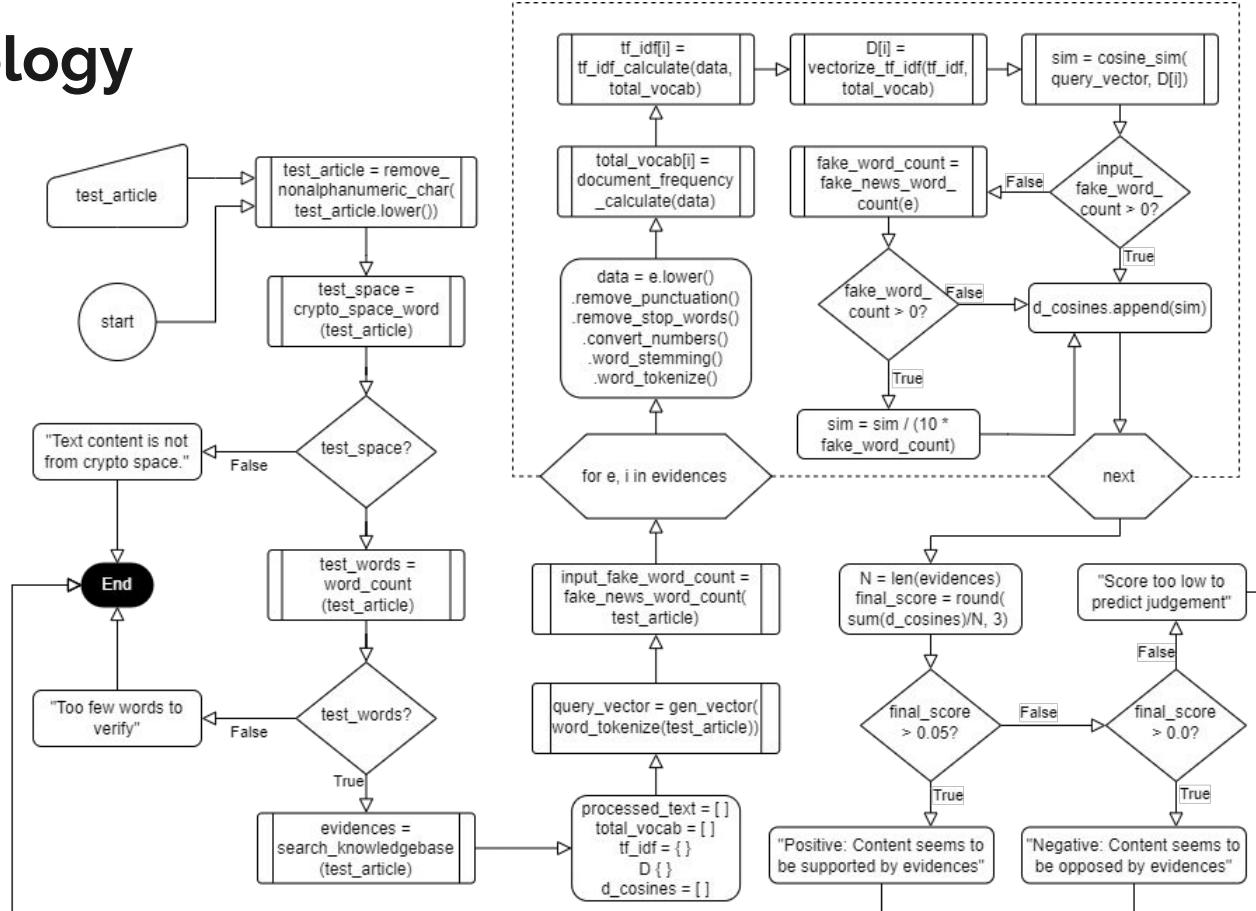
1. Take input from the user some crypto news (i.e. a Tweet or a full article) he/she wants to verify.
2. We apply basic filters to check the article has enough word count and is from the crypto space.
3. If the input passes these filters, we process the input to create a bag of words.
4. We then separately query our KBs (i.e. Google, Twitter, EtherScan) with the prominent key words, retrieving the top results to analyze and compare with the input article using TF-IDF.
5. With these we calculate a cosine similarity score from 0.0 to 1.0 for each query result.

*Testing our approach on some fake news, we observed high similarity scores with articles that reported the input as being fake, so we added the following step:*

6. We divide each similarity score by the number of words found in the evidence which indicate true reporting about fake news, unless the input text contains these words too

# Methodology

"Walmart is announcing it is going to be accepting a cryptocurrency. But it's not Bitcoin, it's Litecoin is what they have chosen. In a news release, they're saying they've entered a major partnership with Litecoin. And they're letting people pay with it."



---

# Test Results

# Results

---

Query Text : india has officially adopted bitcoin as legal tender the government has officially bought 500 btc and is distributing them to all residents of the country

~~~~~Final results~~~~~

Evidence 0 : Indian Prime Minister Narendra Modi's Twitter handle was compromised for a brief period on Saturday. The account was restored after sharing a link promising a Bitcoin BTC/USD giveaway. When it was hacked, misleading information was shared ...

Url: <https://www.benzinga.com/markets/cryptocurrency/21/12/24562022/indian-prime-minister-modis-twitter-handle-hacked-falsely-...>

Cosine Value: 0.05930240666005259

Evidence 1 : A tweet claiming that India has 'officially adopted bitcoin as legal tender' and bought 500 BTC and is distributing them among its residents was put out from itPrime Minister Narendra Modi's Twitter handle was briefly hacked on Sunday, and a ...

Url: <https://www.firstpost.com/india/pm-narendra-modis-personal-twitter-handle-very-briefly-compromised-later-secured-pmo-...>

Cosine Value: 0.04119176549959819

Evidence 2 : The tweet, after Prime Minister Narendra Modi's personal handle was hacked, also claimed that India has officially bought 500 BTC and is distributing them among its residents. Prime Minister Narendra Modi's Twitter handle was briefly hacked ...

Url: <https://www.thehindu.com/news/national/pm-modis-twitter-handle-very-briefly-compromised-later-secured-pmo/article37937402.ece>

Cosine Value: 0.03994367663954965

...

Average cosine similarity value for query and evidences is 0.023

Negative 😞!, Content seems to be opposed by evidences

# Results

---

Query Text : Walmart is announcing it is going to be accepting a cryptocurrency. But it's not Bitcoin, it's Litecoin is what they have chosen. In a news release, they're saying they've entered a major partnership with Litecoin.

~~~~~Final results~~~~~

Evidence 0 : Walmart was the subject of a fake news release issued on Monday, Sept. 13, that falsely stated Walmart announced a partnership with Litecoin (LTC).Walmart had no knowledge of the press release issued by GlobeNewswire, and it is incorrect. ...

Url: <https://corporate.walmart.com/newsroom/2021/09/13/walmart-statement-in-response-to-fake-litecoin-press-release>

Cosine Value: 0.13488480530510094

Evidence 1 : Cryptocurrency litecoin gave up a 20% gain and tumbled back to Earth following a fake press release sent out by GlobeNewswire that referenced a partnership with Walmart.Walmart spokesman Randy Hargrove confirmed that the press release is not authentic.He also said the retailer has been in touch with the newswire company to investigate how the false press release got ...

Url: <https://www.cnbc.com/2021/09/13/walmart-to-accept-payments-with-cryptocurrencies-using-litecoin.html>

Cosine Value: 0.05652246624051879

Evidence 2 : A fake press statement released Monday claimed Walmart was partnering with the cryptocurrency Litecoin, leading to a brief but major stock gain for the company.The press release and the supposed merger was quickly revealed to be untrue, and its inauthenticity was confirmed by a Walmart spokesperson to CNBC.In the press release, the claim was made that Walmart—the biggest...

Url: <https://www.newsweek.com/fake-announcement-that-walmart-will-accept-litecoin-cryptocurrency-sends-stock-soaring-1628492>

Cosine Value: 0.044645119573423736

...

Average cosine similarity value for query and evidences is 0.042

Negative 😞!, Content seems to be opposed by evidences

# Results

---

Query Text : MetaMask has formed a new strategic partnership with four major crypto custodians: Gnosis Safe, Hex Trust, GK8 and Parfin.

~~~~~Final results~~~~~

Evidence 0 : MetaMask has formed a new strategic partnership with four major crypto custodians: Gnosis Safe, Hex Trust, GK8 and Parfin. Decentralized finance (DeFi) wallet and browser extension MetaMask formed a new strategic partnership with four major...

Url: <https://www.bitcoininsider.org/article/159867/metamask-expands-institutional-offering-integrating-new-crypto-custodians>

Cosine Value: 0.17062060717946706

Evidence 1 : MetaMask has formed new strategic partnerships with four major cryptocurrency custodians, namely Hex Trust, Paraffin, Gnosis Safe and GK8. This announcement follows active expansion efforts on the firm's offerings undergone by MetaMask...

Url: <https://coinguora.com/metamask-integrates-with-crypto-custodians-to-expand-offerings/>

Cosine Value: 0.1547151679517015

Evidence 2 : MetaMask expands institutional offering by integrating new crypto custodians 13 Aprile 2022 - 02:00PM

CointelegraphMetaMask has formed a new strategic partnership with four major crypto custodians: Gnosis Safe, Hex Trust, GK8 and ...

Url: <https://it.adfn.com/mercati/COIN/BTCUSD/crypto-news/87824170/metamask-expands-institutional-offering-by-integra>

Cosine Value: 0.12966830357745013

...

Average cosine similarity value for query and evidences is 0.083

Positive 😊!, Content seems to be supported by evidences

# Results

---

Query Text : COTI turns bullish after increased integration with the Cardano ecosystem and the upcoming launch of MultiDAG 2.0 and the Djed stablecoin reflect strengthening fundamentals.

~~~~~Final results~~~~~

Evidence 0 : Development never stops in the fast-paced and competitive crypto sector and COTI is one project that is flashing some bullish signs.VORTECS™ charts from Cointelegraph Markets Pro show that COTI, an enterprise-grade financial technology platform focused on decentralized payments and digitization for any form of currency, could be on the verge of a breakout.The indicator...

Url: <https://cointelegraph.com/news/indicators-flash-bullish-on-coti-ahead-of-its-mainnet-and-djed-stablecoin-launch>

Cosine Value: 0.3272001834850179

Evidence 1 : Quantstamp, DigitalBits and AirSwap climb higher as the wider market stagnates and traders take an early weekend in observance of Good Friday.Continue ReadingAltcoin WatchCryptocurrenciesMarketsmarkets pro COTI turns bullish after increased integration with the Cardano ecosystem and the upcoming launch of MultiDAG 2.0 and the Djed stablecoin reflect strengthening...

Url: <https://bitcoinfox.xyz/tag/markets-pro/>

Cosine Value: 0.1714705835814072

Evidence 2 : Epoxy Composites Market was valued at USD 30.80 billion in 2021 and projected to grow at a CAGR of 6.32% during ...  
Pune India, April 22, 2022 (GLOBE NEWSWIRE) -- The market has been studied for the below mentioned-segmentation and regional...

Url: <https://www.ehtrend.com.br/en/tt/7402224/revenue.html>

Cosine Value: 0.08058385859573304

...

Average cosine similarity value for query and evidences is 0.108

Positive 😊!, Content seems to be supported by evidences

---

# Future Scope

# Future Scope

---

- Adding more knowledge bases, such as Twitter and Etherscan, as we only got around to using Google search to find evidence.
- Using transactional data based on user requested query as a weighted parameter.
- Extending further to use expert-based suggestions as a weighted parameter for the results.
- Applying machine learning models to filter evidences.

# Thank you!



# **ONLINE SOCIAL MEDIA ANALYSIS PROJECT 2**

## **FAKE NEWS DETECTION**

### **Team members:**

Catherine Balraj - A20470315

Varun Raghu - A20461361

# OBJECTIVE

---

The extensive spread of fake news can have a serious negative impact on individuals and society

---

There is a need to change the way people interpret and respond to real news and filter the spread of fake news.

---

Therefore, it is important to detect fake news and misinformation in social media.

---

# DATA PREPROCESSING

Data cleaning methods include:

- Lowercase
- Punctuations
- Tokenization
- Stop words
- Lemmatization

NLTK is a powerful tool with various python modules and libraries to carry out NLP.

# FEATURE EXTRACTION

- For text analysis, a simple text can be converted into features with various techniques such as word embeddings, Bag of Words and TF-IDF.
- For our project we have used the TF-IDF technique which gives the whole context instead of reading the whole content. It works in such a way to eliminate stop words by scoring higher for important words and lower for stop words.

# DATA MODELING

For data modeling, we have used three classification models such as

- Logistic regression
- Naive Bayes
- Multilayer Perceptron Classifier

# DATA EVALUATION

| MODEL                 | TRAINING ACCURACY | VALIDATION ACCURACY |
|-----------------------|-------------------|---------------------|
| LOGISTIC REGRESSION   | 0.85              | 0.81                |
| NAIVE BAYES           | 0.79              | 0.76                |
| MULTILAYER PERCEPTRON | 0.99              | 0.98                |

# MODEL PREDICTION

- MLP classifier had the highest model performance, but it took long time to run due to the number of hidden layers. So, we pickled the model for later use.
- This pickled model was used to predict the labels of the test data and the output results are generated in the .csv format as submission.csv with rows id and label.

# CONCLUSION

- Data Preprocessing using NLTK libraries and feature extraction using the TF-IDF technique.
- Multilayer Perceptron model was the final model used for label prediction in test data due to its higher performance.
- In future, feature extraction can also be done using other methods such as Similarity, Bag of Words, etc. ML algorithms such as LSTM and other NLP models can also be used for modeling.



# Fake News Classification

By:  
Saksham Gulati  
Tanishq Malhotra

# Methodology

---

We first explored the data and understood the type of data we are dealing with.

```
train_data = pd.read_csv("./dataset/train.csv")
train_data.head()
```

```
# test = pd.read_csv("./dataset/test.csv")
# test.head()
```

✓ 0.7s

Python

|   | <b>id</b> | <b>tid1</b> | <b>tid2</b> | <b>title1_en</b>                                  | <b>title2_en</b>                                  | <b>label</b> |
|---|-----------|-------------|-------------|---|---|--------------|
| 0 | 195611    | 0           | 1           | There are two new old-age insurance benefits f... | Police disprove "bird's nest congress each per... | unrelated    |
| 1 | 191474    | 2           | 3           | "If you do not come to Shenzhen, sooner or lat... | Shenzhen's GDP outstrips Hong Kong? Shenzhen S... | unrelated    |
| 2 | 25300     | 2           | 4           | "If you do not come to Shenzhen, sooner or lat... | The GDP overtook Hong Kong? Shenzhen clarifi...   | unrelated    |
| 3 | 123757    | 2           | 8           | "If you do not come to Shenzhen, sooner or lat... | Shenzhen's GDP overtakes Hong Kong? Bureau of ... | unrelated    |
| 4 | 141761    | 2           | 11          | "If you do not come to Shenzhen, sooner or lat... | Shenzhen's GDP outpaces Hong Kong? Defending R... | unrelated    |

# Assumptions and Preprocessing

---

From the figure above we comfortably made a few conclusions :

1. Column label is the classification which is derived mostly from the “title1\_en” and “title2\_en” columns. It is a classification problem.
2. Other columns need not be compared in order to get final predictions so columns t1 and t2 were dropped
3. Data in “title1\_en” and “title2\_en” is english language and hence it can be assumed that this is a Natural Language Classification Problem.
4. Since this was an NLP Problem we did basic step in processing which is to clean data of any special characters.

# Methodology

---

We completed the prediction in two phases :

## **Phase 1 : Creating Baseline Model**

Baseline model was designed to make show that final classifier was good enough for prediction. It is the threshold for the actual model and it is used as a reference in order to evaluate how good the actual model.

## **Phase 2 : Predicting using BERT Classifier**

Bert Classifier is the actual classifier which is used to create the final results.

# Phase 1 : Creating a Baseline Model



```
10  # This class is designed to extend Text Preprocessing capabilities
11
12  #author : Saksham Gulati
13
14
15
16  class PreprocessText():
17      # dataframe = pd.DataFrame
18      # columns = []
19
20      def __init__(self, dataframe, columnsList=[]):
21          self.dataframe = dataframe
22          self.columns = columnsList
23          nltk.download('stopwords')
24          nltk.download('punkt')
25
26      def tokenise(self, df=None, columnsList=[]):
27          if (df is not None and columnsList):
28              self.dataframe = df
29              self.columns = columnsList
30
31          for item in self.columns:
32              tokenised_column = 'tokenised_' + str(item)
33              self.dataframe[str(tokenised_column)] = self.dataframe.apply(
34                  lambda row: word_tokenize(row[item]), axis=1)
35
36              self.dataframe[str(tokenised_column)] = self.dataframe[str(
37                  tokenised_column)].apply(lambda row: ' '.join(row))
38
39          return self.dataframe
40
41      # Cleans a string: Lowercasing, trimming, removing non-alphanumeric
42      def clean(self, df=None, columnsList=[]):
43          if (df is not None and columnsList):
44              self.dataframe = df
45              self.columns = columnsList
46
47          for item in self.columns:
48              # tokenised_column = 'tokesined_' + str(item)
49              try:
50                  self.dataframe[str(item)] = self.dataframe[str(item)].apply(
51                      lambda row: " ".join(re.findall(r'\w+', row, flags=re.UNICODE)).lower())
52              except Exception as e:
53                  print(
54                      "Exiting with error :: The input does not seems to be string try cleaning data before tokenising it.")
55
56
57      return self.dataframe
```

For Baseline model we chose a simple LSTM network with two identical layers  
The process for creating the baseline model was as follows :

1. Tokenization : Tokenization is the process to break a sentence into small chunks. Image on the left is code implementation of class which we wrote for preprocessing
2. Creating Corpus : Corpus is a collection of all the vocabulary which needs to be created. In our case our corpus consists of combined content title1\_en and title2\_en

# Models - Specifications and Implementations

---

- After preprocessing the data and splitting into training and testing sets, we are now categorizing the labels by assigning them numerical values such as 0 for unrelated, 1 for agreed and 2 for disagreed so that it is for the model to differentiate and classify the desired output.
- Following layers were used to setup the LSTM Model :
  - Two Input Layers
  - One Embedding Layer
  - One LSTM Layer
  - One Concatenate Layer
  - One Dense Layer

- 
- We have used the following configurations as per available resources after a few rounds of brute force experimentation :
    - Batch Size = 128
    - Epochs = 10
    - Sequence Length = 20
    - Learning Rate = 0.01
    - Optimizer = Adam

# Models - Specifications and Implementations

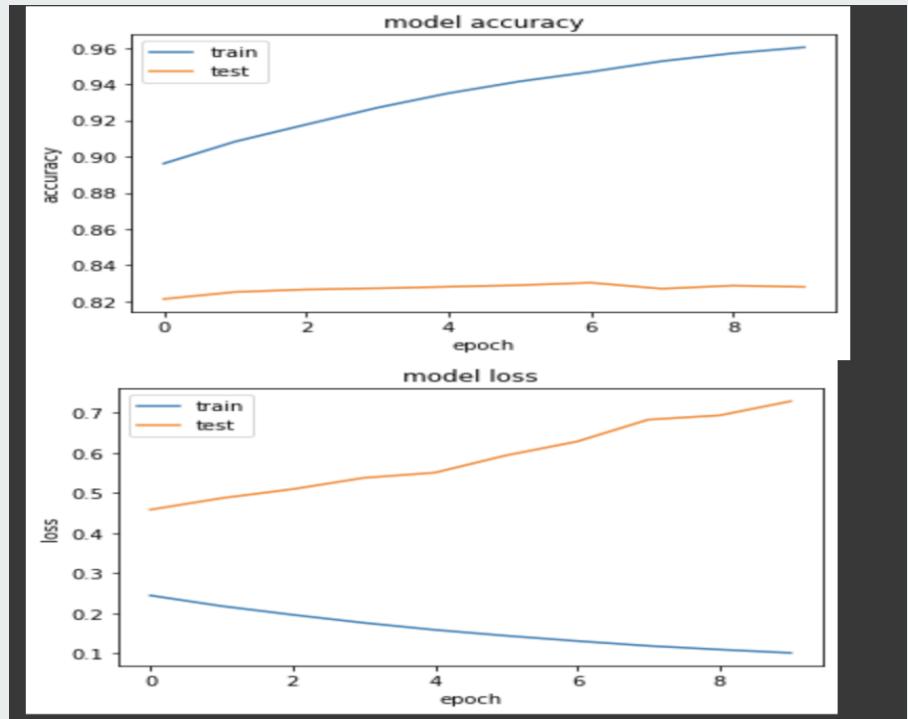
---

| Layer (type)                | Output Shape    | Param # | Connected to                               |
|-----------------------------|-----------------|---------|--|
| <hr/>                       |                 |         |  |
| input_3 (InputLayer)        | [(None, 20)]    | 0       | []   |
| input_4 (InputLayer)        | [(None, 20)]    | 0       | []   |
| embedding_1 (Embedding)     | (None, 20, 256) | 2560000 | ['input_3[0][0]', 'input_4[0][0]']         |
| lstm_1 (LSTM)               | (None, 128)     | 197120  | ['embedding_1[0][0]', 'embedding_1[1][0]'] |
| concatenate_1 (Concatenate) | (None, 256)     | 0       | ['lstm_1[0][0]', 'lstm_1[1][0]']           |
| dense_1 (Dense)             | (None, 3)       | 771     | ['concatenate_1[0][0]']                    |
| <hr/>                       |                 |         |  |
| Total params: 2,757,891     |                 |         |  |
| Trainable params: 2,757,891 |                 |         |  |
| Non-trainable params: 0     |                 |         |  |

# Results

---

We can see from the image below that the results are not very reliable as there is not much change in accuracy and there is huge gap between values of validation and training sets. Hence we used a better model for our final predictions.



# Phase 2 : Final Classifier using BERT

---

For Final Model we chose Bert - Bidirectional Encoder Representation. BERT is an excellent pretrained model for language processing.

1. BERT is type of transformer model of encoder decoder type. It also uses self attention model which takes into consideration, the relationship between words. We have used Bert base which has 12 encoding layers.
2. The tokenization for BERT model was done using Bert Tokenizer
3. The no. of epochs we have used are 2. The resources required are huge and only a limited epochs can be run.
4. BERT uses the processes of transfer learning where a pre-trained model is used to transfer the knowledge on new data in order to achieve higher accuracy with faster speeds than legacy models.
5. We are using Bert-Base-Uncased where uncased means it does not differentiate between english and English

# Models - Specifications and Implementations

```
Output exceeds the size limit. Open the full output data in a text editor
BertForSequenceClassification(
    (bert): BertModel(
        (embeddings): BertEmbeddings(
            (word_embeddings): Embedding(30522, 768, padding_idx=0)
            (position_embeddings): Embedding(512, 768)
            (token_type_embeddings): Embedding(2, 768)
            (LayerNorm): BertLayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
        )
        (encoder): BertEncoder(
            (layer): ModuleList(
                (0): BertLayer(
                    (attention): BertAttention(
                        (self): BertSelfAttention(
                            (query): Linear(in_features=768, out_features=768, bias=True)
                            (key): Linear(in_features=768, out_features=768, bias=True)
                            (value): Linear(in_features=768, out_features=768, bias=True)
                            (dropout): Dropout(p=0.1, inplace=False)
                        )
                        (output): BertSelfOutput(
                            (dense): Linear(in_features=768, out_features=768, bias=True)
                            (LayerNorm): BertLayerNorm()
                            (dropout): Dropout(p=0.1, inplace=False)
                        )
                    )
                )
            ...
        )
        (dropout): Dropout(p=0.1, inplace=False)
        (classifier): Linear(in_features=768, out_features=3, bias=True)
    )
)
```

1. BERT Base Model uses 12 layers on encoders
1. This model has one dropout layer in the last.
1. The final layer of the model is the classification layer with three features namely : unrelated, agreed and disagreed.
1. We have used the following configurations :
  - Epochs = 2
  - Optimizer = Adam
  - Learning Rate = 5e5
  - Batch Size = 128
  - Max Sequence = 32

# Models - Specifications and Implementations

```
1 ✓ BertForSequenceClassification(  
2 ✓   (bert): BertModel(  
3 ✓     (embeddings): BertEmbeddings(  
4 ✓       (word_embeddings): Embedding(30522, 768, padding_idx=0)  
5 ✓       (position_embeddings): Embedding(512, 768)  
6 ✓       (token_type_embeddings): Embedding(2, 768)  
7 ✓     (LayerNorm): BertLayerNorm()  
8 ✓     (dropout): Dropout(p=0.1, inplace=False)  
9   )  
10 ✓   (encoder): BertEncoder(  
11 ✓     (layer): ModuleList(  
12 ✓       (0): BertLayer(  
13 ✓         (attention): BertAttention(  
14 ✓           (self): BertSelfAttention(  
15 ✓             (query): Linear(in_features=768, out_features=768, bias=True)  
16 ✓             (key): Linear(in_features=768, out_features=768, bias=True)  
17 ✓             (value): Linear(in_features=768, out_features=768, bias=True)  
18 ✓             (dropout): Dropout(p=0.1, inplace=False)  
19 ✓           )  
20 ✓           (output): BertSelfOutput(  
21 ✓             (dense): Linear(in_features=768, out_features=768, bias=True)  
22 ✓             (LayerNorm): BertLayerNorm()  
23 ✓             (dropout): Dropout(p=0.1, inplace=False)  
24 ✓           )  
25 ✓         )  
26 ✓       (intermediate): BertIntermediate(  
27 ✓         (dense): Linear(in_features=768, out_features=3072, bias=True)  
28 ✓       )  
29 ✓       (output): BertOutput(  
30 ✓         (dense): Linear(in_features=3072, out_features=768, bias=True)  
31 ✓         (LayerNorm): BertLayerNorm()  
32 ✓         (dropout): Dropout(p=0.1, inplace=False)  
33 ✓       )  
34     )  
35   (1): BertLayer(  
36     (attention): BertAttention(  
37       (self): BertSelfAttention(  
38         (query): Linear(in_features=768, out_features=768, bias=True)  
39         (key): Linear(in_features=768, out_features=768, bias=True)  
40         (value): Linear(in_features=768, out_features=768, bias=True)  
41         (dropout): Dropout(p=0.1, inplace=False)  
42       )  
43       (output): BertSelfOutput(  
44         (dense): Linear(in_features=768, out_features=768, bias=True)  
45         (LayerNorm): BertLayerNorm()  
46         (dropout): Dropout(p=0.1, inplace=False)  
47     )
```

```
48   )  
49   (intermediate): BertIntermediate(  
50     (dense): Linear(in_features=768, out_features=3072, bias=True)  
51   )  
52   (output): BertOutput(  
53     (dense): Linear(in_features=3072, out_features=768, bias=True)  
54     (LayerNorm): BertLayerNorm()  
55     (dropout): Dropout(p=0.1, inplace=False)  
56   )  
57 )  
58 (2): BertLayer(  
59   (attention): BertAttention(  
60     (self): BertSelfAttention(  
61       (query): Linear(in_features=768, out_features=768, bias=True)  
62       (key): Linear(in_features=768, out_features=768, bias=True)  
63       (value): Linear(in_features=768, out_features=768, bias=True)  
64       (dropout): Dropout(p=0.1, inplace=False)  
65     )  
66     (output): BertSelfOutput(  
67       (dense): Linear(in_features=768, out_features=768, bias=True)  
68       (LayerNorm): BertLayerNorm()  
69       (dropout): Dropout(p=0.1, inplace=False)  
70     )  
71   (intermediate): BertIntermediate(  
72     (dense): Linear(in_features=768, out_features=3072, bias=True)  
73   )  
74   (output): BertOutput(  
75     (dense): Linear(in_features=3072, out_features=768, bias=True)  
76     (LayerNorm): BertLayerNorm()  
77     (dropout): Dropout(p=0.1, inplace=False)  
78   )  
79 )  
80 (3): BertLayer(  
81   (attention): BertAttention(  
82     (self): BertSelfAttention(  
83       (query): Linear(in_features=768, out_features=768, bias=True)  
84       (key): Linear(in_features=768, out_features=768, bias=True)  
85       (value): Linear(in_features=768, out_features=768, bias=True)  
86       (dropout): Dropout(p=0.1, inplace=False)  
87     )  
88     (output): BertSelfOutput(  
89       (dense): Linear(in_features=768, out_features=768, bias=True)  
90       (LayerNorm): BertLayerNorm()  
91       (dropout): Dropout(p=0.1, inplace=False)  
92     )  
93   )
```

# Results

---

1. With BERT Model which used in phase 2, we were able to achieve approximately 89% accuracy.
2. The BERT model required much more resources than the basic LSTM model. And much more time just to train 2 epochs.

accuracy: 0.8962

1. As a result we conclude that BERT is a better model for finding similarity in sentences and deal with natural language data.

# References

---

1. [1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
1. Pretrained Transformers for Text Ranking: BERT and Beyond | Proceedings of the 14th ACM International Conference on Web Search and Data Mining
1. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A Search Space Odyssey," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222-2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.
1. Libraries
  - Pandas: <https://pandas.pydata.org/pandas-docs/stable/>
  - Numpy: <https://numpy.org/doc/stable/>
  - Pytorch: <https://pytorch.org/docs/stable/index.html>
  - Tensorflow: [https://www.tensorflow.org/api\\_docs](https://www.tensorflow.org/api_docs)
  - Keras: <https://faroit.com/keras-docs/1.2.0/>
  - Matplotlib: <https://devdocs.io/matplotlib~3.1/>
  - NLTK : <https://www.nltk.org/>

## **Project 2 (Option 1)**

Sai Tharun Pathakota   Miles Bakenhus

4/27/2022

# Pre-processing/Feature extraction

**GOAL:** Classify titles  $A$  &  $B$  as: “agreed”, “disagreed”, or “unrelated”

**Pre-processing steps were taken prior to learning the classification models:**

- Punctuation removed from titles.
- Characters set to lowercase
- Stopwords removed from titles e.g. remove “the” and “a” [Loper and Bird, 2002]
- Lemmatization was performed e.g. “is” changed to “be” [Loper and Bird, 2002]
- Numeric characters were converted to words [Dupras, 2021].

**Features were extracted using sklearn [Pedregosa et al., 2011]:**

- Smoothed tf-idf feature matrices generated for respective titles:  $X_A, X_B$
- Smoothed idf is slightly different from textbook definition:

$$\text{idf}(t) = \log \frac{1 + n}{1 + \text{df}(t)} + 1$$

- Cosine similarity was computed for each instance  $i$ :

$$\sigma_i = \sigma_{\cos} ((X_A)_i^T, (X_B)_i^T)$$

**Final feature matrix (same process for  $X_{\text{test}}$ ):**

$$X_{\text{train}} = (X_A \quad X_B \quad \sigma)$$

### Linear Support Vector Classification (LSVC) and Multinomial Logistic Regression (MLR):

- Used an  $l_2$  penalty term in cost minimization:

$$\frac{1}{2} w^T w$$

- Trained with labels  $Y_{\text{train}}$  and the feature matrix  $X_{\text{train}}$

#### MLR Specific:

- Log-linear model for multinomial probabilities
- "saga" solver: variation of a Stochastic Average Gradient descent

#### LSVC Specific:

- Classifies data by splitting support into hyperplanes with maximal separation
- Performs optimization on dual of cost minimization problem
- "one-vs-rest" classifier

# Results

Models were evaluated for total accuracy and k-fold cross validation:

- 16 folds
- Computed accuracy and weighted  $F_1$  for each fold
- Weighted  $F_1$  used because of training label imbalance

| Model       | Total Accuracy | mean(KFold Accuracy) | s.d.(KFold Accuracy) | mean(KFold $F_1$ ) | s.d.(KFold $F_1$ ) |
|-------------|----------------|----------------------|----------------------|--------------------|--------------------|
| MLR         | 0.869          | 0.839                | 0.002                | 0.835              | 0.003              |
| <b>LSVC</b> | <b>0.889</b>   | <b>0.842</b>         | <b>0.003</b>         | <b>0.839</b>       | <b>0.003</b>       |

**LSVC chosen to predict  $Y_{\text{test}}$  using  $X_{\text{test}}$  feature matrix based on total accuracy.**

## Python Packages Used

- pandas [pandas development team, 2020]
- numpy [Harris et al., 2020]
- nltk [Loper and Bird, 2002]
- num2words [Dupras, 2021]
- scipy [Virtanen et al., 2020]
- sklearn [Pedregosa et al., 2011].

# References

Virgil Dupras. num2words. <https://github.com/savoirfairelinux/num2words>, 2021.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.

Edward Loper and Steven Bird. NLTK: The natural language toolkit. *CoRR*, cs.CL/0205028, 2002. URL <http://dblp.uni-trier.de/db/journals/corr/corr0205.html#cs-CL-0205028>.

The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL <https://doi.org/10.5281/zenodo.3509134>.

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.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 10.1038/s41592-019-0686-2.

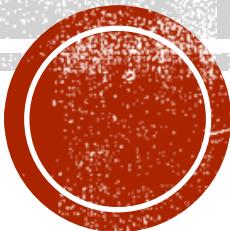
# **FAKE NEWS CLASSIFICATION**

**Done under- Professor Kai Shu**

**Submitted by:**

**Radhika Malhotra (A20491601)**

**Aman Singh (A20491333)**



# TASK AT HAND

- 2 ‘.csv files’ have been provided to us, namely ‘TRAIN.csv ’ and ‘TEST.csv ’.
- Train.csv contains our training data. It has 2 sets of news stories – title1\_en (existing fake news), & title2\_en (new incoming news article).
- Since this file is the training data, labels were provided based on the following rule:
  - “Unrelated” - if the incoming news has no relation to the existing one
  - “Agreed” - if the incoming news talks about the same news as the existing one
  - “Disagreed” - if the incoming news is contrary in information to the existing one
- We were supposed to come up with a code that could do this classification properly. So, based on the requirements, we chose to use 2 approaches, both very similar to each other, but different enough in the respect that it made a significant effect on the accuracy of our model for this task at hand.
- We’ve used different approaches of Keras’ Functional API for LSTM.



# OUR APPROACH

- Exploring the data:
  - As explained on the previous slide.
  - Additionally, figuring out the size & shape (dimensionality) of the dataset.
  - Determined the polarity of our data here
- Data Cleaning:
  - First, we remove null values from our training data & the index is reset.
- Preprocessing:
  - Cleaning data to replace all data that is not an alphabet, to a space.
- Tokenization:
  - This is one of the most important steps of our code. It's the process of breaking down larger texts/ sentences into individual pieces - words, or tokens.
  - We performed the task of tokenization with the help of the NLTK (Natural Language Tool Kit) library.



- **Stop Words Removal:**
  - Now, we remove the Stop Words, that are the most common words in English, for e.g., 'a', 'an', 'the' etc. These don't necessarily add a lot of meaning to the sentence, so their removal won't affect our model's performance.
  - But, it will help with reduction in processing time, and will require less storage which will be a major advantage as it can take hours to train the model with large datasets like the one we have at hand.
- **Lemmatization:**
  - The process where we remove ends of words, to reduce words to their root-word.
- **WordCloud Creation:**
  - This is a technique of visualizing the most frequent words occurring in the given data.
- **Model Creation & Fitting:**
  - Finally, we create our model by adding in all the layers.
  - Then, we fit the data to the model, set our batch size and epochs, and get the accuracy of the model.

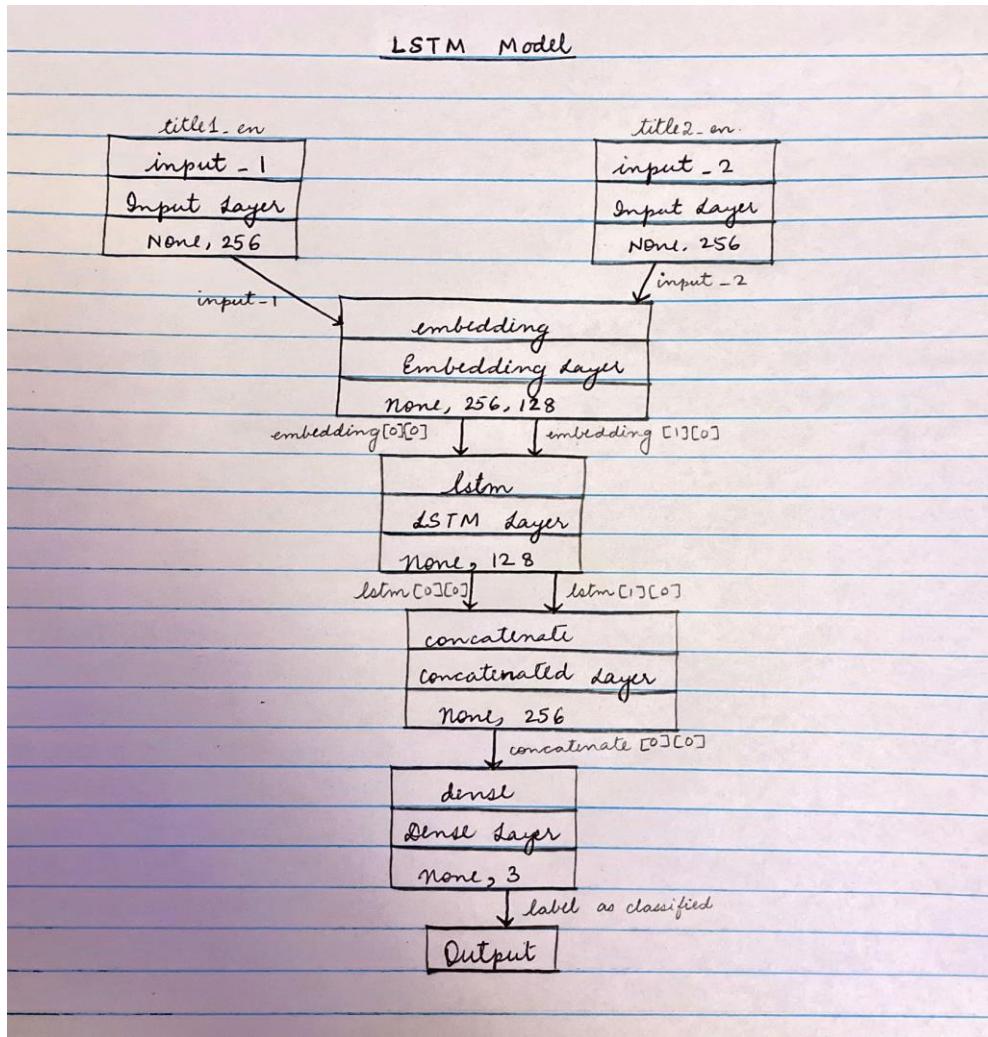


# THE MODEL

- Since this is a classification problem which has to accept 2 input values (title1\_en, title2\_en), and process them simultaneously, no Machine Learning model was being able to process this data in this required manner in its original state. So, upon doing some research, we found that there can be 2 approaches to solving a problem like this, namely:
  - Using a Sequential Model - either we combine the multiple inputs and feed them to the 'feedforward' network, or we process them separately, but sequentially. Therefore, this approach couldn't work for us, since we need both inputs to be processed simultaneously, since the comparison to test the model's accuracy is dependent on both of them
  - Using Functional API - Here, we are able to provide our model with multiple inputs at the same time, make layers for those multiple inputs and combine the outputs at the end. So, basically, branching or sharing of layers is allowed here, which is why it is perfect for our current problem. Hence, we chose to use this approach.



# KERAS' FUNCTIONAL API



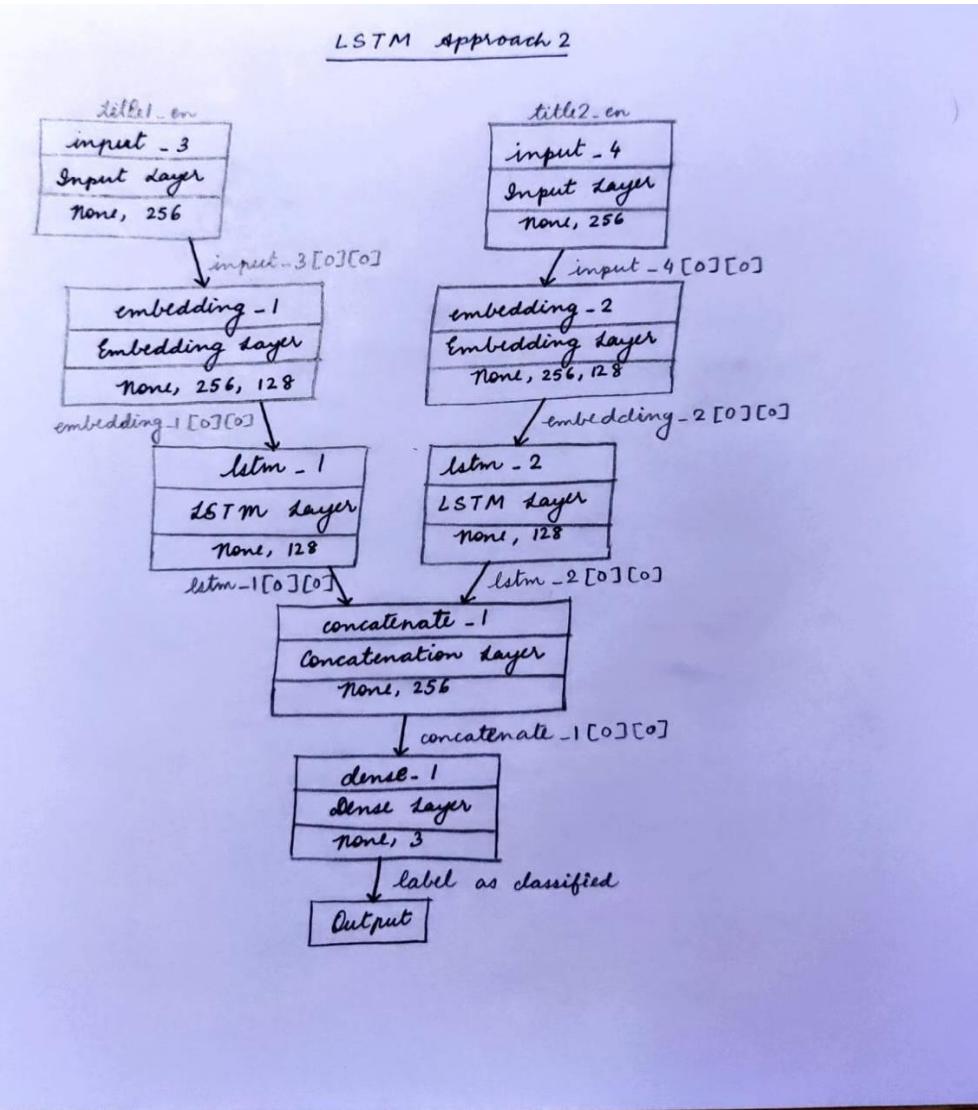
```
13 model.summary()
```

Model: "model"

| Layer (type)              | Output Shape     | Param # | Connected to                           |
|---------------------------|------------------|---------|--|
| input_1 (InputLayer)      | [None, 256]      | 0       | []                                     |
| input_2 (InputLayer)      | [None, 256]      | 0       | []                                     |
| embedding (Embedding)     | (None, 256, 128) | 256000  | ['input_1[0][0]', 'input_2[0][0]']     |
| lstm (LSTM)               | (None, 128)      | 131584  | ['embedding[0][0]', 'embedding[1][0]'] |
| concatenate (Concatenate) | (None, 256)      | 0       | ['lstm[0][0]', 'lstm[1][0]']           |
| dense (Dense)             | (None, 3)        | 771     | ['concatenate[0][0]']                  |

Total params: 388,355  
Trainable params: 388,355  
Non-trainable params: 0





```

13 Pred_Data = DenseLayer(Concat_Layer)
14 modelB = Model(inputs=[InputA, InputB], outputs=Pred_Data)
15 modelB.summary()

Model: "model_1"

Layer (type)          Output Shape       Param #     Connected to
=====
input_3 (InputLayer)   [(None, 256)]      0           []
input_4 (InputLayer)   [(None, 256)]      0           []
embedding_1 (Embedding) (None, 256, 128)  256000    ['input_3[0][0]']
embedding_2 (Embedding) (None, 256, 128)  256000    ['input_4[0][0]']
lstm_1 (LSTM)          (None, 128)        131584     ['embedding_1[0][0]']
lstm_2 (LSTM)          (None, 128)        131584     ['embedding_2[0][0]']
concatenate_1 (Concatenate) (None, 256)      0           ['lstm_1[0][0]', 'lstm_2[0][0]']
dense_1 (Dense)         (None, 3)          771        ['concatenate_1[0][0]']

Total params: 775,939
Trainable params: 775,939
Non-trainable params: 0

```

- The previous slide shows the ‘model summary’ of the Keras’ Functional API LSTM model we’ve currently implemented.
- Since the amount of time taken to run each epoch is very large, for the purpose of explaining the working, we have only run it for a small epoch of 5. Still, we got an accuracy of 82.99% for the 1<sup>st</sup> approach. If we give it plenty of time to execute, we will have a much better accuracy in a matter of a few hours.

```
✓ [24] 1 Batch_size = 128
17m 2 Epoch = 5
3
4 historyA = modelA.fit(x = [x1_train, x2_train], y = y_train, batch_size = Batch_size, epochs = Epoch)
5
Epoch 1/5
1603/1603 [=====] - 201s 125ms/step - loss: 0.4937 - accuracy: 0.7628 - val_
Epoch 2/5
1603/1603 [=====] - 200s 124ms/step - loss: 0.4609 - accuracy: 0.7824 - val_
Epoch 3/5
1603/1603 [=====] - 199s 124ms/step - loss: 0.4317 - accuracy: 0.8006 - val_
Epoch 4/5
1603/1603 [=====] - 205s 128ms/step - loss: 0.4040 - accuracy: 0.8153 - val_
Epoch 5/5
1603/1603 [=====] - 199s 124ms/step - loss: 0.3774 - accuracy: 0.8299 - val_
```



- In the 2nd approach, with 5 epochs, we obtained an accuracy of 80.83%.

```
✓ [25] 1 Batch_size = 128
       2 Epoch = 5
       3
       4 historyB = modelB.fit(x = [x1_train, x2_train], y = y_train, batch_size = Batch_size, epochs = Ep
17m
```

Epoch 1/5  
1603/1603 [=====] - 206s 126ms/step - loss: 0.5605 - accuracy: 0.7270 - val\_

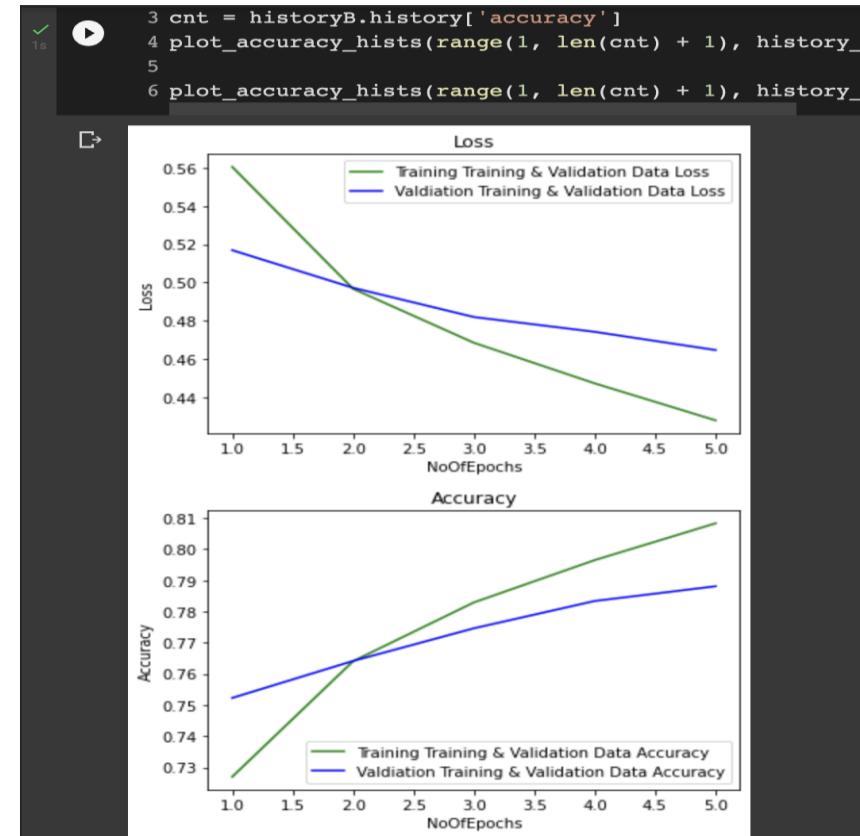
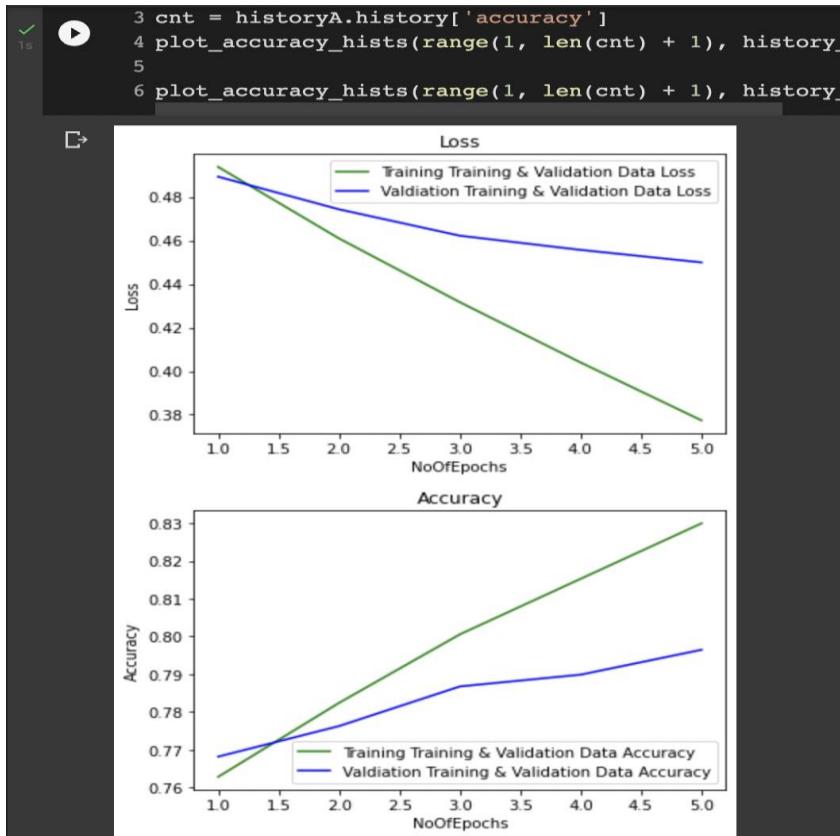
Epoch 2/5  
1603/1603 [=====] - 200s 125ms/step - loss: 0.4967 - accuracy: 0.7640 - val\_

Epoch 3/5  
1603/1603 [=====] - 201s 125ms/step - loss: 0.4685 - accuracy: 0.7829 - val\_

Epoch 4/5  
1603/1603 [=====] - 201s 125ms/step - loss: 0.4472 - accuracy: 0.7965 - val\_

Epoch 5/5  
1603/1603 [=====] - 201s 125ms/step - loss: 0.4279 - accuracy: 0.8083 - val\_

# SCREENSHOTS OF LOSS & ACCURACY GRAPH PLOTS FOR THE 2 APPROACHES



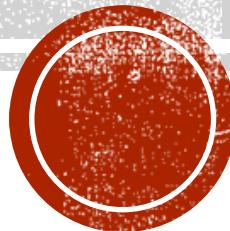
# REFERENCES

- <https://jon-dagdagan.medium.com/fake-news-detection-pre-processing-text-d9648a2854e5>
- [https://keras.io/guides/functional\\_api/](https://keras.io/guides/functional_api/)
- <https://stackoverflow.com/questions/59907559/making-inputs-to-keras-rnn-written-in-functional-api>
- [https://www.tensorflow.org/text/tutorials/text\\_classification\\_rnn](https://www.tensorflow.org/text/tutorials/text_classification_rnn)
- <https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce>
- <https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/>
- <https://machinelearningmastery.com/stacked-long-short-term-memory-networks/>
- [file:///Users/radhikamalhotra/Desktop/MTech%201-2/Online%20Social%20Network%20Analysis%20\(579\)/project%202/information-10-00150-v2.pdf](file:///Users/radhikamalhotra/Desktop/MTech%201-2/Online%20Social%20Network%20Analysis%20(579)/project%202/information-10-00150-v2.pdf)
- [https://keras.io/api/layers/core\\_layers/embedding/](https://keras.io/api/layers/core_layers/embedding/)
- <https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/>
- Etc.





**THANK YOU!**



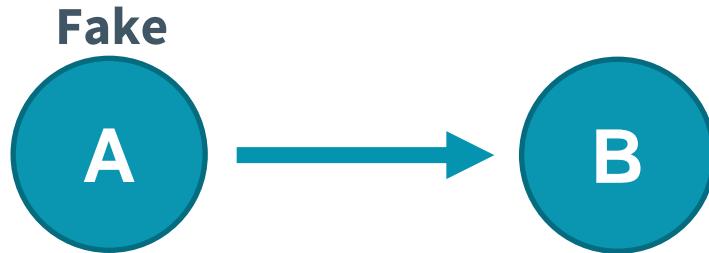
# Fake News Classification



Antonio Gonzalez-Cuellar Taboada  
Jorge Rosco Martin

- ▶ Problem description
- ▶ Preprocessing
- ▶ Models & Results
- ▶ Chosen Models
- ▶ Conclusion

## Problem description



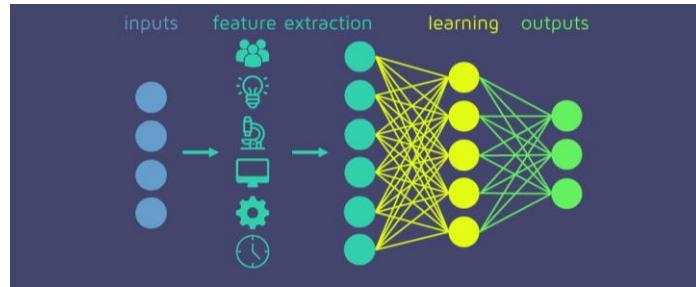
- ▶ Agreed – B talks about the same fake news as A.
- ▶ Disagreed – B refutes the fake news in A.
- ▶ Unrelated – B is unrelated to A

## Preprocessing

- ▶ Data cleansing
  - ▶ Expansion
  - ▶ Reduction
  - ▶ Tokenization
  - ▶ Padding
  - ▶ Splitting dataset
- 
- ```
graph TD; A["▶ 43It's an example45"] --> B["▶ It's an example"]; B --> C["▶ It is an example"]; C --> D["▶ It is example"]; D --> E["▶ [[3] [12] [36]]"]; E --> F["▶ [[3] [12] [36] [0] [0]]"]
```
- ▶ 43It's an example45
  - ▶ It's an example
  - ▶ It is an example
  - ▶ It is example
  - ▶ [[3] [12] [36]]
  - ▶ [[3] [12] [36] [0] [0]]

## Models & Results

- ▶ Model 1
  - ▶ SimpleRNN layer
  - ▶ Dense layers:
    - ◆ 512 neurons
    - ◆ 256 neurons
    - ◆ 128 neurons
    - ◆ 64 neurons
    - ◆ 32 neurons
    - ◆ 16 neurons
    - ◆ 8 neurons
    - ◆ 4 neurons
    - ◆ 3 neurons
  
- ▶ Model 2
  - ▶ SimpleRNN layer
  - ▶ Dense layers:
    - ◆ 256 neurons
    - ◆ 32 neurons
  - ▶ Dropout layer
  - ▶ Dense layer:
    - ◆ 3 neurons



### Relu

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

### Softmax

$$\text{Softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

## Models & Results

### ► Models 3 – 11

| Model                      | Accuracy_Training_Set | Accuracy_Test_Set | Precision | Recall   | f1_score | Training Time (secs) |
|----------------------------|-----------------------|-------------------|-----------|----------|----------|----------------------|
| RandomForestClassifier     | 0.999781              | 0.752501          | 0.752501  | 0.752501 | 0.752501 | 151.95               |
| GradientBoostingClassifier | 0.698805              | 0.700345          | 0.700345  | 0.700345 | 0.700345 | 2766.13              |
| AdaBoostClassifier         | 0.684216              | 0.686989          | 0.686989  | 0.686989 | 0.686989 | 204.30               |
| BernoulliNB                | 0.683251              | 0.68582           | 0.68582   | 0.68582  | 0.68582  | 2.05                 |
| DecisionTreeClassifier     | 0.999781              | 0.659752          | 0.659752  | 0.659752 | 0.659752 | 37.03                |
| SGDClassifier              | 0.58239               | 0.585603          | 0.585603  | 0.585603 | 0.585603 | 874.78               |
| LinearSVC                  | 0.571101              | 0.567958          | 0.567958  | 0.567958 | 0.567958 | 203.02               |
| GaussianNB                 | 0.29166               | 0.289711          | 0.289711  | 0.289711 | 0.289711 | 1.87                 |
| MultinomialNB              | 0.234235              | 0.229991          | 0.229991  | 0.229991 | 0.229991 | 1.18                 |

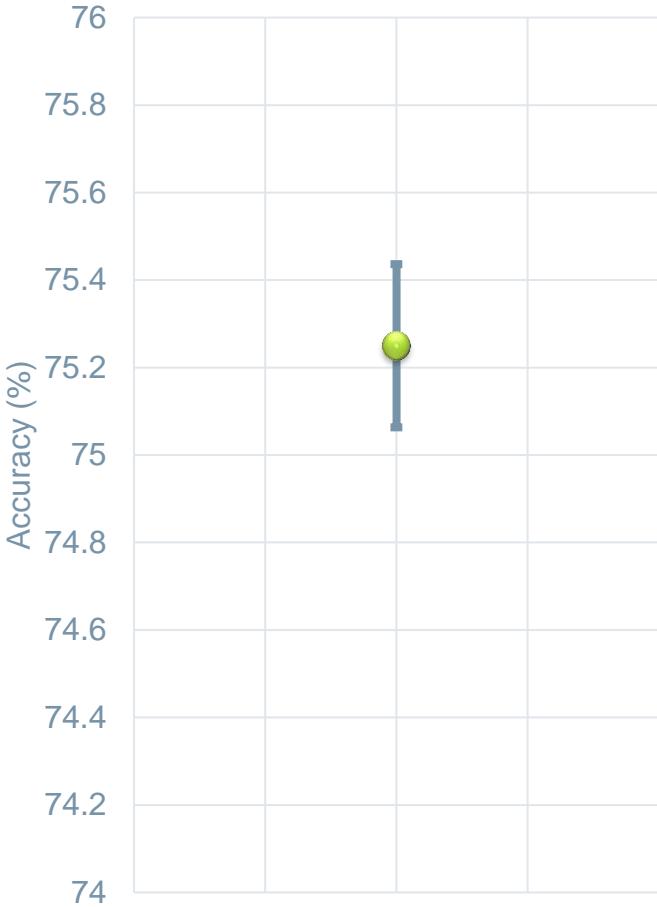
## Models & Results

### ► Models 3 – 11



## Chosen Model

- ▶ Random Forest Classification
  - ▶ Accuracy rate = 75,25%
  - ▶ Error band = [75.06 – 75.44]



## Conclusions

- ▶ A large amount of data has been preprocessed to improve the efficiency and implementation of the model.
  
- ▶ A model has been designed that predicts the next product a customer will buy with an accuracy of 75%

# Fake News Classification

CS579 – Project 2

Cyprien Gueyraud – Haroon Arif

# Similarity Score

**Step 1 :**

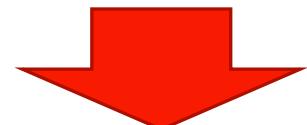
```
1 from sentence_transformers import SentenceTransformer, util  
2 import numpy as np
```



```
Sentence 1: I like Python because I can build AI applications  
Sentence 2: I like Python because I can do data analytics  
Similarity score: 0.8015284538269043
```



```
Agreed : 0.5423055588533836  
Disagreed : 0.5712711215019226  
UnRelated : 0.32446138422690435
```



We can separate Agreed/Disagreed and Unrelated

## Step 2 :

Key words

```
Agreed : 74238
Disagreed : 6606
Unrelated : 175598
Disagreed Words : [[4041, 'rumor'], [1151, 'rumour'], [763, 'ten'], [739, 'ice'], [729, 'ear'], [726, 'car'], [703, 'ne
Agreed Words : [[11251, 'one'], [10670, 'ear'], [9891, 'eat'], [7929, 'ate'], [7446, 'day'], [7170, 'tio'], [7029, 'har
Unrelated Words : [[42521, 'rumor'], [28531, 'ing'], [22374, 'eat'], [17285, 'ear'], [17268, 'ill'], [16997, 'ate'], [1
```



Rumor/Rumour present in about 80% of Disagreed messages



We can separate Agreed and Disagreed

## Repetition of words

Step 3 :

Example :

False news 1 : His **name** was written on it.  
News 2 : My **name** is Bond, James Bond.



1 word is repeated



↳ Agreed : 74238  
Disagreed : 6606  
Unrelated : 175598  
Disagreed average similar Words : 1.8380260369361188  
Agreed average similar Words : 3.131388237829683  
Unrelated average similar Words : 0.9745896878096562



Another way to find Agreed messages

# Test and Results

Some results we get :

Sample : 100

Disagreed : 3  
Agreed : 26  
Unrelated : 71

True : 75

Sample : 100

Disagreed : 2  
Agreed : 29  
Unrelated : 69

True : 75

Sample : 100

Disagreed : 1  
Agreed : 30  
Unrelated : 69

True : 72

Sample : 100

Disagreed : 0  
Agreed : 28  
Unrelated : 72

True : 72

Sample : 100

Disagreed : 10  
Agreed : 25  
Unrelated : 65

True : 78

Sample : 100

Disagreed : 10  
Agreed : 25  
Unrelated : 65

True : 74

# FAKE NEWS CLASSIFICATION

CS579- Project#2

Member:

Yiyi Lee

Yang Bai

# Result of Different batch size :

Batch size : 10

```
print(end - start)
```

```
device: cuda
epoch 1 ==> loss: 3411.513, accuracy: 0.8430
epoch 2 ==> loss: 2240.582, accuracy: 0.8503
epoch 3 ==> loss: 1381.465, accuracy: 0.8475
epoch 4 ==> loss: 886.531, accuracy: 0.8489
The time used to execute this is given below
6061.691191673279
```

Batch size : 50

```
print(end - start)
```

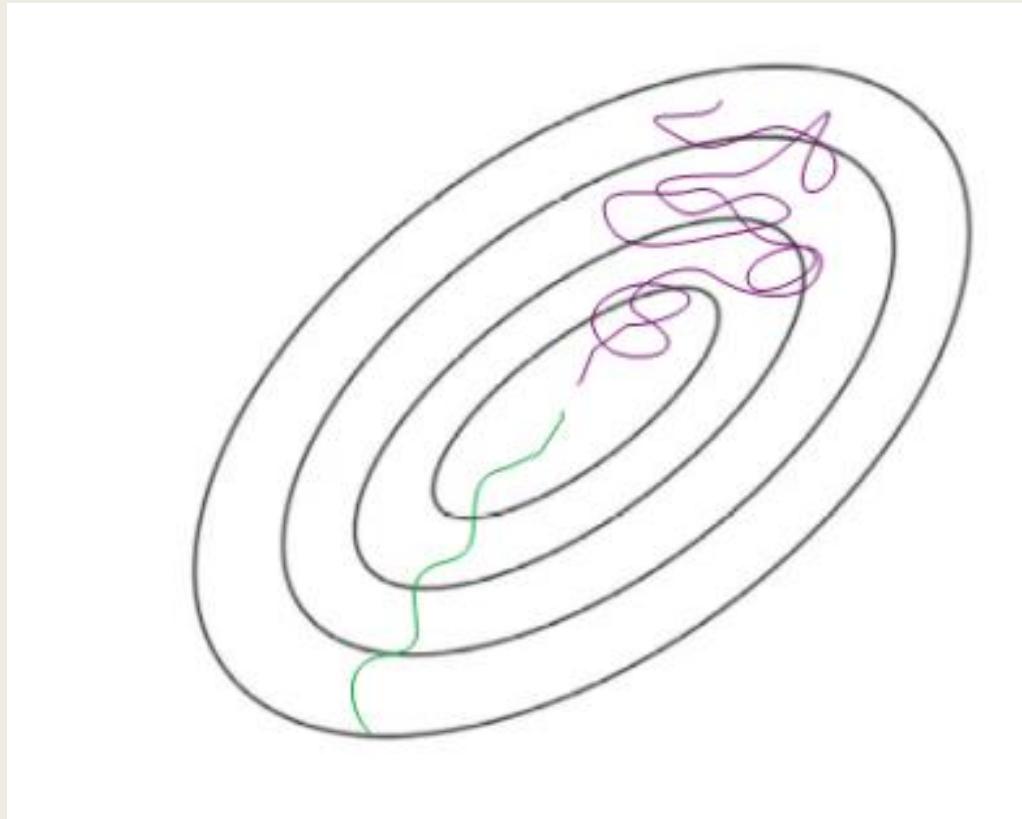
```
device: cuda
epoch 1 ==> loss: 728.340, accuracy: 0.8353
epoch 2 ==> loss: 523.077, accuracy: 0.8461
epoch 3 ==> loss: 385.420, accuracy: 0.8497
epoch 4 ==> loss: 277.219, accuracy: 0.8495
The time used to execute this is given below
5200.354317188263
```

Batch size : 100

```
print(end - start)
```

```
device: cuda
epoch 1 ==> loss: 387.340, accuracy: 0.8300
epoch 2 ==> loss: 282.857, accuracy: 0.8389
epoch 3 ==> loss: 222.719, accuracy: 0.8325
epoch 4 ==> loss: 169.759, accuracy: 0.8329
The time used to execute this is given below
3008.864597558975
```

# Batch size and Loss



Small batch size : It takes long time to converge.

Large batch size : It takes few steps to converge.

# Fake News Classification

Aman Sahu (A20492367)

# Introduction

- ▶ It is important to detect fake news in social media as it can have serious negative impact on individuals and society.
- ▶ Given the title of a fake news article A and the title of a coming news article B, this project can be used to classify the title of the coming news article into three categories:
  - ▶ **agreed:** B talks about the same fake news as A implying B is also a fake news.
  - ▶ **disagreed:** B refutes the fake news in A implying B is not a fake news.
  - ▶ **unrelated:** B is unrelated to A implying the authenticity of B cannot be determined.

# Data Pre-processing

- ▶ First, the datasets have been read into a two separate pandas dataframes: dataset (for training dataset) and test\_dataset (for testing dataset).
- ▶ Then the text is converted into array of numbers. These array objects are converted to lists.
- ▶ The labels given in the training dataset are one hot encoded and then fitted into an array.
- ▶ Then the raw text is broken into words (tokens) using tokenization. This helps in interpreting the meaning of the text by analyzing the sequence of the words.
- ▶ Finally, the training dataset is split into training (2/3) and validation (1/3) subsets.

# Models Used

- ▶ **Simple Recurrent Neural Network (RNN)**

Recurrent neural networks (RNN) are a class of neural networks that is powerful for modeling sequence data such as time series or natural language.

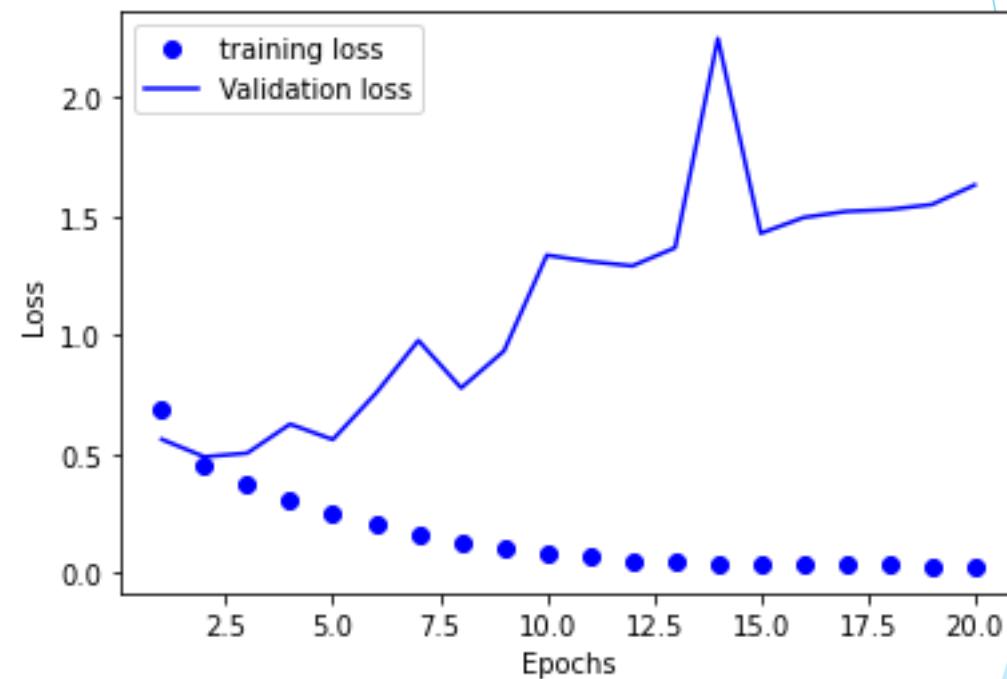
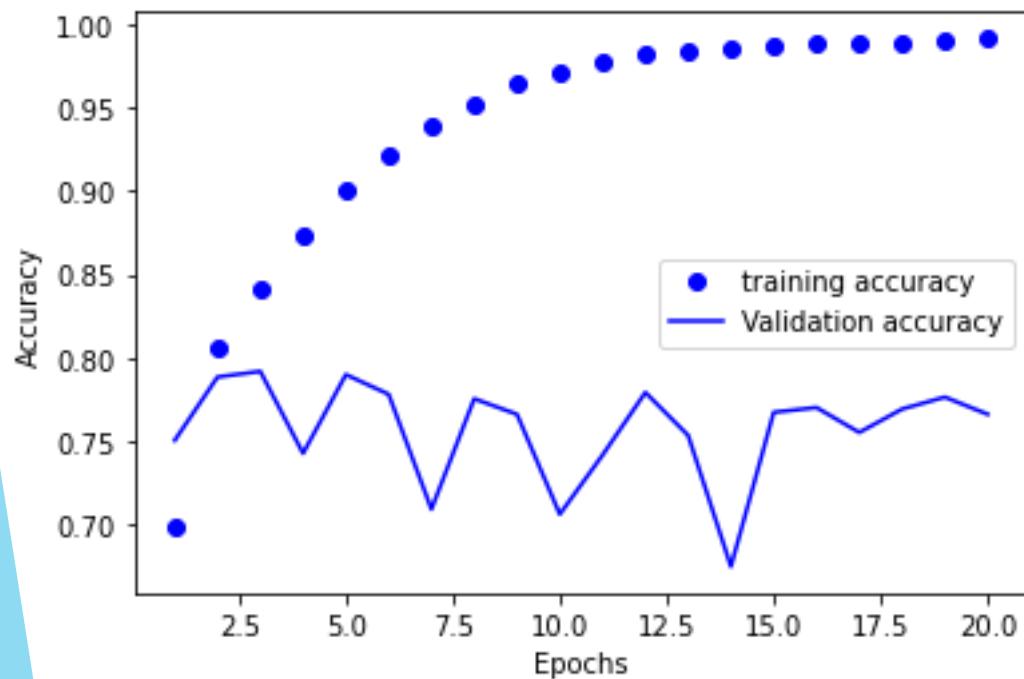
- ▶ **Long short-term memory (LSTM)**

It is an artificial recurrent neural network (RNN) architecture used in the field of deep learning (DL). Unlike standard feedforward neural networks, LSTM has feedback connections.

- ▶ **Bidirectional long-short term memory (biLSTM)**

It is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction.

# Simple RNN Model

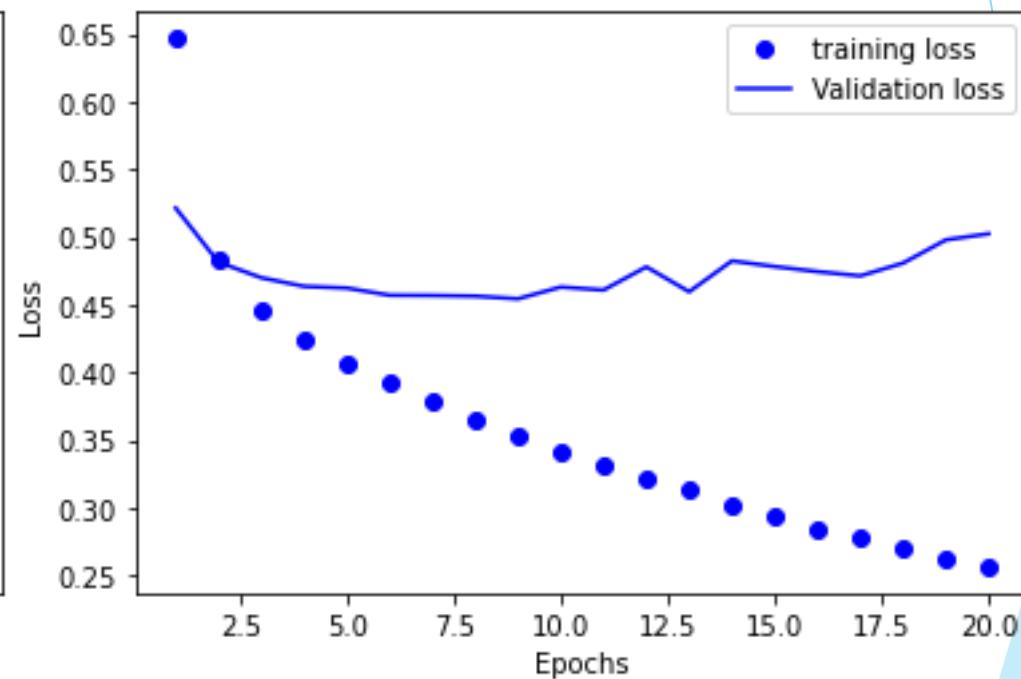
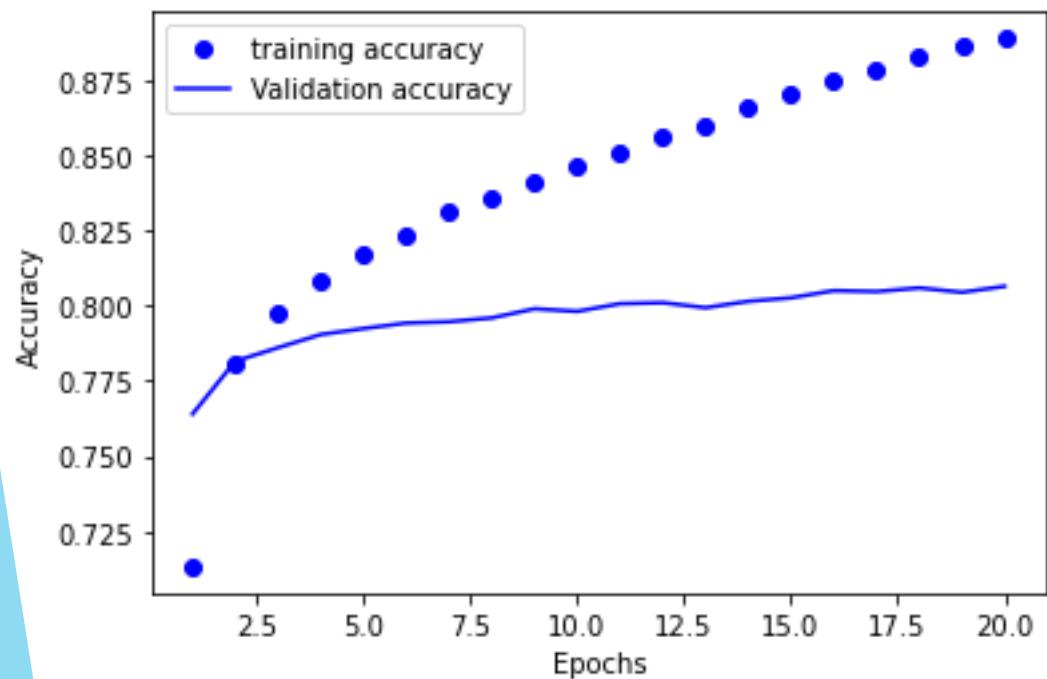


# RNN Performance

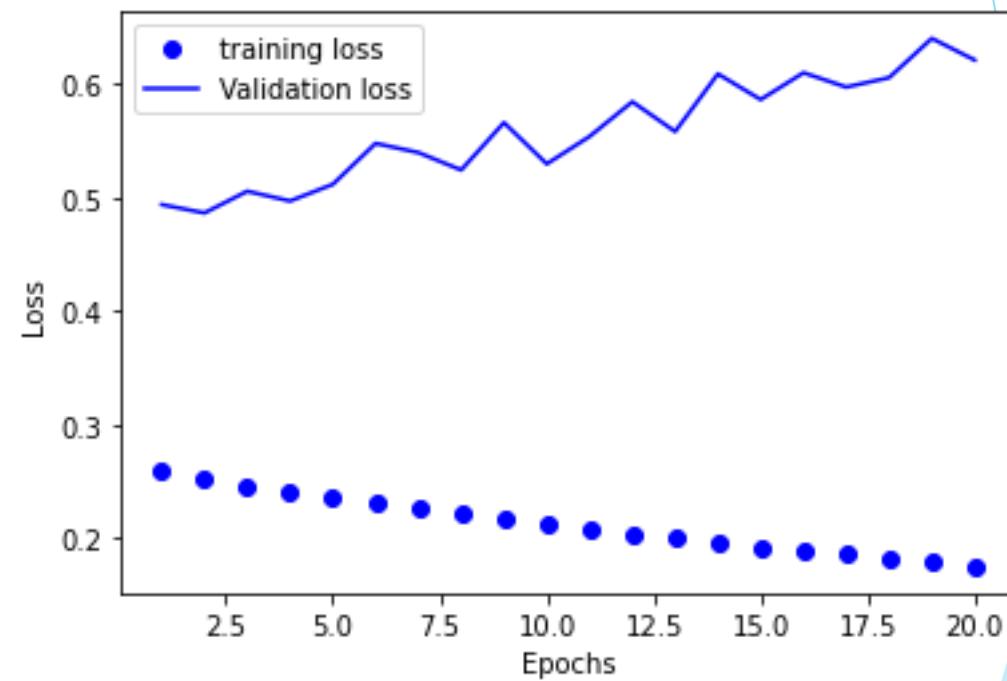
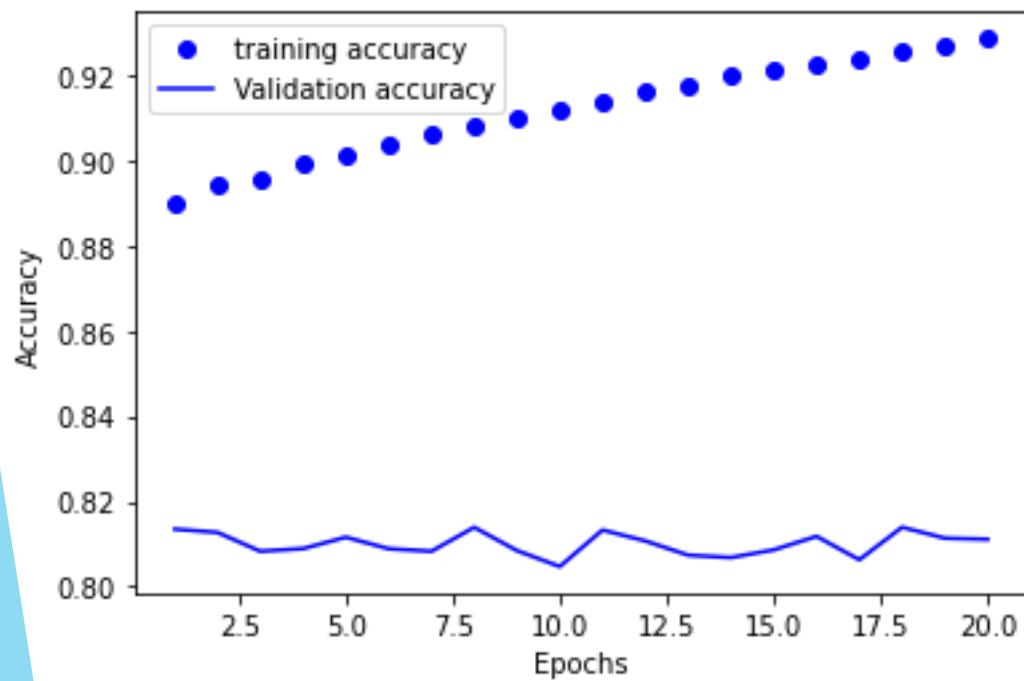
- ▶ On validation with training dataset an accuracy of 77% was achieved for RNN model.

```
2672/2672 [=====] - 8s 3ms/step - loss: 1.6312 - acc: 0.7664
[[16973 167 7689]
 [ 103 894 1296]
 [ 9244 1471 47644]]
(array([0.64487082, 0.35308057, 0.84133571]), array([0.6835958 , 0.38988225, 0.81639507]), array([0.66366889, 0.37056995, 0.82867778]), array([24829, 2293, 58359], dtype=int64))
precision    recall    f1-score   support
      0       0.64      0.68      0.66     24829
      1       0.35      0.39      0.37     2293
      2       0.84      0.82      0.83     58359
accuracy                           0.77     85481
macro avg       0.61      0.63      0.62     85481
weighted avg    0.77      0.77      0.77     85481
```

# LSTM Model



# Bidirectional LSTM Model



# Bidirectional LSTM Performance

- ▶ On validation with training dataset an accuracy of 81% was achieved for Bidirectional LSTM model.

```
2672/2672 [=====] - 499s 187ms/step - loss: 0.6210 - acc: 0.8111
[[18364 46 6419]
 [ 53 969 1271]
 [ 7561 797 50001]]
(array([0.70690584, 0.53476821, 0.86670365]), array([0.73961899, 0.42259049, 0.85678302]), array([0.72289251, 0.47210719, 0.861
71478]), array([24829, 2293, 58359], dtype=int64))
      precision    recall   f1-score   support
          0       0.71      0.74      0.72     24829
          1       0.53      0.42      0.47     2293
          2       0.87      0.86      0.86     58359
accuracy                           0.81    85481
macro avg       0.70      0.67      0.69    85481
weighted avg    0.81      0.81      0.81    85481
```

# Predictions on Test Dataset

- ▶ The trained models are saved after validation.
- ▶ The biLSTM model is then used to predict the labels for the test dataset and classify the titles of news articles (represented as numbers) into agreed, disagreed and unrelated.
- ▶ The prediction results are finally saved into a comma separated value file named “submission.csv”.

Thank You

# Fake News Classification

Hajar Mohammed (A20479402)  
Harshitha Reddy Talusani (A20472820)

# Introduction

Internet and social media made the access to the news information much easier and comfortable. In this project we used a simple fake news classification method based on one of the artificial intelligence algorithms – naive Bayes classifier. The goal of the research is to examine how this particular method works for this particular problem given a manually labeled news dataset and to support (or not) the idea of using artificial intelligence for detecting the fake news.

# Dataset

As per the dataset provided by the professor we have 3 CSV files:

train.csv: Training data, test.csv: Test data , sample submission.csv: Expected submission format

With columns as - id: the id of each news pair.

tid1: the id of fake news title 1

tid2: the id of news title 2.

title1\_en: the fake news title 1 in English.

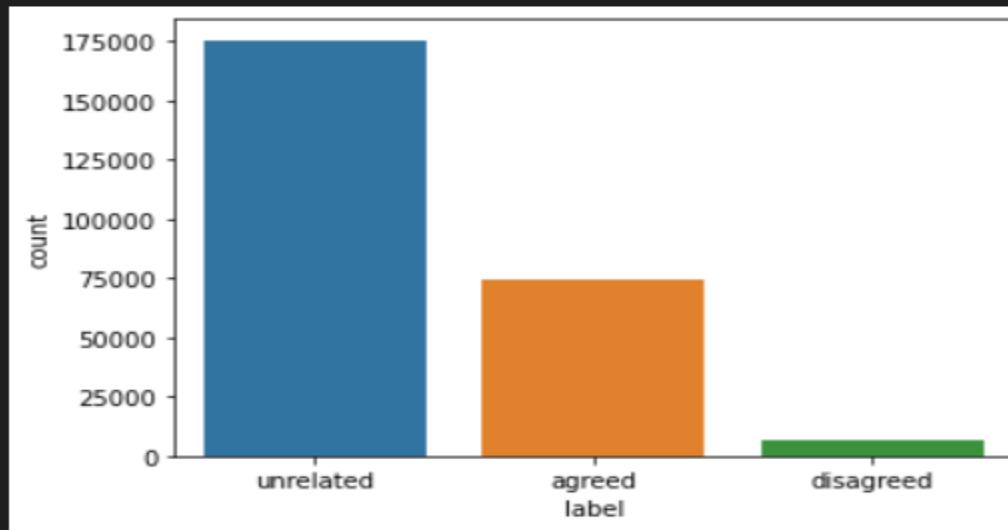
title2\_en: the news title 2 in English.

label: indicates the relation between the news pair:  
agreed/disagreed/unrelated.

| 1  | id     | tid1 | tid2  | title1_en               | title2_en | label     |
|----|--------|------|-------|-------------------------|-----------|-----------|
| 2  | 195611 | 0    | 1     | There are Police disp   | unrelated | unrelated |
| 3  | 191474 | 2    | 3     | "If you do Shenzhen's   | unrelated | unrelated |
| 4  | 25300  | 2    | 4     | "If you do The GDP o    | unrelated | unrelated |
| 5  | 123757 | 2    | 8     | "If you do Shenzhen's   | unrelated | unrelated |
| 6  | 141761 | 2    | 11    | "If you do Shenzhen's   | unrelated | unrelated |
| 7  | 132794 | 6    | 15    | "How to d It's very pr  | agreed    | agreed    |
| 8  | 126388 | 6    | 17    | "How to d Differentia   | agreed    | agreed    |
| 9  | 6314   | 6    | 18    | "How to d stir-fried g  | agreed    | agreed    |
| 10 | 162344 | 7    | 14    | It took 30 A single pie | unrelated | unrelated |
| 11 | 28221  | 7    | 16    | It took 30 Use a garli  | agreed    | agreed    |
| 12 | 255346 | 7    | 18    | It took 30 stir-fried g | agreed    | agreed    |
| 13 | 12412  | 7    | 39141 | It took 30 Quick iden   | unrelated | unrelated |
| 14 | 160593 | 9    | 10    | "if you eat Durian can  | unrelated | unrelated |
| 15 | 36755  | 10   | 88182 | Durian can The durian   | unrelated | unrelated |

# Data preprocessing

After reading the data from the files, we are cleaning the data by removing stop words, special characters, numbers and punctuations. After which we visualized the label features.



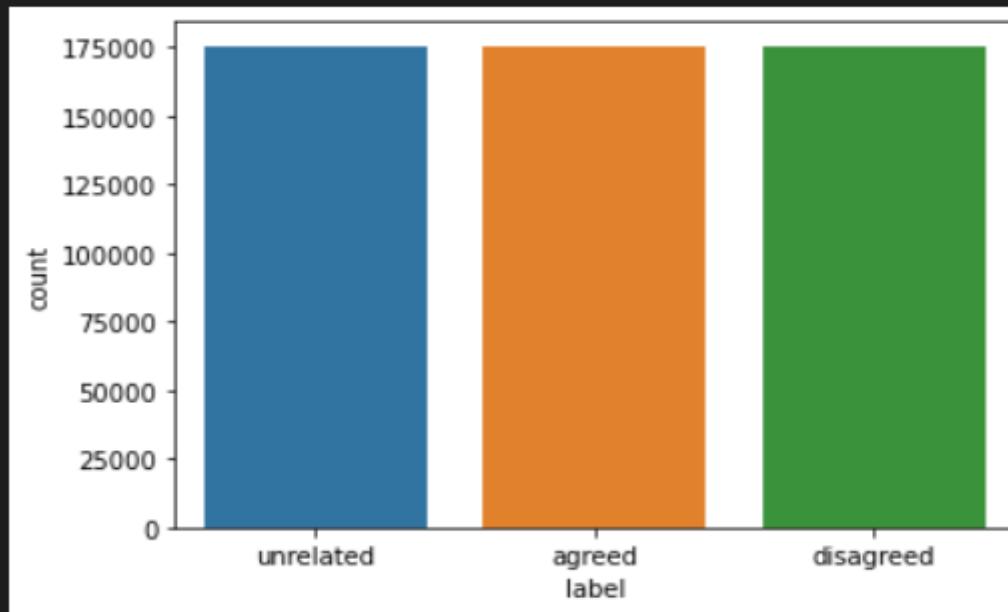
# TF-IDF VECTORIZER (converting data into vectors)

To convert the data into vectors, we are basically using the TF-IDF vectorizer

```
vectorizer = TfidfVectorizer(max_features=4000)
X_train = vectorizer.fit_transform(df_train.Complete_Title)
X_test = vectorizer.fit_transform(df_test.Complete_Title)
y_train = df_train.label
```

# Splitting train test data

After splitting the data we are balancing the data.

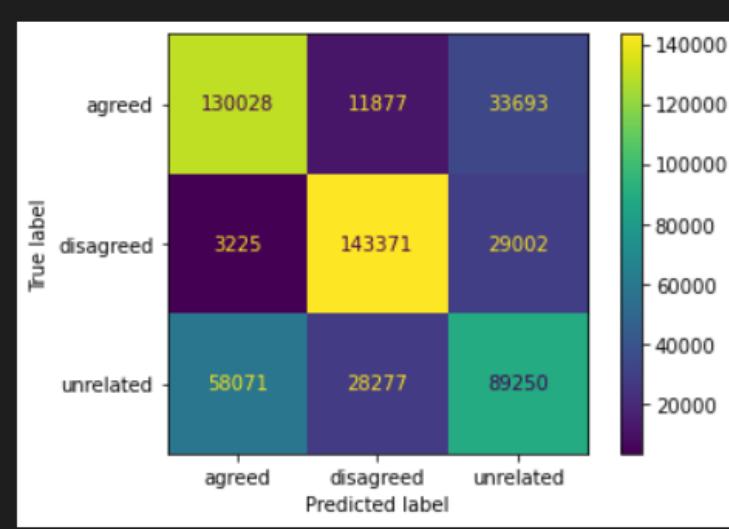


# Training model

For the model, we have used Multinomial Naive Bayes Classifier, accuracy and confusion matrix is given below.

```
model.score(X_train, y_train)
```

```
0.6884076128429709
```



# Results

```
results.to_csv("submission.csv")  
  
results  
  
   id    label  
0  256442 unrelated  
1  256443 unrelated  
2  256444 unrelated  
3  256445 unrelated  
4  256446 unrelated  
... ... ...  
64105 320547 unrelated  
64106 320548 unrelated  
64107 320549 agreed  
64108 320550 agreed  
64109 320551 unrelated  
  
64110 rows × 2 columns
```

|    |    | id     | label     |
|----|----|--------|-----------|
| 1  |    | 256442 | unrelated |
| 2  | 0  | 256443 | unrelated |
| 3  | 1  | 256444 | unrelated |
| 4  | 2  | 256445 | unrelated |
| 5  | 3  | 256446 | unrelated |
| 6  | 4  | 256447 | disagreed |
| 7  | 5  | 256448 | agreed    |
| 8  | 6  | 256449 | unrelated |
| 9  | 7  | 256450 | agreed    |
| 10 | 8  | 256451 | unrelated |
| 11 | 9  | 256452 | agreed    |
| 12 | 10 | 256453 | agreed    |
| 13 | 11 | 256454 | unrelated |
| 14 | 12 | 256455 | unrelated |
| 15 | 13 | 256456 | agreed    |
| 16 | 14 | 256457 | agreed    |
| 17 | 15 | 256458 | unrelated |
| 18 | 16 | 256459 | unrelated |
| 19 | 17 | 256460 | unrelated |
| 20 | 18 |        |           |

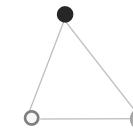
Thank you!

2022.04.26

# Fake news classifier based on neural network .

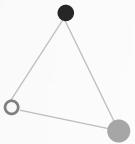
How do we do that?

Report person: Hong Hanbin A20449716  
Zhang Yuanbo A20478825



# 01

Team members



Let's first take a look at two of us.

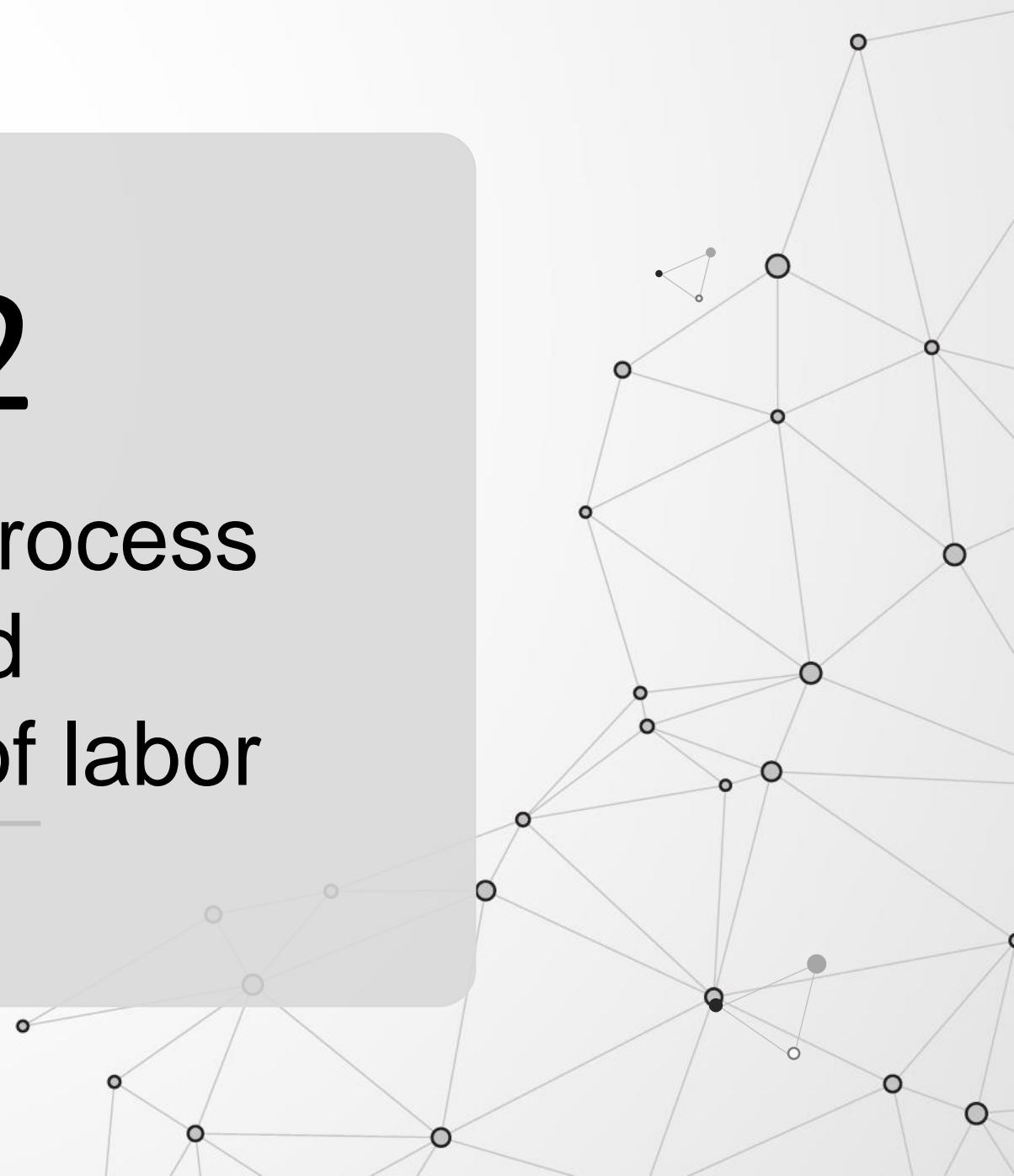
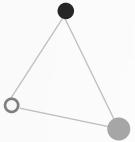
T E A M

Hong Hanbin A20449716, CS Ph.D. , Major: AI security



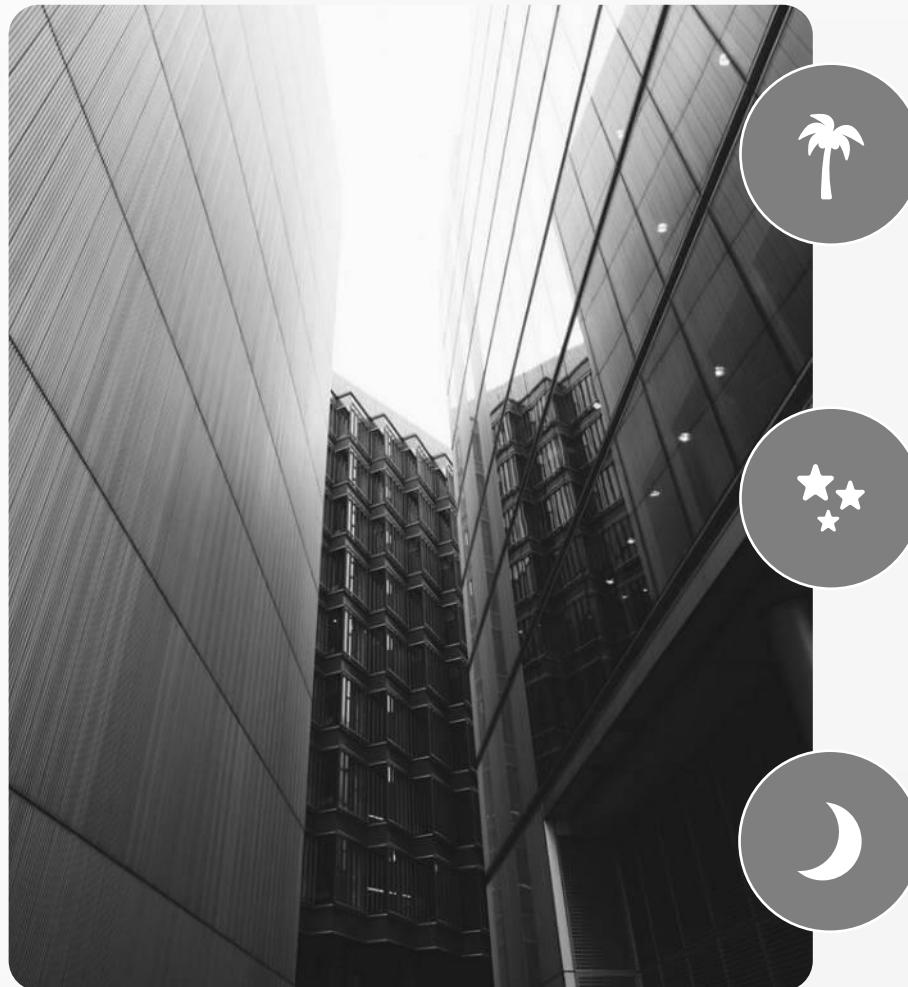
# 02

## Project process and division of labor



Which

## Processing



We held a meeting to determine the division of labor

The data processing

Train and test the model



# Let's see what we've done.



## Group Contribution

**Hanbin Hong's contribution is mainly on model designing and training, and report writing.**

**Yuanbo Zhang's contribution is mainly on data pre-process, slides preparing, and presentation.**

# The technology used



## Separate the words

The first step is to separate the words in a sentence. We use Jieba package to do this step and remove all punctuation from the results. (<https://github.com/fxsjy/jieba>)



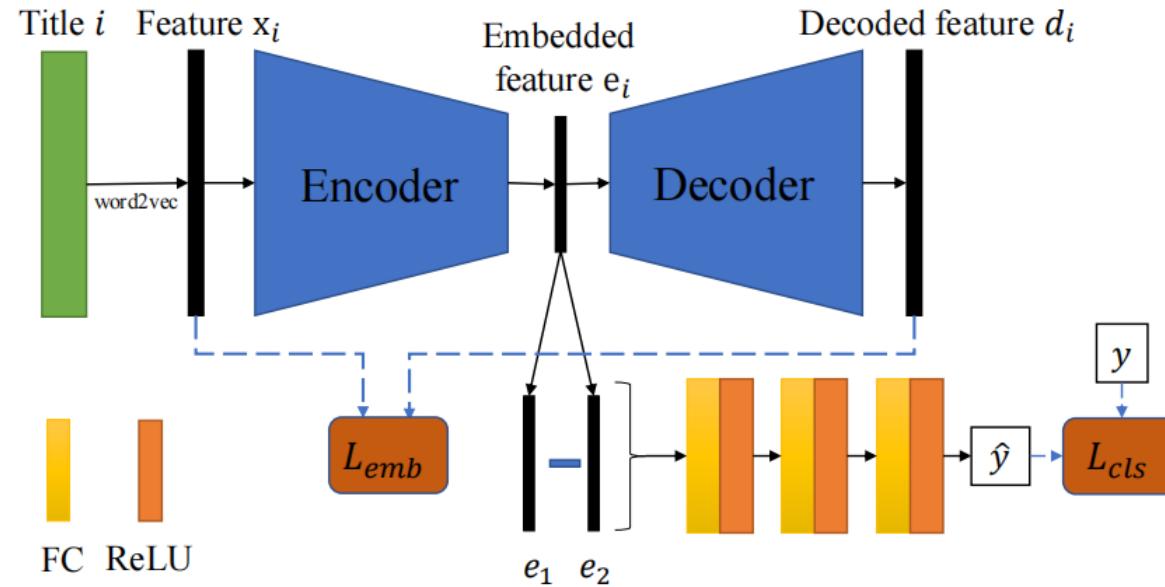
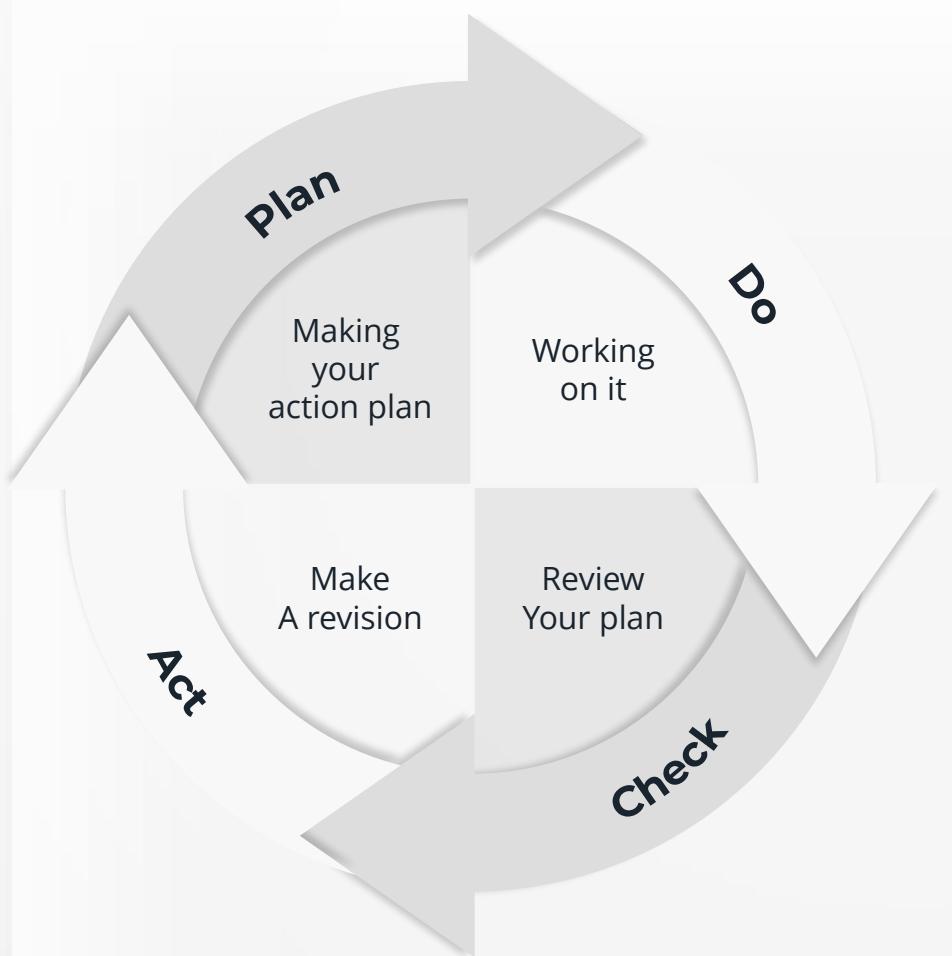
## Transform the words to a value vector

We use word2vec to do this step, in which the words are transform to vectors with k elements, and the word vectors in a sentence will be sum up to represent the sentence.



## Encoder-Decoder network and MLP

To classify the fake news, we use an Encoder-Decoder network to learn an embedding of the features and a MLP to classify the embedded features.



## The Framework

To classify the fake news, we first learn an embedding network to reduce the dimension of the input and keep the useful information, then two titles will be embedded and the difference between the embedded features will be input to a Multi-Layer Perception (MLP) for the classification.

## Fake News Classification

We use a MLP to classify the embedded features. Given the features  $x_1$  and  $x_2$ , they will be first embedded to  $e_1$  and  $e_2$ . The MLP take  $e_1 - e_2$  as the input and classify whether they are agreed, disagreed, or unrelated.

$$y = \text{MLP}(e_1 - e_2) \quad (4)$$

To train the MLP, we use the cross entropy loss to measure the distance between the prediction  $\hat{y}$  and the label  $y$ .

$$\mathcal{L}_{cls} = \sum_j y \log \hat{y} \quad (5)$$

## Feature Embedding

Denote the feature for title A and B as  $x_1$  and  $x_2$ , respectively. Denote the class as  $y \in \{0, 1, 2\}$ , where 0 denotes Disagreed, 1 denotes Unrelated, and 2 denotes Agreed.

Since the dimension of the features are high, e.g., 2500 dimensions vector, we will embed the features to a latent space with less dimensions. We leverage an Encoder-Decoder structure to reduce the dimension. We use a fully-connected layer as the Encoder  $E$  to map the input features from  $k$  dimension to embedding dimension  $l$ . Also, the Decoder  $D$  is also a fully-connected layer that map the embedded features from  $l$  dimension back to  $k$  dimension. the embedded features is:

$$e_1 = E(x_1) \quad (1)$$

$$e_2 = E(x_2) \quad (2)$$

We use a reconstruction loss to lead this embedding process:

$$\mathcal{L}_{emb} = \sqrt{(x_i^2 - D(E(x_i))^2)} \quad (3)$$

This is because that if the decoded features can be similar to the original features, the embedding learns the important information that is equally contained by the decoded features, while reducing the dimension of the feature.

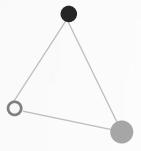
## Problem Setting

Given the title of a fake news article A and the title of a coming news article B, we aim to classify B into one of the three categories:

- (1) Agreed: B talks about the same fake news as A.
- (2) Disagreed: B refutes the fake news in A.
- (3) Unrelated: B is unrelated to A.

## Pre-processing

The neural networks take values as input, so we need to transform the English title to value features..



# 03

## Results and Conclusions

---



# Experiment

We use the whole training set as the training data and ignore the validation.

The feature dimension is set to 2,500 and the embedded feature dimension is set to 500.

We use a 4-layer MLP and the output dimension for each layers are 128, 128, 64, and 3. We use Adam algorithm to train the networks.

The learning rate is set to 0.001. We train the networks with 500 epoch and the batch size is set to 512.

The codes are implemented with Python and Pytorch, and are running on a NVIDIA RTX 3080 GPU.

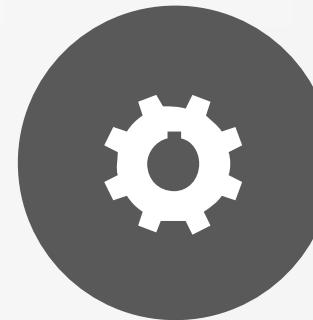
Codes are available in the attached files.

Some training log are shown in Table 1, a full training log is also attached to the files.

It shows that the classification accuracy on the training set is as high as 94.29%.



# Result



Some training log are shown in Table 1, a full training log is also attached to the files.

It shows that the classification accuracy on the training set is as high as 94.29%.

Epoch 1, embedding loss 1: 0.0060, embedding loss 2: 0.0059, classify loss: 0.6227, accuracy: 0.6962  
Epoch 2, embedding loss 1: 0.0030, embedding loss 2: 0.0030, classify loss: 0.5615, accuracy: 0.7278

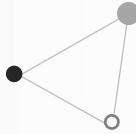
...

Epoch 498, embedding loss 1: 0.0101, embedding loss 2: 0.0097, classify loss: 0.1379, accuracy: 0.9414  
Epoch 499, embedding loss 1: 0.0065, embedding loss 2: 0.0064, classify loss: 0.1360, accuracy: 0.9426  
Epoch 500, embedding loss 1: 0.0087, embedding loss 2: 0.0085, classify loss: 0.1345, accuracy: 0.9429

Table 1: Training log

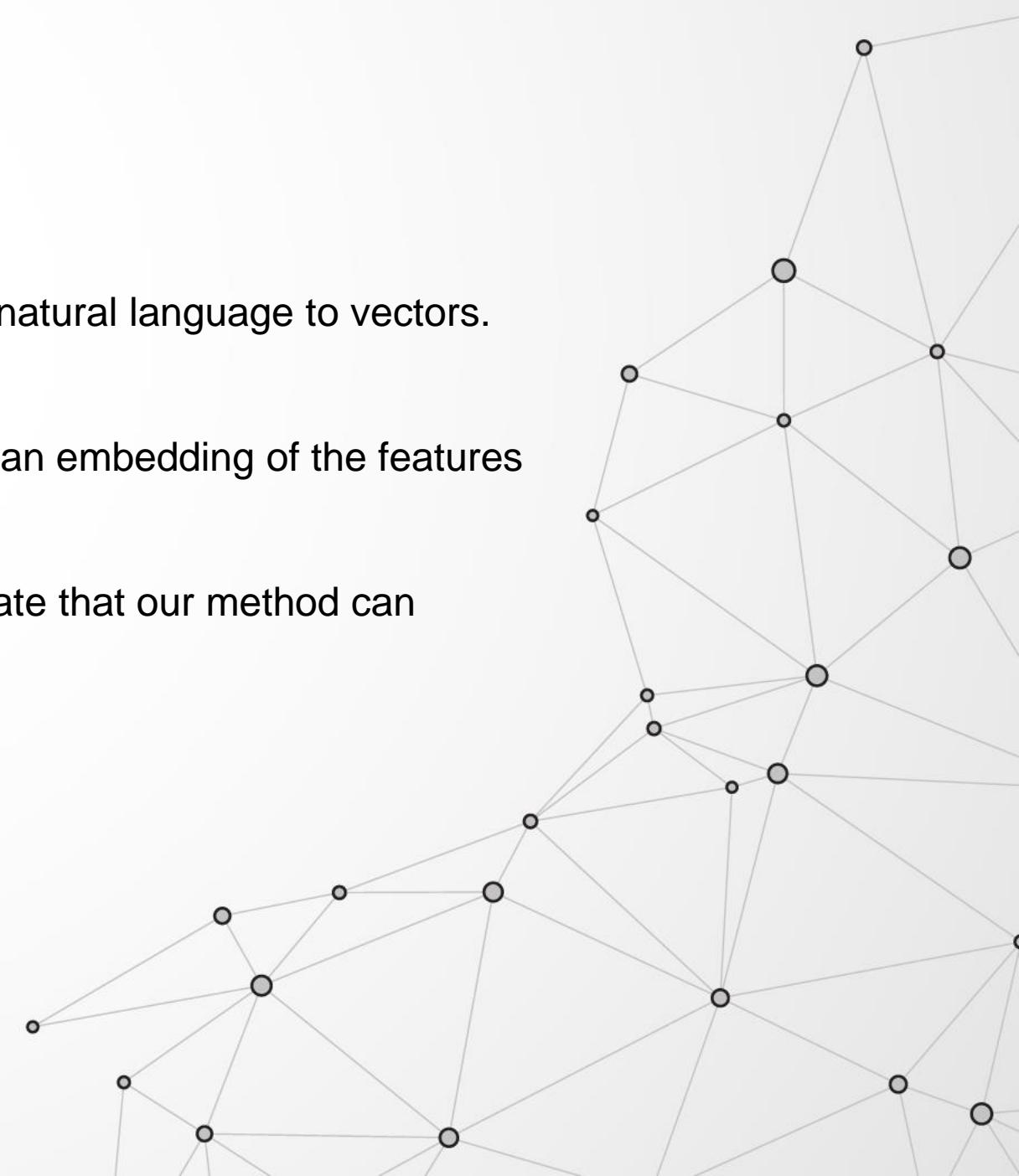
# Conclusion

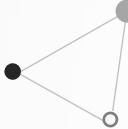
To process the fake news quantitatively,  
we use Jieba and Word2Vec to transform the natural language to vectors.



To classify the fake news,  
we use an Encoder-Decoder network to learn an embedding of the features  
and a MLP to classify the embedded features.

The experimental results on training set illustrate that our method can  
achieve a high classification accuracy.





# Thank you

---

$O(n \_\_ n)$





# Fake News Classification

UDAY RAMESH

ONLINE SOCIAL NETWORK ANALYSIS

# Problem Statement

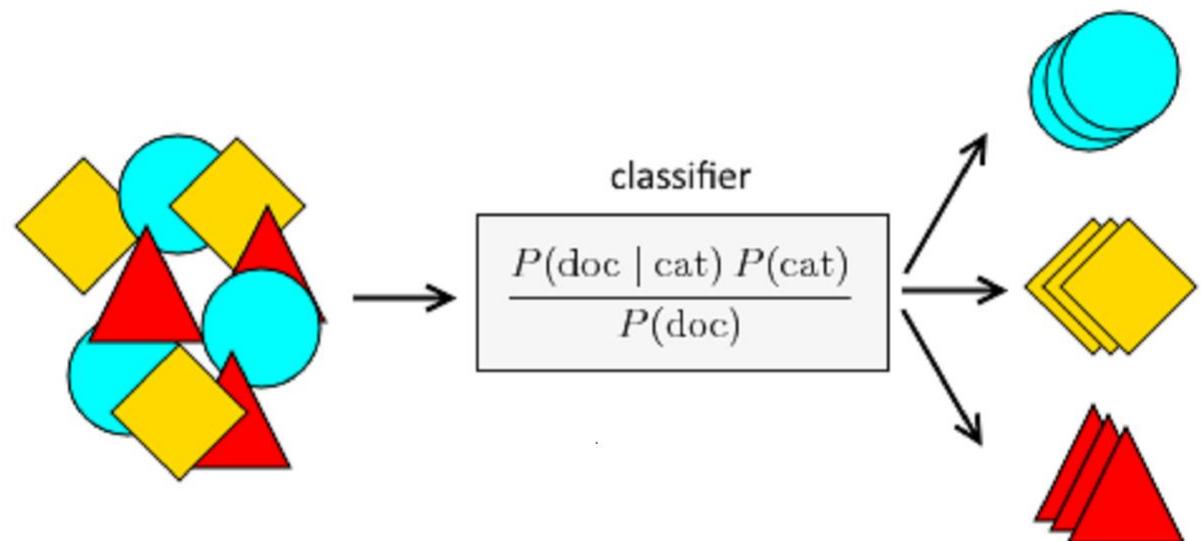
- ▶ Given the titles of two articles, where A is a fake news article and B is the next article, classify the relationship between the two as :
  - ▶ A – Agreed: discuss the same fake news.
  - ▶ B – Disagreed: B refutes the news presented in A.
  - ▶ C – Unrelated articles.
- ▶ There are a total of 256,422 training pairs provided and 64,110 testing pairs.

# Natural Language Processing

- NLP is required to parse the titles of the articles.
- Uses the Python - Natural Language Toolkit (NLTK).
- Requires extensive preprocessing.
  - Remove special characters.
  - Remove stop words.
  - Stemming to remove prefixes and suffixes.
- Tokenization helps separating the inputted string into words for ease of analysis.

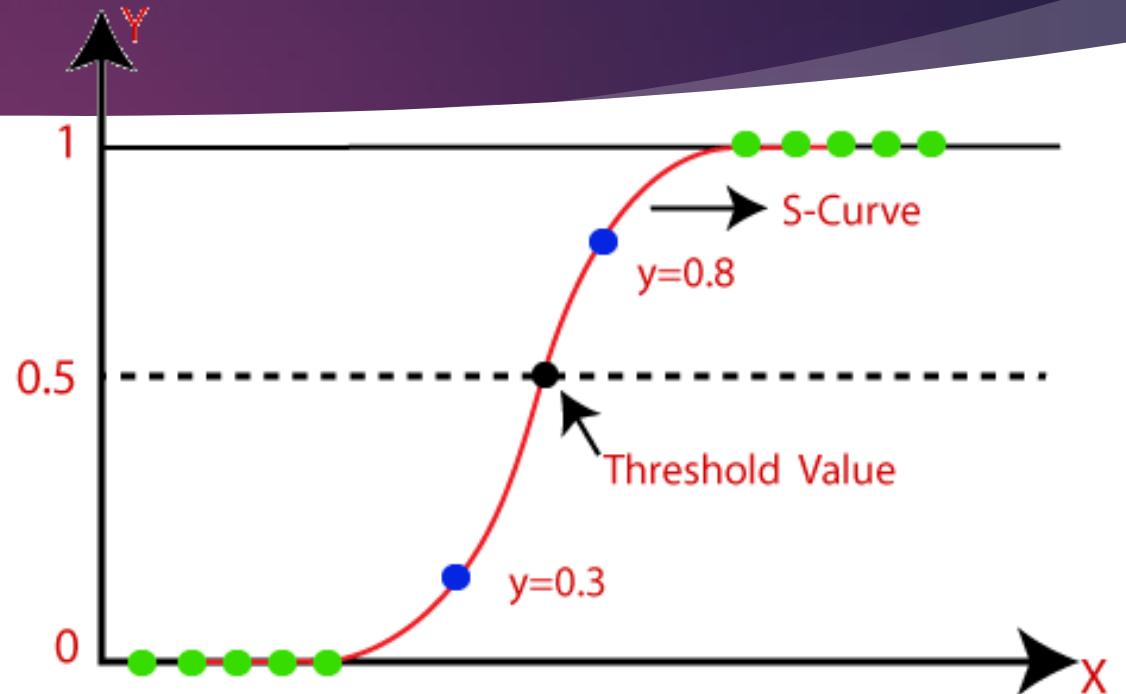
# Classifier: Naïve Bayes

- Relies on probability theory.
- Classifies new entries based on prior knowledge.
- Assumes all features are independent.
- Works nicely with high-dimensional data.

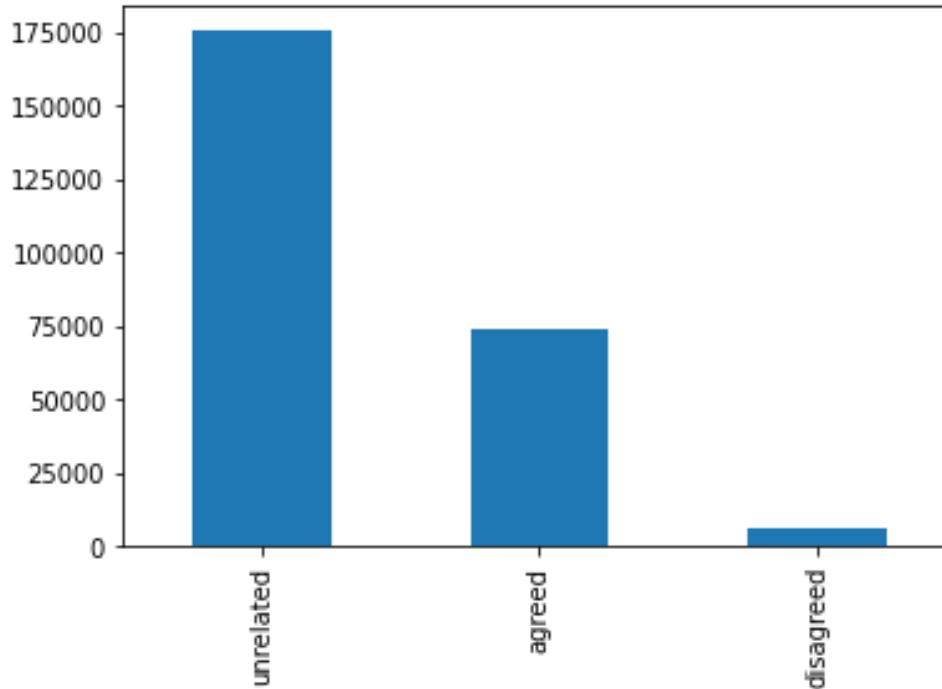


# Classifier: Logistic Regression

- Most often used for categorical labels.
- Very fast to implement and train.
- Assumes linearity between the dependent and independent variables.

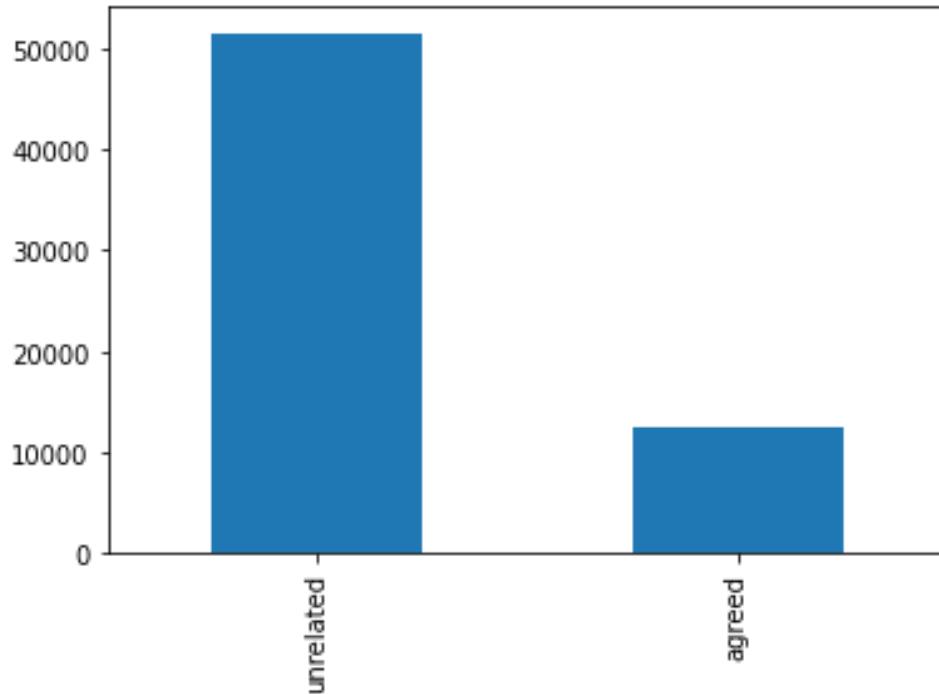


## ► *Training Data*



- Total number of training datasets that are trained under Unrelated, Agreed and Disagreed.

## ► *Submission Data*



- After processing the data, here is the result that obtained using the datasets.

# Classification Results

- Naïve Bayes classifier: 68% accuracy.
- Logistic Regression classifier: 75% accuracy.
- The logistic regression classifier was used to perform predictions on the testing dataset as its training accuracy was higher .

# Conclusion and Limitations

- Ultimately, a natural language processor fake news classifier was trained and applied to testing data. This required data preprocessing and resulted in a model with relatively strong accuracy.
- One key limitation of the analysis is the skew in the training dataset labels. Most of the dataset consisted of unrelated observations, with the potential to create bias in the model, rendering it more likely to create an unrelated classification in new articles.

# References

- Brownlee, J. (2016, April 1). Logistic Regression for Machine Learning. *Machine Learning Mastery*. Retrieved from: <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>.
- Kumari, K. (2021, July 19). Detecting Fake News with Natural Language Processing. *Analytics Vidhya*.
- Jurafsky, D. & Martin, J. (2021, December 29). Speech and Language Processing, 3<sup>rd</sup> edition. Retrieved from: <https://web.stanford.edu/~jurafsky/slp3/>.
- Menczer, F., & Hills, T. (2020, December 1). Information Overload Helps Fake News Spread, and Social Media Knows It. *Scientific American*.
- NLTK Project. (2022, Feb 09). Documentation. <https://www.nltk.org>.
- Shukla, P., & Iriondo, R. (2020, July 22). Natural Language Processing (NLP) with Python. *Towards AI* .



Thank You

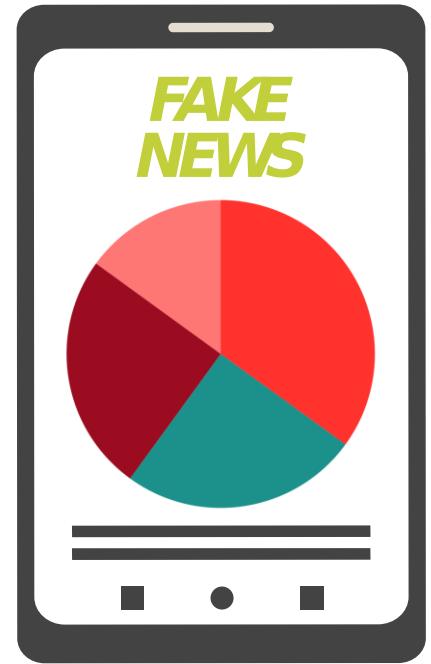
---

- + . CS579 PROJECT II:
- FAKE NEWS  
CLASSIFICATION

Pablo Perez Rodriguez  
A20497421

# Scenario

- Training data is composed of:
  - News article title A
  - News article title B
  - Label: B agrees, disagrees or is unrelated to A
- Test data same structure but unlabeled
- Task: construct a model to predict labels for the test data



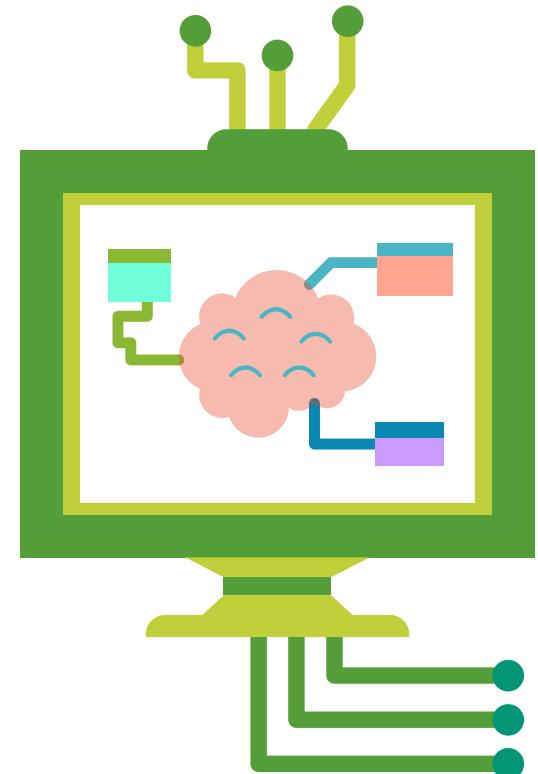
# Data preparation

- Data pre-processing
  - Lowercase
  - Special characters
  - Extra whitespaces
  - Stopwords
  - Lemmatization
  - Numbers to text
- Feature extraction
  - TF-IDF
- Split train and validation data
  - 18% of the labeled data used as validation set



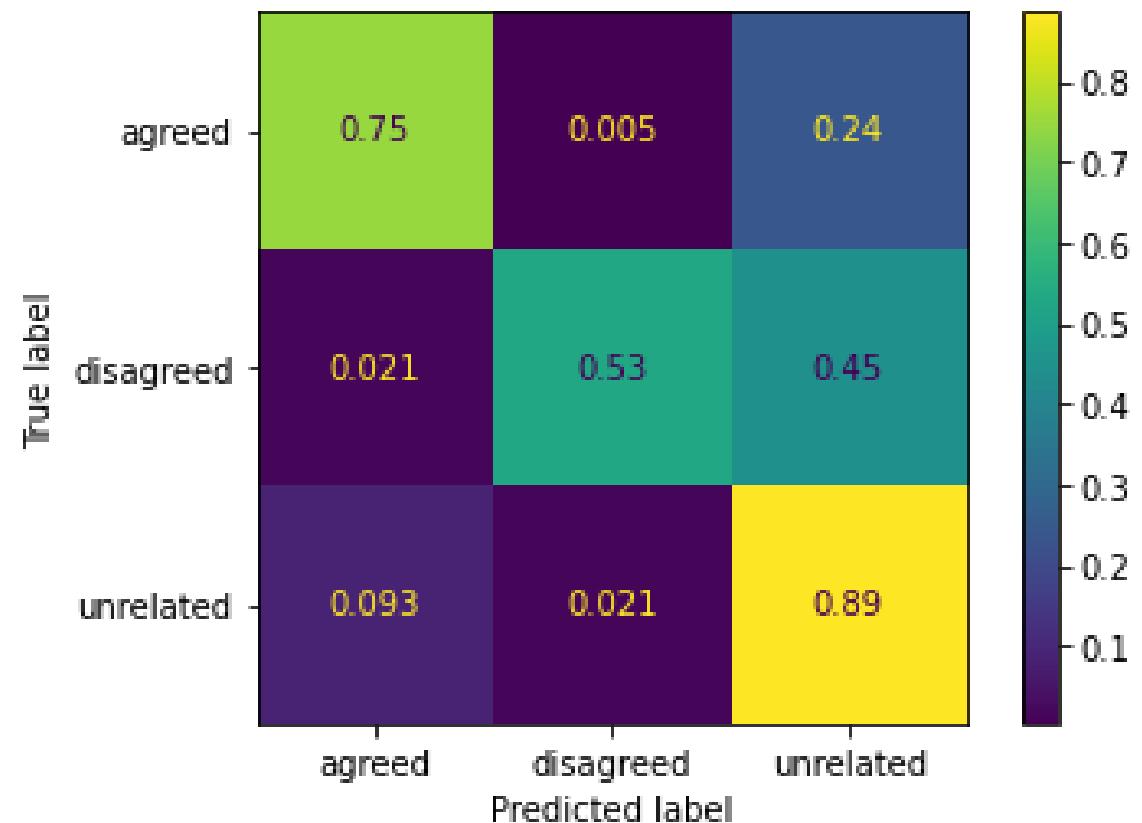
# Model

- Multilayer Perceptron Classifier (MLP)
  - 3 hidden layers (150, 100, 50)
  - *Relu* activation function
  - *Adam* solver for weight optimization
  - Max iterations set to 20 (5 hours training)



# Results

- Accuracy (test data): **84.7%**
- Differences in the classification of the different classes
  - Data imbalance



---

- + . CS579 PROJECT II:
- FAKE NEWS  
CLASSIFICATION

Pablo Perez Rodriguez  
A20497421

# Data-efficient Graph Anomaly Detection

Xiongxiao Xu

# Model

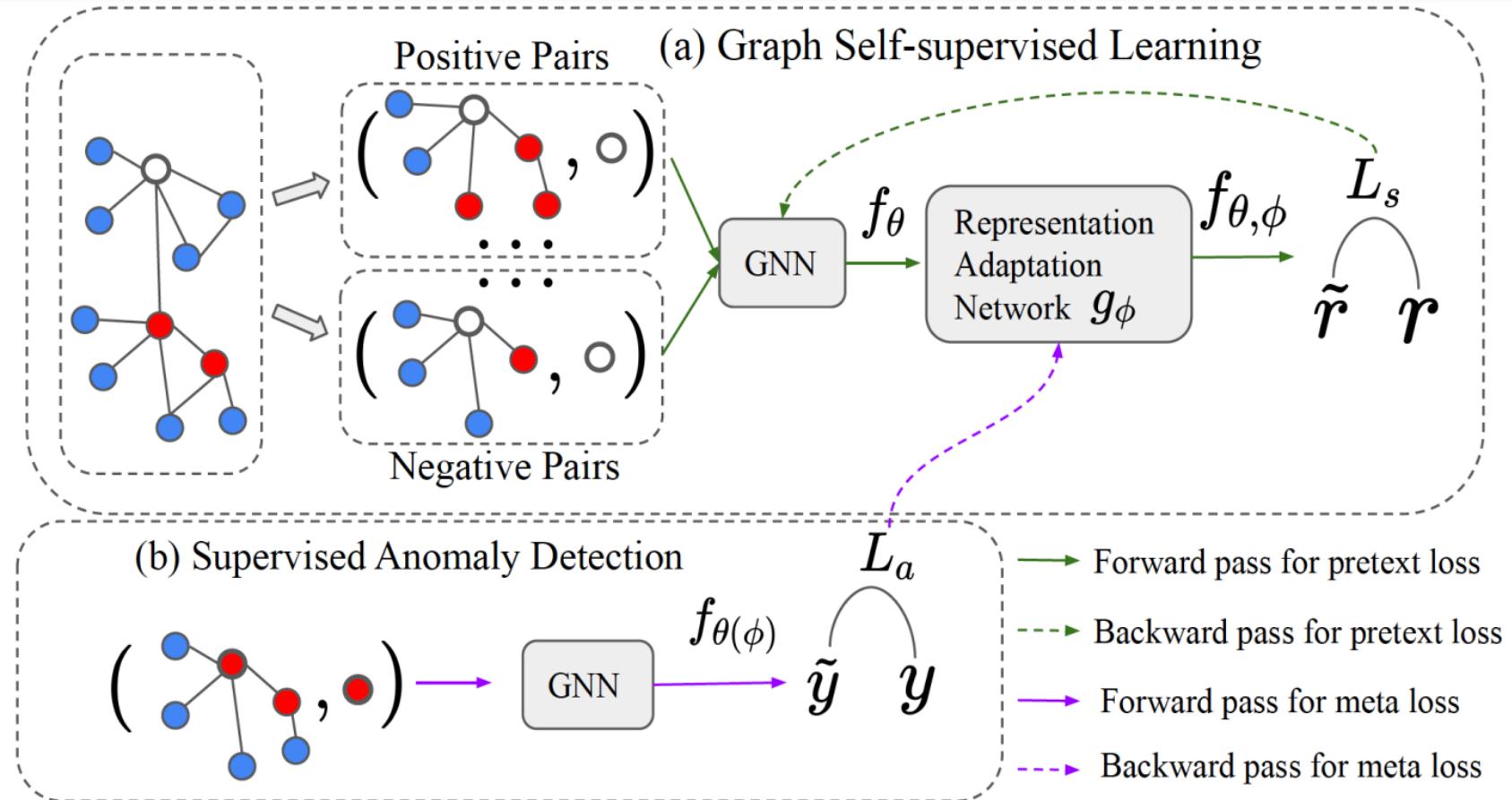


Figure 2. Data-Efficient Graph Anomaly Detection.

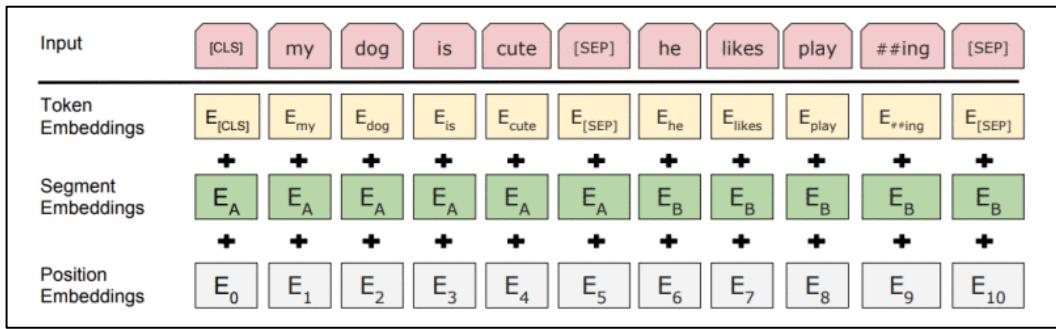
Thank You!

# BERT Fake News Classification

Haroon Gharwi, Truong Pham

# First Model - BERT Huggingface

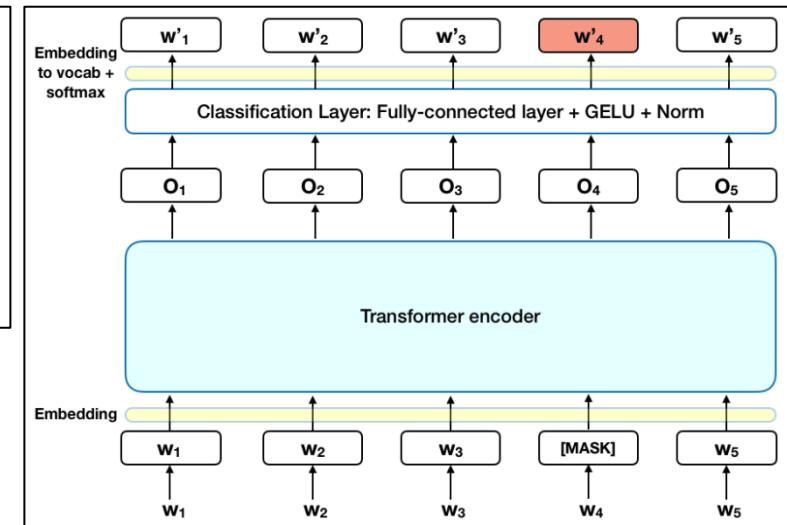
BERT's Architecture



BERT's inputs

total parameters: 109.484.547

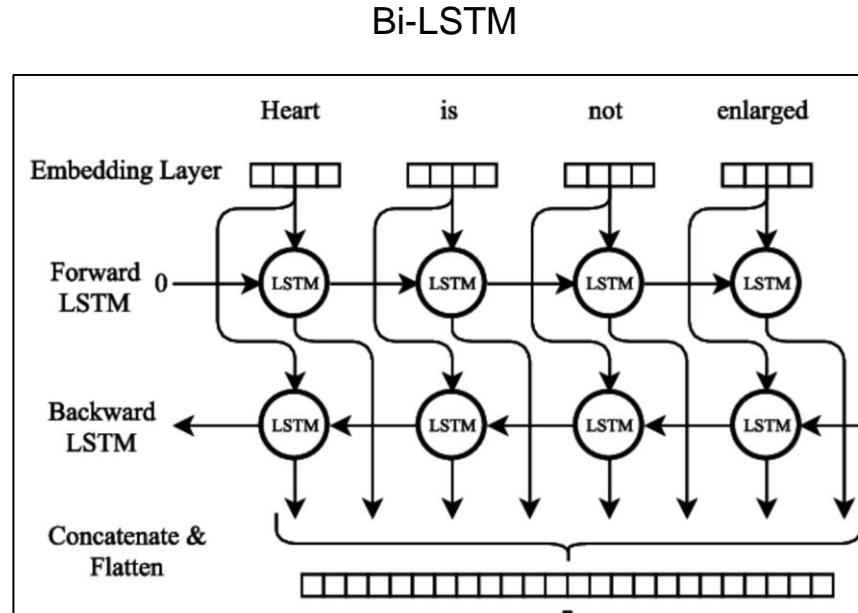
trainable parameters: 2307



# Second Model - BERT + BiLSTM

Total parameters:  
331.767.043

Trainable  
parameters:  
112.802.563



# Results

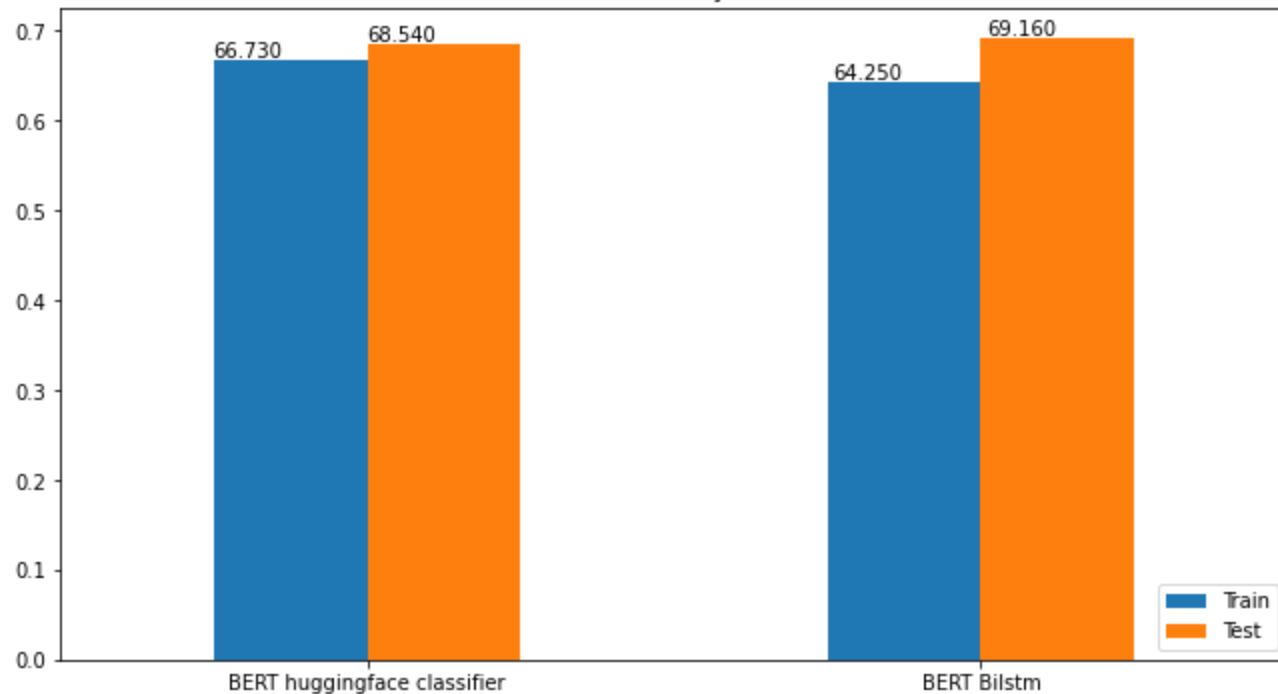
First model - BERT huggingface classifier:

- Train accuracy: 0.6673
- Validation accuracy: 0.6854
- Training time: 3h40m

Second model - BERT bilstm:

- Train accuracy: 0.6425
- Validation accuracy: 0.6916
- Training time: 5h40m

Accuracy

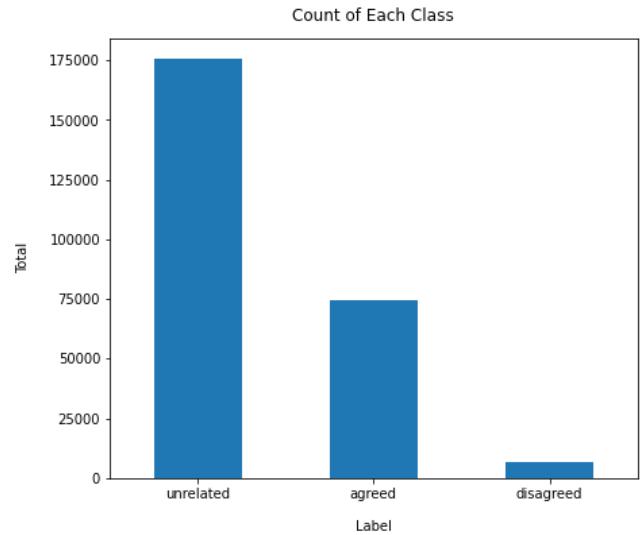


# Conclusion

Class imbalance affected results

Should Increasing the training time

Other methods should also be explored



# Fake News Classifier

---

CS 579 - Spring 2022

Muhammad Umar (A20455931)  
Arvil Dey (A20404230)

# Problem

---



# Solution

---

- Manually fact check news everytime its heard
- Make a computer do it for you
- Use statistical knowledge of past fact checking to predict what may be fake news based on the vocabulary - our proposed solution

# Methods

---

- Naive Bayes
- Logistic Regression
- Multilayer Perceptron Model

# Steps

---

1. Clean the input text
2. Lemmatize the text
3. Obtain a TF-IDF matrix by vectorizing it
4. Train models using this matrix
5. Predict on testing data

# Results

---

| Method                          | Training Accuracy |
|---------------------------------|-------------------|
| Naive Bayes                     | 77.3%             |
| Logistic Regression             | 81.1%             |
| Multilayer Perceptron (Keras)   | 98.3%             |
| Multilayer Perceptron (Sklearn) | 99.5%             |

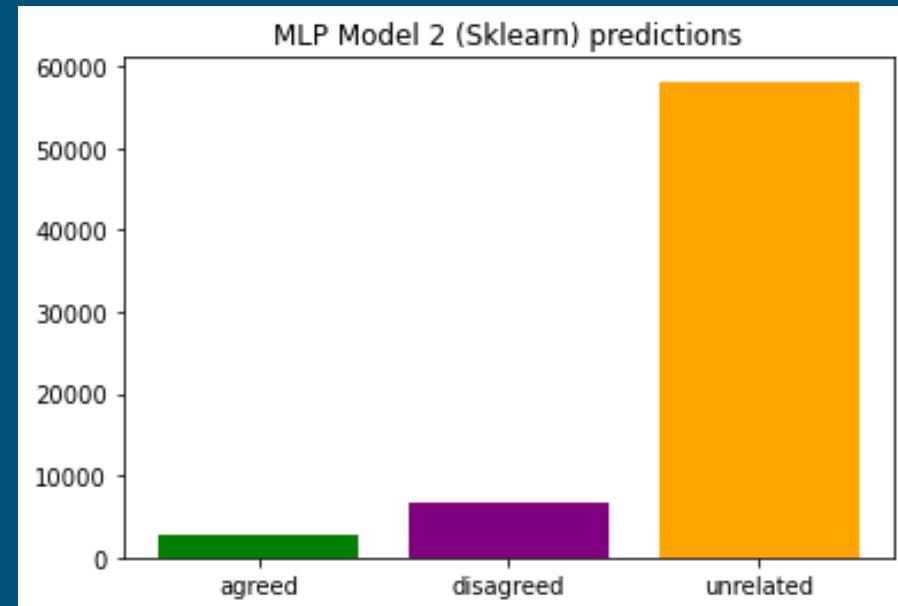
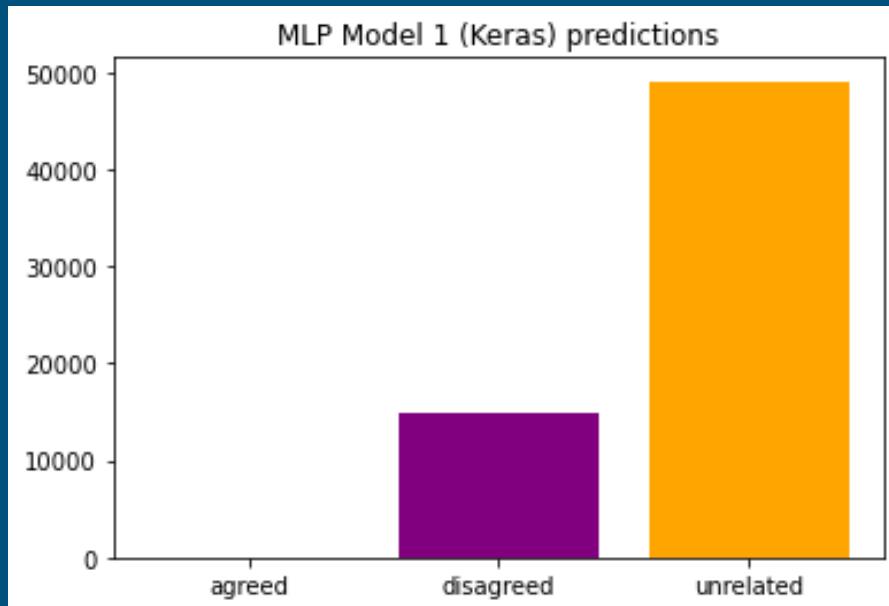
# MLP Model differences

---

- Keras model
  - Dense (16) -> Dropout (10%) -> Dense (32) -> Dropout (10%) -> Dense (3) = output
- Sklearn model
  - Dense (128) = output

# Test Results

---



# Observations

---

- MLP model had the best results
- Data is highly imbalanced in favor of "unrelated"
- Sklearn training is slow vs Keras
- Testing data TS-IDF matrix had to be padded

---

Thank you!