ISE 540: Text Analytics

2020 Election Hashtags Classification & Sentiment Analysis

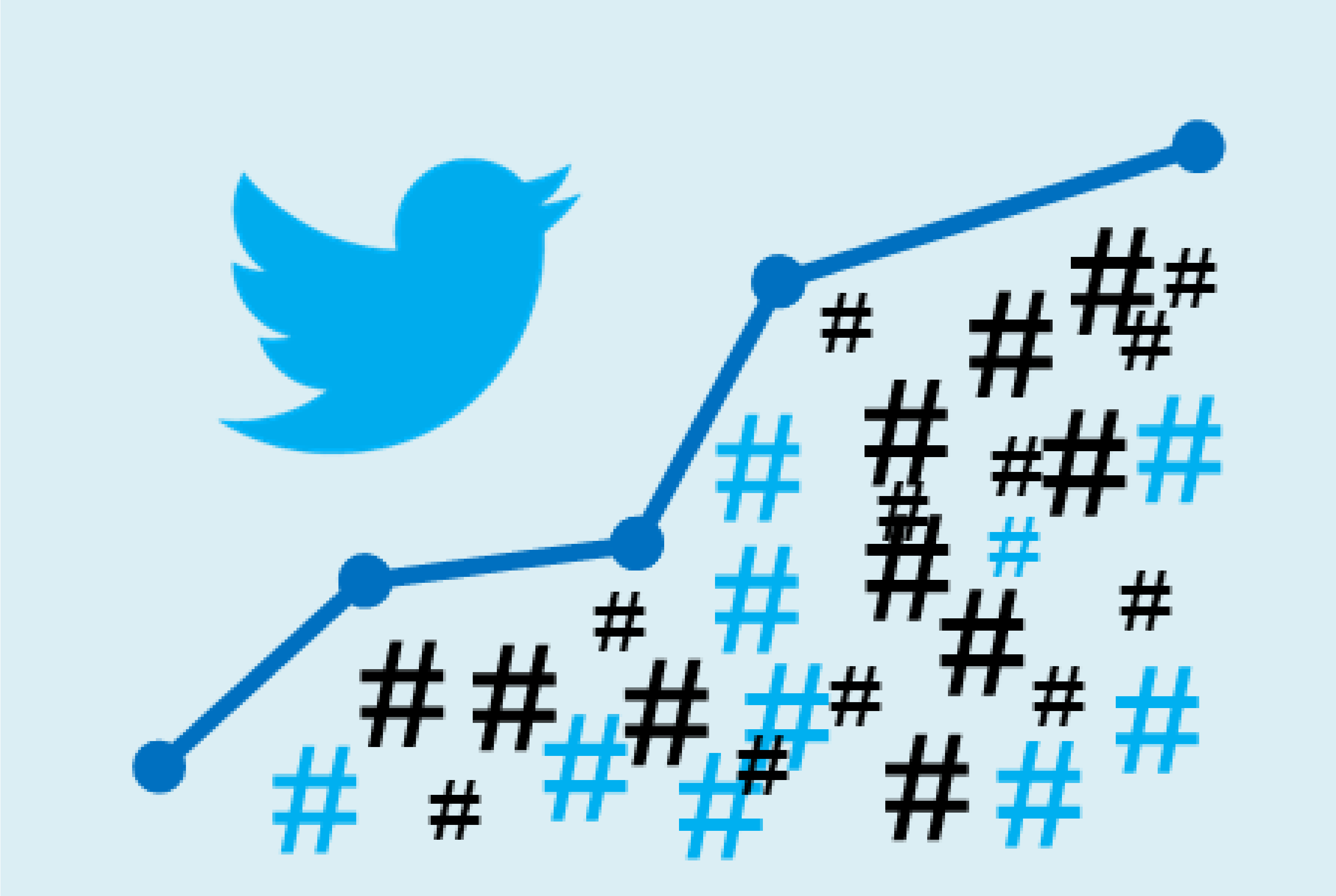
Instructor: Professor Mayank Kejriwal

Team: Influencers
Jiawei Huang
Boxue Li
Hsuan-Hsuan Wu
Junteng Zheng

November 16, 2020

USC Viterbi

Problem Statement (1/2)



Problem Statement (2/2)

Goal:

To identify the most accurate hashtag to a tweet and understand how intense people feel about the topic hashtags about the 2020 Election.

1. To identify the most accurate hashtag for a tweet's content
2. To understand how people feel about top hashtags on Twitter by sentiment analysis.

Data Collection

- Source: Twitter API
- Search Keywords:
  #DonaldTrump, #JoeBiden, #2020Election, #Vote, #Debates2020
- Collected 274,069 raw data from 10/21/2020 to 11/08/2020.

1321964788839145474 2020-10-29 23:59:01 {'hashtags': [{'text': 'Debates2020', 'indices': [96, 108]}], 'symbols': [], 'user_mentions': [{'screen_name': 'Reuters', 'name': 'Reuters', 'id': 1652541, 'id_str': '1652541', 'indices': [3, 11]}], 'urls': [], 'media': [{'id': 1319703944831094784, 'id_str': '1319703944831094784', 'indices': [109, 132], 'media_url': 'http://pbs.twimg.com/amplify_video_thumb/1319703944831094784/img/tFjoqiltawc2ZN8V.jpg', 'media_url_https': 'https://pbs.twimg.com/amplify_video_thumb/1319703944831094784/img/tFjoqiltawc2ZN8V.jpg', 'url': 'https://t.co/xaMdbd9j4m', 'display_url': 'pic.twitter.com/xaMdbd9j4m', 'expanded_url': 'https://twitter.com/Reuters/status/1319706191832977408/video/1', 'type': 'photo', 'sizes': {'thumb': {'w': 150, 'h': 150, 'resize': 'crop'}, 'medium': {'w': 1200, 'h': 675, 'resize': 'fit'}, 'small': {'w': 680, 'h': 383, 'resize': 'fit'}, 'large': {'w': 1280, 'h': 720, 'resize': 'fit'}}, 'source_status_id': 1319706191832977408, 'source_status_id_str': '1319706191832977408', 'source_user_id': 1652541, 'source_user_id_str': '1652541'}]} RT @Reuters: Watch: Biden asks Trump what he's hiding and asks him to release his tax returns 👇 #Debates2020 https://t.co/xaMdbd9j4m

Data Cleaning

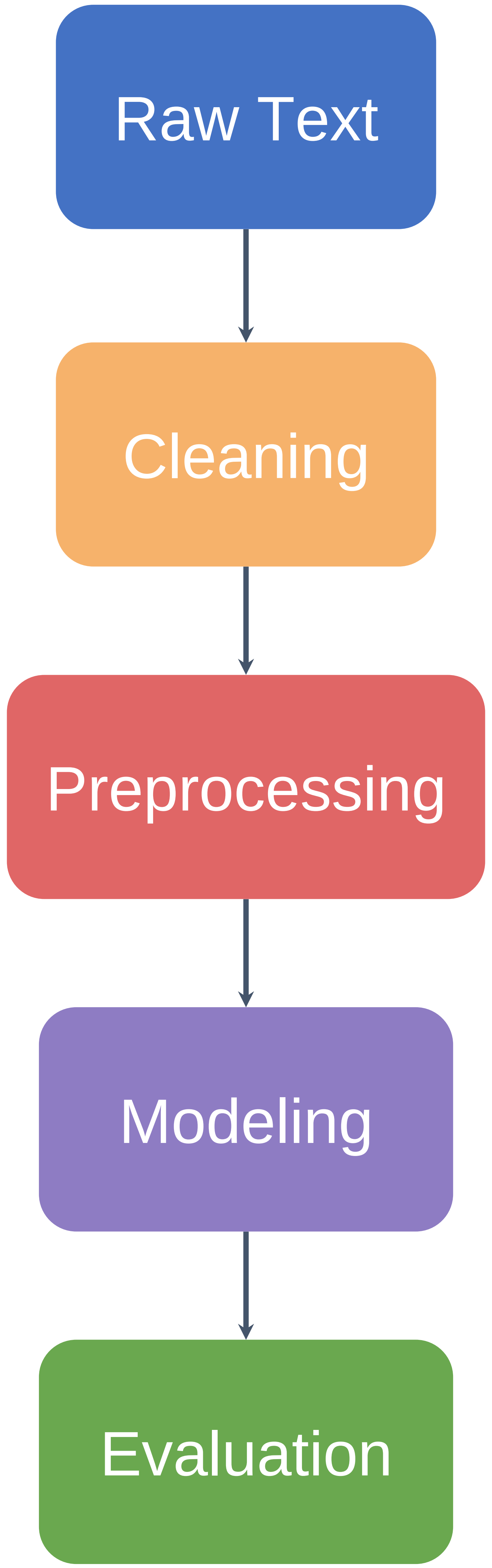
- Remove html links, RT, @username, and punctuations
- Remove rows with no hashtags and no tweets
- Remove distinct hashtags that less than 20 observations
- Duplicate tweets and explode hashtags

	id	date	hashtags	tweet
0	1320877783984312320	2020-10-26	SCOTUSHearing	They are throwing out precedent. Make them pay
1	1320877783984312320	2020-10-26	2020election	They are throwing out precedent. Make them pay
2	1320877783984312320	2020-10-26	BidenHarris2020	They are throwing out precedent. Make them pay
3	1323776887651446788	2020-11-03	DonaldTrump	gives a last ditch attempt at a re-election campaign as scowls at him.
4	1323776887651446788	2020-11-03	FrankZappa	gives a last ditch attempt at a re-election campaign as scowls at him.
...	...	...	...	...
947570	1323776977241808896	2020-11-04	DonaldTrump	Author Believes Donald Trump May Return To WWE Television Soon
947571	1323776977241808896	2020-11-04	POTUS	Author Believes Donald Trump May Return To WWE Television Soon
947572	1323776977241808896	2020-11-04	Election2020	Author Believes Donald Trump May Return To WWE Television Soon
947573	1323776977241808896	2020-11-04	WWE	Author Believes Donald Trump May Return To WWE Television Soon
947574	1323776977241808896	2020-11-04	WWERaw	Author Believes Donald Trump May Return To WWE Television Soon

947575 rows × 4 columns

Data Preprocessing

- Lowercase Text
- Remove Stop Words
- Lemmatization
- Tokenizing
- Text Features Extraction
  - TF-IDF Vectorizer
  - Count Vectorizer



Solution - Classification Model (1/4)

Multinomial naïve Bayes:

- ➢pure statistical model
- ➢calculate the probabilities of all classes
- ➢basic multi-class text classifier
- ➢fast, less parameters

Probability of event A occured and event B occured

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Probability of event A given B has occured

Probability of event B

Solution - Classification Model (2/4)

Random Forest classifier:

➢classic powerful classifier algorithm

➢grid search parameters(n_estimators, max_depth...

➢have advanced method(xgboost, lightgbm...)

Solution - Classification Model (3/4)

Long Short Term Memory(LSTM):

- ➢a variation of RNN
- ➢good at text problem
- ➢easy implementation using Keras

```
model = Sequential()
model.add(Embedding(len(word_index) + 1,
                    100,
                    weights=[embedding_matrix],
                    input_length=max_len,
                    trainable=False))
model.add(SpatialDropout1D(0.3))
model.add(LSTM(100, dropout=0.3, recurrent_dropout=0.3))

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.8))

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.8))

model.add(Dense(3154))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Solution - Classification Model (4/4)

- **Input:**
  X: TF-IDF/Count vector of top frequency words

MultinomialNB() with TfidfVec

MultinomialNB() with CountVec

Random Forest with TfidfVec

Random Forest with CountVec

LSTM with WordEmbedding

Lightgbm

XGBoost

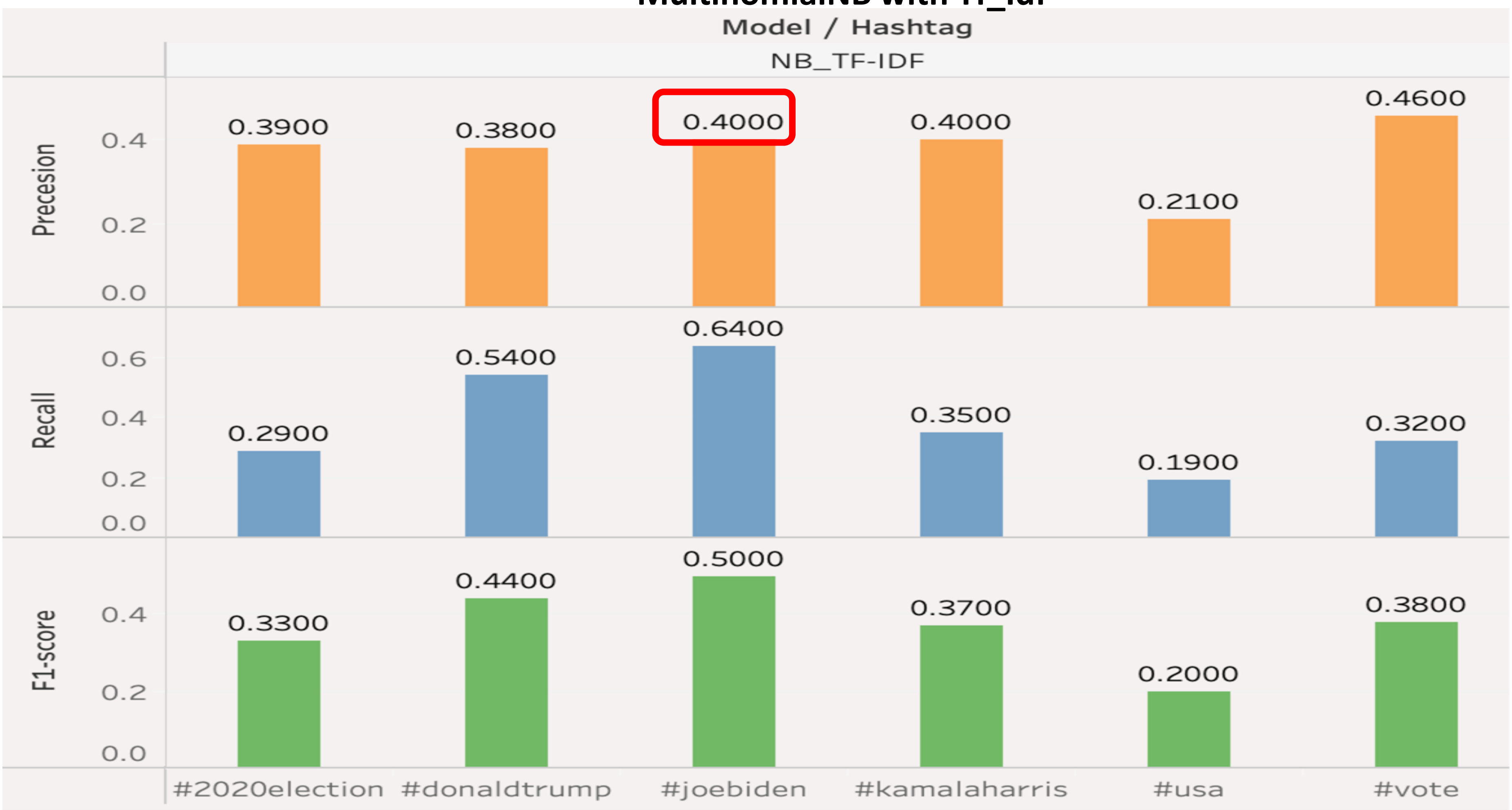
Output:**Predicted Hashtags**

Evaluated by **own hashtags**

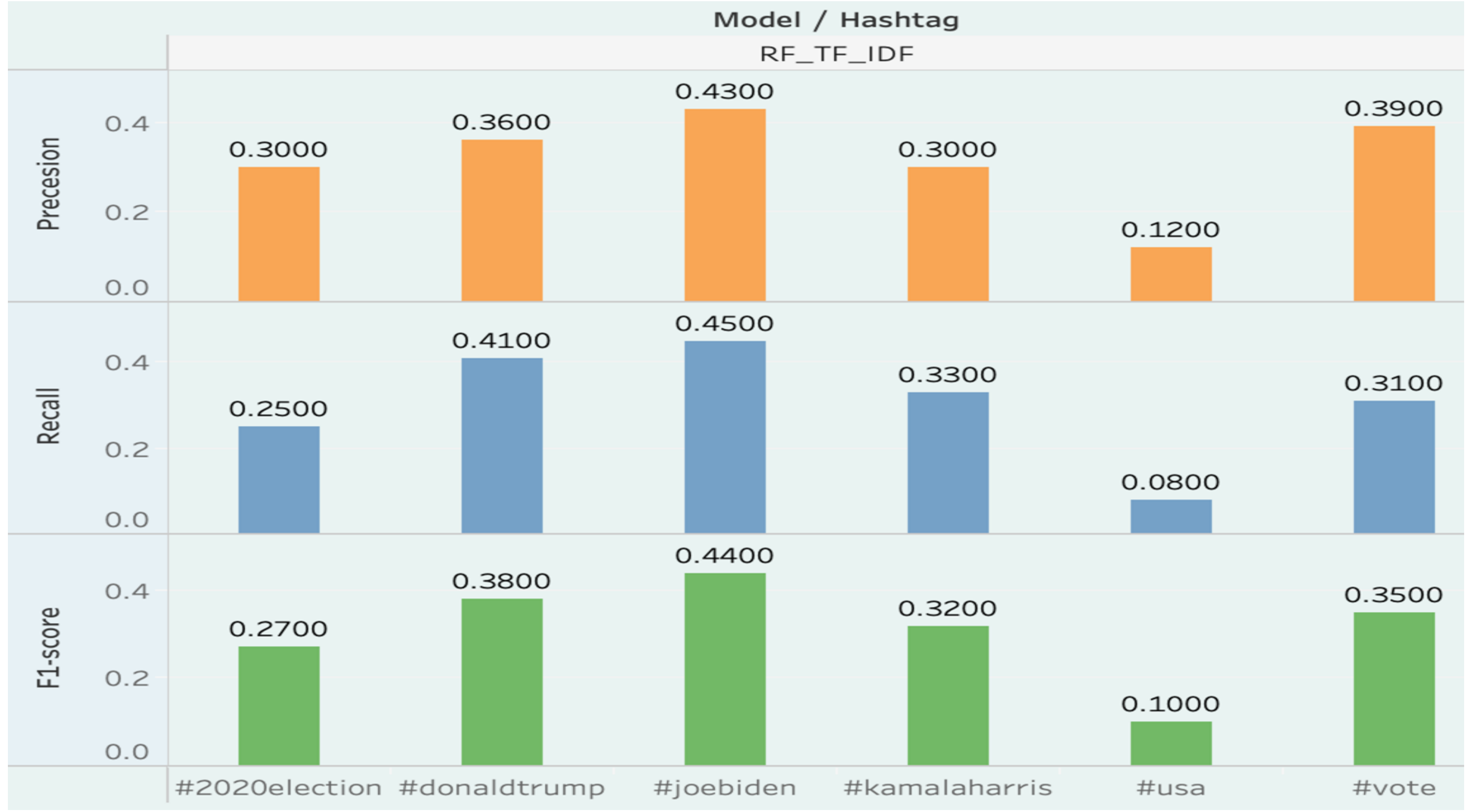
Compare with **baseline model(random assigned hashtag)**

Evaluations & Metrics (1/2)

MultinomialNB with Tf_Idf

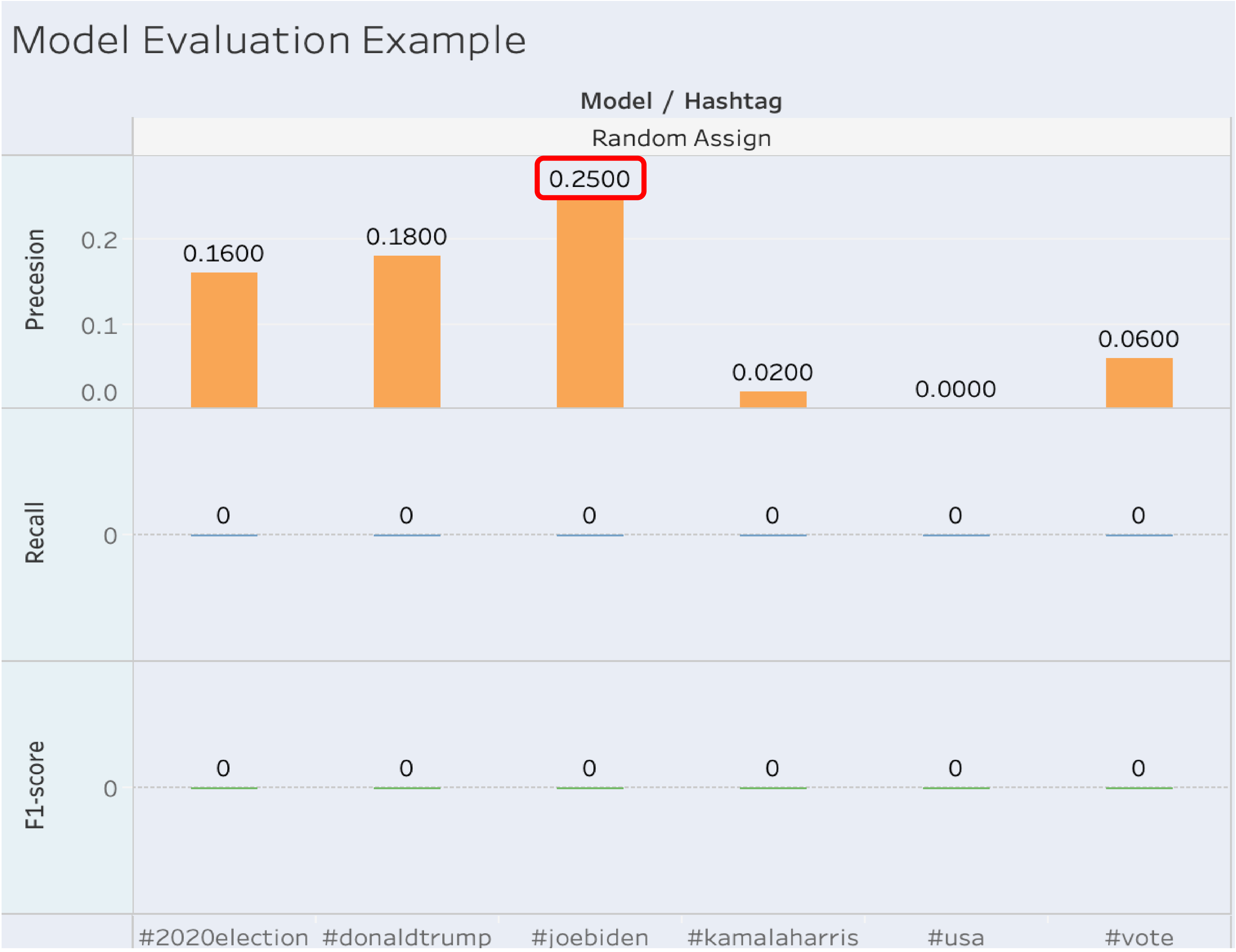


Random Forest with Tf_Idf



Comparison

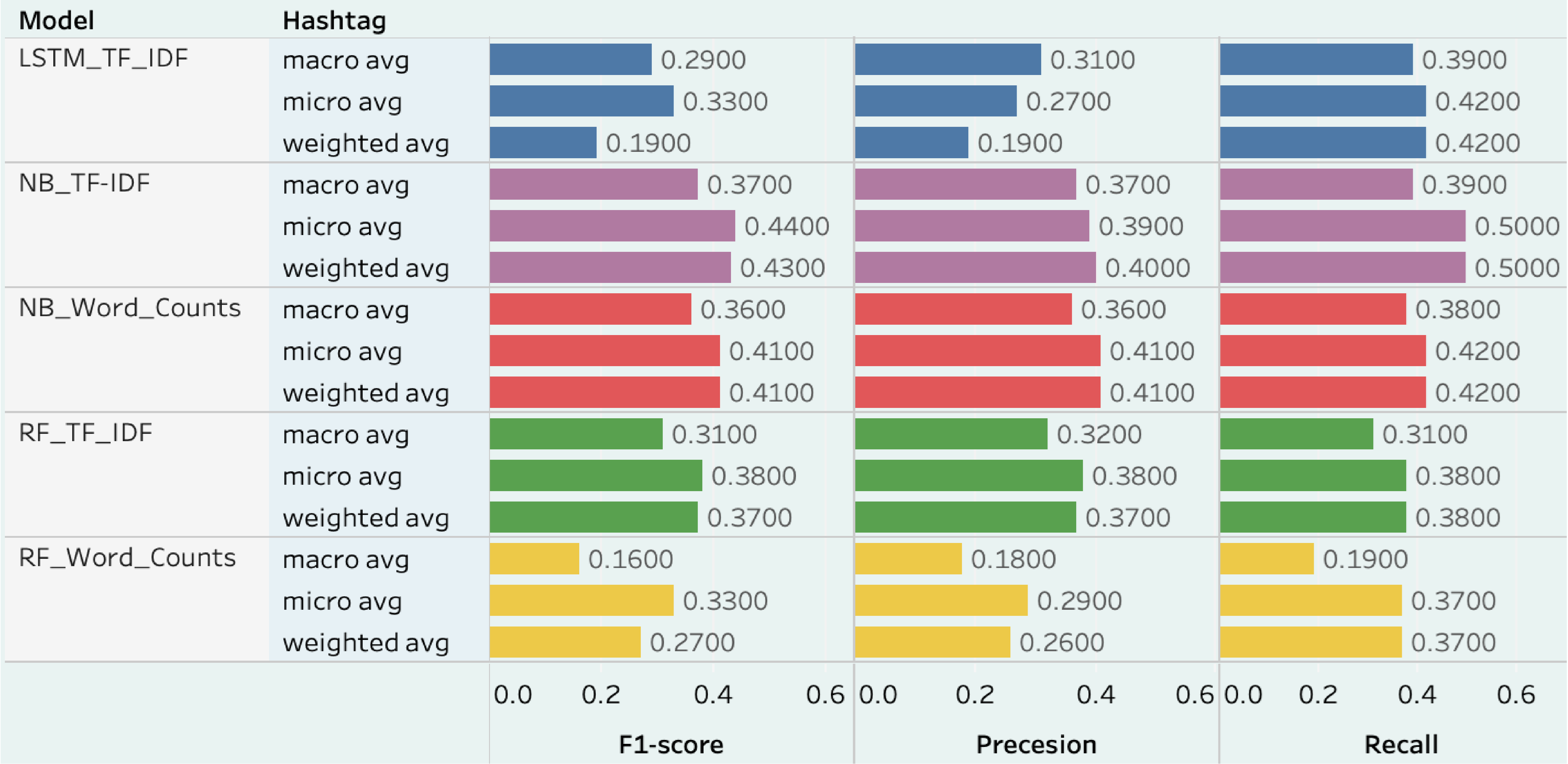
Baseline Model_Random Assign



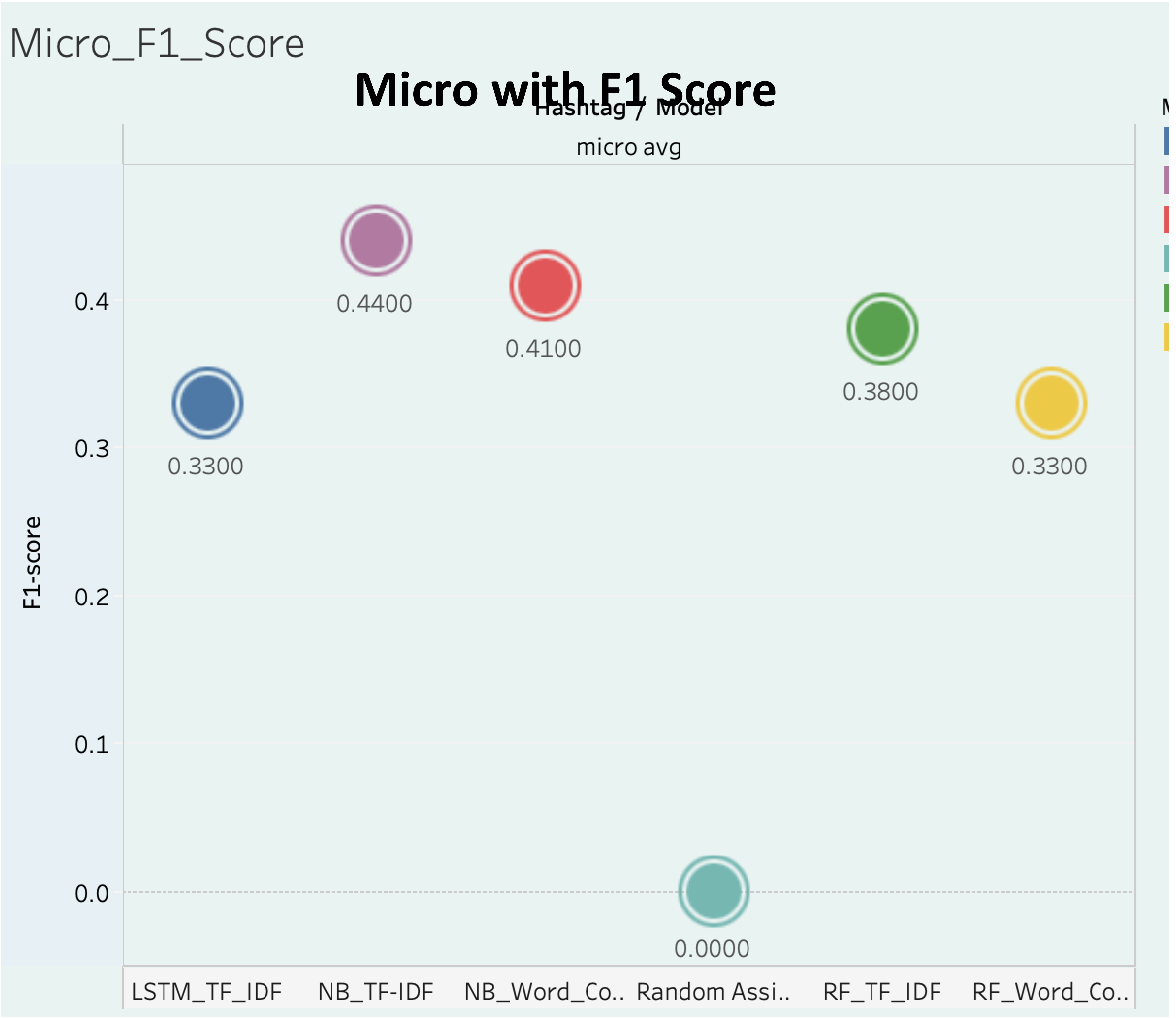
Accuracy, precision,recall,F1_score → Macro & MIcro & weighted average

Evaluations & Metrics (2/2)

Trade off between Macro & Micro & Weighted Average



Micro with F1 Score



Micro formula

$$Micro\_P = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{9}$$

$$Micro\_R = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{10}$$

$$Micro\_F = \frac{2 \times Micro\_P \times Micro\_R}{Micro\_P + Micro\_R}$$

Micro-average Method

In Micro-average method, you sum up the individual true positives, false positives, and false negatives of the system for different sets and the apply them to get the statistics.

USC Viterbi

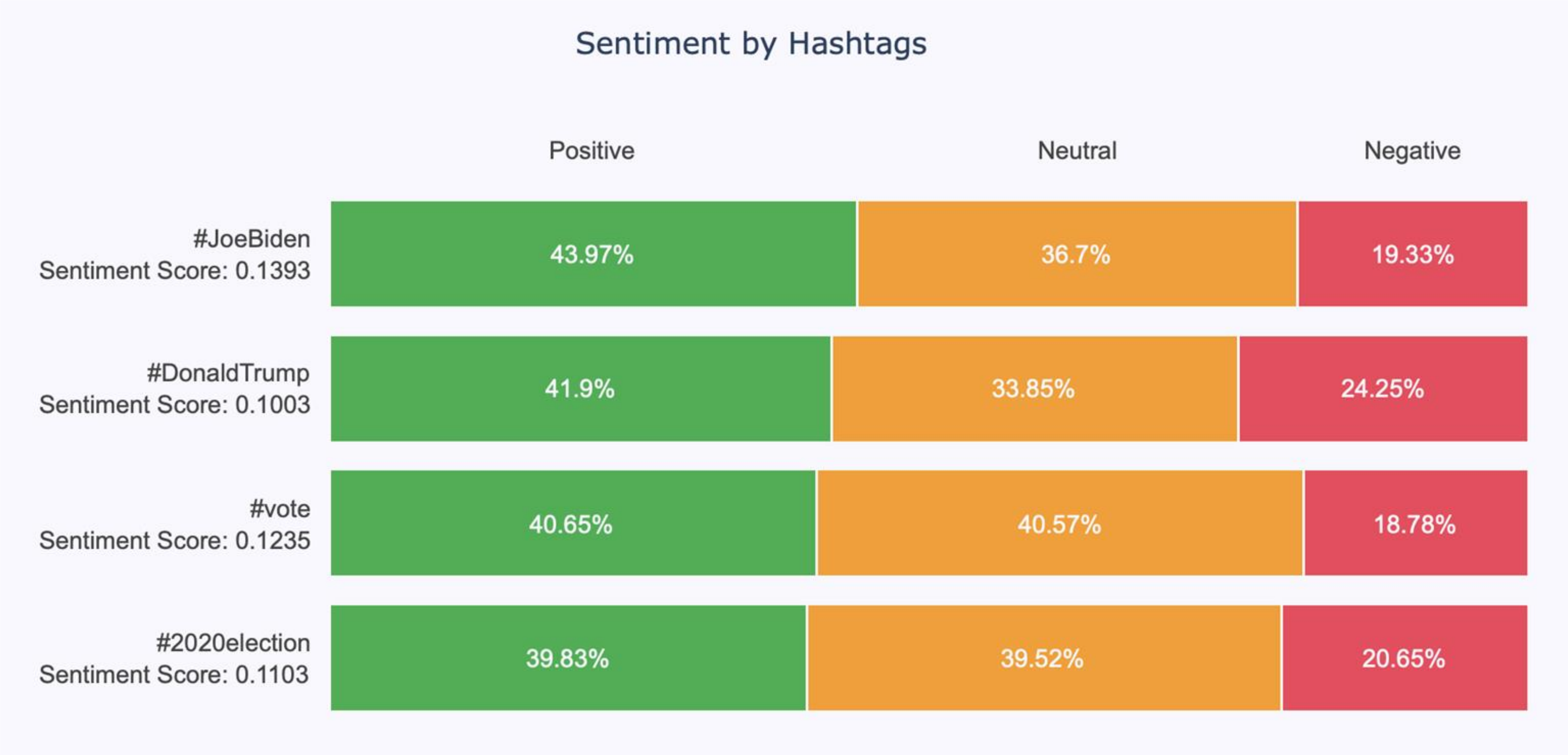
Sentiment Analysis

- Top 4 hashtags: *#JoeBiden, #DonaldTrump, #2020Election, #Vote*
- Tool: VADER (Valence Aware Dictionary and sEntiment Reasoner)
- Example of a given hashtag #JoeBiden:
  - Input Tweet:

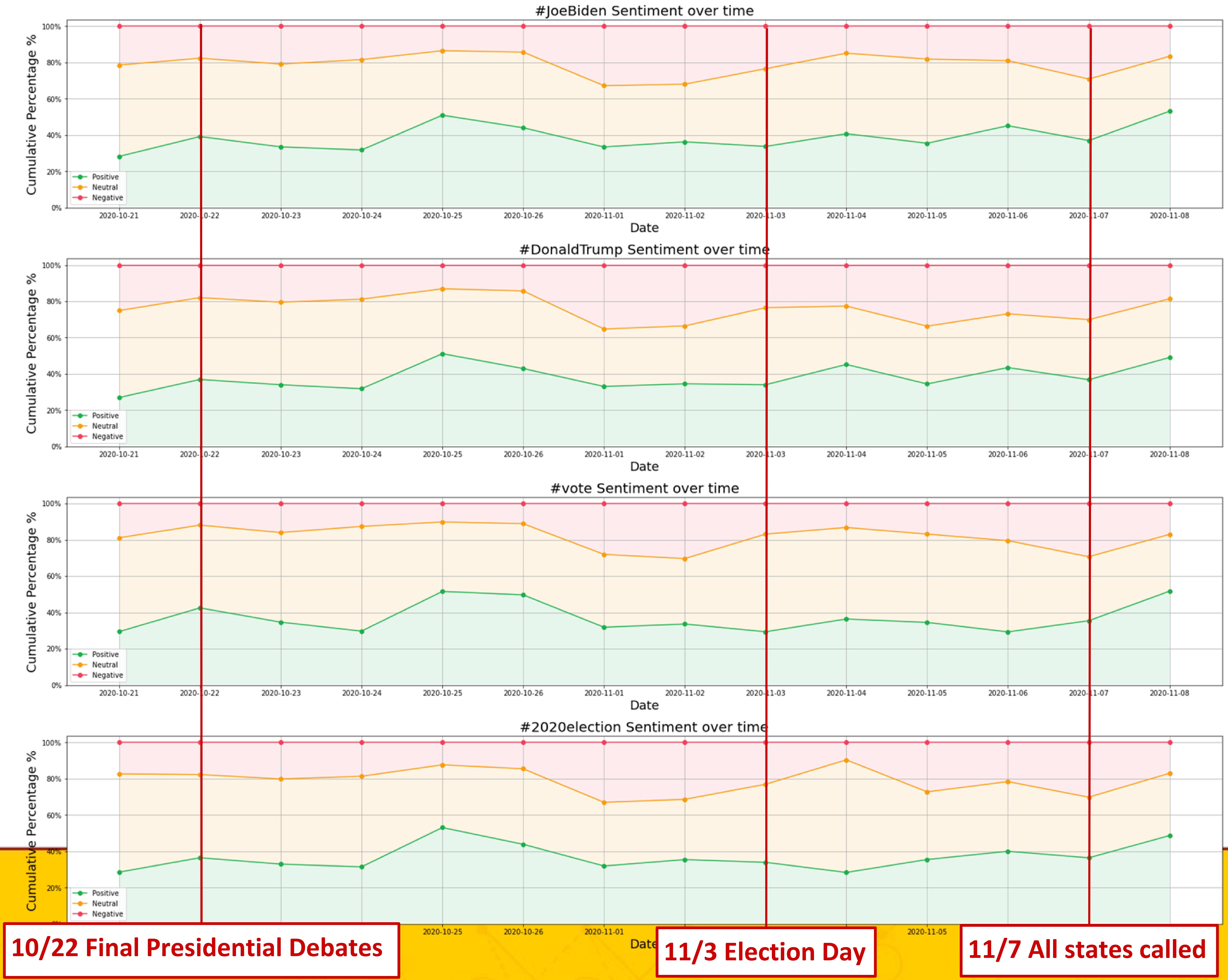
    'god bless who is fighting the good fight for the soul of our country for our humanity and who we are as'
  - Output Sentiment:

    Overall sentiment dictionary is : {'neg': 0.325, 'neu': 0.188, 'pos': 0.488, 'compound': 0.3818}
    sentence was rated as 32.5 % Negative
    sentence was rated as 18.8 % Neutral
    sentence was rated as 48.8 % Positive
    Sentence Overall Rated As Positive

Sentiment Analysis



Sentiment Analysis



USC Viterbi

Sentiment Analysis Conclusion

- For each hashtag, the percentage of positive sentiment is slightly higher.
- Each hashtag sentiment follows a similar pattern over time.
- There is a change in sentiment when an event occurs.

Lessons Learned & Failure

- Classification
    - Address text data using multi-classification algorithm
    - Deep learning framework
    - Fail to invoke the GPU for model computation
- Sentiment Analysis
    - Vader is a powerful sentiment analysis tool.
    - Gauge public opinion before and after election period.
    - People assign hashtags but talk about other things.

Future Work

- Classification
  - Perform xgboost, lightgbm with better computation power
  - Hyperparameter tuning for LSTM(like different batch_size)
- Sentiment Analysis
  - Include emoji
  - Try different methods
  - Sentiment analysis on the predicted hashtags from classification model

Thank you

Q & A

Appendix

Xgboost

```python
def multiclass_logloss(actual, predicted, eps=1e-15):
    # Convert 'actual' to a binary array if it's not already:
    if len(actual.shape) == 1:
        actual2 = np.zeros((actual.shape[0], predicted.shape[1]))
        for i, val in enumerate(actual):
            actual2[i, val] = 1
        actual = actual2

    clip = np.clip(predicted, eps, 1 - eps)
    rows = actual.shape[0]
    vsota = np.sum(actual * np.log(clip))
    return -1.0 / rows * vsota
```

```python
# using xgboost
clf = xgb.XGBClassifier(max_depth=7, n_estimators=2, colsample_bytree=0.8,
                        subsample=0.8, nthread=10, learning_rate=0.1)
clf.fit(train_features, y_train)
predictions = clf.predict_proba(test_features)

print ("logloss: %0.3f " % multiclass_logloss(y_test, predictions))
```

Appendix

LSTM

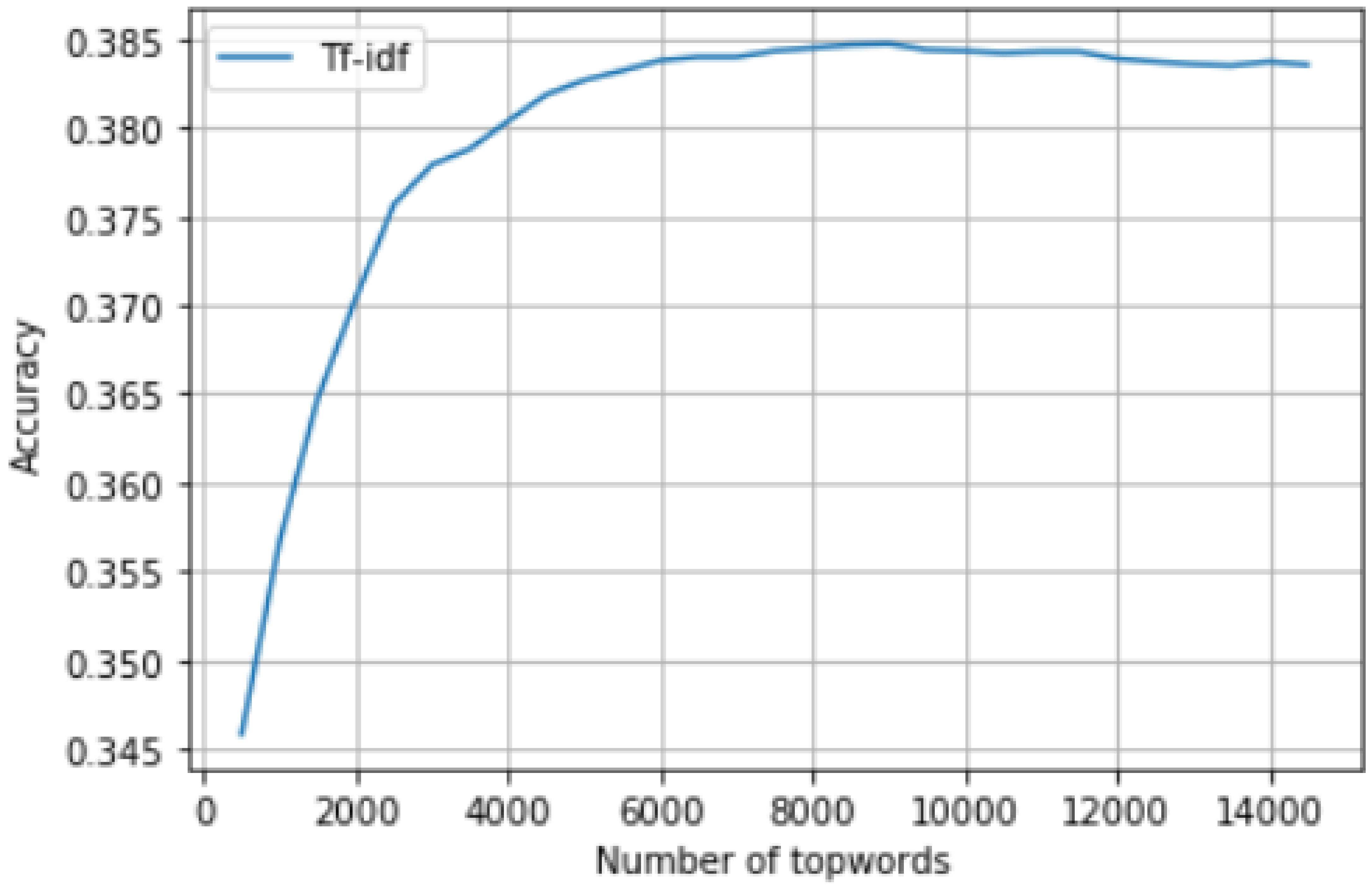
```
model.fit(xtrain_pad, y=ytrain_enc, batch_size=128, epochs=100, verbose=1, validation_data=(xvalid_pad, yvalid_enc))
Epoch 92/100
2497/2497 [==============================] - 302s 121ms/step - loss: 3.5753 - val_loss: 3.5888
Epoch 93/100
2497/2497 [==============================] - 301s 121ms/step - loss: 3.5766 - val_loss: 3.5910
Epoch 94/100
2497/2497 [==============================] - 301s 120ms/step - loss: 3.5771 - val_loss: 3.5883
Epoch 95/100
2497/2497 [==============================] - 301s 120ms/step - loss: 3.5771 - val_loss: 3.5882
Epoch 96/100
2497/2497 [==============================] - 299s 120ms/step - loss: 3.5766 - val_loss: 3.5897
Epoch 97/100
2497/2497 [==============================] - 300s 120ms/step - loss: 3.5760 - val_loss: 3.5903
Epoch 98/100
2497/2497 [==============================] - 302s 121ms/step - loss: 3.5762 - val_loss: 3.5875
Epoch 99/100
2497/2497 [==============================] - 299s 120ms/step - loss: 3.5753 - val_loss: 3.5862
Epoch 100/100
2497/2497 [==============================] - 300s 120ms/step - loss: 3.5761 - val_loss: 3.5884

<tensorflow.python.keras.callbacks.History at 0x1a92422630>
```

	precision	recall	f1-score	support
joebiden	0.27	1.00	0.43	21643
donaldtrump	0.00	0.00	0.00	14757
2020election	0.35	0.02	0.03	11744
vote	0.00	0.00	0.00	4397
giveaway	0.40	0.94	0.56	78
coronavirus	0.30	0.16	0.21	126
foia	1.00	1.00	1.00	51
breaking	0.16	0.04	0.06	134
micro avg	0.27	0.42	0.33	52930
macro avg	0.31	0.39	0.29	52930
weighted avg	0.19	0.42	0.19	52930

Appendix

Finding the best number of word counts for each model



Appendix

Grid Search for random forest:

```
cv_rf = RandomForestClassifier(random_state = 42,bootstrap = True)
param_grid = {
    'n_estimators': [20,50,100],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [5,7],
    'criterion' :['gini', 'entropy']
    }
```

```
[56] cv_rf = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, verbose = 2)
     cv_rf.fit(train_features, y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split
  % (min_groups, self.n_splits)), UserWarning)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurre
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=20
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=20
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=20
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   34.2s remainin
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=20
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=20
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=20
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=50
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=50
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=50
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=50
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=50
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=50
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=100
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=10
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=100
[CV]  criterion=gini, max_depth=5, max_features=sqrt, n_estimators=10
[CV] criterion=gini, max_depth=5, max_features=sqrt, n_estimators=100
```

```
[57] cv_rf.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 7,
 'max_features': 'sqrt',
 'n_estimators': 20}
```

```
rf_final = RandomForestClassifier(random_state = 42,bootstrap = True, criterion='gini', max_depth=7,
                                  max_features='sqrt', n_estimators=20)
rf_final.fit(train_features, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=7, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

Appendix

Remove less than 20 observation hashtags

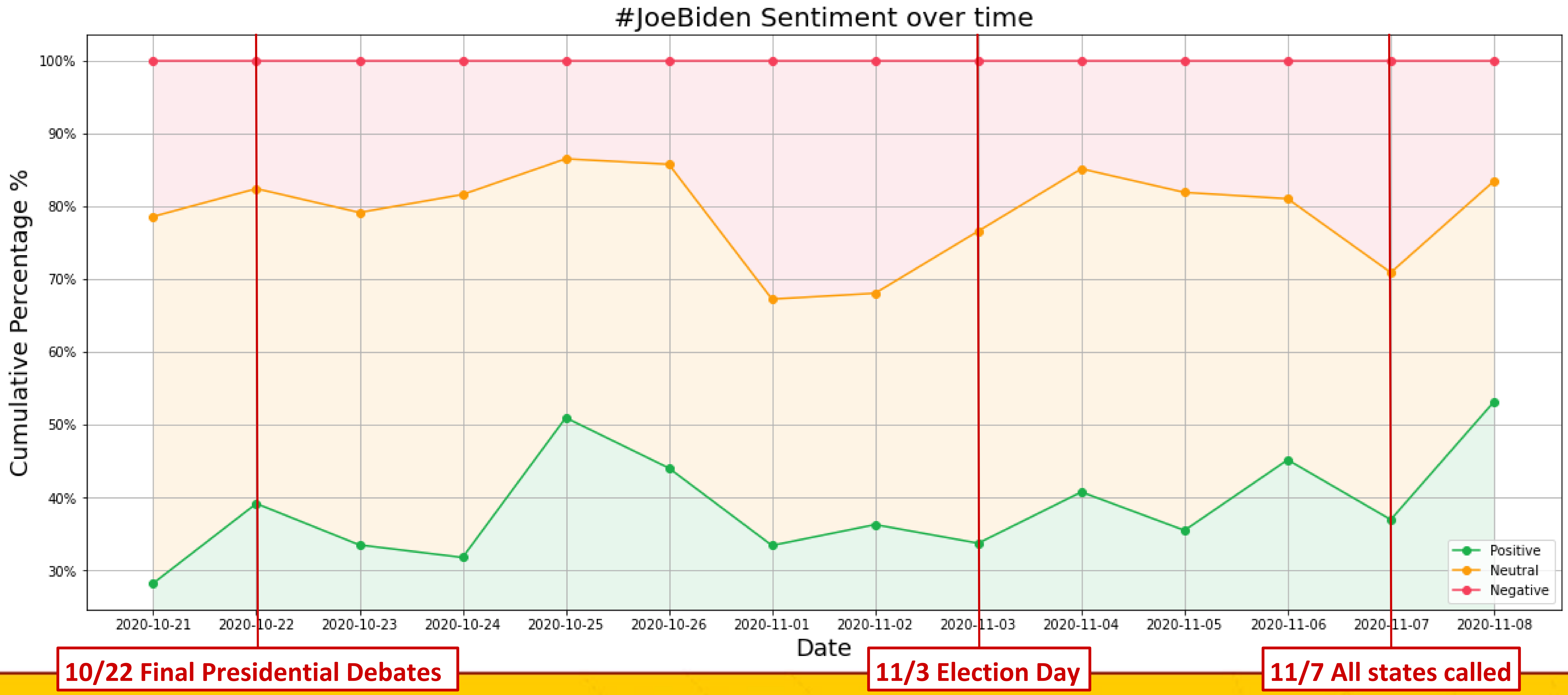
```
for i in range(len(hash_4)):
    df.drop(df.index[df['hashtags'] == hash_4[i]], inplace = True)
```

```
df['hashtags'].value_counts()
```

```
joebiden         108773
donaldtrump       74374
2020election      58384
vote              21983
kamalaharris      12151
                  ...
ballotdropbox        20
46th                 20
address              20
fba                  20
gbwhatsapp           20
Name: hashtags, Length: 925, dtype: int64
```

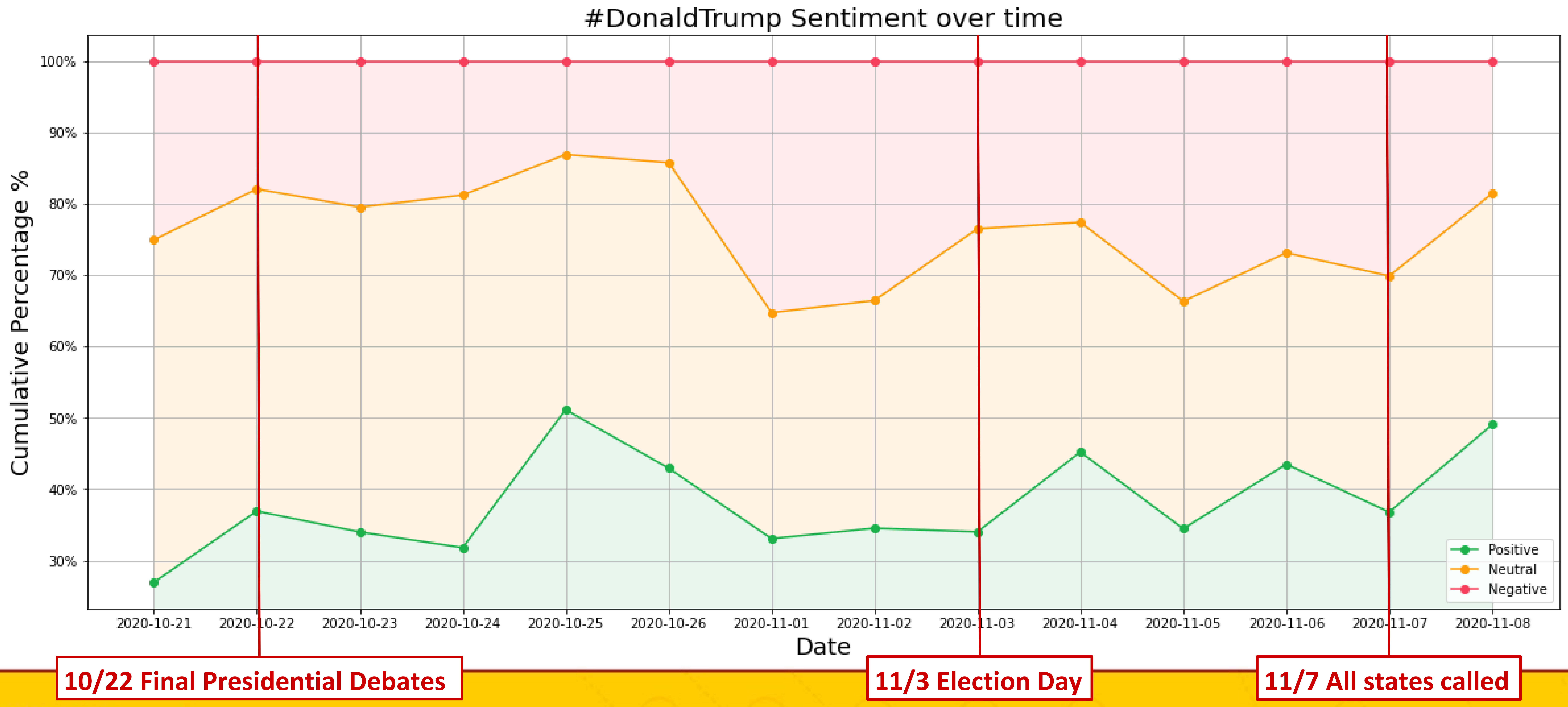
USC Viterbi

Appendix

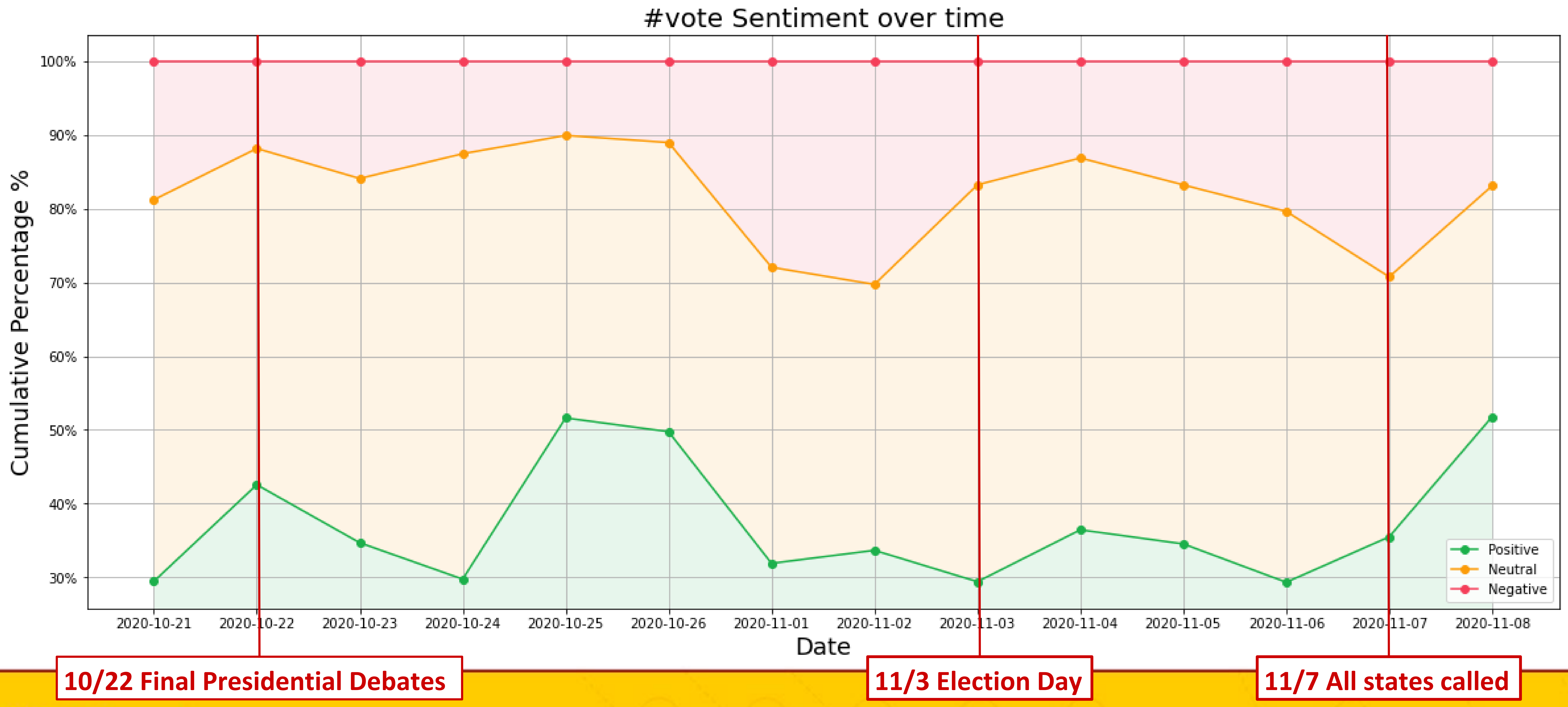


#JoeBiden Sentiment over time
Cumulative Percentage %
100%
90%
80%
70%
60%
50%
40%
30%
2020-10-21
2020-10-22
2020-10-23
2020-10-24
2020-10-25
2020-10-26
2020-11-01
2020-11-02
2020-11-03
2020-11-04
2020-11-05
2020-11-06
2020-11-07
2020-11-08
Date
Positive
Neutral
Negative
10/22 Final Presidential Debates
11/3 Election Day
11/7 All states called

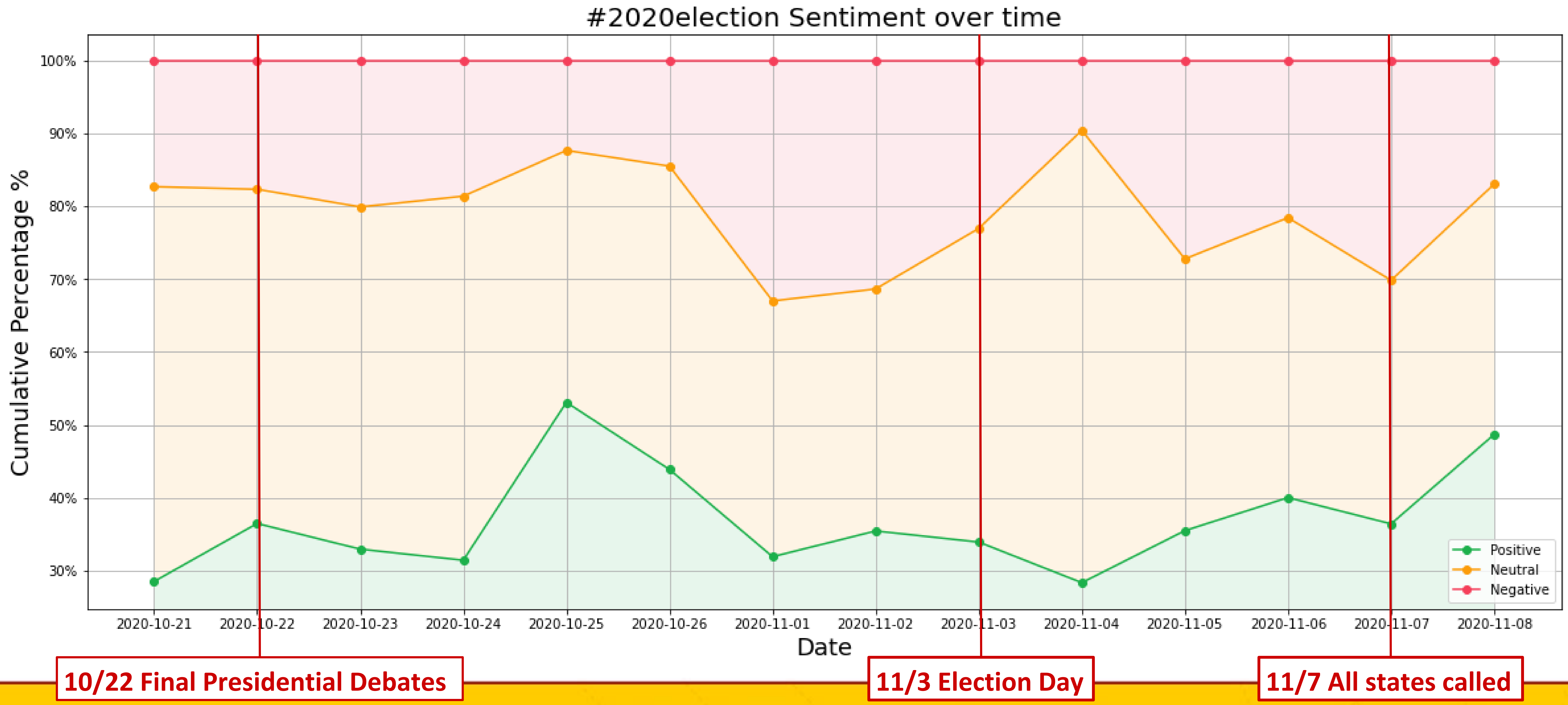
Appendix


#DonaldTrump Sentiment over time
100%
90%
80%
70%
60%
50%
40%
30%
Cumulative Percentage %
2020-10-21
2020-10-22
2020-10-23
2020-10-24
2020-10-25
2020-10-26
2020-11-01
2020-11-02
2020-11-03
2020-11-04
2020-11-05
2020-11-06
2020-11-07
2020-11-08
Date
Positive
Neutral
Negative
10/22 Final Presidential Debates
11/3 Election Day
11/7 All states called

Appendix


#vote Sentiment over time
Cumulative Percentage %
100%
90%
80%
70%
60%
50%
40%
30%
2020-10-21
2020-10-22
2020-10-23
2020-10-24
2020-10-25
2020-10-26
2020-11-01
2020-11-02
2020-11-03
2020-11-04
2020-11-05
2020-11-06
2020-11-07
2020-11-08
Date
Positive
Neutral
Negative
10/22 Final Presidential Debates
11/3 Election Day
11/7 All states called

Appendix


#2020election Sentiment over time
Cumulative Percentage %
100%
90%
80%
70%
60%
50%
40%
30%
2020-10-21
2020-10-22
2020-10-23
2020-10-24
2020-10-25
2020-10-26
2020-11-01
2020-11-02
2020-11-03
2020-11-04
2020-11-05
2020-11-06
2020-11-07
2020-11-08
Date
Positive
Neutral
Negative
10/22 Final Presidential Debates
11/3 Election Day
11/7 All states called

Appendix

$$Macro\_P = \frac{1}{n}\sum_{i=1}^{n} P_i \qquad (5)$$

$$Macro\_R = \frac{1}{n}\sum_{i=1}^{n} R_i \qquad (6)$$

$$Macro\_F = \frac{1}{n}\sum_{i=1}^{n} F_i \qquad (7)$$

$$Macro\_F = \frac{2 \times Macro\_P \times Macro\_R}{Macro\_P + Macro\_R} \qquad (8)$$

$$Micro\_P = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \qquad (9)$$

$$Micro\_R = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \qquad (10)$$

$$Micro\_F = \frac{2 \times Micro\_P \times Micro\_R}{Micro\_P + Micro\_R} \qquad (11)$$

Appendix

USC Viterbi