Fake News Detection

Project Advisor: Professor Magdalini Eirinaki
Group: Team G33

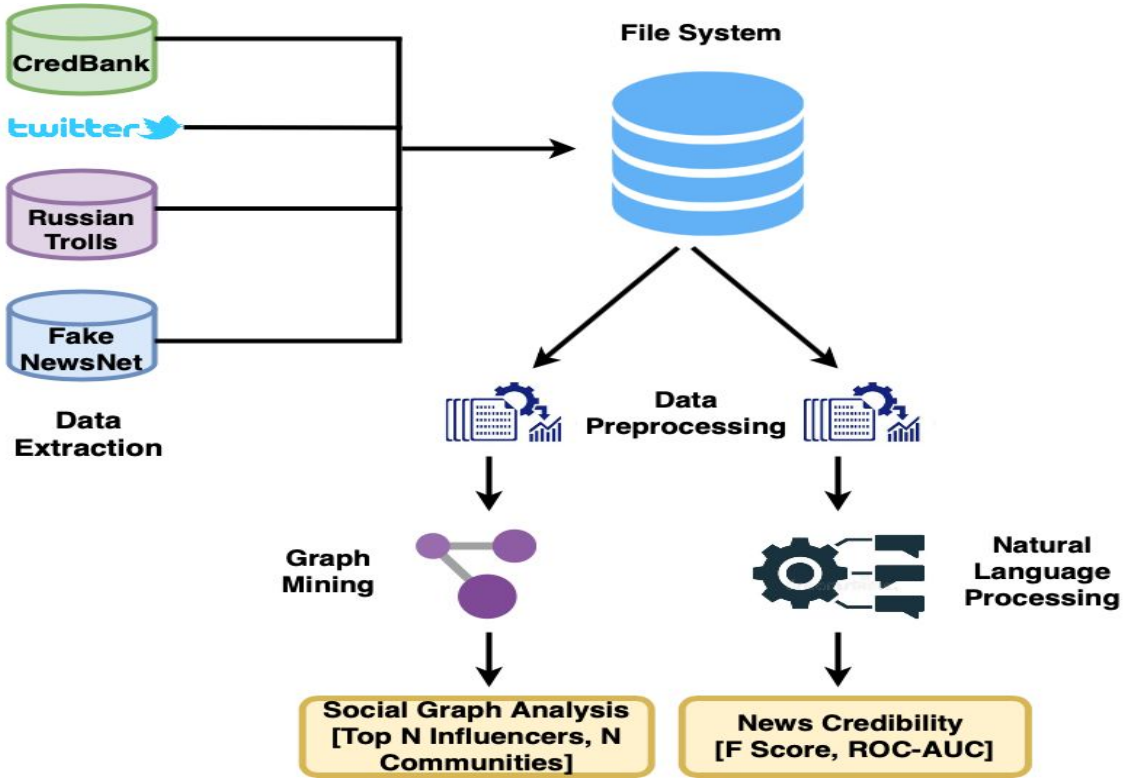
Team Members:

- **Abhinav Balasubramanian**
- **Aayushi Gupta**
- **Vineet Khatwal**
- **Priyanka Raju**

Project Description

- Authenticity and reliability of the news that we receive from any communication media these days is highly questionable
- Aspects of given news article: News Reporting agencies, Crowdsourcing real time content, fact verifications like wikipedia, state laws and sentiment analysis
- Current technologies provides many ways to hamper, evaluate and modify the various data sources and produce the so called "fake news". Thus, combatting this problem is inevitable for today's society
- Our project aims at analyzing twitter social media to detect if a news shared is fake or not.

System Architecture

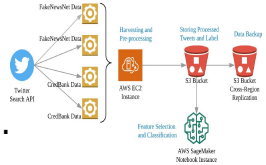


Dataset

- Fake News Net
  1. 500K Fake news propagating user IDs and their follower IDs
  2. 500K Real news propating user IDs and their follower IDs

- CredBank
  1. Credibility Annotation File - Rating file for news related to trending Topics
  2. Searched Tweet File (80 million tweets) - Provides us with the tweet id for for the tweets related to different topics.

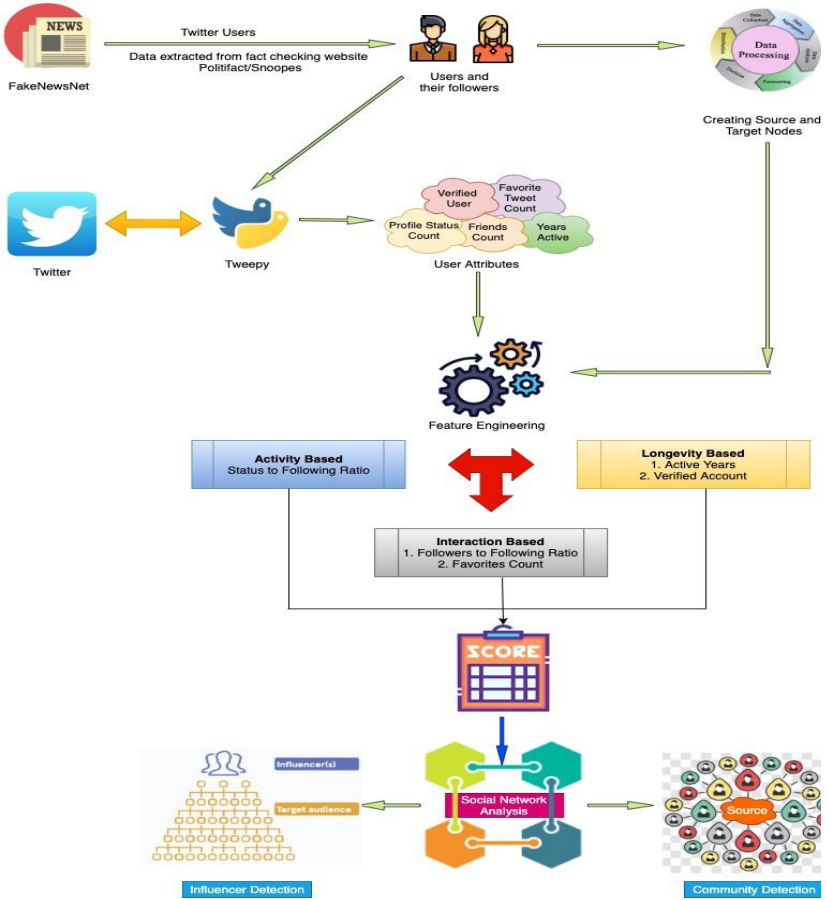
  We used both the files to create a dataframe with the tweet and it's corresponding average user rating.



- Twitter(Tweepy)
  1. Extracted the User Attributes for the 1M user IDs (500K Fake + 500K Real)
  2. Extracted the tweets based on tweet id from CREDBANK dataset. Used AWS EC2 to overcome the limitation of twitter rate limiting for fetching tweets.

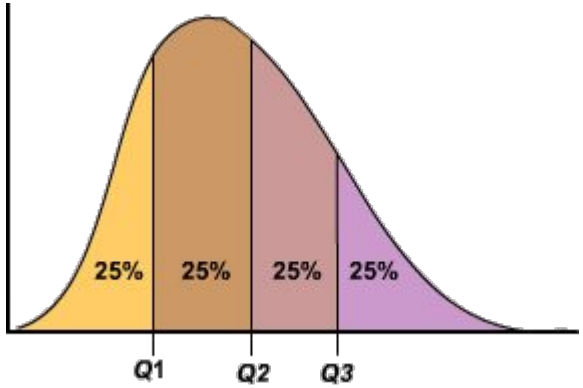
- Russian Trolls
  1. With more than 200K records, this dataset contains two CSV files. tweets.csv includes details on individual tweets, while users.csv includes details on individual accounts

Fake News Propagation Analysis



Edge Weight Computation using User Attributes

Attributes	Category	Quantile Range	Score (percentage)
Active years	Longevity Based	Min to Quantile 1	5
		Quantile 1 to Quantile 2	10
		Quantile 2 to Quantile 3	15
		Quantile 3 to Max	20
Verified status	Longevity Based	False	0
		True	30
Status to Following Ratio	Activity Based	Min to Quantile 1	5
		Quantile 1 to Quantile 2	10
		Quantile 2 to Quantile 3	15
		Quantile 3 to Max	20
Followers to Following Ratio	Interaction Based	Min to Quantile 1	5
		Quantile 1 to Quantile 2	10
		Quantile 2 to Quantile 3	15
		Quantile 3 to Max	20
Favorites Count	Interaction Based	Min to Quantile 1	2.5
		Quantile 1 to Quantile 2	5
		Quantile 2 to Quantile 3	7.5
		Quantile 3 to Max	10



Influencer & Community Detection Results

FAKE NEWS

Node	Degree Centrality	Page Rank	Betweenness Centrality	Closeness Centrality
19672966	412	167.8201207	167.8201207	0.2251199232
21237884	378	165.8195036	165.8195036	0.1836982333
20974554	386	158.559881	158.559881	0.2232423176
22024242	375	158.03263	158.03263	0.2283327927
20984200	358	157.9282189	157.9282189	0.1663397921

REAL NEWS

Node	Degree Centrality	Page Rank	Betweenness Centrality	Closeness Centrality
57419364	343	142.0523125	7718611.921	0.2079099437
16255515	235	99.98408311	7562899.863	0.2035352719
1426071613	224	99.45528214	24976	1
627845910	224	99.23058991	3064020	0.1281323134
454826519	223	99.01363314	24753	1

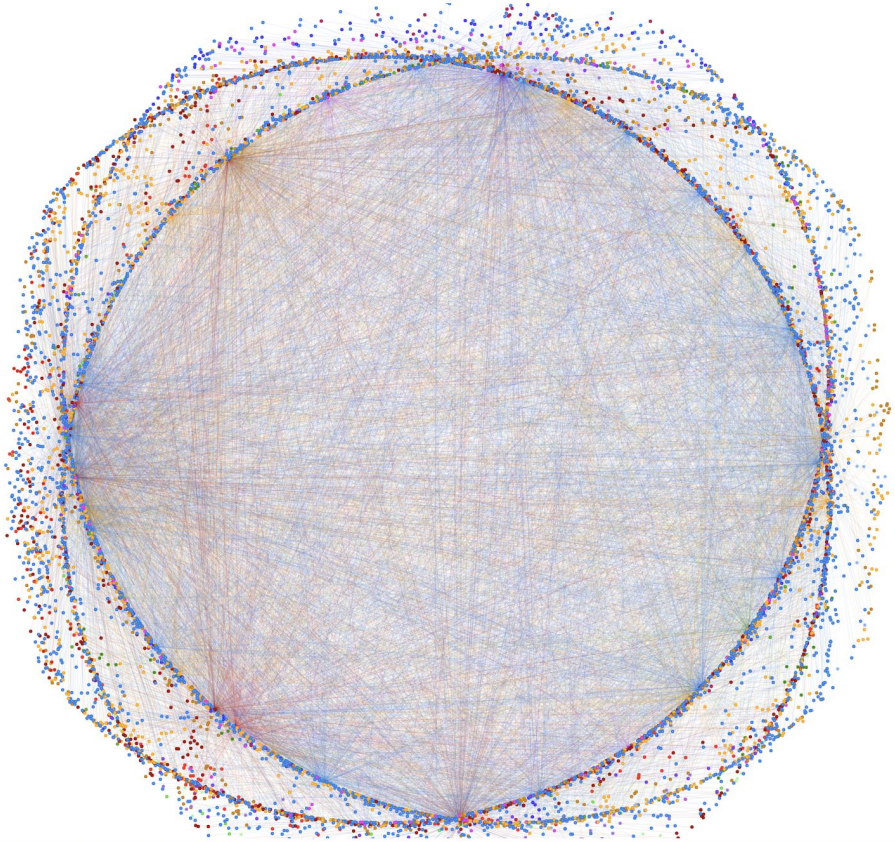
Comparative Analytics of Community Detection Results

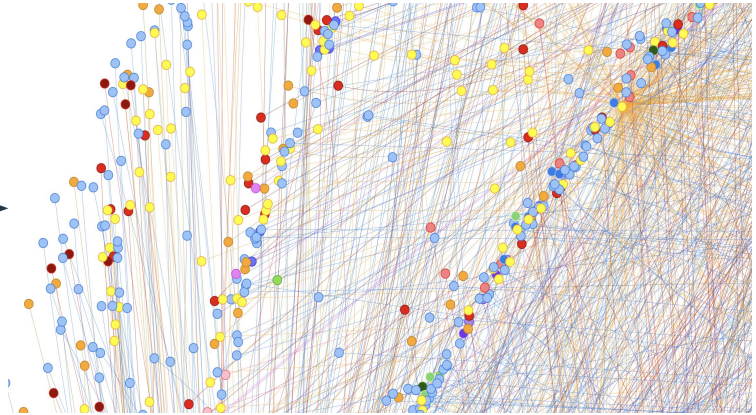
Community Algorithm	Type of Users	Total Number of Communities	Size of the largest Community	Size of the smallest community	Number of overlapping communities formed
Label Propagation	Fake	47	4031	7	N/A
	Real	41	8805	2	N/A
Louvain Modularity	Fake	122	373	7	40
	Real	127	303	2	37
Connected Components	Fake	24	14080	7	N/A
	Real	26	13853	2	N/A

Significance of Influencers in Community Detection Results

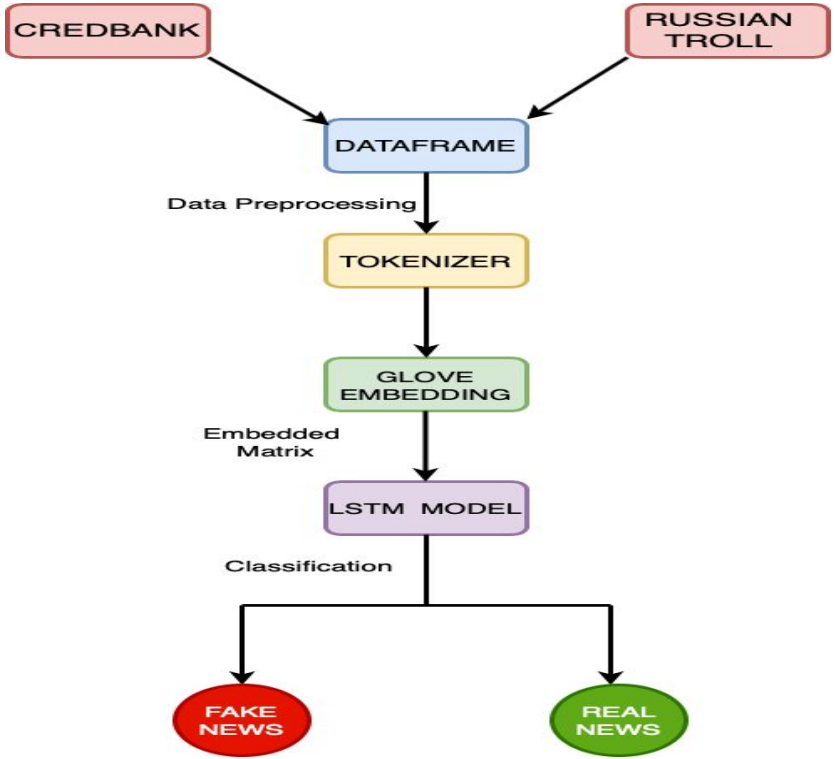
Community Algorithm	Type of Users	Top 3 Influencers (user id)	Community label in which Influencer node is present	Size of the community	Community Rank based on size (Nth largest community)
Label Propagation	Fake	19672966	21104	4031	1
		21237884	18748	2177	3
		20974554	21104	4031	1
	Real	57419364	1	8805	1
		1426071613	150	1610	2
		16255515	14889	225	8
Louvain Modularity	Fake	19672966	6509	350	4
		21237884	9440	373	1
		20974554	3865	338	6
	Real	57419364	8420	303	1
		1426071613	14889	225	2
		16255515	2592	218	13
Connected Components	Fake	19672966	0	14080	1
		21237884	0	14080	1
		20974554	0	14080	1
	Real	57419364	0	13853	1
		1426071613	14888	255	3
		16255515	0	13853	1

Community Detection Visualization

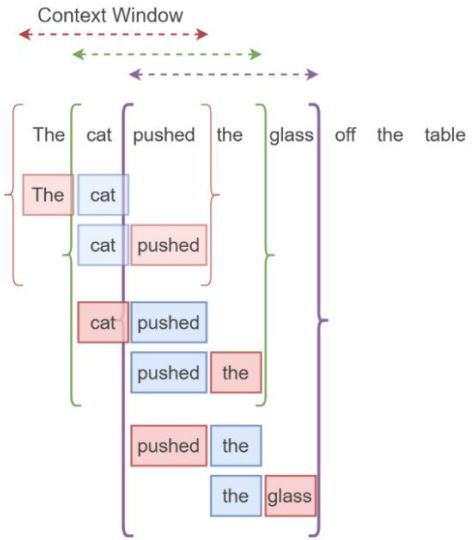




Tweet content analysis - Architecture



GloVe Embedding



Creating data for Word2vec

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

The co-occurrence matrix for the sentence "the cat sat on the mat" with a window size of 1. As you probably noticed it is a symmetric matrix.

```
# word embedding - Word embedding algorithms can figure out tons of relationships from the text data.
# They use the idea of context and learn by seeing what word occurs near other words.
embeddings_index = {}
f = open(GLOVE_DIR, encoding='utf8')
print('Loading Glove from:', GLOVE_DIR,'...', end='')
for line in f:
    values = line.split()
    word = values[0]
    embeddings_index[word] = np.asarray(values[1:], dtype='float32')
f.close()
```

```
Loading Glove from: glove/glove.twitter.27B.100d.txt ...
```

```
print('Done.\n Proceeding with Embedding Matrix…', end='')
embedding_matrix = np.random.random((len(word_index) + 1, EMBEDDING_DIM))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
if embedding_vector is not None:
    embedding_matrix[i] = embedding_vector
print('completed!')
```

```
Done.
 Proceeding with Embedding Matrix…completed!
```

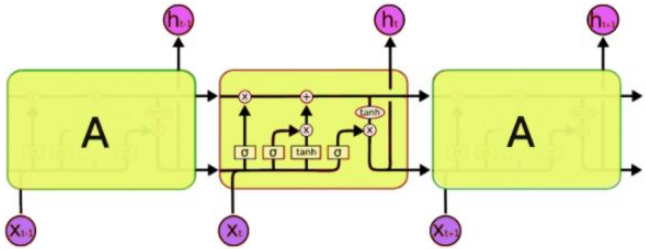
```
embedding_matrix
```

```
array([[ 0.615817  ,  0.77440973,  0.54176615, ...,  0.55195329,
         0.24508226,  0.73264507],
       [ 0.36865596,  0.32552834,  0.40117384, ...,  0.34201113,
         0.90814332,  0.75305113],
       [ 0.57012606,  0.76853613,  0.79023725, ...,  0.65372734,
         0.28810636,  0.92334335],
       ...,
       [ 0.27039668,  0.04949559,  0.44999879, ...,  0.41649154,
         0.67924104,  0.92719648],
       [ 0.46115317,  0.08536362,  0.94084957, ...,  0.80663213,
         0.39114618,  0.69319253],
       [ 0.01125   ,  0.29374   ,  0.33192   , ..., -0.35411   ,
         0.088161  ,  0.83493   ]])
```

```
embedding_matrix.shape
```

```
(75123, 100)
```

LSTM Model



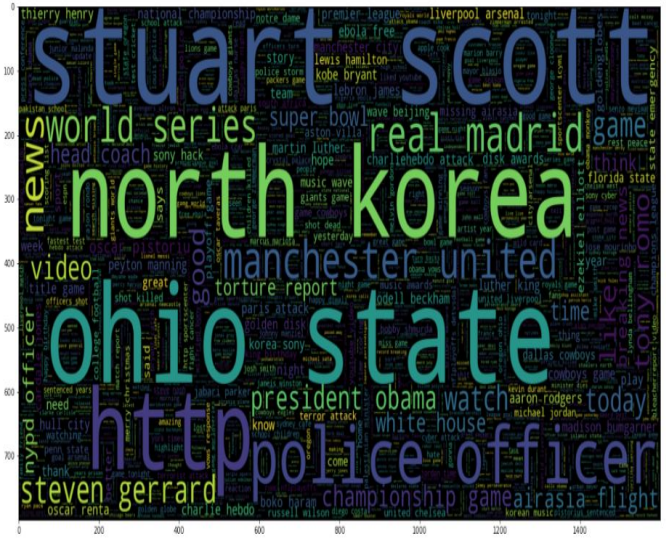

```
#building LSTM model

model = Sequential()
model.add(Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32'))
model.add(Embedding(len(word_index) + 1,
 EMBEDDING_DIM,
 weights = [embedding_matrix],
 input_length = MAX_SEQUENCE_LENGTH,
 trainable=False,
 name = 'embeddings'))
model.add(LSTM(60, return_sequences=True,name='lstm_layer'))
model.add(GlobalMaxPool1D())
model.add(Dropout(0.1))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2, activation='sigmoid'))
```

```
# compiling the model
model.compile(optimizer='adam', loss='binary_crossentropy',metrics = ['accuracy'])
```

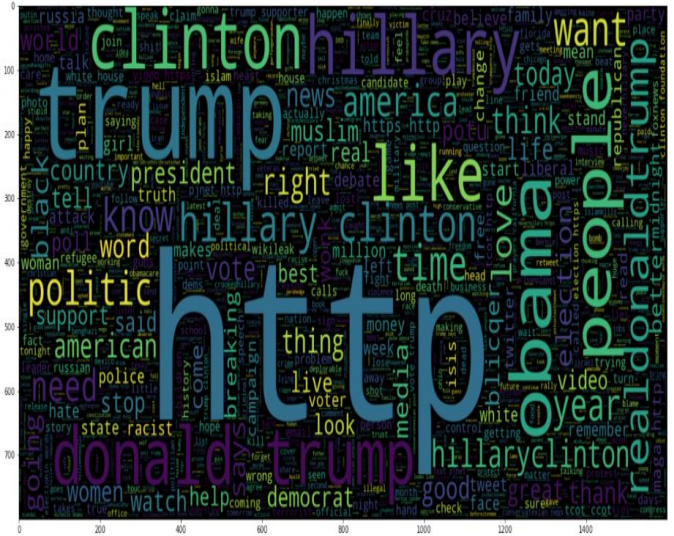
```
history = model.fit(x_train, y_train_val, epochs = 10, batch_size=128, validation_data=(x_val, y_val))
```

```
Epoch 1/10
408/408 [==============================] - 13s 33ms/step - loss: 0.4070 - accuracy: 0.8095 - val_loss: 0.3003 - val
_accuracy: 0.8687
Epoch 2/10
408/408 [==============================] - 12s 30ms/step - loss: 0.2929 - accuracy: 0.8706 - val_loss: 0.2591 - val
_accuracy: 0.8867
Epoch 3/10
408/408 [==============================] - 13s 33ms/step - loss: 0.2597 - accuracy: 0.8879 - val_loss: 0.2458 - val
_accuracy: 0.8916
Epoch 4/10
408/408 [==============================] - 14s 35ms/step - loss: 0.2303 - accuracy: 0.9019 - val_loss: 0.2054 - val
_accuracy: 0.9140
Epoch 5/10
408/408 [==============================] - 17s 41ms/step - loss: 0.2101 - accuracy: 0.9122 - val_loss: 0.2142 - val
_accuracy: 0.9093
Epoch 6/10
408/408 [==============================] - 17s 41ms/step - loss: 0.2009 - accuracy: 0.9159 - val_loss: 0.1842 - val
_accuracy: 0.9248
Epoch 7/10
```

Content analysis - Word Clouds

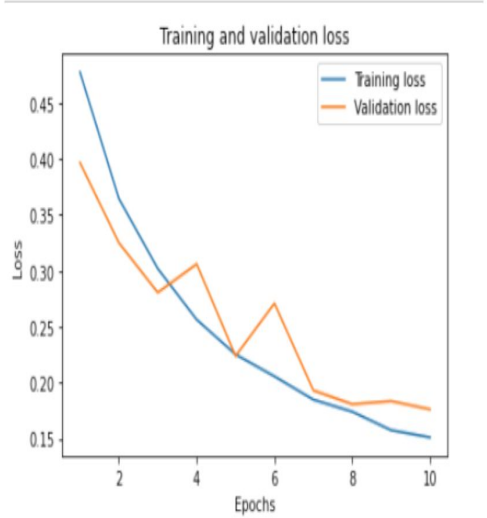


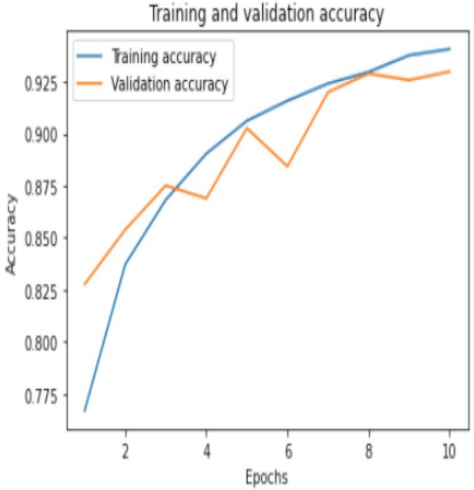
Real News



Fake News

Content analysis results - Learning curve





Content analysis - Results: Confusion Matrix

ACTUAL

PREDICTED	CONFUSION MATRIX	FAKE	REAL
	FAKE	TP 7908	FP 292
	REAL	FN 1068	TN 7033

Future Work

- Improving the influencer detection using weighted centrality measures
- Improving the community detection by building an ensemble community detection algorithm
- Improving the content based model by obtaining larger and less biased data (current dataset contains fake news around the 2016 elections)
- Integrating the two approaches to obtain a more robust system

Thank You