# Twitter Topic Classification

Julius Olson & Philip Andersson

# Task Specification

- Classification problem
- Classes = Topics
- Documents = Tweets

- Given a tweet, can we determine what its topic is?

# How?

- Twitter datasets about different topics
- Naive Bayes as a baseline
- Neural Net approach
- Compare results and draw conclusions

# Datasets

## Kaggle - Public Datasets

- Airlines
- Republican Candidate Debate - US Election
- Us Election Day 2016
- Bitcoin
- Financial market

# Datasets - Pre processing

- Metadata
- Mentions
- Hashtags
    - Very Common
    - Ordinary
- Hyperlinks
- Lowercase

- Labeled training and testing tweets

```
Satoshi Vision Tokyo Conference is live!

SBI Bits Jerry Chan is up first!

#BitcoinCash is #Bitcoin

https://t.co/br0Z7umhl4
```

```
satoshi vision tokyo conference is live!

sbi bits jerry chan is up first!

bitcoincash is

LINK
```

# Data cleaning

- Tokenization - *nltk*
- Lemmatization - *nltk* - *WordNetLemmatizer*
- Remove unusual words -- hyperparameter
- Remove common words -- hyperparameter

- Cleaned training and testing documents

```
['piper', 'jaffray', 'company', 'weighs', 'in', 'on', 'capital', 'one',
'financial', 'corp.', '’', 's', 'q4', 'NUMERIC', 'earnings', 'PUNCTUATION', 'cof', 'LINK']
```

# Naive Bayes

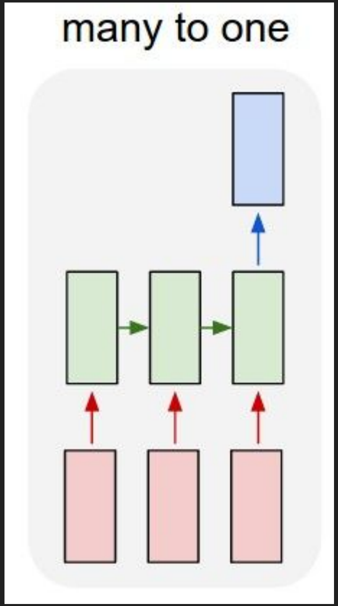
- Build feature sets
  - Training
  - Testing
- Bag of words
- Bag of Bigrams
- NLTK - NaiveBayesClassifier

Prior Likelihood

$$predicted\ label = argmax_{topic \in topics} \underbrace{P(topic)}_{\text{Prior}} \Pi_{token \in tweet} \underbrace{P(token|topic)}_{\text{Likelihood}}$$

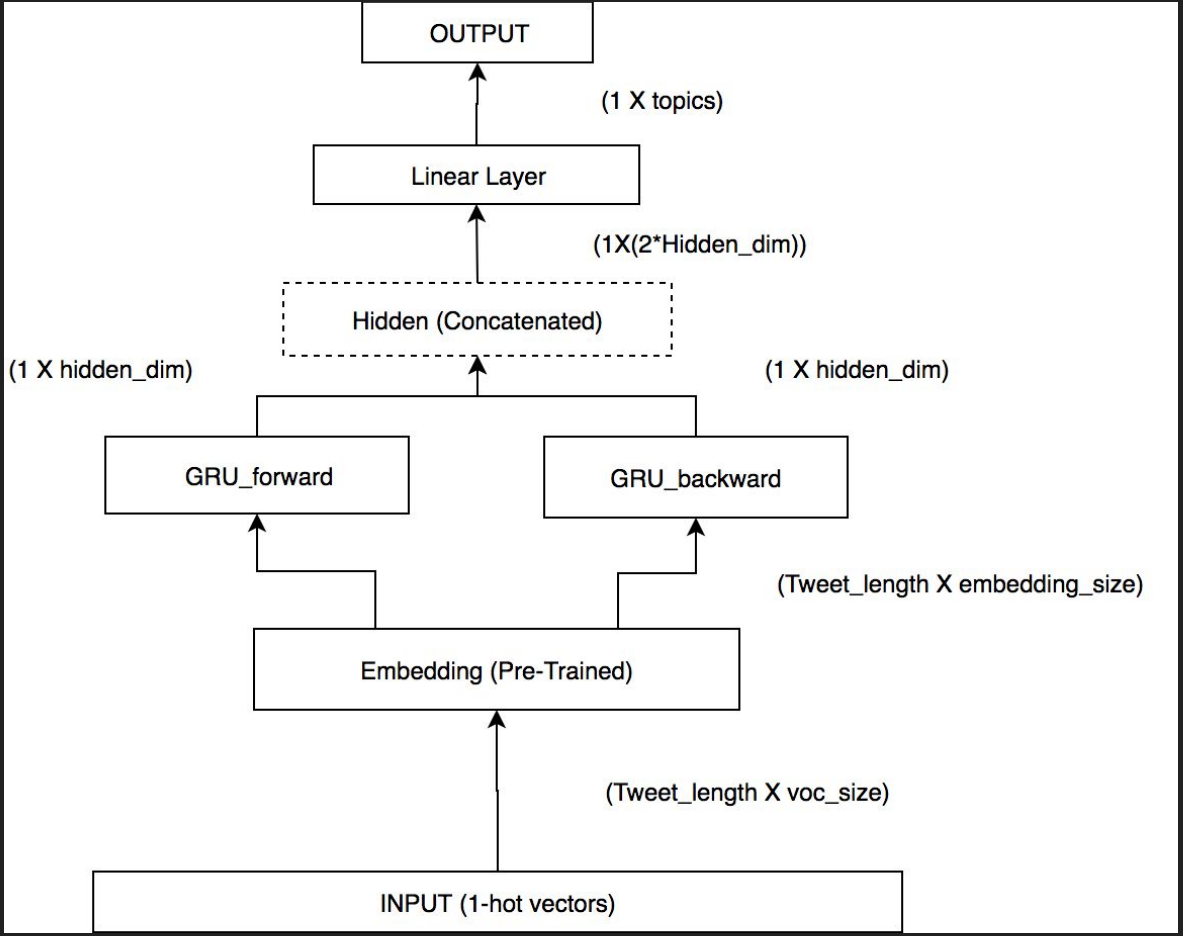
# Neural Network - Gated Recurrent Unit

- Verison of Recurrent Neural Network
- Tackles issues of vanishing/exploding gradient
- Contains update gate and reset gate
- The cells control rate of information transferred between time steps
- Allows us to look at context instead of just bag of words (this is however true with all RNNs)



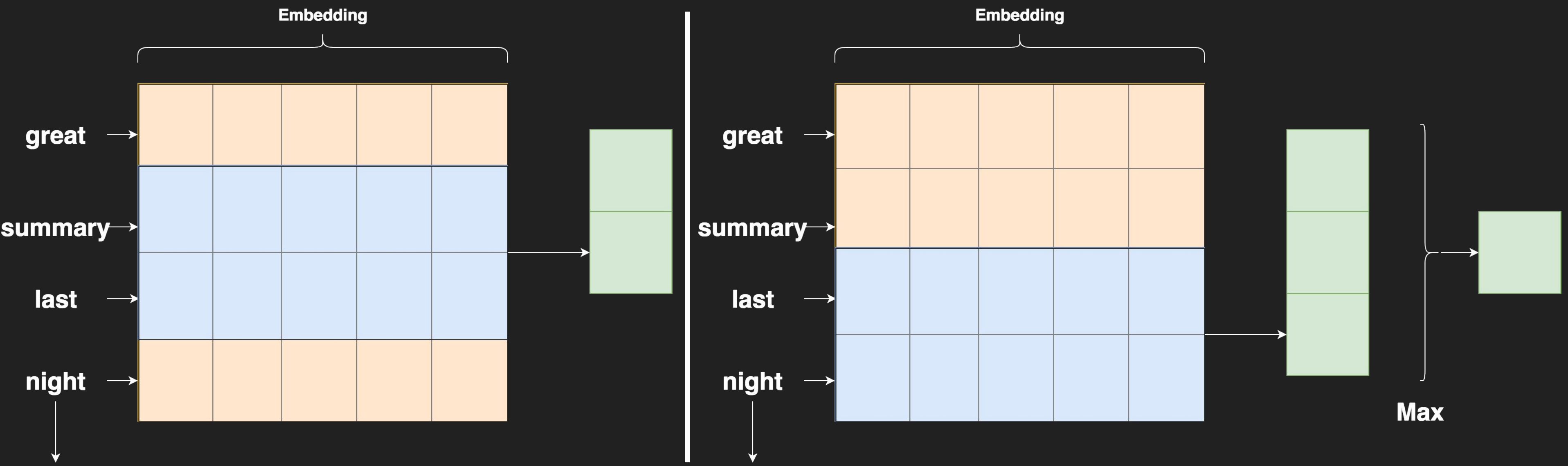
# Neural Network - GRU

- Activation function:
  RELU
- In GRU:
  tanh()
  sigmoid()



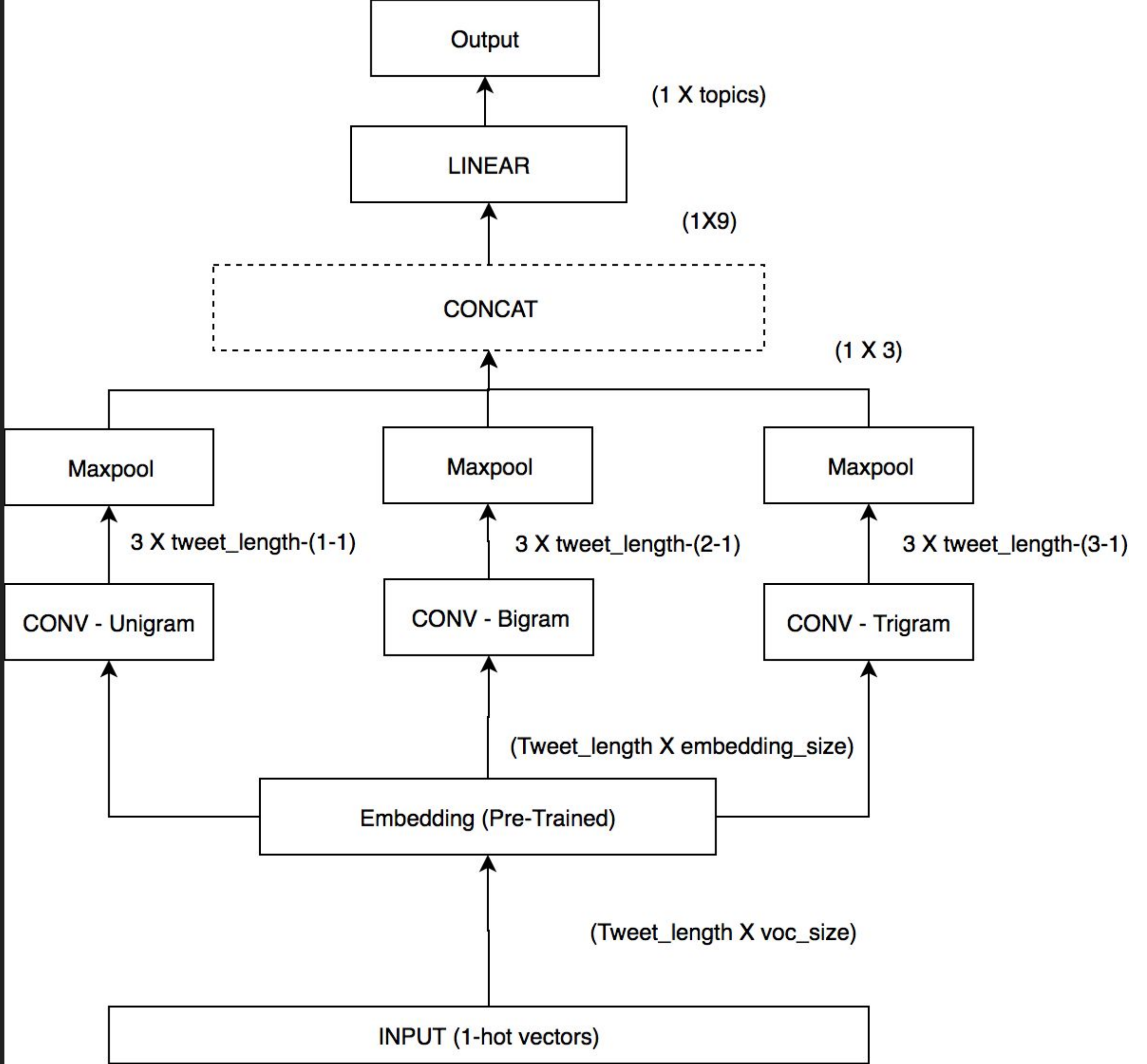
# Neural Network - Convolutional Neural Network

- Non-recurrent -> easier to train
- Filter with learnable parameters
- Can be thought of as a representation of n-grams



# CNN

- Uses n-grams
- Activation function: RELU



# Results

- Classification
- Confusion matrix
- Accuracy, Precision
- F1-score

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

# Naive Bayes - Most important features

Bag of Words

| Features | Classes |
|---|---|
| bitcoin = 1 | btc : airlin = 4429.9 : 1.0 |
| flight = 1 | airlin : stocks = 3896.2 : 1.0 |
| congress = 1 | electi : btc = 2596.7 : 1.0 |
| cancelled = 1 | airlin : btc = 1867.6 : 1.0 |
| senate = 1 | electi : btc = 1373.0 : 1.0 |
| candidate = 1 | GOPDeb : btc = 1354.3 : 1.0 |
| debate = 1 | GOPDeb : stocks = 1319.7 : 1.0 |
| jeb = 1 | GOPDeb : btc = 1142.6 : 1.0 |
| gopdebate = 1 | GOPDeb : electi = 941.7 : 1.0 |
| inc. = 1 | stocks : electi = 866.1 : 1.0 |

Bag of Bigrams

| Features | Classes |
|---|---|
| bitcoin = 1 | btc : airlin = 4429.9 : 1.0 |
| flight = 1 | airlin : stocks = 3896.2 : 1.0 |
| congress = 1 | electi : btc = 2596.7 : 1.0 |
| cancelled = 1 | airlin : btc = 1867.6 : 1.0 |
| flight PUNCTUATION = 1 | airlin : btc = 1376.6 : 1.0 |
| senate = 1 | electi : btc = 1373.0 : 1.0 |
| candidate = 1 | GOPDeb : btc = 1354.3 : 1.0 |
| debate = 1 | GOPDeb : stocks = 1319.7 : 1.0 |
| jeb = 1 | GOPDeb : btc = 1142.6 : 1.0 |
| flight UNKNOWN = 1 | airlin : btc = 1059.4 : 1.0 |

# Results - No common words removed

| Type | Accuracy | Avg. Recall | Avg. F1score | Avg. Precision |
|---|---|---|---|---|
| Bayes BOW | 95.25% | 95.66% | 94.82% | 94.16% |
| Bayes BOB | 95.40% | 96.02% | 94.77% | 93.79% |
| Neural GRU | **97.55%** | **97.48%** | **97.58%** | **97.70%** |
| Neural CNN | 97.01% | 96.98% | 97.11% | 97.23% |

# Results - 100 most common words removed

| Type | Accuracy | Avg. Recall | Avg. F1score | Avg. Precision |
| --- | --- | --- | --- | --- |
| Bayes BOW | 90.78% | 90.07% | 89.37% | 88.85% |
| Bayes BOB | 91.00% | 90.84% | 89.38% | 88.45% |
| Neural GRU | **93.28%** | **91.94%** | **91.92%** | **91.97%** |
| Neural CNN | 91.89% | 90.18% | 90.26% | 90.37% |

# Conclusions

- Naive Bayes - better than expected!
- Build more precise datasets
- Broader topics / Try other topics
- GRU and CNN - Many hyperparameters to tweak.
    - Backpropagation
    - Dropout
    - Number of layers
    - Hidden dimensions
    - etc