# Predicting Genres From Movie Summaries

Using IMDb as a Data Source

A CS5242 Project By:
A0254355A Niharika Shrivastava
A0250681H Sri Bhuvaneswari S
A0250635J Shreyas Sridhar

# Overview

- Problem Description
- Proposed Solution
- Data Collection
- Data Cleaning
- Data Visualisation
- Neural Networks Implemented
  - MLP
  - CNN
  - RNN
- Observations and Analysis
- Conclusions

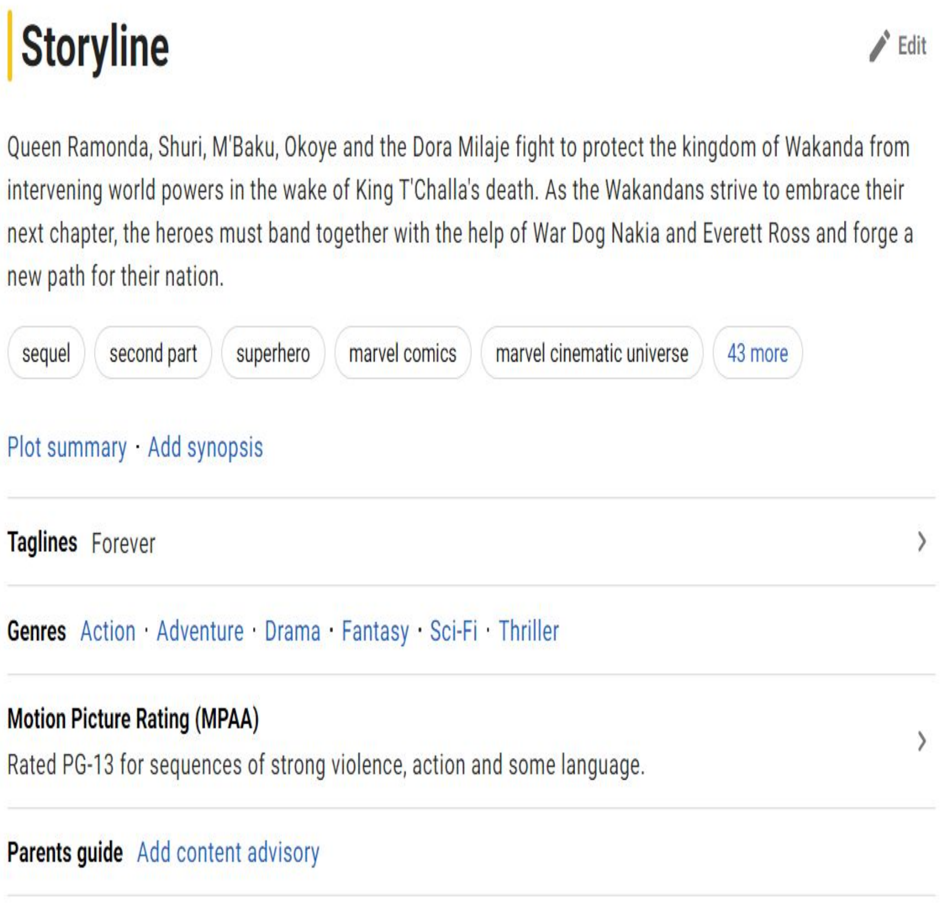
## Problem Description

- Movie genres play an important role both in determining the likelihood of a potential viewer to watch the movie and to find it interesting. It is not possible to figure out the genre of a movie from a simple summary.

- Therefore, we aim to develop a model which predicts the movie genre using IMDb movie summaries. It eliminates the manual process of finding genre tags. As a plot summary conveys much information about a movie, we explore different neural network models to predict the movie genre using movie summaries.

# An Overview of the Scraper

- The source of data for this project is IMDb.
- Custom scraper built using Beautiful Soup which makes use of HTML tags.
- Collect:
    - Movie name
    - Summary
    - Genres



Storyline

Edit

Queen Ramonda, Shuri, M'Baku, Okoye and the Dora Milaje fight to protect the kingdom of Wakanda from intervening world powers in the wake of King T'Challa's death. As the Wakandans strive to embrace their next chapter, the heroes must band together with the help of War Dog Nakia and Everett Ross and forge a new path for their nation.

sequel · second part · superhero · marvel comics · marvel cinematic universe · 43 more

Plot summary · Add synopsis

Taglines Forever

Genres Action · Adventure · Drama · Fantasy · Sci-Fi · Thriller

Motion Picture Rating (MPAA)

Rated PG-13 for sequences of strong violence, action and some language.

Parents guide Add content advisory

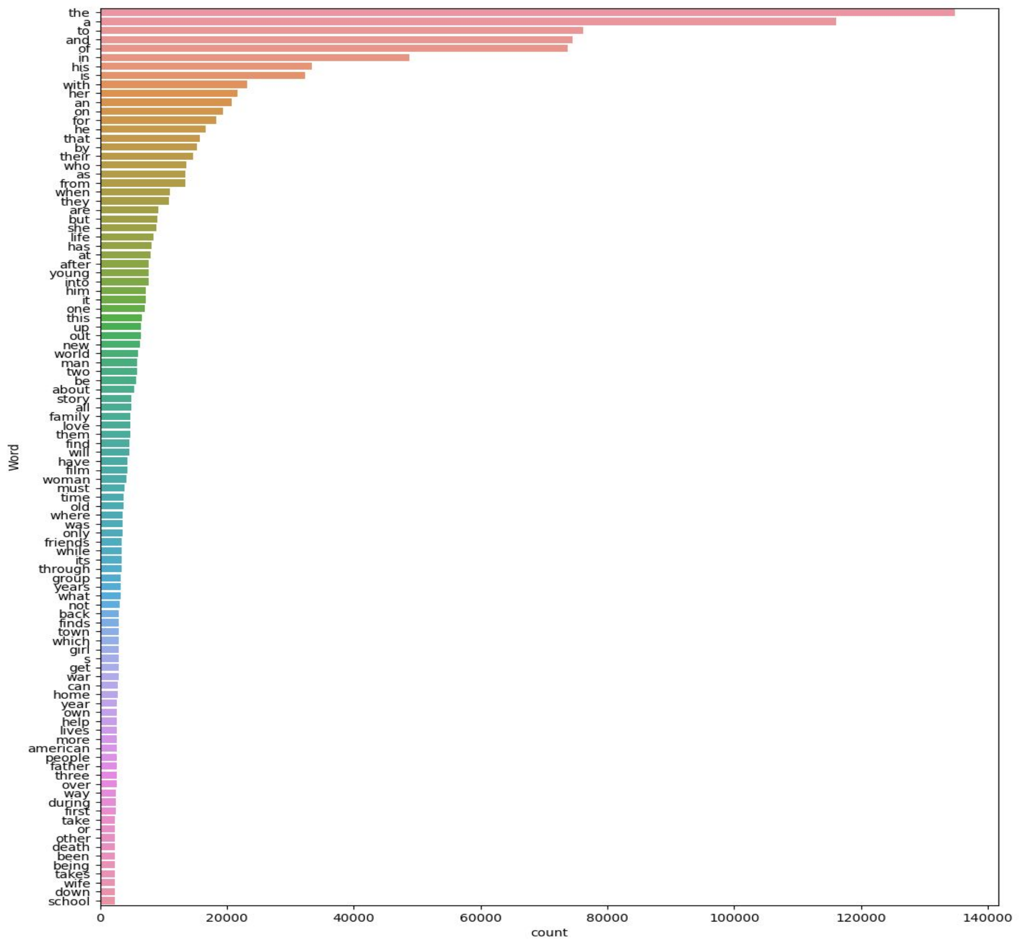
## Data Cleaning and Preprocessing

- Remove special characters and whitespaces
- Convert summary into lowercase
- Delete duplicates
- Delete movies without a plot (they have ‘Add a Plot’ section instead)
- Stemming
- Remove all stopwords from the summary

# Most Frequent Words in Summaries

- We plot the frequency of the most frequently occuring words in the summaries.

- We observe that:
- Words such as the, a, and, to, etc.(stopwords) occupy approximately the first 50 most frequent words.
- The word-frequency distribution is heavily skewed due to these stopwords. Therefore, a model trained on these summaries will give more importance to them, over actual genre-defining words. This will lead to noisy predictions

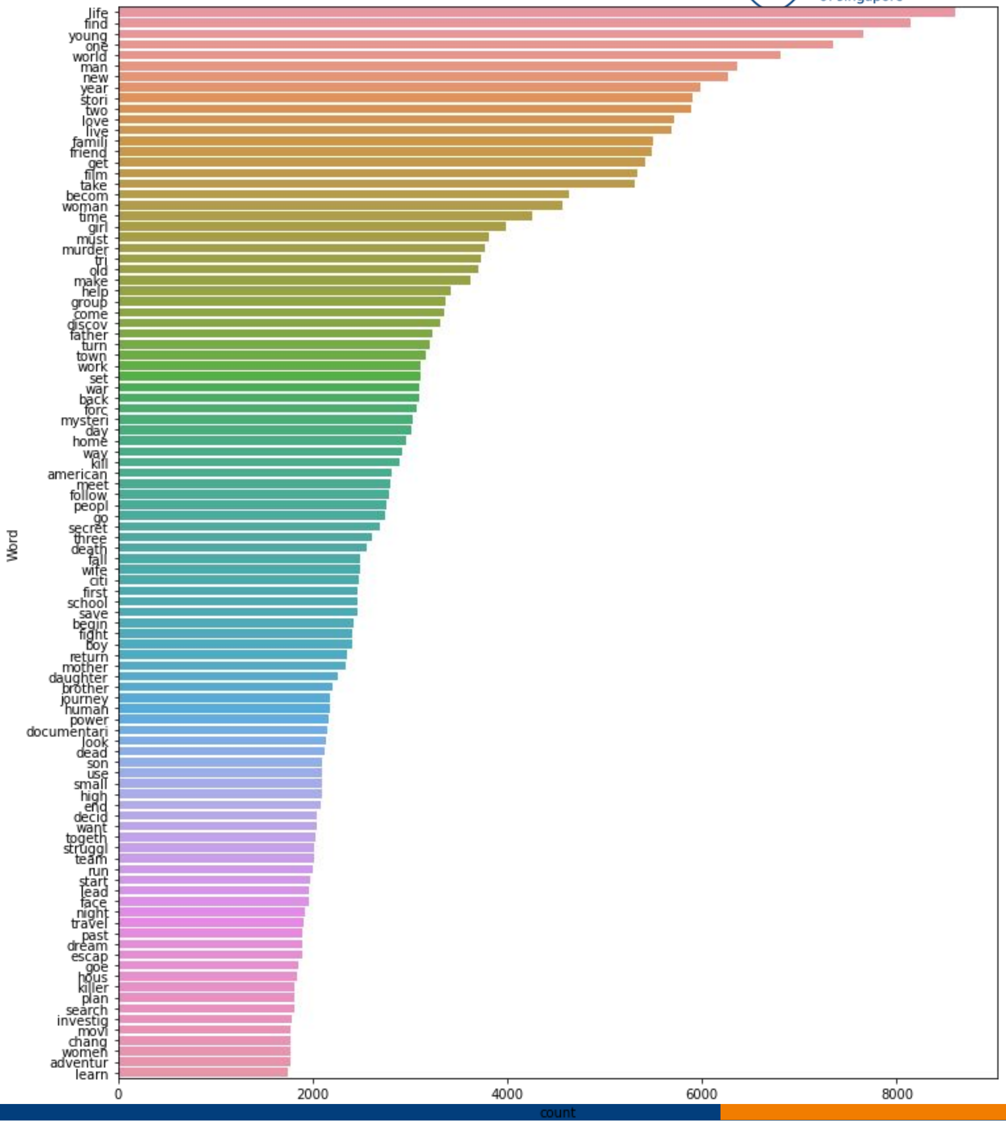


# Data Visualisation

1. After removal of stopwords, more significant words such as **life, world, love, family, documentary** will get attention from the models to predict genre.

2. Each word is stemmed into their base-form to standardize the data without losing its overall meaning. This reduces noisiness (unnecessary variance) in the data even when the meaning is essentially the same. For e.g.:
   - **takes** and **taken** becomes **take**.
   - **running**, **run**, **ran** becomes **run**.

3. The frequency distribution is highly uniform after the first top 10 words, thereby reducing bias.

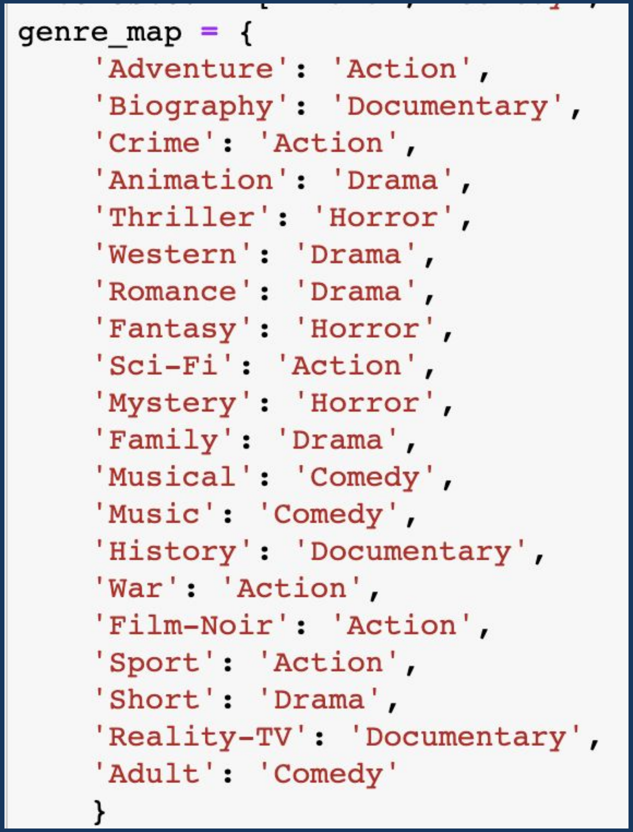


## Data Visualisation (cont.)

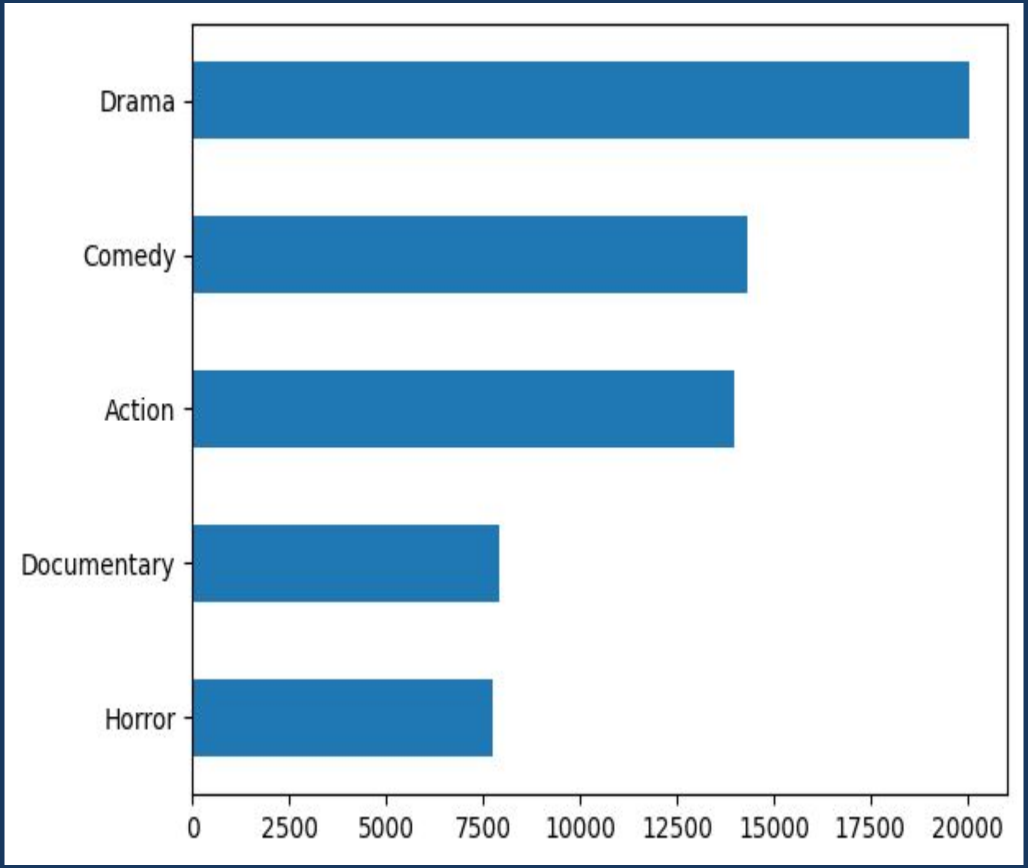
- Each summary is of variable length.
- The maximum number of words in a summary is 612, whereas 75% of the summaries are 23 words or less.
- Hence, we can safely assume ~25 features in each summary without losing generality.

| | |
|---|---|
| count | 64014.000000 |
| mean | 22.236042 |
| std | 20.181776 |
| min | 1.000000 |
| 25% | 12.000000 |
| 50% | 17.000000 |
| 75% | 23.000000 |
| max | 612.000000 |

# Data Visualisation

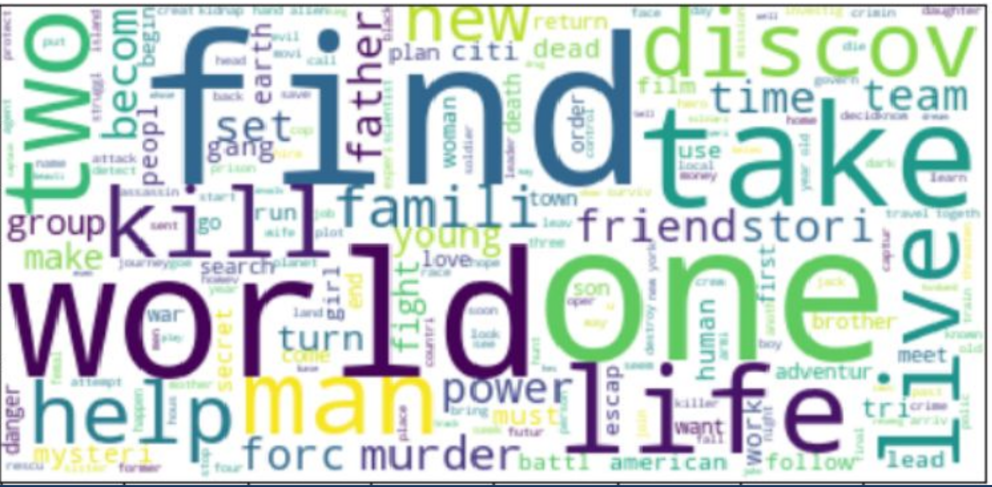


```
genre_map = {
    'Adventure': 'Action',
    'Biography': 'Documentary',
    'Crime': 'Action',
    'Animation': 'Drama',
    'Thriller': 'Horror',
    'Western': 'Drama',
    'Romance': 'Drama',
    'Fantasy': 'Horror',
    'Sci-Fi': 'Action',
    'Mystery': 'Horror',
    'Family': 'Drama',
    'Musical': 'Comedy',
    'Music': 'Comedy',
    'History': 'Documentary',
    'War': 'Action',
    'Film-Noir': 'Action',
    'Sport': 'Action',
    'Short': 'Drama',
    'Reality-TV': 'Documentary',
    'Adult': 'Comedy'
    }
```



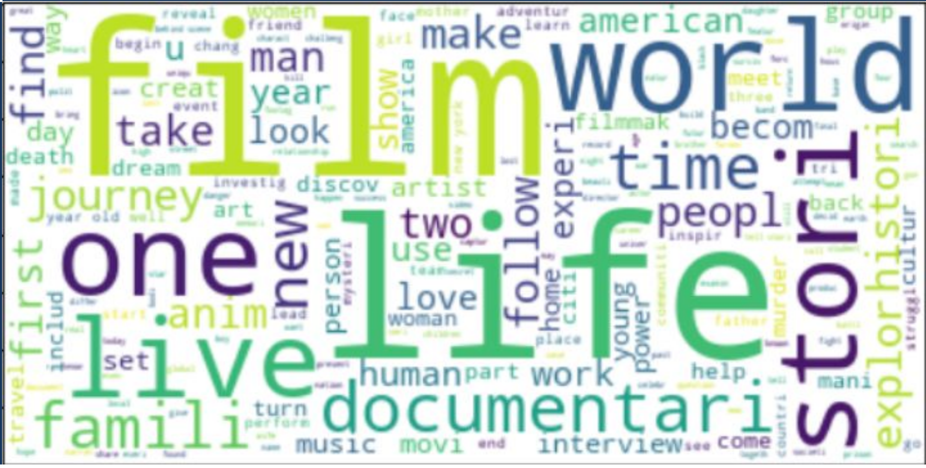
CS5242 - Predicting Genres From Movie Summaries

# Data Visualisation (cont.)



Action

Documentary

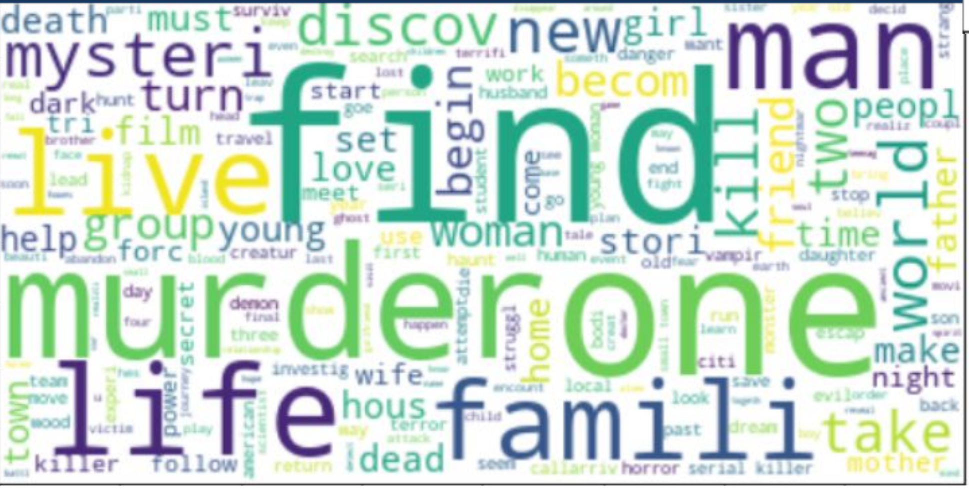


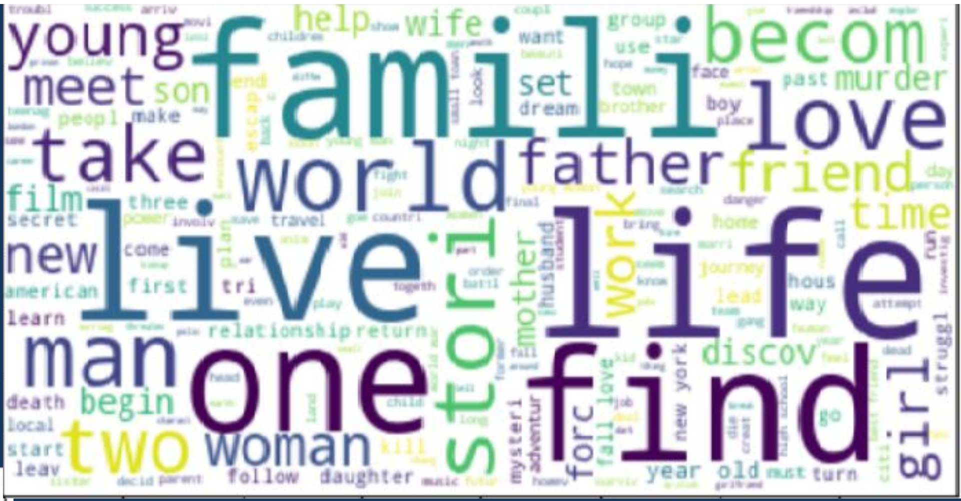
Most Common Words by Genre

# Data Visualisation (cont.)

## Most Common Words by Genre

Horror

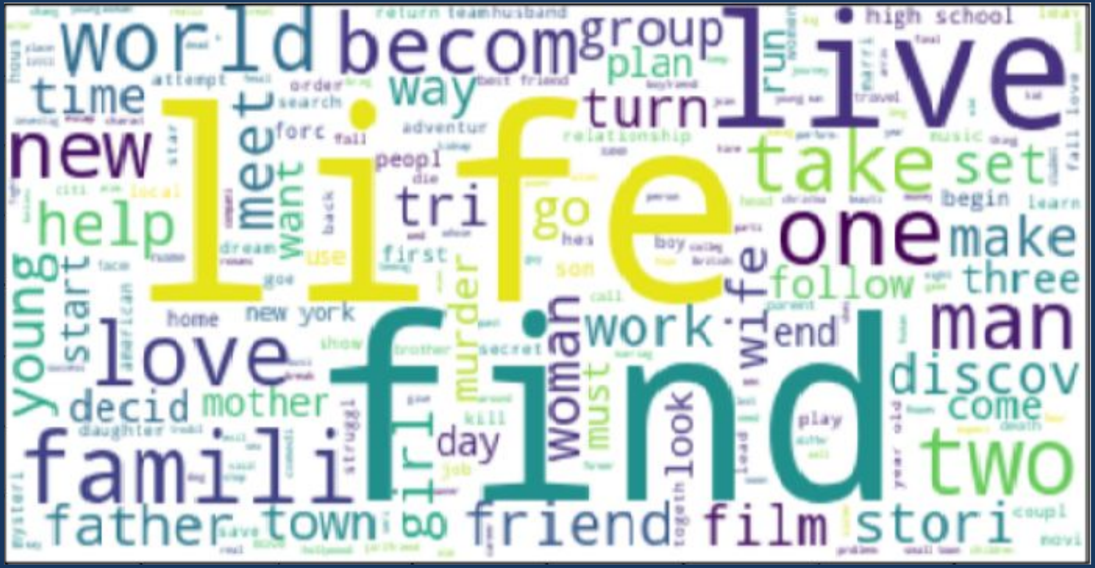




Drama

## Data Visualisation (cont.)

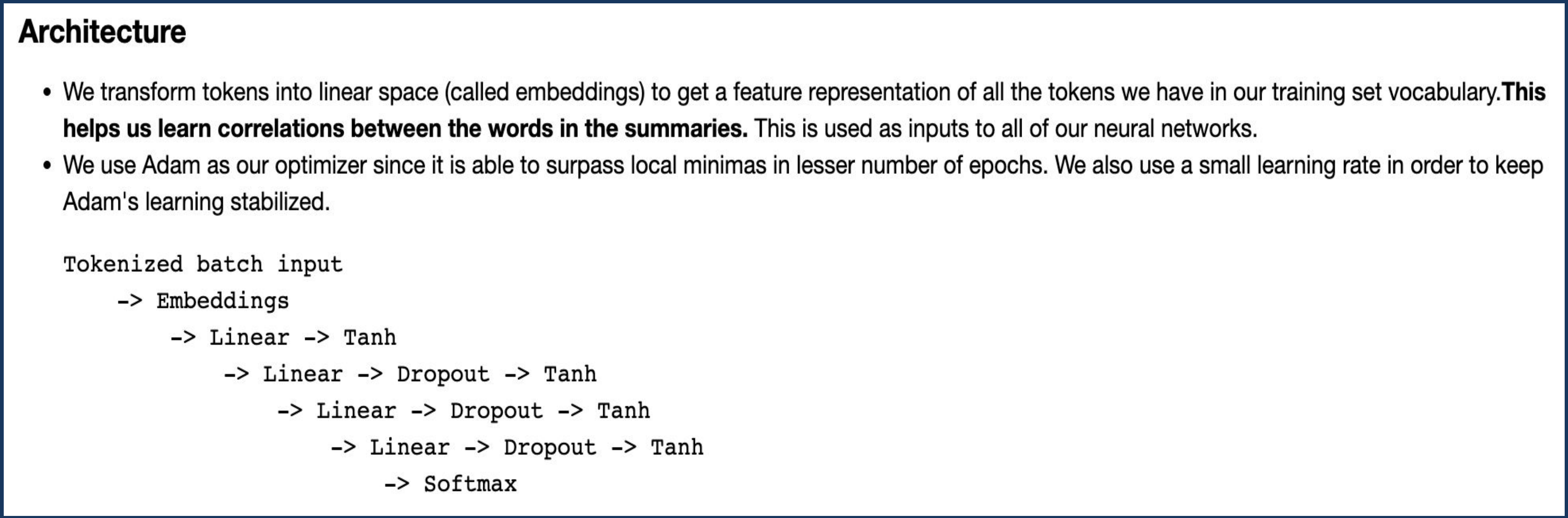
Comedy



Most Common Words by Genre

## Neural Network Architectures Implemented

# Multi Layer Perceptron



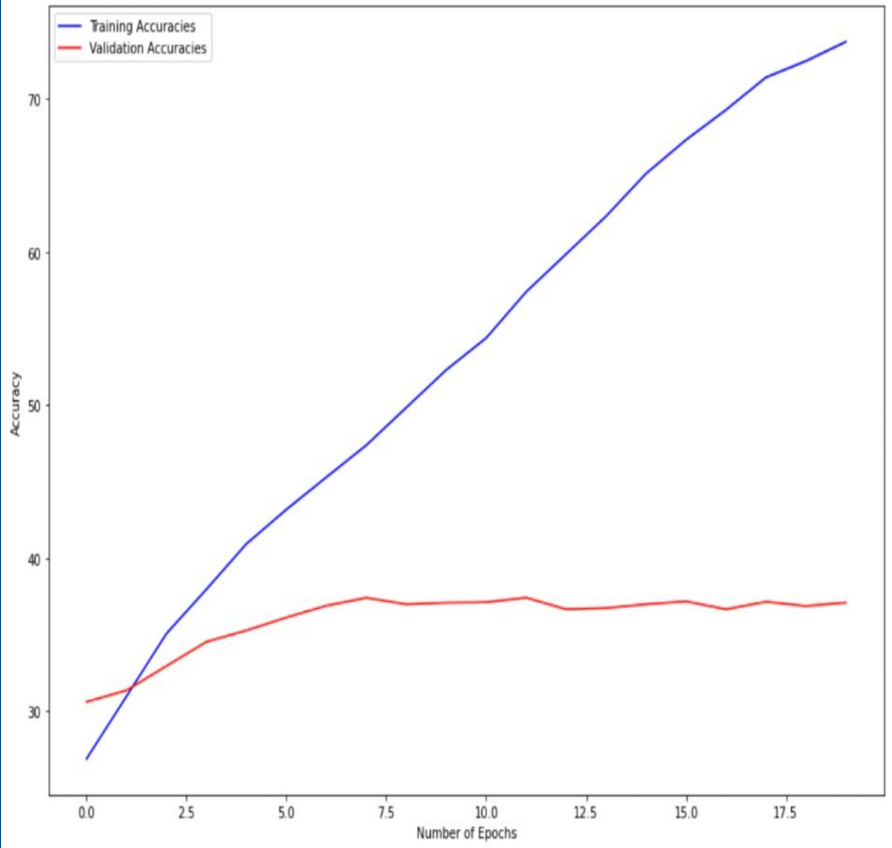
## Architecture

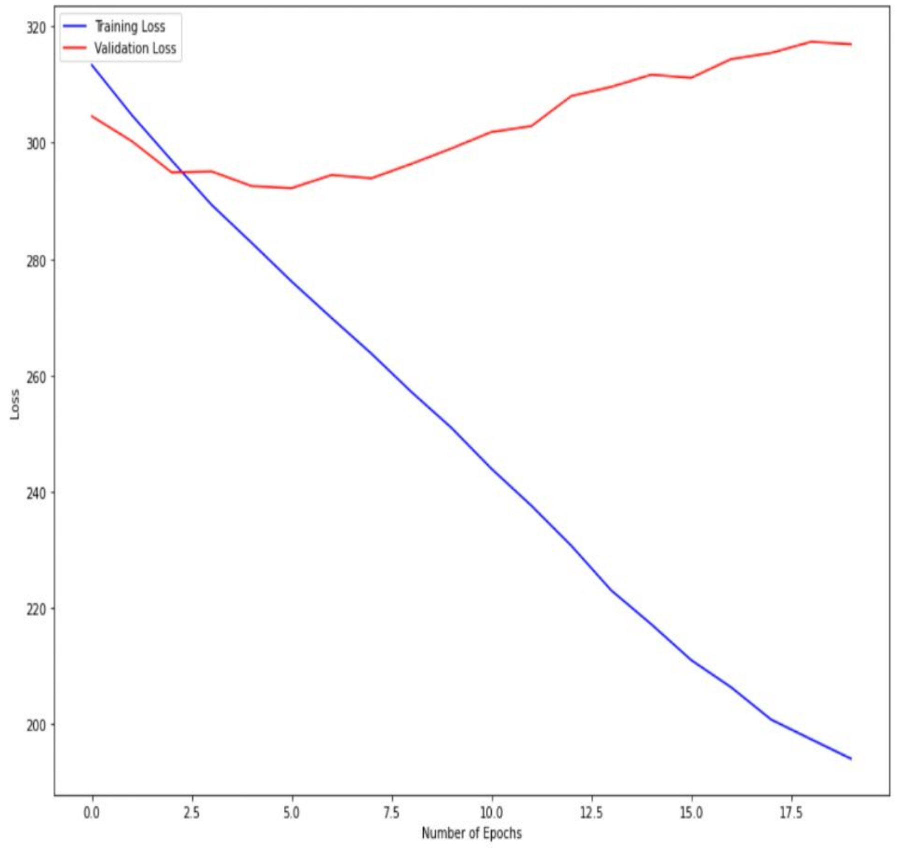
- We transform tokens into linear space (called embeddings) to get a feature representation of all the tokens we have in our training set vocabulary.**This helps us learn correlations between the words in the summaries.** This is used as inputs to all of our neural networks.
- We use Adam as our optimizer since it is able to surpass local minimas in lesser number of epochs. We also use a small learning rate in order to keep Adam's learning stabilized.

```
Tokenized batch input
    -> Embeddings
        -> Linear -> Tanh
            -> Linear -> Dropout -> Tanh
                -> Linear -> Dropout -> Tanh
                    -> Linear -> Dropout -> Tanh
                        -> Softmax
```

MLP for text analysis

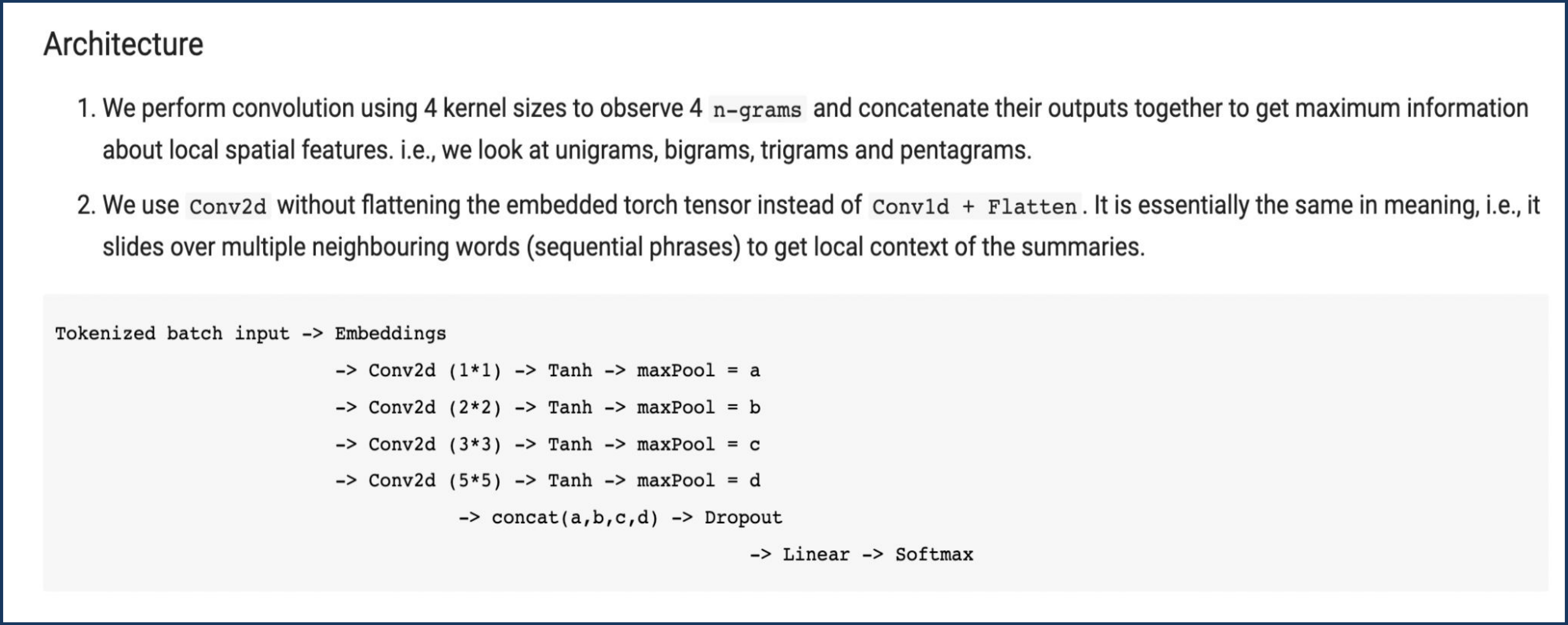
# MLP Model Metrics





Training Values are in Blue and Validation Values are in Red

# Convolutional Neural Network



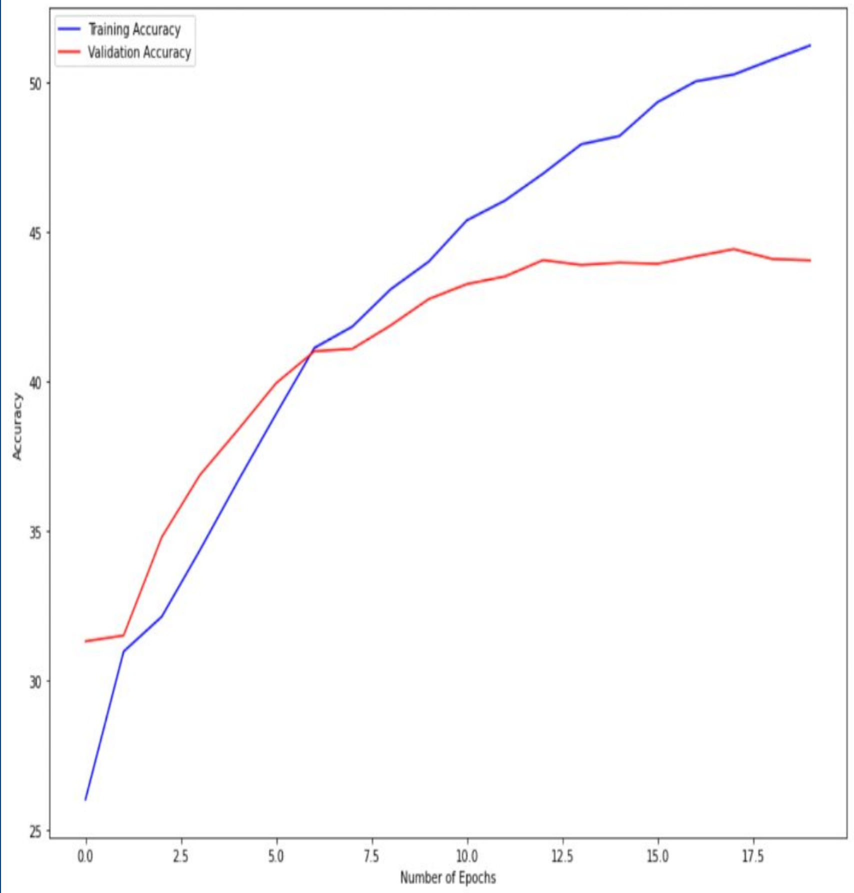
## Architecture

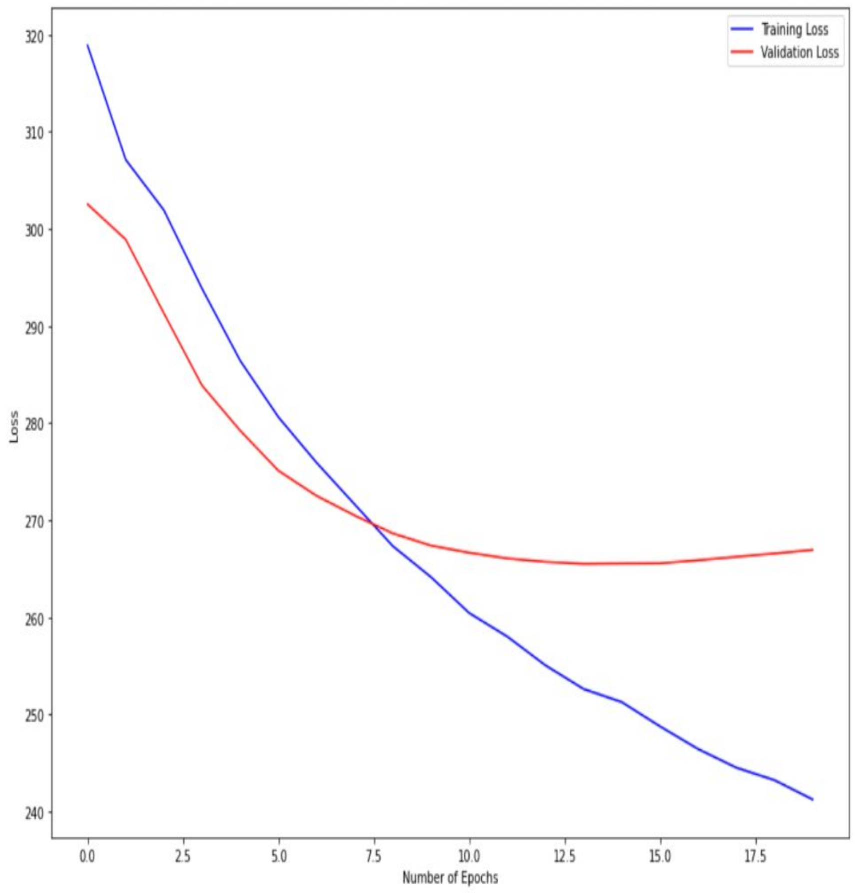
1. We perform convolution using 4 kernel sizes to observe 4 `n-grams` and concatenate their outputs together to get maximum information about local spatial features. i.e., we look at unigrams, bigrams, trigrams and pentagrams.
2. We use `Conv2d` without flattening the embedded torch tensor instead of `Conv1d + Flatten`. It is essentially the same in meaning, i.e., it slides over multiple neighbouring words (sequential phrases) to get local context of the summaries.

```
Tokenized batch input -> Embeddings
                         -> Conv2d (1*1) -> Tanh -> maxPool = a
                         -> Conv2d (2*2) -> Tanh -> maxPool = b
                         -> Conv2d (3*3) -> Tanh -> maxPool = c
                         -> Conv2d (5*5) -> Tanh -> maxPool = d
                                    -> concat(a,b,c,d) -> Dropout
                                                          -> Linear -> Softmax
```

CNN for text analysis

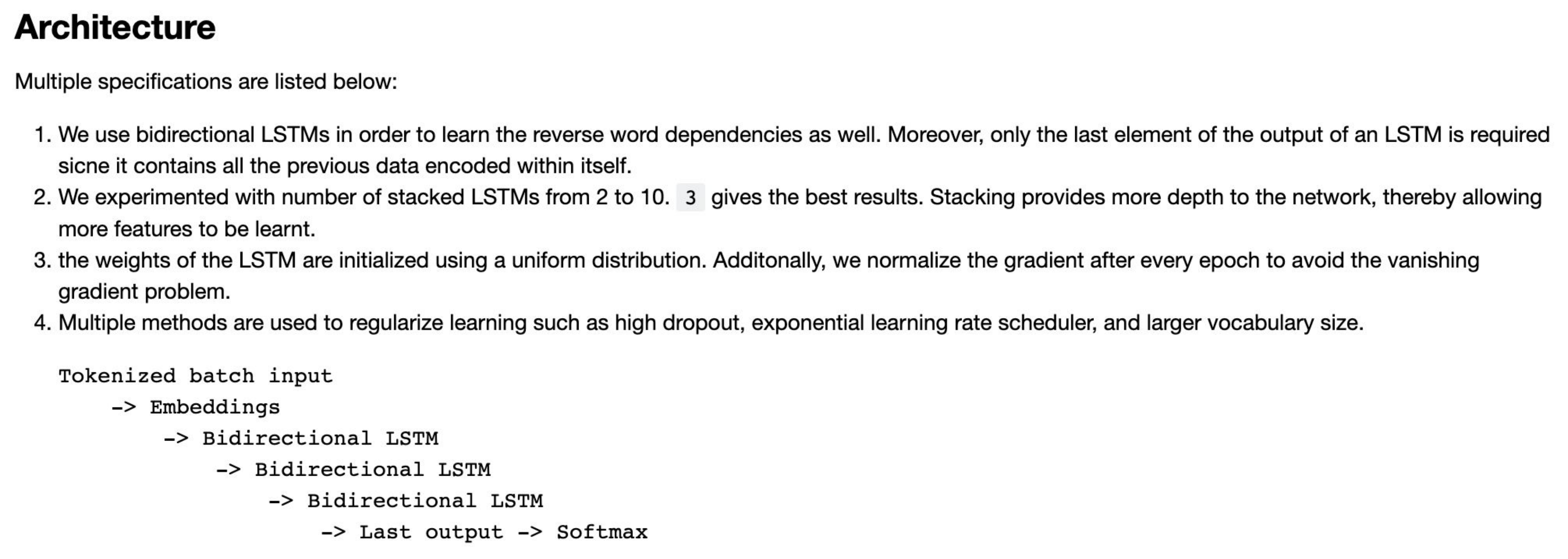
## CNN Model Metrics





Training Values are in Blue and Validation Values are in Red

# Recurrent Neural Network



## Architecture

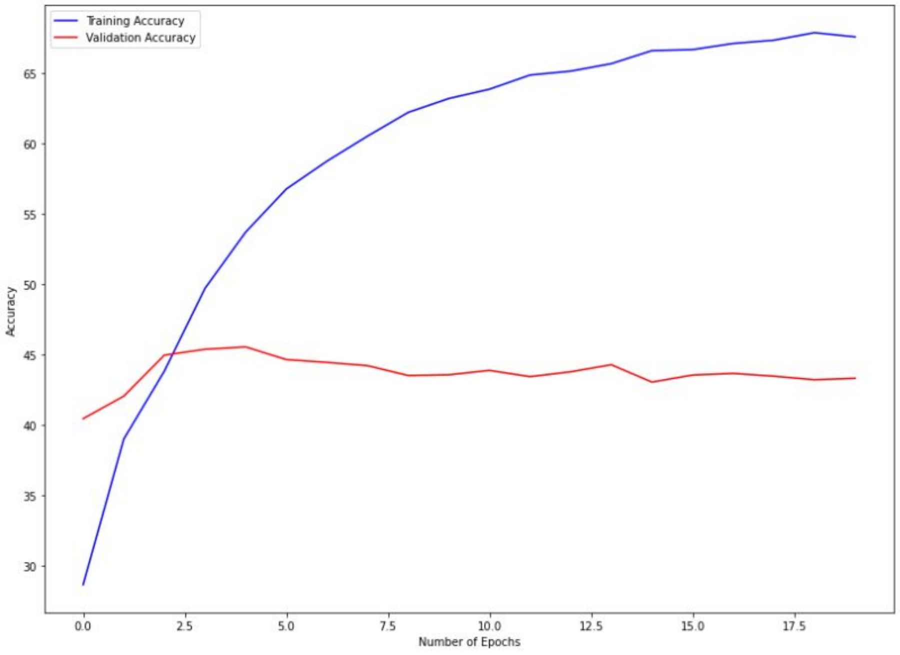
Multiple specifications are listed below:

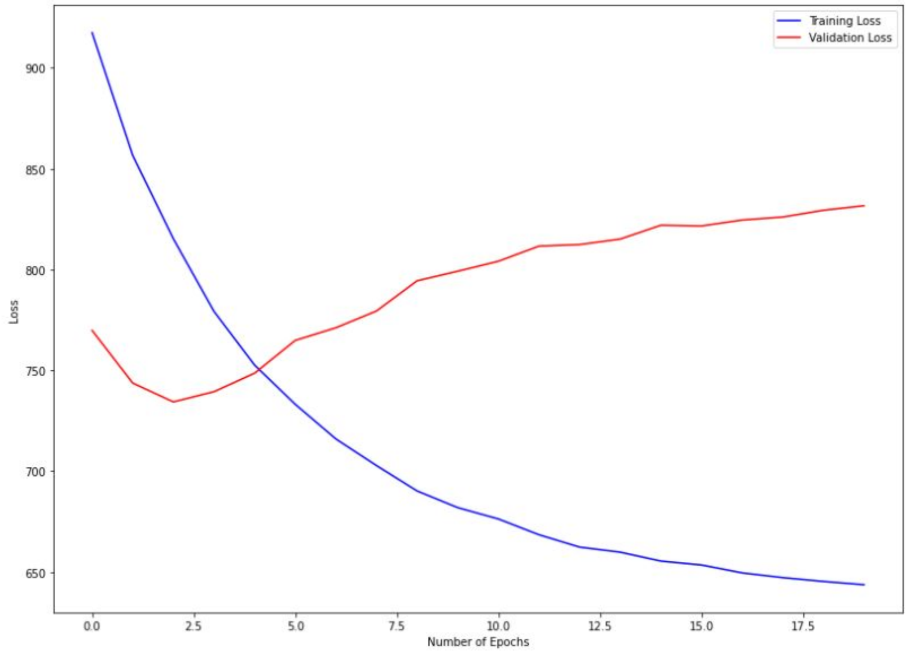
1. We use bidirectional LSTMs in order to learn the reverse word dependencies as well. Moreover, only the last element of the output of an LSTM is required sicne it contains all the previous data encoded within itself.
2. We experimented with number of stacked LSTMs from 2 to 10. `3` gives the best results. Stacking provides more depth to the network, thereby allowing more features to be learnt.
3. the weights of the LSTM are initialized using a uniform distribution. Additonally, we normalize the gradient after every epoch to avoid the vanishing gradient problem.
4. Multiple methods are used to regularize learning such as high dropout, exponential learning rate scheduler, and larger vocabulary size.

```
Tokenized batch input
    -> Embeddings
        -> Bidirectional LSTM
            -> Bidirectional LSTM
                -> Bidirectional LSTM
                    -> Last output -> Softmax
```

RNN for text analysis

# RNN Model Metrics





Training Values are in Blue and Validation Values are in Red

## Observations and Results

# Architecture Comparison (FILL)

| Multi Layer Perceptron | Convolutional Neural Network | Recurrent Neural Network |
|---|---|---|
| 37% Accuracy | 44% Accuracy | 45% Accuracy |
| Training Time: 33 s | Training Time: 34 s | Training Time: 1 m 45 s |
| Overfitting: High | Overfitting: Low | Overfitting: High |

## Conclusions

- The Convolutional Neural Network model seems to be the best fit for this problem as shown by the model's accuracy and its aversion towards overfitting.
- Some reasons why CNN models work better are:
  - They process the words as morphemes
  - They consider groups of words in close vicinity
  - They try to extract features from the summaries in terms of word groups and morpheme groups
- This allows the CNN model to better understand the relationship between summaries and their genres.
- The MLP model is not capable of learning the deeper connections between the words and the summaries, and is, therefore, not as powerful.
- The RNN tries to find the correlation within the sequence of the words in the summaries, which is not as important in our problem, since we are trying to classify the text rather than understand the relationship between consecutive words.

## Further Improvements (FILL)

| | Neural Network | Shortcomings and Possible Enhancements |
|---|---|---|
| 1 | MLP | ● Fast learning<br>● Tends to overfit<br>● Requires improvement through generalisation |
| 2 | CNN | ● Low parameter count, moderately fast learning<br>● Doesn't tend to overfit in this case<br>● Can be improved with more data, resources and time |
| 3 | RNN | ● Slow learning<br>● Highly overfits in this case<br>○ Due to lesser importance of the sequence of words in the summary |

NUS
National University of Singapore

# Thank You