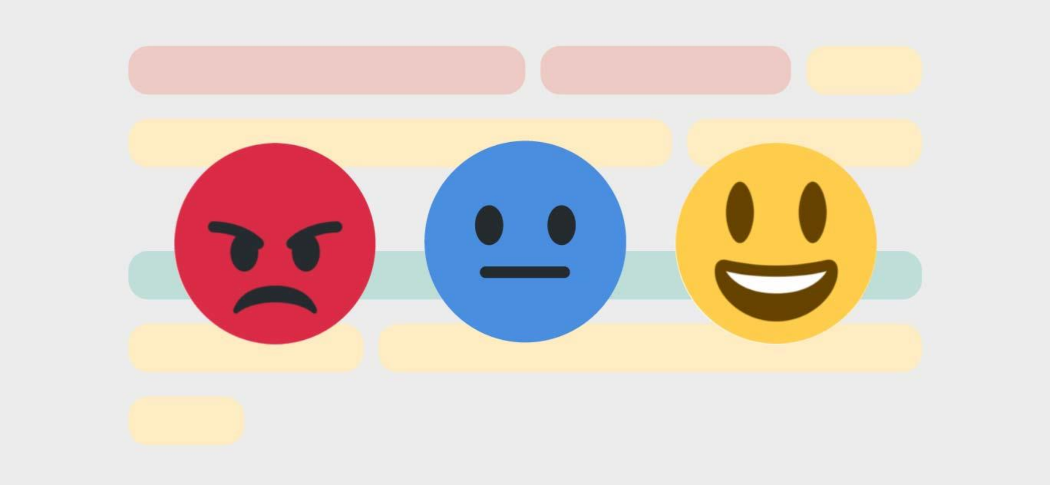
# Project

Text Prediction on Movie Reviews

Jason Maloney jpmalone@syr.edu,
Bing-Je Wu bwu117@syr.edu,
Maya Mileva mileva@syr.edu,
Antonio Llorens allloren@syr.edu
IST 736 – Text Mining

# Table of Contents

## Introduction



The popularity of online reviews is the natural consequence of the digital age. For the film industry, an online review of critical audiences plays an important role. On the one hand, the good comments of a movie can attract more audiences in general. On the other hand, good comments do not necessarily mean high box revenue and vice versa.

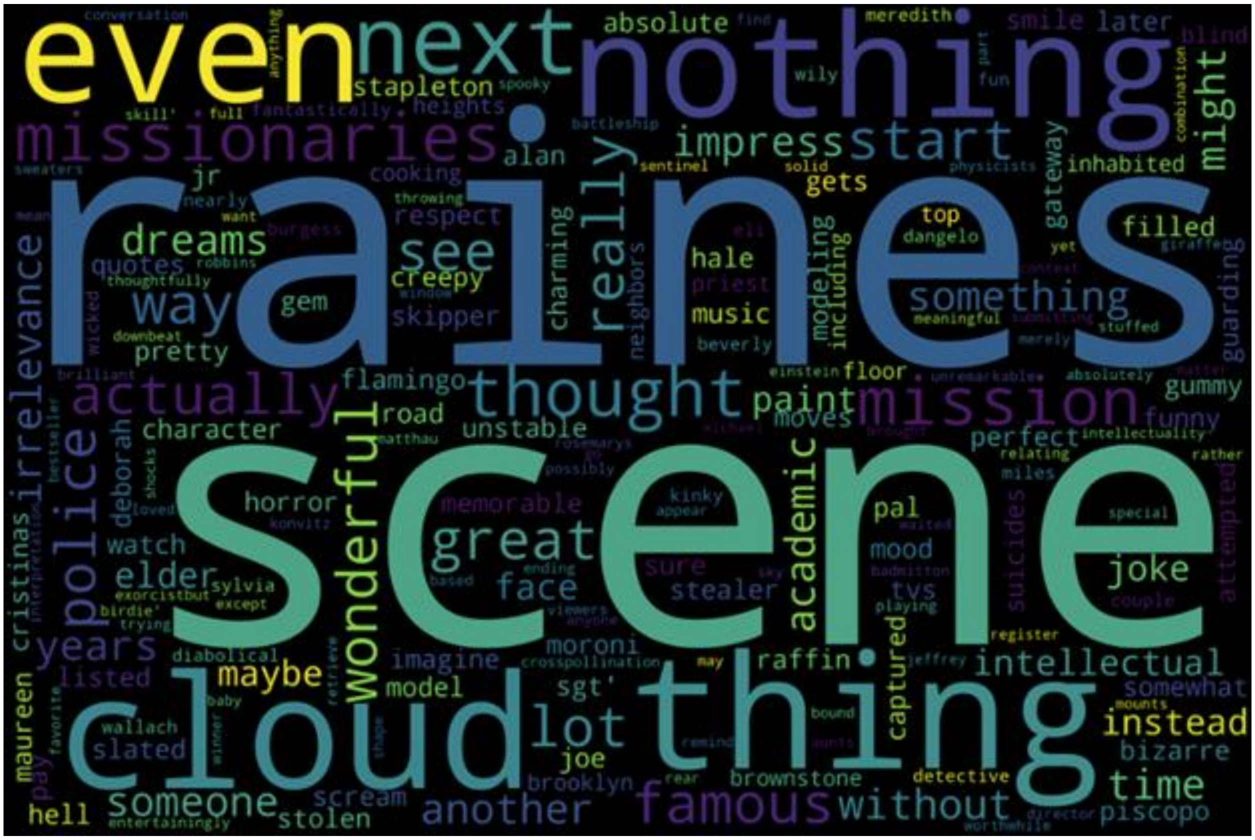
Several websites allow Internet users to submit movie reviews and aggregate them into an average. Community-driven review sites have allowed the typical movie goer to express their opinion on films. On these online review sites, users generally only have to register with the website to submit reviews. This means that they are a form of open access poll, and have the same advantages and disadvantages; notably, there is no guarantee that they will be a representative sample of the film's audience. In some cases, online review sites have produced wildly differing results to the scientific polling of audiences. Likewise, reviews and ratings for many movies can significantly differ between the different review sites, even though certain movies are well-rated (or poorly-rated) across the board.

The Internet Movie Database (IMDb) is a vast repository for image and text data, an excellent source for data analytics and deep learning practice and research. The data is partially related to people, as ratings are crowdsourced opinions. IMDb is the world’s most popular and authoritative source for movie, TV, and celebrity information. Watch trailers, get showtimes, and buy tickets for upcoming films.

Ratings can reflect how the public view the qualities of movies. However, there are certain groups of people who are more likely to rate. For example, people who like or dislike a particular movie or casting are more likely to rate to express their strong opinions. People who are movie lovers are also more likely to rate but in a more objective way. Whether people like it or not, movie reviews are becoming a prominent factor affecting people’s decision to see them in theaters, rent, or purchase.

Customer reviews are a great source of "Voice of the customer" and could offer tremendous insights into what customers like and dislike about a product or service. In other words, customer reviews influence people's booking decisions, which means movie producers should better pay attention to what people are saying about their films. Not only they want good reviews, but also in a way that can help them learn the most about your customers. Reviews can tell film producers if they are keeping up with your customers’ expectations, which is crucial for developing marketing strategies based on the personas of your customers.

However, in reality, it is often not straightforward to determine the proper sentiment or categorization of customer reviews as positive, negative, or neutral. Sentiment Analysis is the task of detecting the tonality of a text. A typical setting aims to categorize a document as "positive," "negative," or "neutral." Some reviews could contain both positive and negative statements at the same time. For instance, the word cloud included is an excellent example of the words used by reviewers and the potential requirements for sentiment analysis.



A key challenge in the area of affective computing is the annotation of emotional labels that describe the underlying expressive behaviors observed during human interactions. The analysis will evaluate the movie reviews provided in the IMDB data set as an input to develop sentiment analysis and assess performance as a positive or negative classifier model. Further tasks such as having a model knowing the next word will be provided and combined with sentiment analysis for potential AI solutions.

## Analysis and Models

The project consists of two parts, the sentiment classification task, and the next word prediction task. For the sentiment classification task, several Machine Learning algorithms, including supervised learning and unsupervised learning, were implemented for building models and having analysis. For the text prediction task, the n-gram models were built for predicting the next word. Models will be evaluated and one of the models will be selected as the best model to combine with the language model to build a web application as an AI solution.

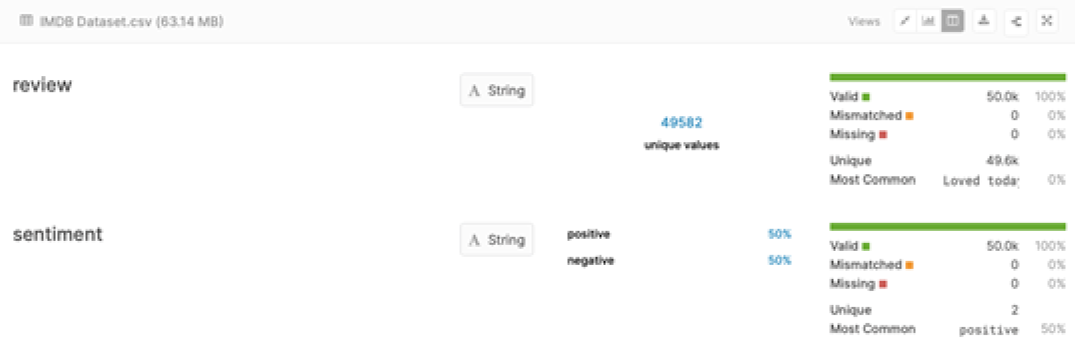
### About the Data

The Large Movie Review Dataset (from Stanford University) consists of 50,000 movie reviews (50% negative and 50% positive). The set is divided into training and validation datasets (each with 25000 movie reviews with an equal number of positive and negative reviews).

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. The Large Movie Review Dataset (often referred to as the IMDB dataset) contains 25,000 highly polar moving reviews (good or bad) for training and the same amount again for testing. The problem is to determine whether a given moving review has a positive or negative sentiment. The data was collected by Stanford researchers and was used in a 2011 paper ( Learning Word Vectors for Sentiment Analysis by Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts) where a split of 50/50 of the data was used for training and test. An accuracy of 88.89% was achieved. The data was also used as the basis for a Kaggle competition titled "Bag of Words Meets Bags of Popcorn" in late 2014 to early 2015. Accuracy was achieved above 97%, with winners achieving 99%.

For more dataset information, please go through the following link, http://ai.stanford.edu/~amaas/data/sentiment/

**Figure 1.** *Data set example*



IMDB Dataset.csv (63.14 MB)

review — String — 49582 unique values

Valid 🟩 50.0k 100%
Mismatched 🟧 0 0%
Missing 🟥 0 0%
Unique 49.6k
Most Common Loved toda; 0%

sentiment — String — positive 50%, negative 50%

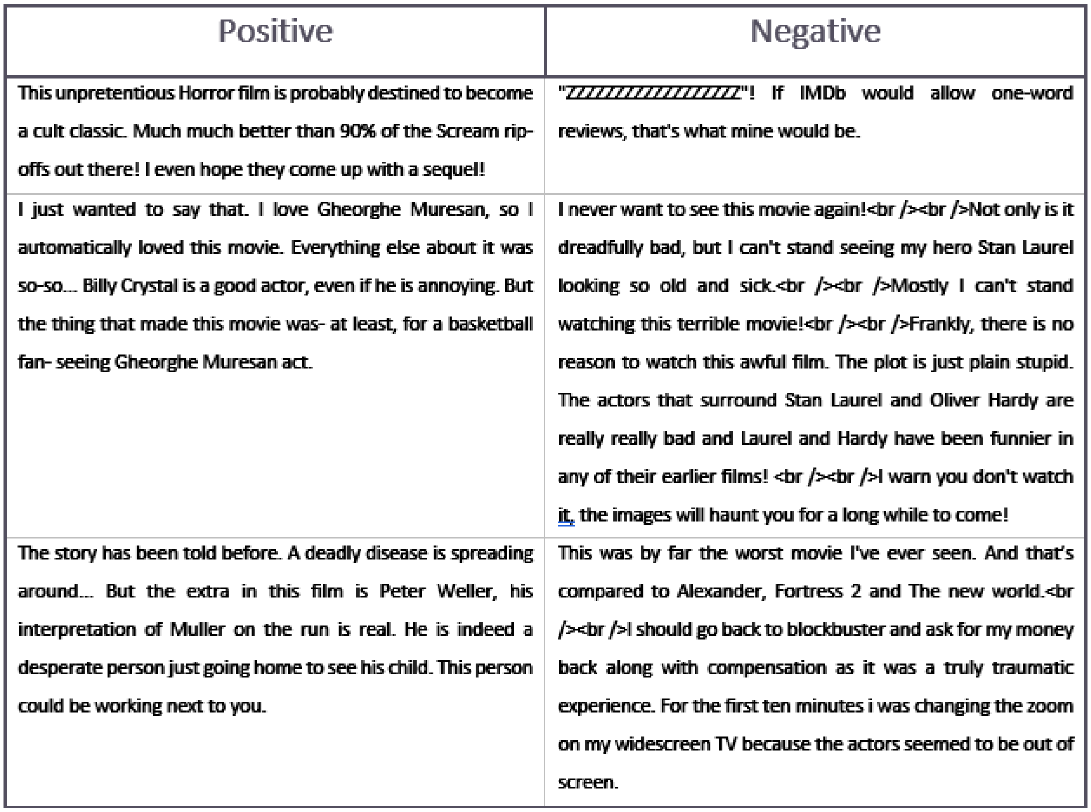
Valid 🟩 50.0k 100%
Mismatched 🟧 0 0%
Missing 🟥 0 0%
Unique 2
Most Common positive 50%

## Getting Data

The original dataset was stored as tar.gz file. To load data to the python program, the command, “gunzip -c aclImdb_v1.tar.gz | tar xopf -”, was used through a terminal to unzip and unpack the tar.gz file. Corpora can be accessed through the folder unpacked from the tar.gz file. A train.csv file and test.csv file were also created through ‘glob’ and ‘pandas’ libraries.

The train and test sets were combined for easier preprocessing. A sample of the original positive and negative reviews is shown in Figure 2 below:

**Figure 2.** *Sample of the movie reviews*



| Positive | Negative |
|---|---|
| This unpretentious Horror film is probably destined to become a cult classic. Much much better than 90% of the Scream rip-offs out there! I even hope they come up with a sequel! | "ZZZZZZZZZZZZZZZZZZZZZZZZ"! If IMDb would allow one-word reviews, that's what mine would be. |
| I just wanted to say that. I love Gheorghe Muresan, so I automatically loved this movie. Everything else about it was so-so... Billy Crystal is a good actor, even if he is annoying. But the thing that made this movie was- at least, for a basketball fan- seeing Gheorghe Muresan act. | I never want to see this movie again!<br /><br />Not only is it dreadfully bad, but I can't stand seeing my hero Stan Laurel looking so old and sick.<br /><br />Mostly I can't stand watching this terrible movie!<br /><br />Frankly, there is no reason to watch this awful film. The plot is just plain stupid. The actors that surround Stan Laurel and Oliver Hardy are really really bad and Laurel and Hardy have been funnier in any of their earlier films! <br /><br />I warn you don't watch it, the images will haunt you for a long while to come! |
| The story has been told before. A deadly disease is spreading around... But the extra in this film is Peter Weller, his interpretation of Muller on the run is real. He is indeed a desperate person just going home to see his child. This person could be working next to you. | This was by far the worst movie I've ever seen. And that’s compared to Alexander, Fortress 2 and The new world.<br /><br />I should go back to blockbuster and ask for my money back along with compensation as it was a truly traumatic experience. For the first ten minutes i was changing the zoom on my widescreen TV because the actors seemed to be out of screen. |

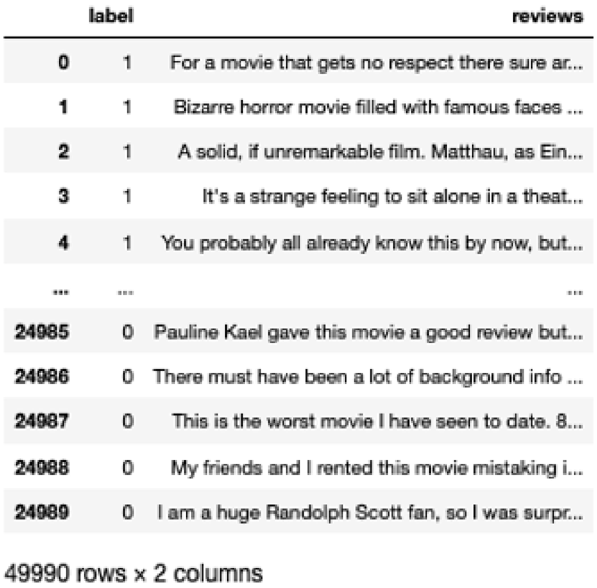
## Sentiment Analysis of IMDB Movie Reviews

Word representations are a critical component of many natural language processing systems. It is common to represent words as indices in vocabulary, but this fails to capture the lexicon's rich relational structure. Vector-based models do much better in this regard. They encode continuous similarities between words as distance or angle between word vectors in a high-dimensional space.

## Exploratory Data Analysis (EDA)

Figure 3 is an example of the data frame with the Label and review columns. Label 0 is negative, and label one is positive. Label distribution is balanced 50% positive and 50% negatives. Figure 4 is a box plot that shows the central tendency measures of the length of reviews. The average review length is over 1 thousand words, with a standard deviation of over nine hundred. This means that some reviews could be over 3 thousand words or others less than one hundred words.

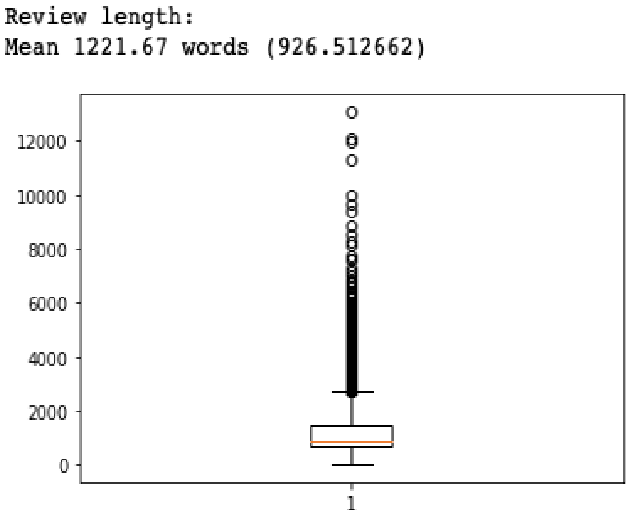
**Figure 3.** *Sample of a data frame structure*



| | label | reviews |
|---|---|---|
| **0** | 1 | For a movie that gets no respect there sure ar... |
| **1** | 1 | Bizarre horror movie filled with famous faces ... |
| **2** | 1 | A solid, if unremarkable film. Matthau, as Ein... |
| **3** | 1 | It's a strange feeling to sit alone in a theat... |
| **4** | 1 | You probably all already know this by now, but... |
| **...** | ... | ... |
| **24985** | 0 | Pauline Kael gave this movie a good review but... |
| **24986** | 0 | There must have been a lot of background info ... |
| **24987** | 0 | This is the worst movie I have seen to date. 8... |
| **24988** | 0 | My friends and I rented this movie mistaking i... |
| **24989** | 0 | I am a huge Randolph Scott fan, so I was surpr... |

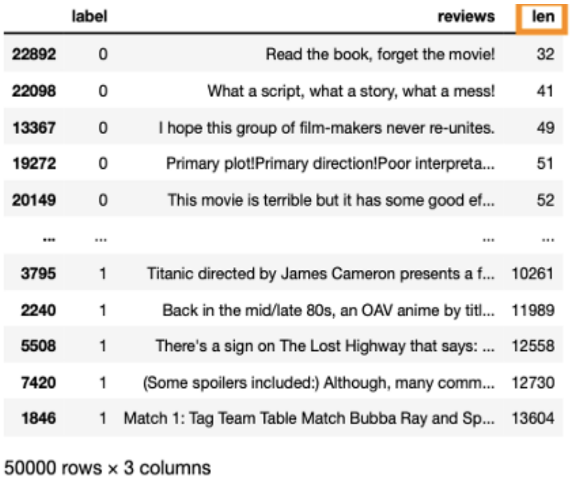
49990 rows × 2 columns

**Figure 4.** *Average review length*



```
Review length:
Mean 1221.67 words (926.512662)
```

**Figure 5.** *Ascending order of review lengths*



| | label | reviews | len |
|---|---|---|---|
| **22892** | 0 | Read the book, forget the movie! | 32 |
| **22098** | 0 | What a script, what a story, what a mess! | 41 |
| **13367** | 0 | I hope this group of film-makers never re-unites. | 49 |
| **19272** | 0 | Primary plot!Primary direction!Poor interpreta... | 51 |
| **20149** | 0 | This movie is terrible but it has some good ef... | 52 |
| **...** | ... | ... | ... |
| **3795** | 1 | Titanic directed by James Cameron presents a f... | 10261 |
| **2240** | 1 | Back in the mid/late 80s, an OAV anime by titl... | 11989 |
| **5508** | 1 | There's a sign on The Lost Highway that says: ... | 12558 |
| **7420** | 1 | (Some spoilers included:) Although, many comm... | 12730 |
| **1846** | 1 | Match 1: Tag Team Table Match Bubba Ray and Sp... | 13604 |

50000 rows × 3 columns

**Figure 6.** *Top 20 words*

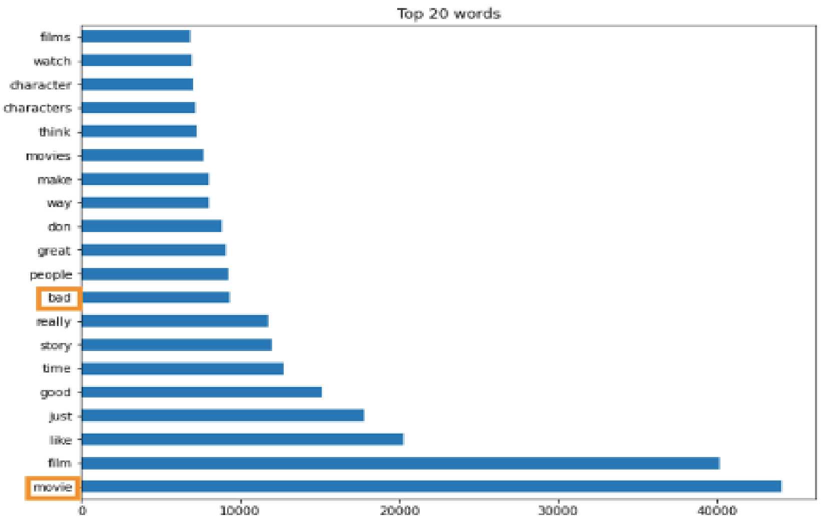


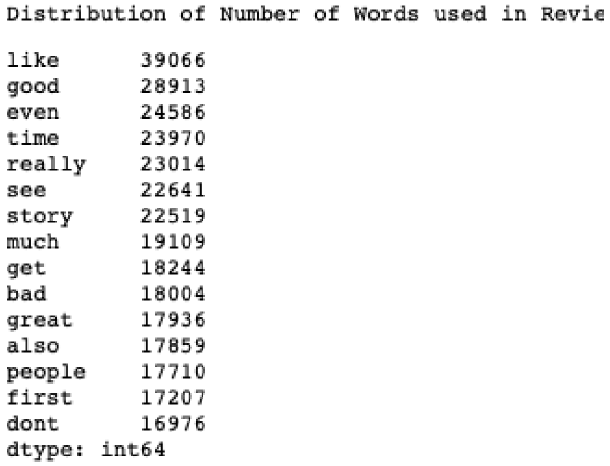
Figure 5 shown above is another data frame example but in ascending order by lengths of reviews. It shows that positive reviews have more words than negative reviews.  It could mean people use fewer words to express the negative sentiment. The bar chart on the right, Figure 6, is the top 20 words useful to identify stops words or any other filter previous to model development.

## Data Preparation Steps

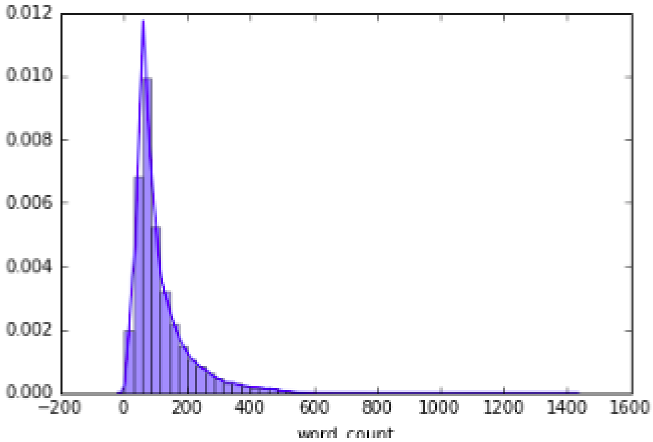
The following steps were applied for data cleaning and preparation before model development:

1. lower case all words
2. remove all email address
3. remove punctuation and special characters
4. remove numbers
5. remove stop words (English plus other no relevant words)
6. converting words to root words (lemmatization)
7. remove rare words

**Figure 7**. Top distribution of words in reviews



```
Distribution of Number of Words used in Revie

like      39066
good      28913
even      24586
time      23970
really    23014
see       22641
story     22519
much      19109
get       18244
bad       18004
great     17936
also      17859
people    17710
first     17207
dont      16976
dtype: int64
```



One review sample after initial cleaning:

```
'saw two grown children although clever shrek thought rather good theatre surrounded
children spring break sound know children liked parents also seemed engaged death
apparent death characters brought appropriate gasps comments hopefully people realize
made kids successful although liked personally liked scrat'
```

**Figure 8.** *Ascending order of review lengths*

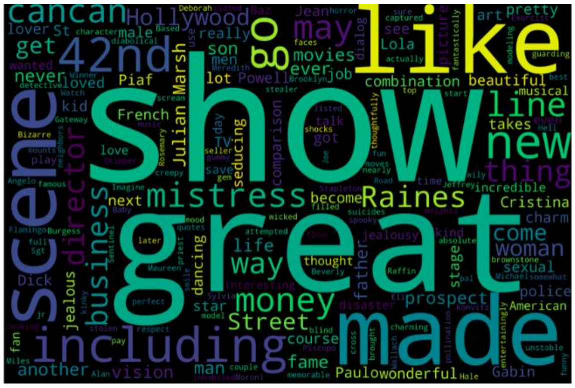
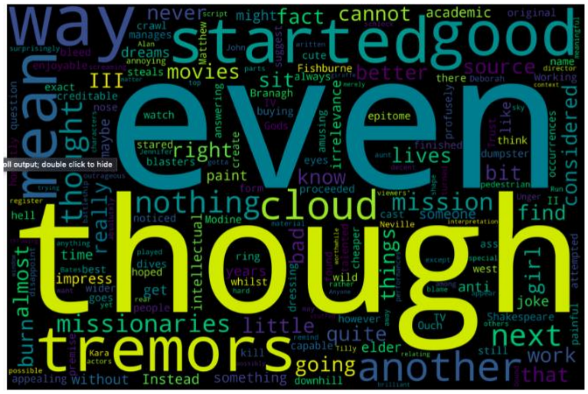


**Figure 9.** *Top 20 words*



## Supervised Learning Models

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and the desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

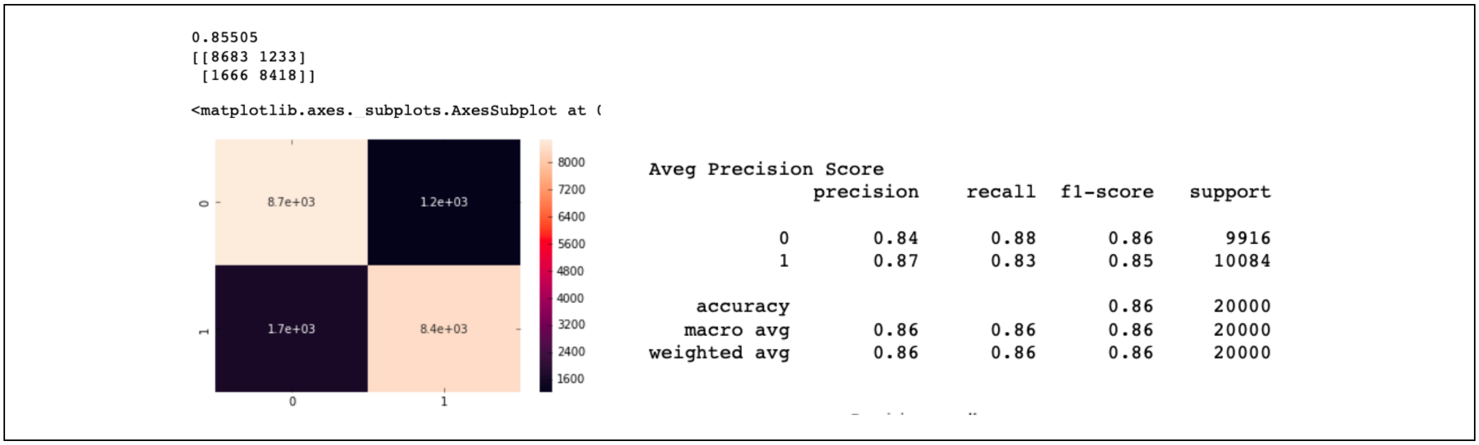
### *Naïve Bayes Classifier (NB)*

The naïve Bayes classifier is a typical generative classifier that can be regarded as a case of Bayesian network classifiers. In general, Bayesian network classifier models first the joint distribution $p(x, y)$ of the measured attributes x and the class labels y factorized in the form $\mathrm{p}(x|y)p(y)$, and then learns the parameters of the model through maximization of the likelihood given by $p(x|y)p(y)$. Due to there is a fundamental assumption that the attributes are conditionally independent given a target class, the naïve Bayes classifier learns the parameters of the model through maximization of the likelihood given by $p(y)\prod j\ p(xj|y)$.

Since the naïve Bayes classifiers optimize the model over the whole dimensionality and are capable of learning even in the presence of some missing values. Furthermore, the naïve Bayes classifier is stable, and its classification result is not significantly changed due to noises or corrupted data.

**(Training set sets to 60 % and test set to 40 %)**

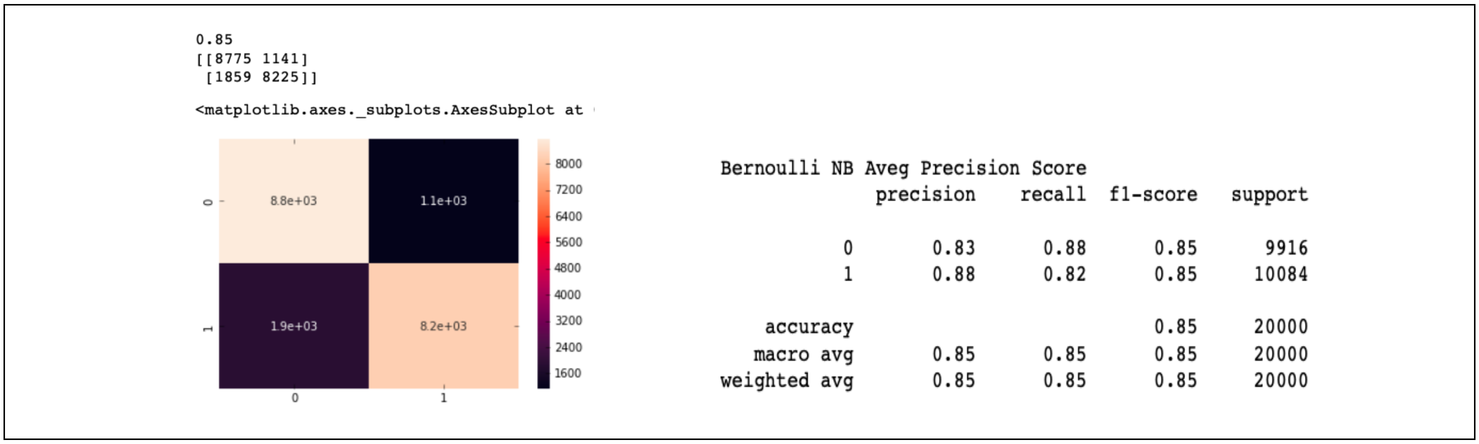
**Table 1.** *Multinomial Naïve Bayes Model Performance Outputs*



```
0.85505
[[8683 1233]
 [1666 8418]]

<matplotlib.axes._subplots.AxesSubplot at (
```

```
Aveg Precision Score
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      9916
           1       0.87      0.83      0.85     10084

    accuracy                           0.86     20000
   macro avg       0.86      0.86      0.86     20000
weighted avg       0.86      0.86      0.86     20000
```

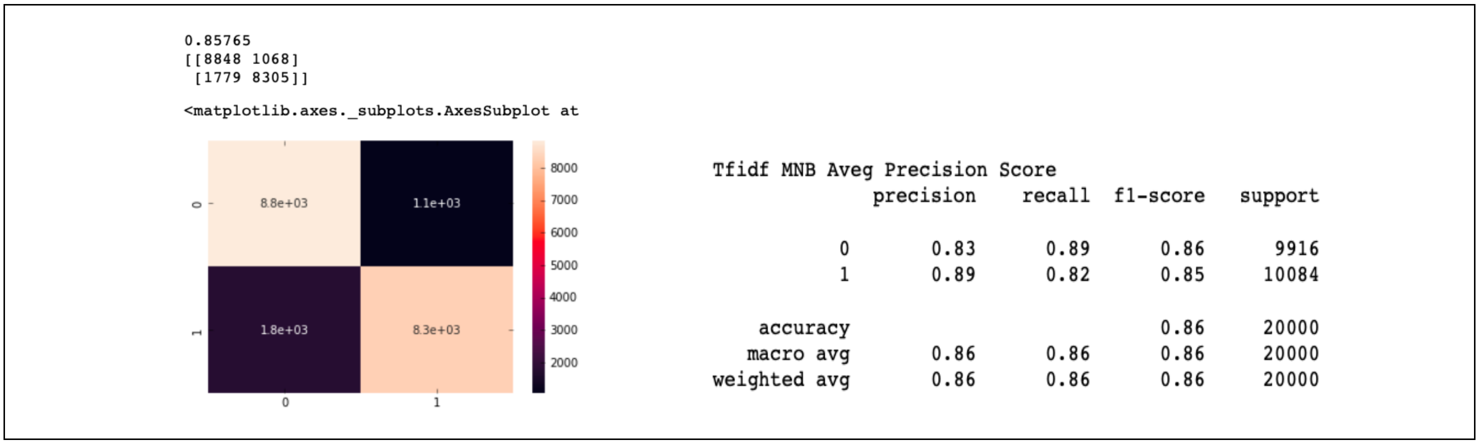
**Table 2.** *Multinomial Naïve Bayes Model Performance Outputs*



```
0.85
[[8775 1141]
 [1859 8225]]

<matplotlib.axes._subplots.AxesSubplot at
```

```
Bernoulli NB Aveg Precision Score
              precision    recall  f1-score   support

           0       0.83      0.88      0.85      9916
           1       0.88      0.82      0.85     10084

    accuracy                           0.85     20000
   macro avg       0.85      0.85      0.85     20000
weighted avg       0.85      0.85      0.85     20000
```

**Table 3.** *Multinomial Naïve Bayes Model with (Tfidf) Performance Outputs*



```
0.85765
[[8848 1068]
 [1779 8305]]

<matplotlib.axes._subplots.AxesSubplot at
```

```
Tfidf MNB Aveg Precision Score
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      9916
           1       0.89      0.82      0.85     10084

    accuracy                           0.86     20000
   macro avg       0.86      0.86      0.86     20000
weighted avg       0.86      0.86      0.86     20000
```

## *Support Vector Machine (SVM) Classifier*

The so-called SVM support vector machine is a supervised learning algorithm that can be used for classification and regression problems as support vector classification (SVC) and support vector regression (SVR).

The SVM classifier is a typical discriminative classifier. Different from the generative classifier, it mainly focuses on how well they can separate the positives from the negatives and does not try to understand the necessary information of the individual classes. The SVM classifier maps first the instance x in training set into a high dimensional space via a function $\Phi$, then computes a decision function of form $f(x) \ = < w, \Phi(x) > \ + b$ by maximizing the distance between the set of points $\Phi(x)$ to the hyperplane or set of hyperplanes parameterized by (w, b) while being consistent on the training set. The SVM classifier builds a single model for all classes, and hence it requires simultaneous consideration of all other classes.

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For high C values, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a minimal amount of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For small C values, the SVM model could show misclassified examples, often also if the training data is linearly separable.

The kernel is a way of computing the dot product of two vectors x and y in some (very high dimensional) feature space, which is why kernel functions are sometimes called “generalized dot product.

Types of kernels:

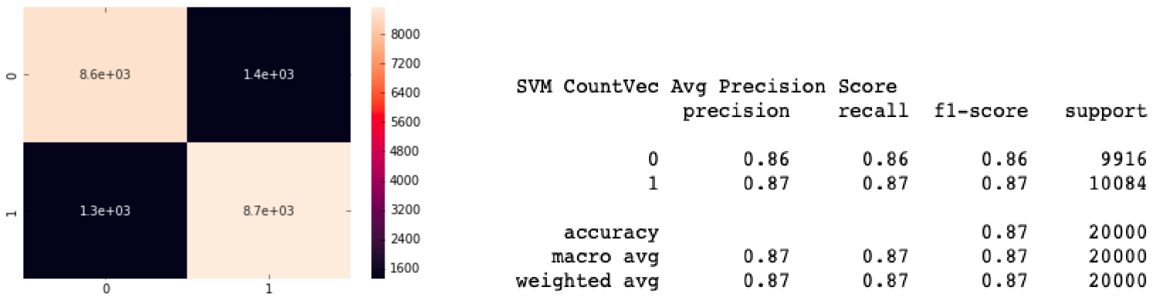
1. linear kernel
2. polynomial kernel
3. Radial basis function kernel (RBF)/ Gaussian Kernel

**(Training set sets to 60 % and test set to 40 %)**

**Table 4.** *Support vector machine (SVM) linear Model Performance Outputs*

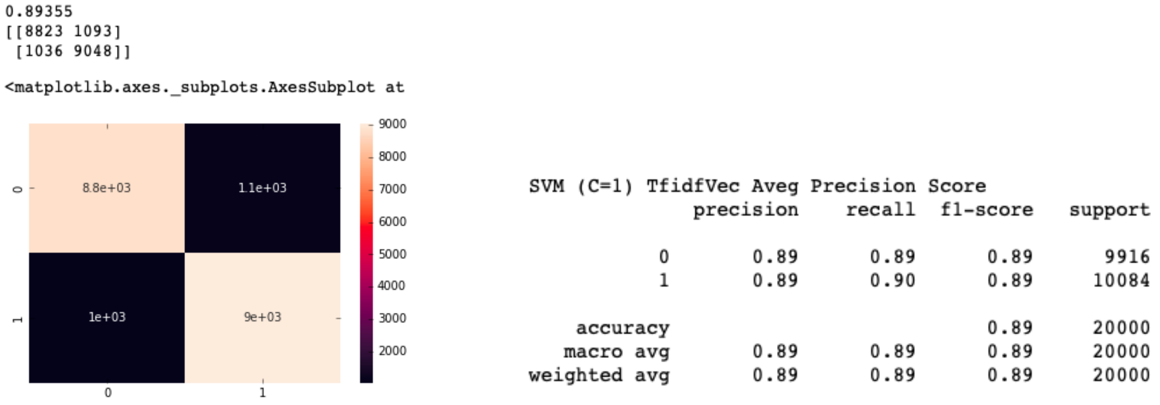
**Cost parameter = 1 with CountVectorizer**

```
0.8656
[[8565 1351]
 [1337 8747]]
```



```
SVM CountVec Avg Precision Score
              precision    recall  f1-score   support

           0       0.86      0.86      0.86      9916
           1       0.87      0.87      0.87     10084

    accuracy                           0.87     20000
   macro avg       0.87      0.87      0.87     20000
weighted avg       0.87      0.87      0.87     20000
```

**Cost parameter = 1 with Tfidf Vectorizer**



```
0.89355
[[8823 1093]
 [1036 9048]]

<matplotlib.axes._subplots.AxesSubplot at
```

```
SVM (C=1) TfidfVec Aveg Precision Score
              precision    recall  f1-score   support

           0       0.89      0.89      0.89      9916
           1       0.89      0.90      0.89     10084

    accuracy                           0.89     20000
   macro avg       0.89      0.89      0.89     20000
weighted avg       0.89      0.89      0.89     20000
```

### *Naïve Bayes vs. SVM*

Naive Bayes Classifier (NBC) and Support Vector Machine (SVM) have different options, including the choice of kernel function for each. They are both sensitive to parameter optimization (i.e., different parameter selection can significantly change their output). Variants of NB and SVM are often used as baseline methods for text classification, but their performance varies greatly depending on the model variant, features used, and task/ dataset.

### *Summary of NB Models and SVM Models*

In a Receiver Operating Characteristic (ROC) curve, the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. In general, an area under the curve (AUC) of 0.5 suggests no discrimination 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding.

Precision and recall are two essential model evaluation metrics. While precision refers to the percentage of your results that are relevant, recall refers to the rate of total relevant results correctly classified by your algorithm. Unfortunately, it is not possible to maximize both these metrics simultaneously as one comes at the cost of another. For simplicity, another metric is called the F-1 score, which is a harmonic mean of precision and recall. Table 5 summarizes the performance metrics of models.

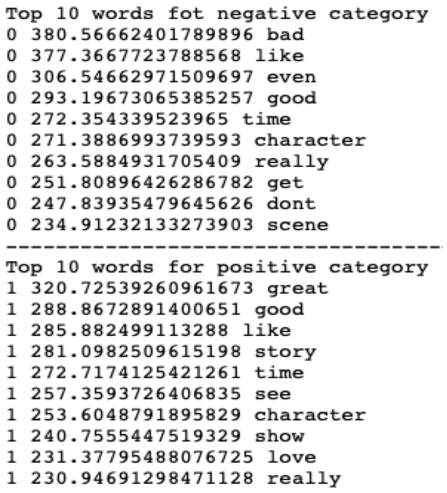
**Table 5.** *Summary of the performance metrics of models*

| Model | Accuracy | AUC Score | F-1 Score |
|---|---|---|---|
| Multinomial NB with CountVect | 0.86 | 0.92 | Pos: 0.85  Neg: 0.86 |
| Bernoulli NB with binary CountVect | 0.85 | 0.93 | Pos: 0.85  Neg: 0.85 |
| **Multinomial NB with TfidfVect** | **0.86** | **0.94** | **Pos: 0.85  Neg: 0.86** |
| SVM linear with CountVect, C=1 | 0.87 | 0.87 | Pos: 0.87  Neg: 0.86 |
| **SVM linear with TfidfVect, C=1** | **0.89** | **0.89** | **Pos: 0.89 Neg: 0.89** |
| SVM with Kernel RBF, and C=1 | 0.58 | 0.58 | Pos: 0.31  Neg: 0.70 |
| **SVM with Kernel RBF, and C= 1000** | **0.89** | **0.89** | **Pos: 0.89 Neg: 0.89** |
| SVM with Kernel POLY, and C=1 | 0.50 | 0.50 | Pos: 0        Neg: 0.66 |

From the above results, the best Naïve Bayes model was the Multinomial with Tfidf. The MNB and BNB modes showed similar performance. However, the SVM model performed better than Naïve Bayes for the movie review data set. The best performing model was the linear SVM with Tfidf Vectorizer and the cost parameter set to 1. Also, with the same output, the non-linear SVM RBF with a cost parameter set to 1000.  Both reached an accuracy of 89%.

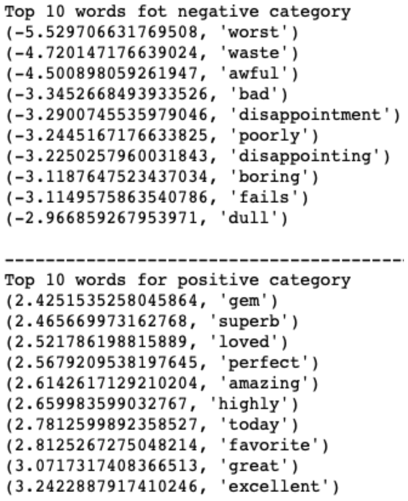
**Table 6.** *Top10 most indicative words for the positive category and the negative category*

**MNB Best Model**



```
Top 10 words fot negative category
0 380.56662401789896 bad
0 377.3667723788568 like
0 306.54662971509697 even
0 293.19673065385257 good
0 272.354339523965 time
0 271.3886993739593 character
0 263.5884931705409 really
0 251.80896426286782 get
0 247.83935479645626 dont
0 234.91232133273903 scene
-----------------------------------
Top 10 words for positive category
1 320.72539260961673 great
1 288.8672891400651 good
1 285.882499113288 like
1 281.0982509615198 story
1 272.7174125421261 time
1 257.3593726406835 see
1 253.6048791895829 character
1 240.7555447519329 show
1 231.37795488076725 love
1 230.94691298471128 really
```

**SVM Best Model**



```
Top 10 words fot negative category
(-5.529706631769508, 'worst')
(-4.720147176639024, 'waste')
(-4.500898059261947, 'awful')
(-3.3452668493933526, 'bad')
(-3.2900745535979046, 'disappointment')
(-3.2445167176633825, 'poorly')
(-3.2250257960031843, 'disappointing')
(-3.1187647523437034, 'boring')
(-3.1149575863540786, 'fails')
(-2.966859267953971, 'dull')

----------------------------------------
Top 10 words for positive category
(2.4251535258045864, 'gem')
(2.465669973162768, 'superb')
(2.521786198815889, 'loved')
(2.5679209538197645, 'perfect')
(2.6142617129210204, 'amazing')
(2.659983599032767, 'highly')
(2.7812599892358527, 'today')
(2.8125267275048214, 'favorite')
(3.0717317408366513, 'great')
(3.2422887917410246, 'excellent')
```

### *Neural Network (Keras)*

Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers, and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

In ANN implementations, the "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold.

The model used for predicting the sentiment of the reviews is Keras. The whole dataset was used.

### Why Keras?

There are many deep learning frameworks available in the market like TensorFlow, Theano. So why Keras was preferred? The most important reason is its Simplicity. Keras is a top-level API library where you can use any framework as your backend and is easy to learn and easy to use.

Simple steps were followed to build the model:

- get bigrams and unigrams from the data
- encode it using tf-idf
- select the top 20000 features from the vector
- discard features that occur less than two times

The neural network is created by stacking layers—this requires two main architectural decisions:

1. How many layers to use in the model?
2. How many hidden units to use for each layer?

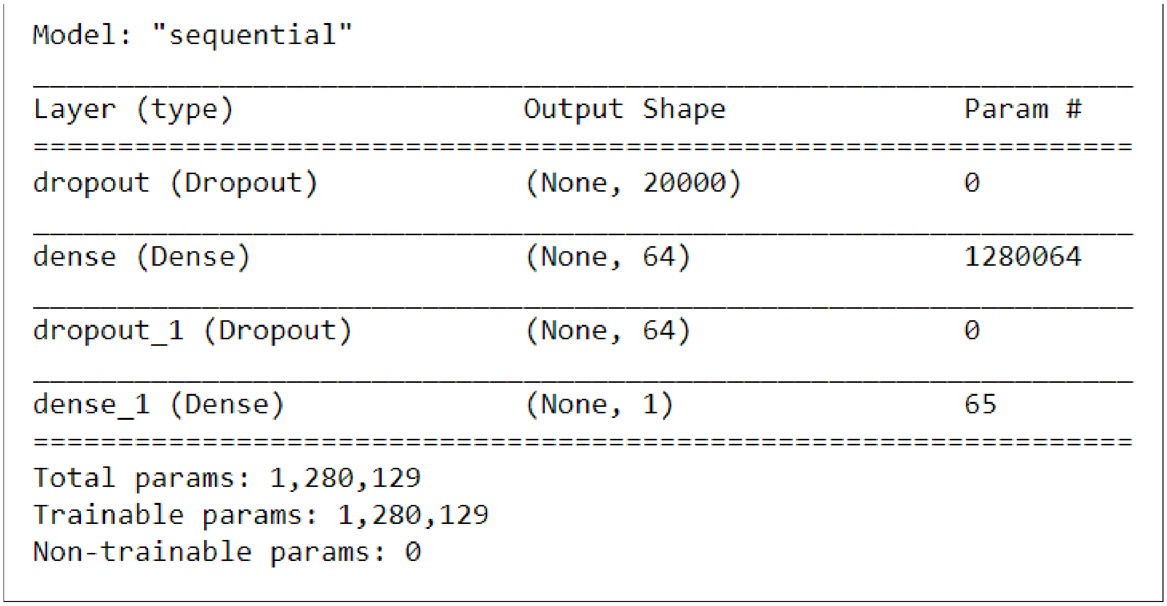
The input data consists of an array of word-probabilities. The labels to predict are either 0 or 1.

### Build the model

Four layers were created: two Dropout and two Dense.

**(Training set sets to 80 % and test set to 20 %)**

**Figure 10.** Structure of the model



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dropout (Dropout)            (None, 20000)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1280064
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,280,129
Trainable params: 1,280,129
Non-trainable params: 0
```

**Compile the model**
Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

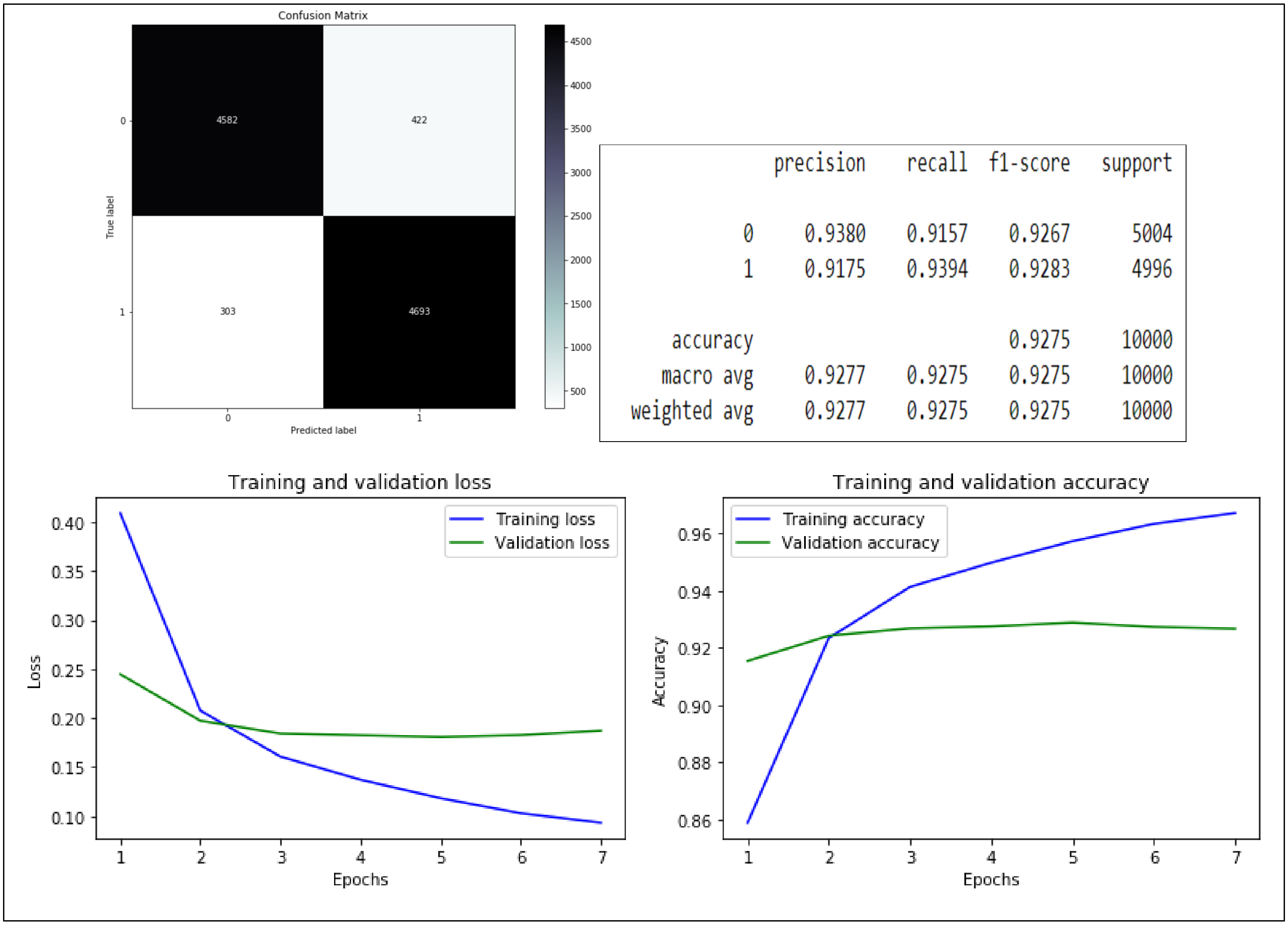
- Loss function —These measures how accurate the model is during training. We want to minimize this function to "steer" the model in the right direction.
- Optimizer —This is how the model is updated based on the data it sees and its loss function.
- Metrics —Used to monitor the training and testing steps. The following example uses accuracy, the fraction of the images that are correctly classified.

Since this is a binary classification problem and the model outputs logits (a single-unit layer with a linear activation), we'll use the binary_crossentropy loss function. ‘adam’ optimizer was used.

**Train the model**
Callback for early stopping on validation loss was created (if the loss does not decrease on two consecutive tries, it stops training.)

**Evaluate the model**
The confusion matrix is one of the best ways to visualize the accuracy of a model. Table 7 shown below indicates this model produced the best accuracy. Almost 94% of the positive and 92% of the negative reviews were correctly classified.

**Table 7.** *Performance of the neural network (Keras) model*



```
              precision    recall  f1-score   support

           0     0.9380    0.9157    0.9267      5004
           1     0.9175    0.9394    0.9283      4996

    accuracy                         0.9275     10000
   macro avg     0.9277    0.9275    0.9275     10000
weighted avg     0.9277    0.9275    0.9275     10000
```

From the training and validation plot shown in Table 7, conclusions about the model training performance can be made. The perfect scenario is that they both decrease toward the minima. The accuracy plot shows a little overfitting.

## Unsupervised Learning Models

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. In contrast to supervised learning that usually makes use of human-labeled data, unsupervised learning, also known as self-organization, allows for modeling of probability densities over inputs. It forms one of the three main categories of machine learning, along with supervised and reinforcement learning.

### *LDA*

Topic Models are a type of statistical language models used for uncovering hidden structure in a collection of texts. In a practical and more intuitively, you can think of it as a task of:

- Dimensionality Reduction, where rather than representing a text T in its feature space as {Word_i: count(Word_i, T) for Word_i in Vocabulary}, can be represented  it in a topic space as {Topic_i: Weight(Topic_i, T) for Topic_i in Topics}
- Unsupervised Learning, where it can be compared to clustering, as in the case of clustering, the number of topics, like the number of clusters, is an output parameter. By doing topic modeling, clusters of words are built rather than clusters of texts. A text is thus a mixture of all the topics, each having a specific weight
- Tagging, abstract “topics” that occur in a collection of documents that best represents the information in them.

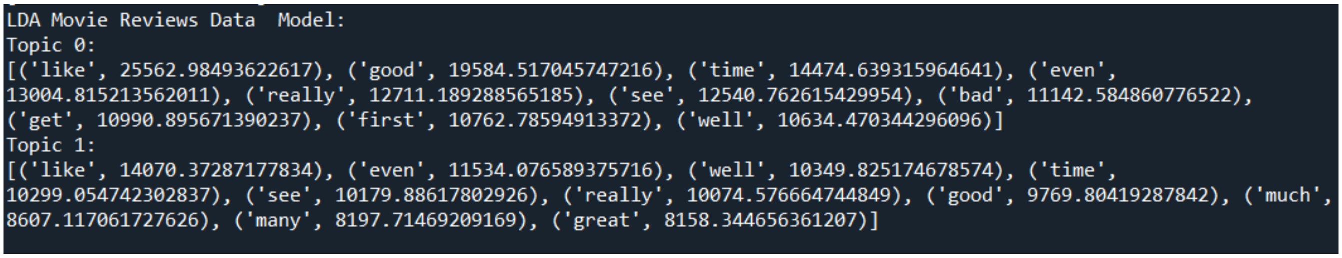
Several existing algorithms can be used to perform topic modeling. The most common of it are, Latent Semantic Analysis (LSA/LSI), Probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities. The generative process of LDA can be described as, given the M number of documents, N number of words, and prior K number of topics, the model trains to output:

- **psi, the distribution of words for each topic K**
- **phi, the distribution of topics for each document i**

For this project, LDA was used in an attempt to discover the positive and negative reviews as two main topics. Countvectorizer was used to transform the text and remove stop words. An experiment with and without stemming was performed.  Top 10 most common words with and without stemming were visualized shown as Table 9 and Table 9 below. From Table 8 and Table 9, it is easy to notice some changes in the words and word order after stemming.

**Table 8.** *Not stemmed*



```
LDA Movie Reviews Data  Model:
Topic 0:
[('like', 25562.98493622617), ('good', 19584.517045747216), ('time', 14474.639315964641), ('even',
13004.815213562011), ('really', 12711.189288565185), ('see', 12540.762615429954), ('bad', 11142.584860776522),
('get', 10990.895671390237), ('first', 10762.78594913372), ('well', 10634.470344296096)]
Topic 1:
[('like', 14070.37287177834), ('even', 11534.076589375716), ('well', 10349.825174678574), ('time',
10299.054742302837), ('see', 10179.88617802926), ('really', 10074.576664744849), ('good', 9769.80419287842), ('much',
8607.117061727626), ('many', 8197.71469209169), ('great', 8158.344656361207)]
```

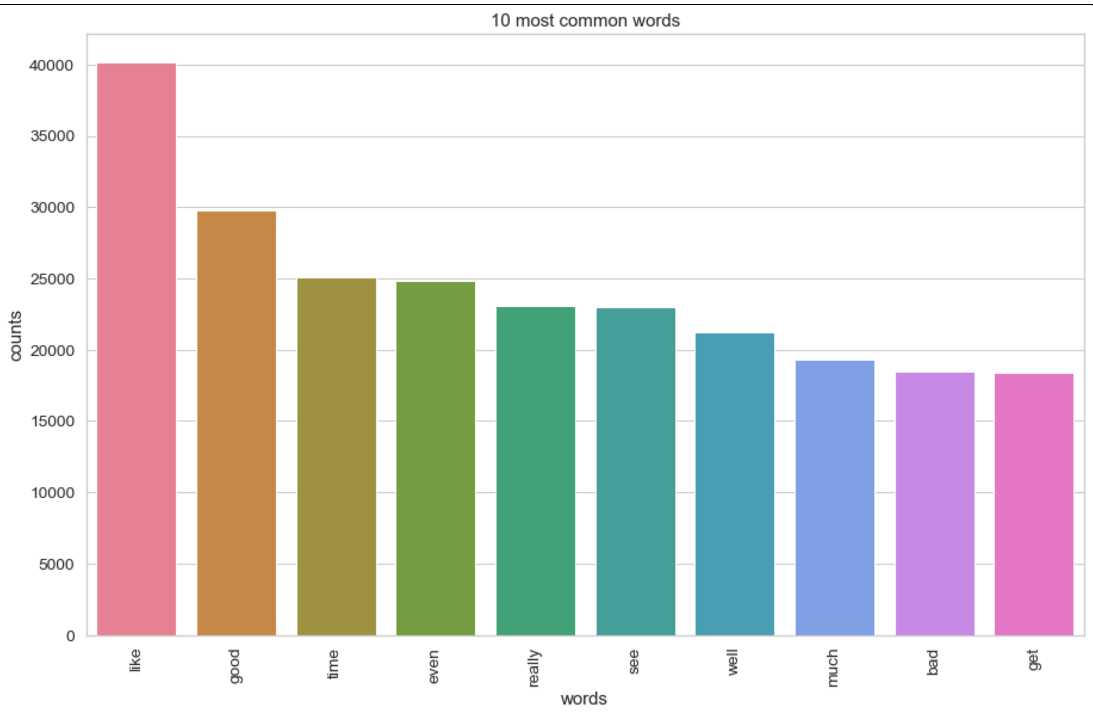
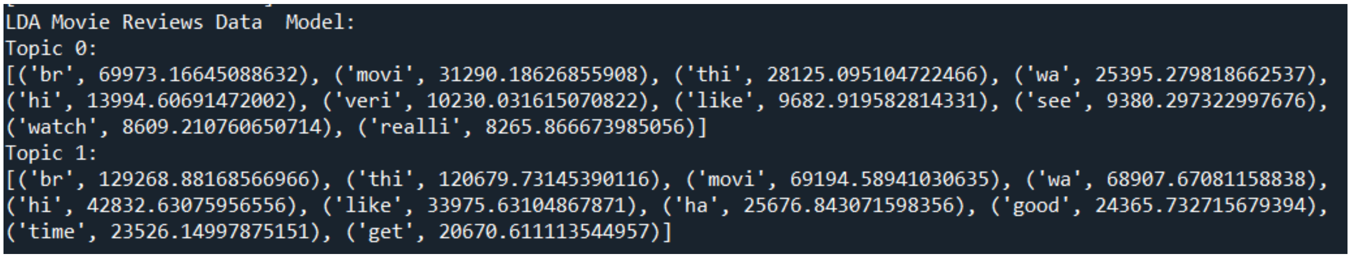
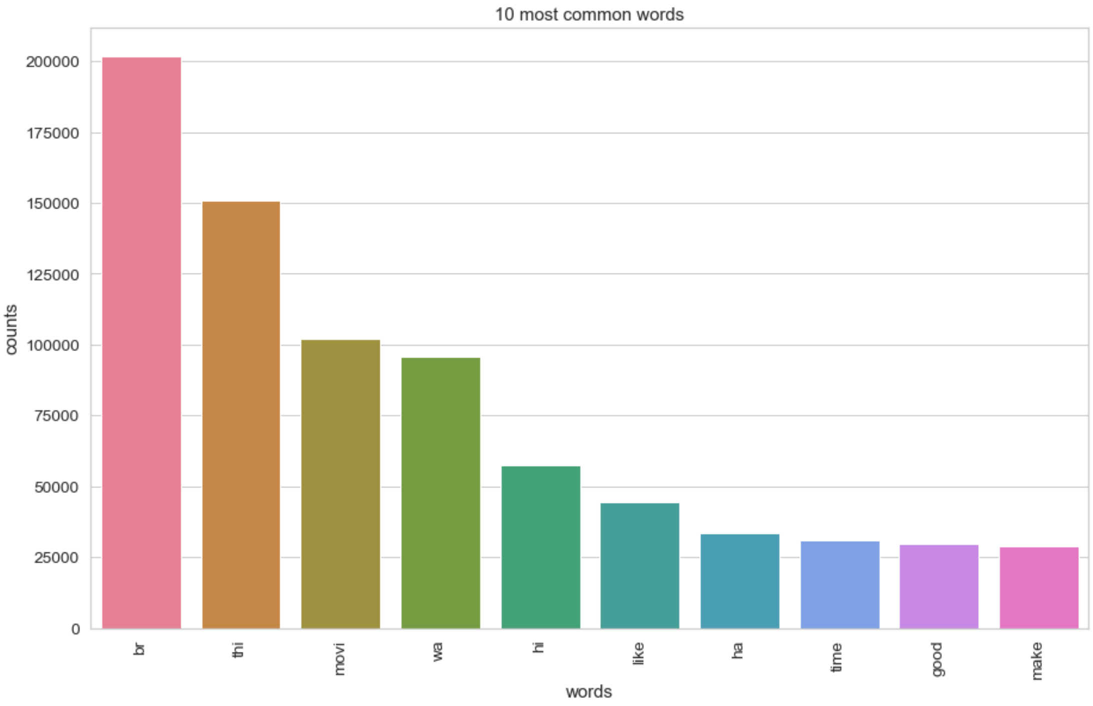


**Table 9.** *Stemmed*



```
LDA Movie Reviews Data  Model:
Topic 0:
[('br', 69973.16645088632), ('movi', 31290.18626855908), ('thi', 28125.095104722466), ('wa', 25395.279818662537),
('hi', 13994.60691472002), ('veri', 10230.031615070822), ('like', 9682.919582814331), ('see', 9380.297322997676),
('watch', 8609.210760650714), ('realli', 8265.866673985056)]
Topic 1:
[('br', 129268.88168566966), ('thi', 120679.73145390116), ('movi', 69194.58941030635), ('wa', 68907.67081158838),
('hi', 42832.63075956556), ('like', 33975.63104867871), ('ha', 25676.843071598356), ('good', 24365.732715679394),
('time', 23526.14997875151), ('get', 20670.611113544957)]
```



Again, the number of topics was set to two. In the not stemmed version, the topics are more obvious - bad reviews and good reviews.

**Table 10**. Feature words

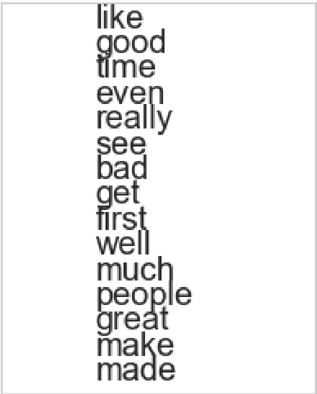
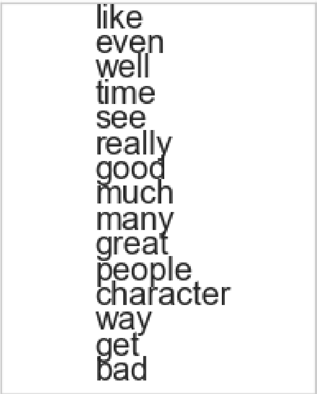
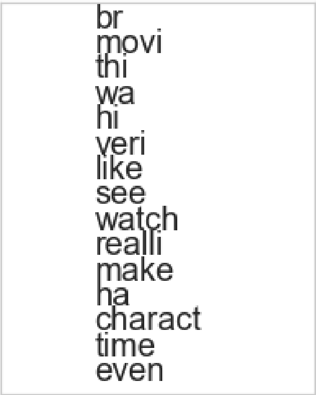
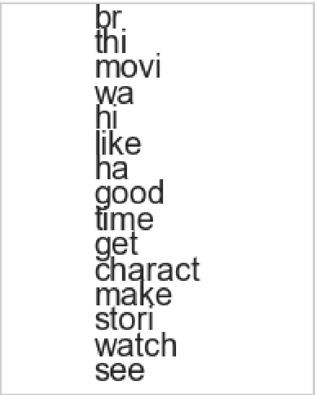
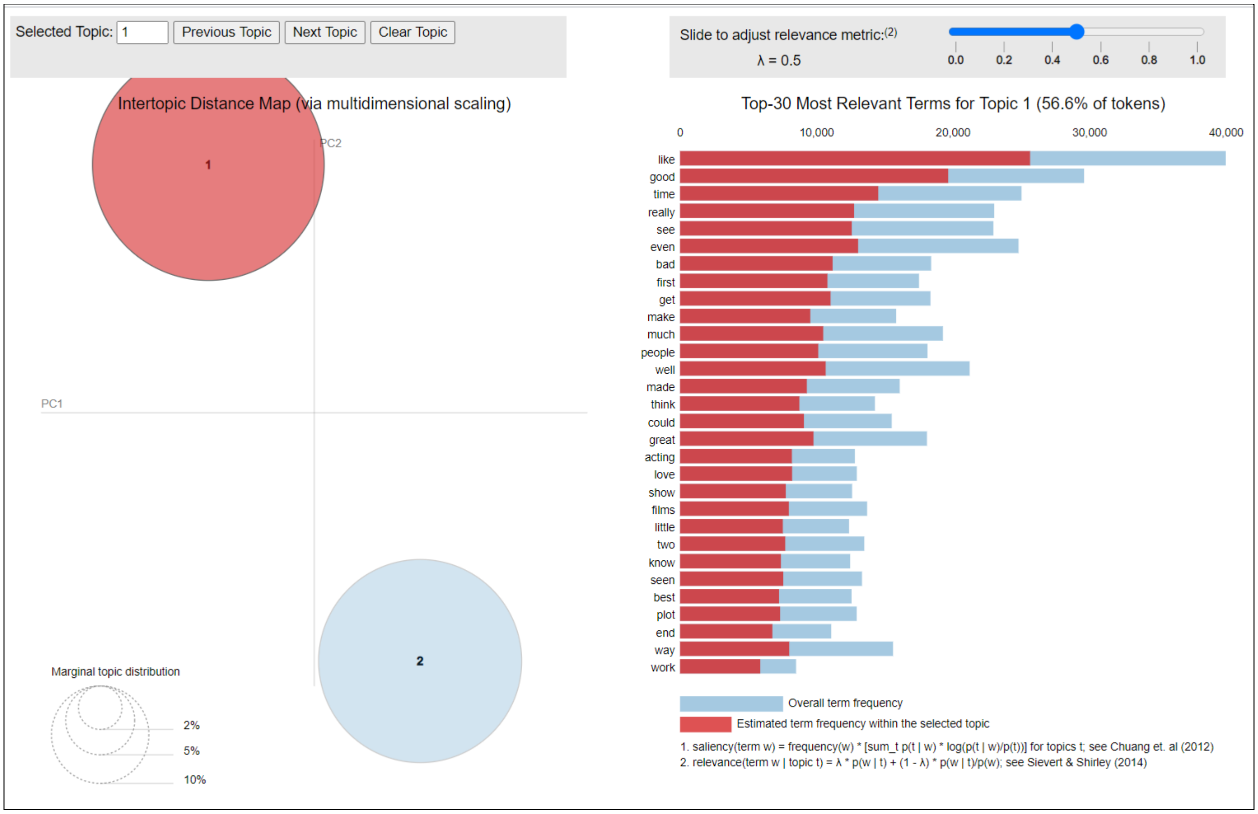
| **Not stemmed:**<br>Most of the words overlap, but there is the word 'great' in Topic #0 and the word 'bad' in Topic#1. | **Stemmed:** In the stemmed version, the words appear different and it is really hard to define the topics. |
|---|---|
| Topic #0  like<br>good<br>time<br>even<br>really<br>see<br>bad<br>get<br>first<br>well<br>much<br>people<br>great<br>make<br>made<br>Topic #1  like<br>even<br>well<br>time<br>see<br>really<br>good<br>much<br>many<br>great<br>people<br>character<br>way<br>get<br>bad | Topic #0  br<br>movi<br>thi<br>wa<br>hi<br>veri<br>like<br>see<br>watch<br>realli<br>make<br>ha<br>charact<br>time<br>even<br>Topic #1  br<br>thi<br>movi<br>wa<br>hi<br>like<br>ha<br>good<br>time<br>get<br>charact<br>make<br>stori<br>watch<br>see |

**Figure 11.** *Interactive visualization from pyLDAvis*



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

From Figure 11, it is noticeable that the two topics are almost evenly distributed. There are a lot of words that overlapping in both topics. Further stop-words modification or lemmatization can produce better results

### *K-means Clustering*

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. The algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster while keeping the centroids as small as possible. The 'means' in the K-means refers to averaging of the data. Its goal is to find the centroid.

**How the K-means algorithm works**

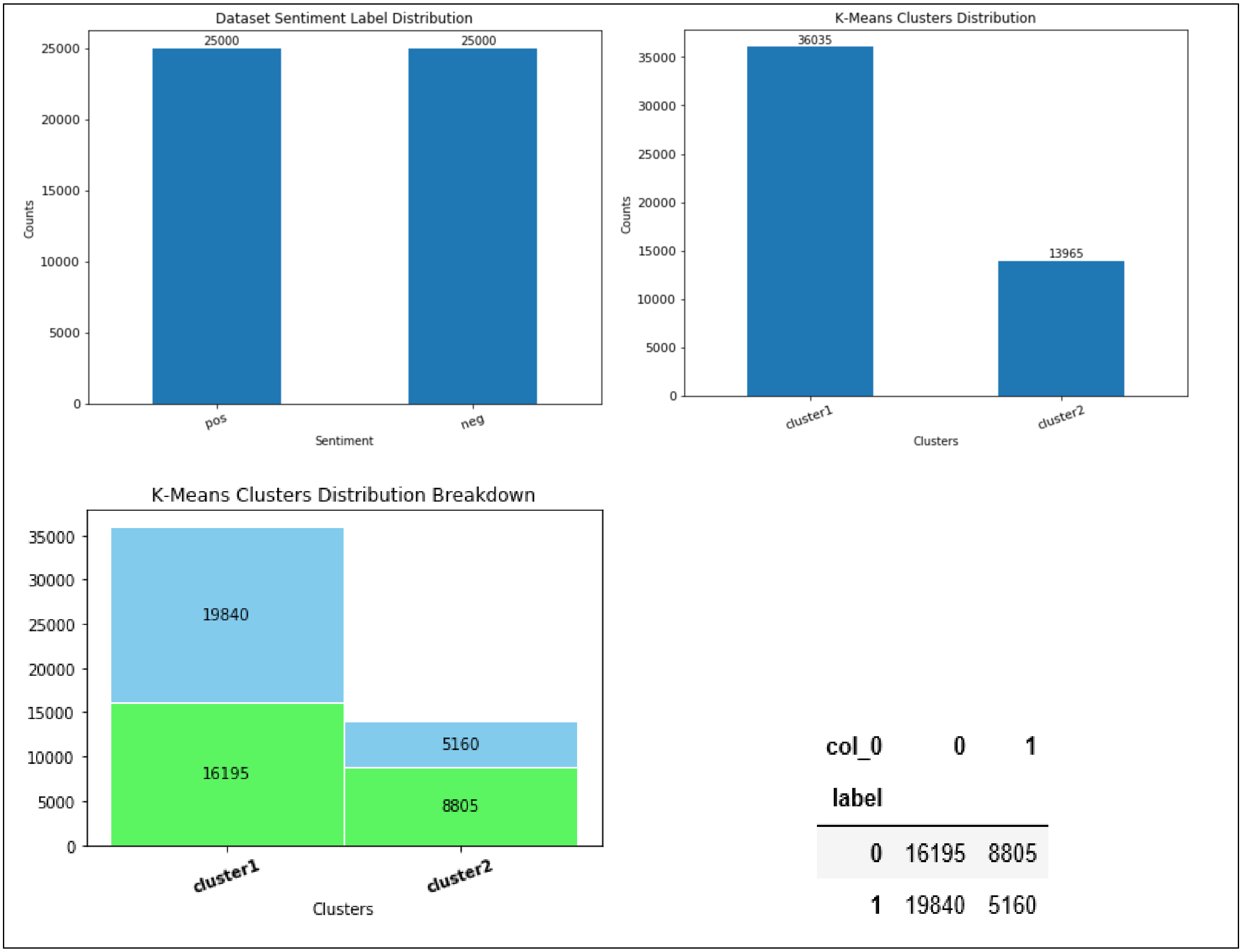
To process the learning data, the K-means algorithm in data mining starts with the first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

In the case of sentiment in movie reviews, the number of centroids k was set to 2 for two-level of sentiment, positive and negative. English stop words and tokens have a frequency below five times have been excluded with Tfidf vectorizer.

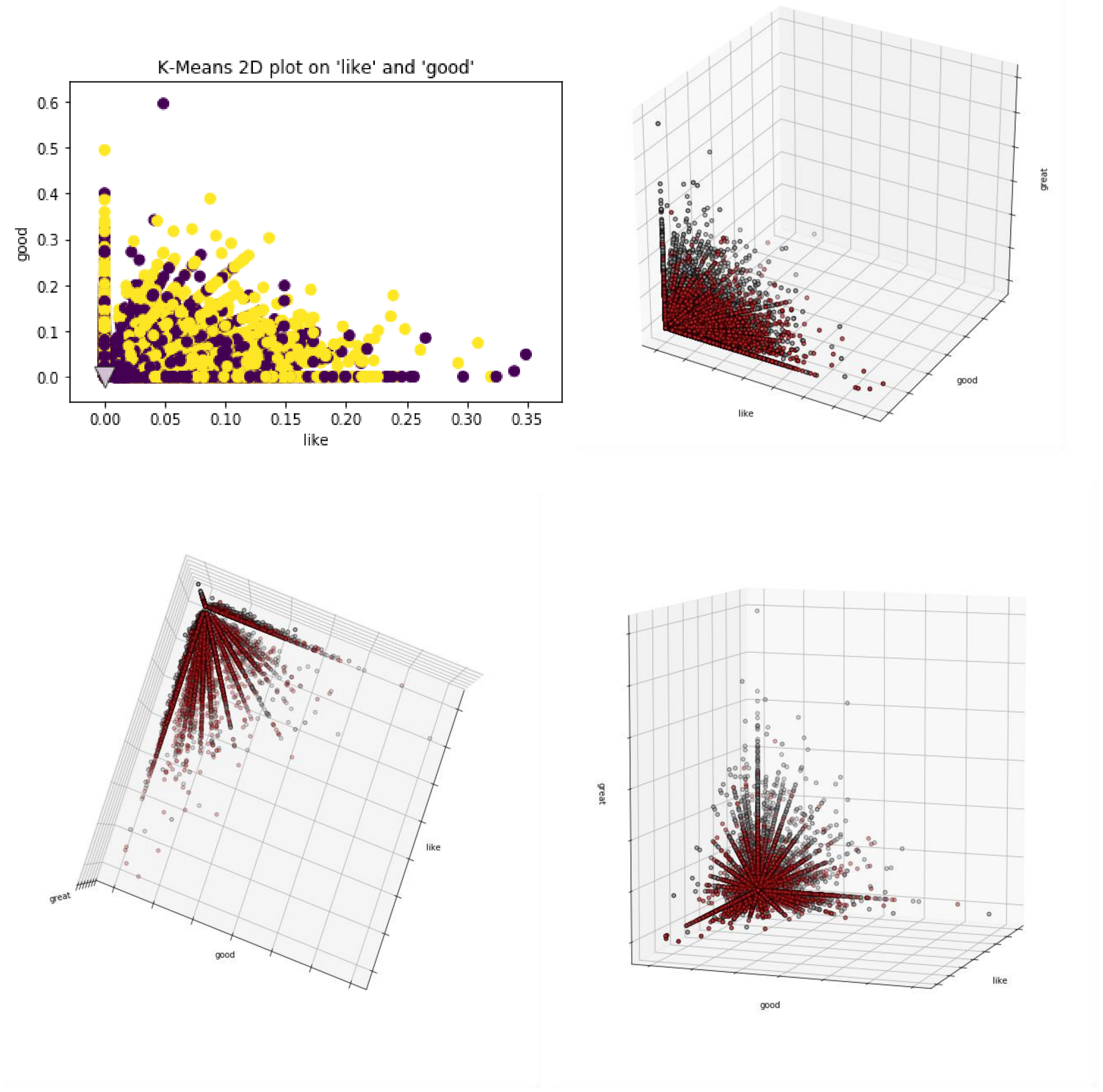
**Table 11**. *K-means cluster distribution*



| col_0 | 0 | 1 |
|---|---|---|
| label | | |
| 0 | 16195 | 8805 |
| 1 | 19840 | 5160 |

Table 11 shows how the reviews were labeled by the k-means algorithm and how it was different from the ground truth. Each of the sentiments should have 25000 counts. However, cluster 1 consists of 19840 positive reviews and 16195 negative reviews; cluster 2 has 5160 positive reviews and 8805 negative reviews. It seems like the K-means algorithm cannot successfully separate reviews into two clusters based on the sentiment levels.

**Table 12**. K-means scatter plots in 2D and 3D (‘like’, ‘good’ and ‘great’ features)



From Table 12, it confirms that it is hard to separate data by implementing the k-means algorithm. The 2-D plot shows that the centroid, triangles, are located at the origin in the Cartesian coordinate system. The 3-D plots affirm the difficulty of separating data in k-means.

## Next Word Prediction for Movie Reviews

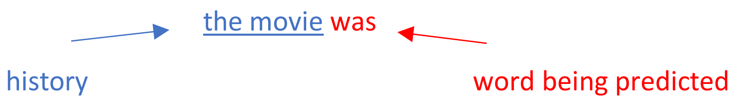
### Language Model

#### *Data Cleaning*

The glob package reads in the 50,000 different reviews from four different files: positive train, negative train,  positive test, and negative test. As each review is read in as a string. Before the reviews are stored in a list, all text is changed to lower case and regular expressions filter line breaks (\n) and other HTML markups like <br />. To tokenize the words, the four lists are combined to be one long string that contains all 50,000 reviews that make up the corpus. Finally, the string is tokenized with RegexpTokenizer to get all the word-like objects. This tokenizer separates tokens at spaces and punctuation causing contractions to be separated. For example, “don’t” is tokenized to be two tokens, “don” and “t”. This creates a corpus with 11,772,360 total tokens and a vocabulary with 104,138 unique tokens.

#### *N-gram Language Model*

An n-gram is a group of n tokens that appear together in a corpus. For the n-gram text prediction function, previous words are used to predict the next word using conditional probabilities. The next word is predicted using the contextual probabilities from the corpus. For example,



where “the movie” is the context or history and “was” is the word to predict. This example would be considered a trigram due to the three words that make up the phrase. The context or history of an n-gram has n-1 words or tokens and predicts the nth word. These conditional probabilities calculated using the chain rule for probability (Rizvi, 2019).

Let $W_1^{n-1}$ represent the string of tokens $w_1, w_2, \ldots, w_{n-1}$. Using the chain rule of probability

$$(1)\ P(w_1 \ldots w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) \ldots P(w_n|w_1 \ldots w_{n-1})$$

where each conditional probability is calculated using equation (2).

$$(2)\quad P(w_n|w_1 \ldots w_{n-1}) = \frac{P(w_1 w_2 \ldots w_n)}{P(w_1\ w_2\ \ldots w_{n-1})}$$

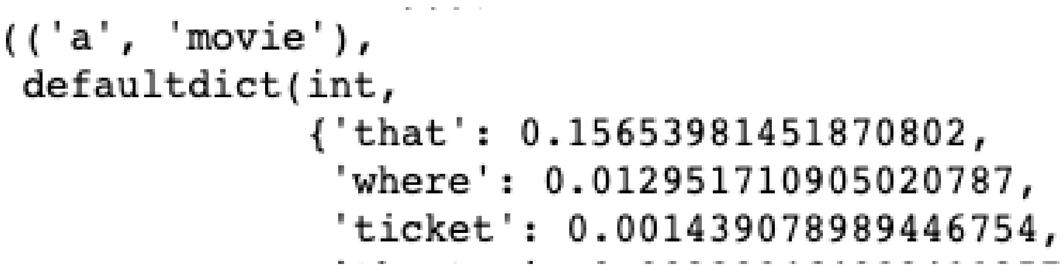
The two models used for the application are the bigram and trigram prediction models. These probabilities are calculated using equations (3) and (4), respectively

$$(3)\ P(w_2|w_1) = \frac{P(w_1, w_2)}{P(w_1)}$$

$$(4)\ P(w_3|w_1,w_2) = \frac{P(w_1\ w_2,w_3)}{P(w_1 w_2)}$$

To find these probabilities, the n-gram package from the NLTK library finds all the different n-grams in the corpus. For example, consider the trigram “a movie ___”. The bigram “a movie” is used as the key and the values are an embedded dictionary where each key is a possible third word with the value being the likelihood of that word as a conditional probability. A snapshot of the first entries is shown below in Figure 12.

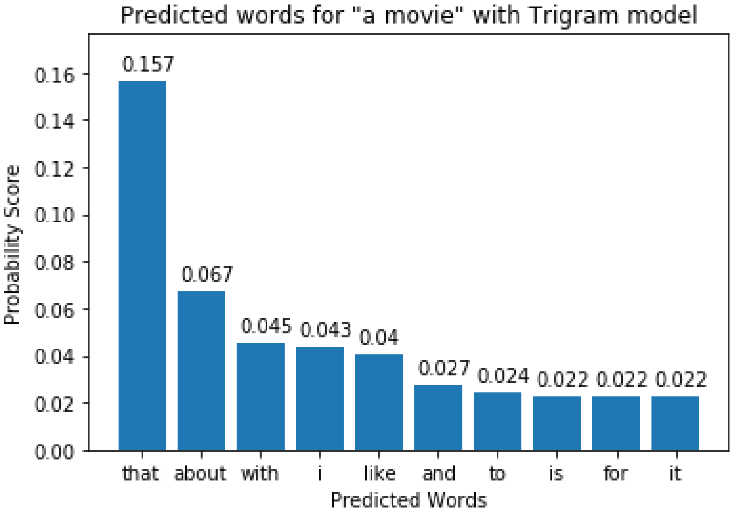
**Figure 12:** *An example of a dictionary from the trigram model to predict the next word after the bigram “a movie”.*



```
(('a', 'movie'),
 defaultdict(int,
                  {'that': 0.15653981451870802,
                   'where': 0.012951710905020787,
                   'ticket': 0.001439078989446754,
```

In general, for each n-gram model, the (n-1)-gram is stored as a key for a dictionary where the embedded dictionary contains all possible nth words and their probabilities as the key, value pairs. To predict the nth word, the text prediction model returns the key that has the highest probability, as it is the most likely word to occur next, given the corpus. The bar chart below in Figure 13 shows the 10 most likely words to follow the bigram “a movie” with “that” having the largest probability of 0.157, thus the predicted word is “that”.

**Figure 13:** *Bar graph of 10 most likely words to follow the bigram “a movie”.*



The application allows users to enter as many words as they would like by applying a back-off approach. Each model uses at most two words as history to predict the next word, thus when users enter in more than two words, the final two words are used to predict the third. This technique uses the Markov Assumption for Natural Language Processing which uses only the most recent words to predict the next word. The conditional probabilities are then calculated using equation (5)

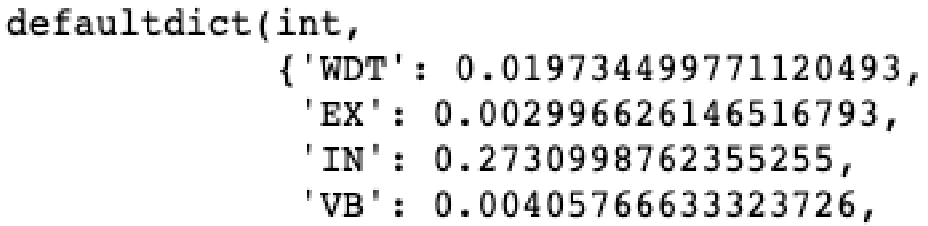
$$(5)\ \ P(w_n|w_1 \dots w_{n-1}) = P(w_n|w_{n-1}\ w_{n-2})$$

The n-gram models have two main challenges to overcome.

1. There is no prediction available because the (n-1)-gram is not in the corpus
2. There are multiple predictions due to several unique nth words having equal conditional probabilities.

To handle the case of no prediction, each token's part of speech is tagged and a dictionary identical to the word dictionary is created for bigrams and trigrams. A snapshot of an entry for the part of speech bigram "DT, NN" (determiner, noun) is shown in Figure 14.

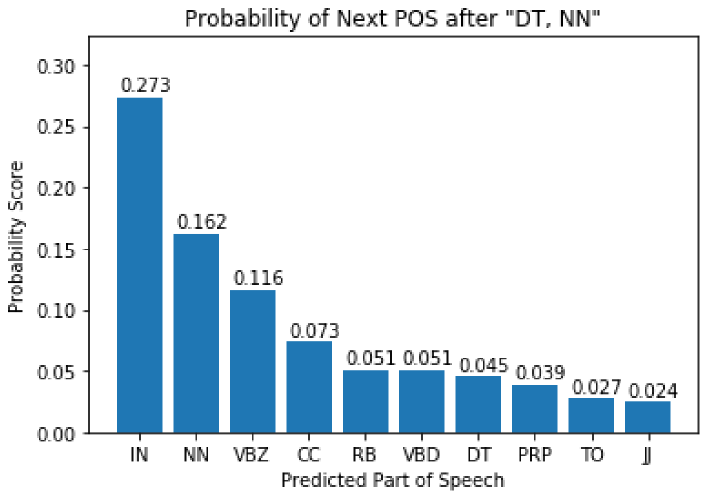
**Figure 14:** *An example of a dictionary from the trigram part of speech model to predict the next part of speech after the bigram "DT NN"*



```
defaultdict(int,
            {'WDT': 0.019734499771120493,
             'EX': 0.0029966261465167 93,
             'IN': 0.2730998762355255,
             'VB': 0.004057666333237 26,
```

The part of speech prediction model returns the part of speech that has the largest conditional probability, thus most likely to occur next. The model then uses a predefined dictionary of the most common word in the corpus for each part of speech to return the most common word for the predicted part of speech. A bar graph showing the 10 most likely parts of speech following "DT NN" is shown in Figure 15. This model predicts that the part of speech "IN", a preposition, is most likely to follow. The most common preposition in the corpus is "of", thus the model predicts the word "of" for the third word of the trigram.

**Figure 15:** *Bar graph of 10 most likely parts of speech to follow the bigram "DT NN".*



This method will help to always predict a word, even if the input (n-1)-gram is not in the corpus, but it does produce more phrases that sound awkward to a native speaker. For example, if a user enters "the oven", this bigram is not in the corpus thus the model reverts to using parts of speech to give the next word which is "of". This is reasonable from the part of speech point of view, the most common trigram that begins with a bigram of "DT NN" is "DT NN IN". However, "the oven of" may not be the most likely combination of words to use from a native speaker's point of view.

To handle the cases where several unique words are predicted, the model uses the frequency distribution of the corpus and returns the most frequent word of all the nth word predictions. For example, consider the bigram “dont see” to predict the third word of the trigram. The corpus has five different trigrams that begin with “dont see”, each having an equally likely third word to return. The third words and their probabilities are shown below in Figure 16. The frequency distribution of the five possible words to return are shown in the bar graph in Figure 17.

**Figure 16:** *Predicted third words from the bigram “dont see", each with a likelihood of 0.2.*

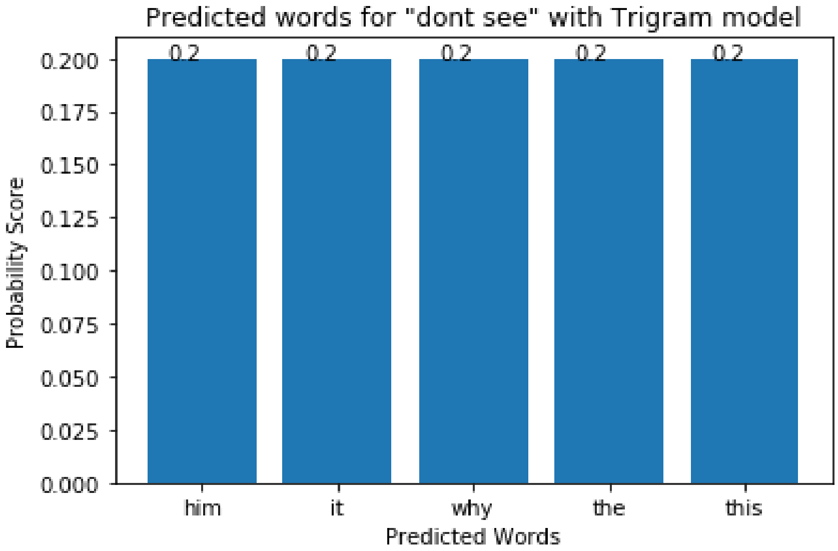
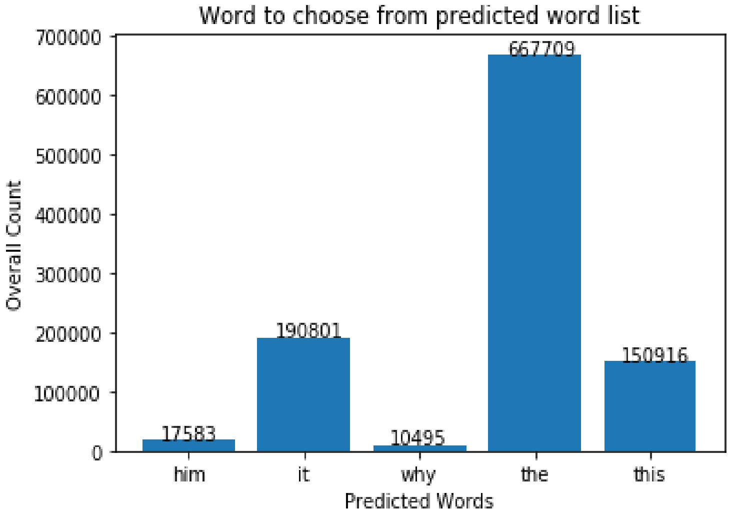


**Figure 17:** *Frequency of the 5 possible words to complete the trigram “dont see __”*



To summarize the text prediction application, the model operates over X steps. First, the user's input is tokenized using the RegexpTokenizer that separates tokens by spaces and/or punctuation. The last two tokens are the input for the trigram prediction model if this model predicts more than one word, the most frequent word of the predicted words in the corpus is returned. If this model does not return an input, the last token is used for the input in the bigram model.

This uses the back-off approach and the Markov assumption for predicting the nth word. If the bigram model still does not have a word to return as a prediction, the trigram part of speech prediction model is used to predict the next part of speech given the final two tokens' part of speech. This then returns the most common word from the predicted part of speech.

One way to improve the predictions of the model would be to add more text to the vocabulary and increase the number and variety of n-grams to account for different combinations of words users can input. Another way to help ensure that a word is predicted is by prompting the user to enter in a movie review. Because the corpus consists of 50,000 movie reviews, prompting the user to use words found in a movie review will improve the prospect of a model providing a predicted word because it is more likely that the user will enter in a phrase that is contained in the corpus. Finally, this model can be further improved by adding an option for the user to select if the predicted word is correct. These selections can be stored and used to give weights to different probabilities of predicted words for different inputs. It will also provide the opportunity for the model to learn from experience to give more weight to words that are correct and less weight to incorrect predictions. By using (n-1) gram history of the context, the n-gram models can use conditional probabilities to predict the nth word.

# Results

## Best models for application

To deploy a model on the application for any potential AI solution, models need to be compared and evaluated. Then, the way to evaluate models is to find the best metric that meets the business requirement. The application for this project is to have the sentiment analysis in real-time and to have the ability to predict the next word in aid of writing a review for users. Usually, people will think of using an accuracy rate as the comparison metric intuitively. However, when an application is built based on a certain business goal, other metrics such as precision and recall become equally important. Companies usually want their products to have fewer false negatives, having more true cases that were correctly predicted, and to have a higher precision rate, having a higher number of predicted cases that were truly true. In the case, the F-1 score, the harmonic mean of precision and recall, can be used for finding the best model by putting the tradeoff into consideration.

According to Table 5 and Table 7, the neural network model has the highest F-1 score on each of the classes, positive and negative, among others. It has 92.67% F1-score on negative class and 92.83% F1-score on positive class. Thus, the neural network model has been selected to deploy on the application for sentiment analysis tasks.

For the next word prediction task, the language model, from bigrams model to five grams model, was implemented on the application with a back-off approach, starting from the complex model to the simplest model. Due to the time pressure, the part of speech tagging feature was not implemented in the application.

**Table 13**. *Models used in production for the application*

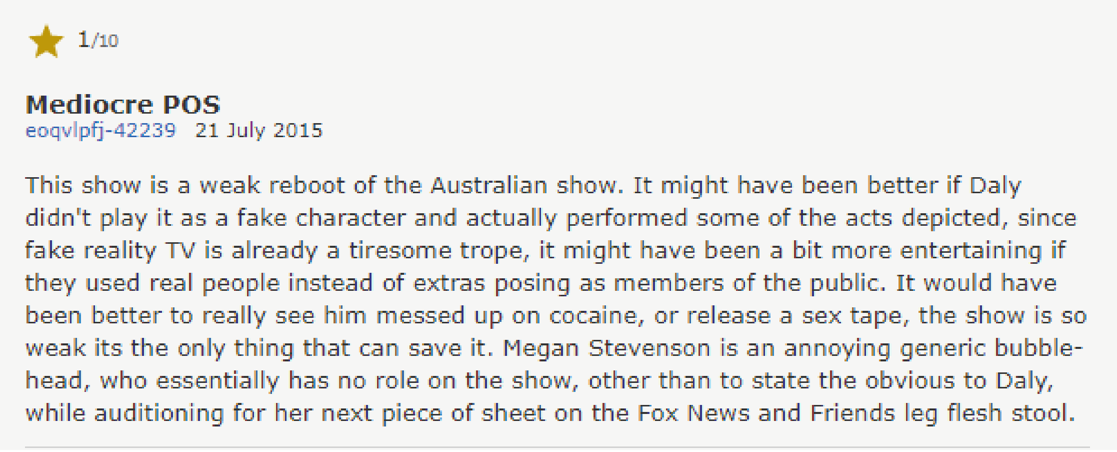
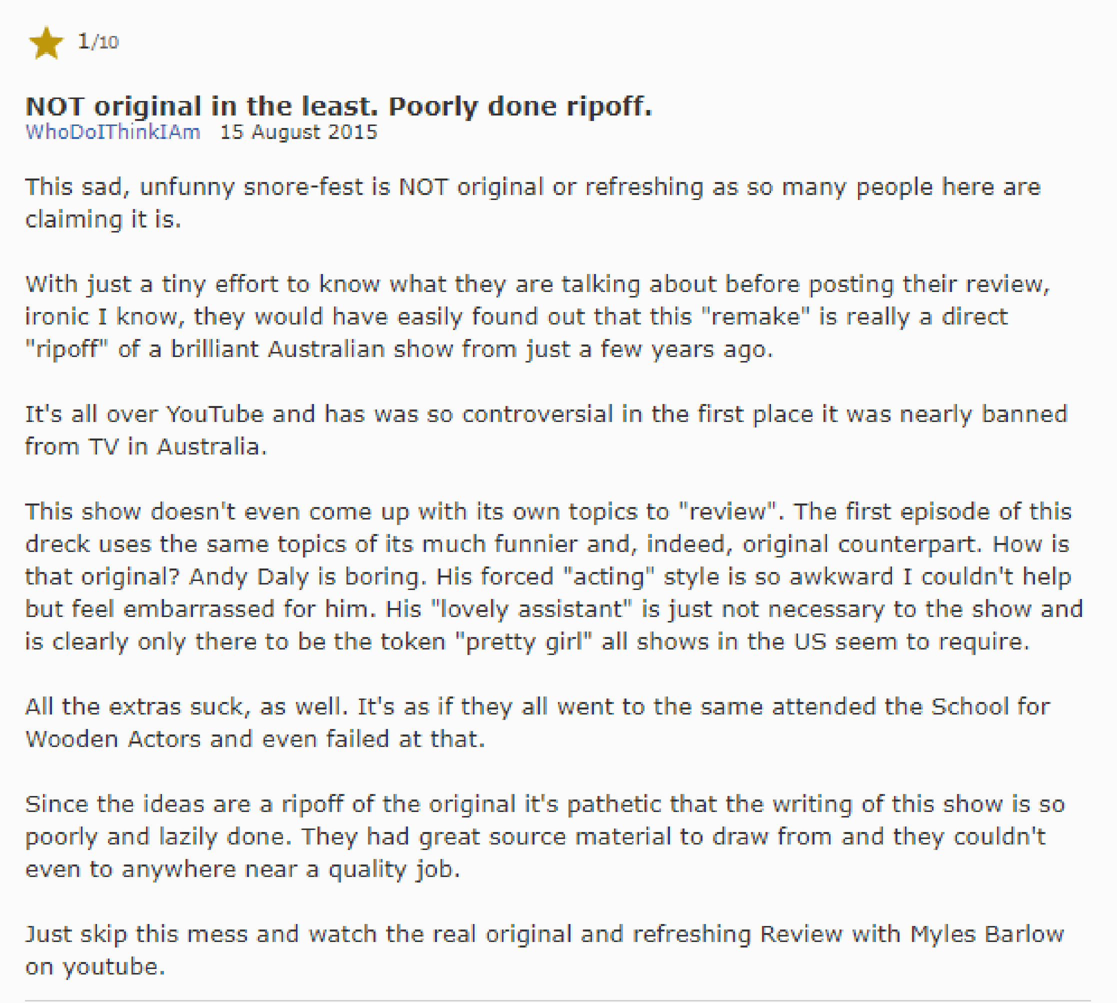
| Sentiment Analysis Task | Next Word Prediction Task |
|---|---|
| Neural Network Model, 93% accuracy | Bigrams, trigrams, four grams, and five grams model, combined with the back-off approach |

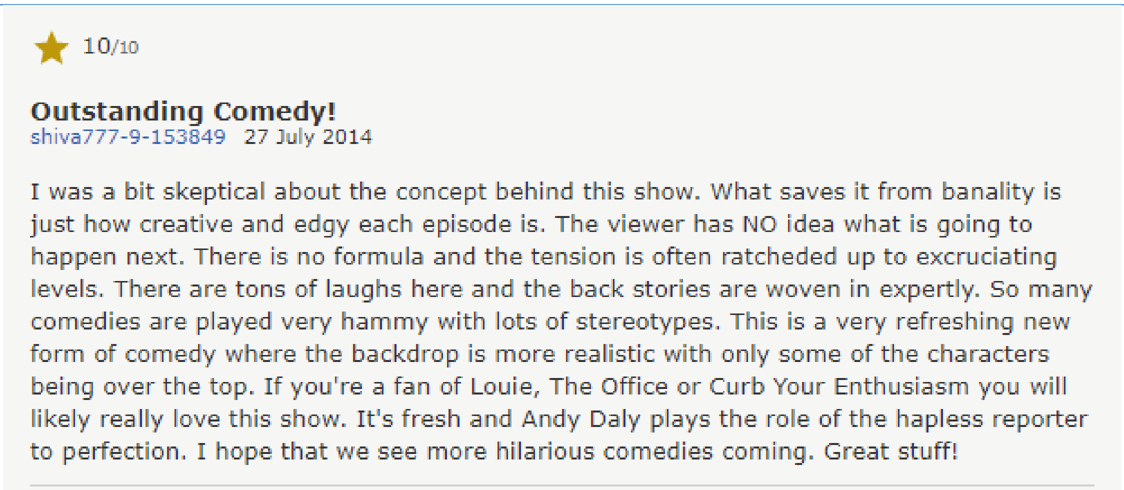
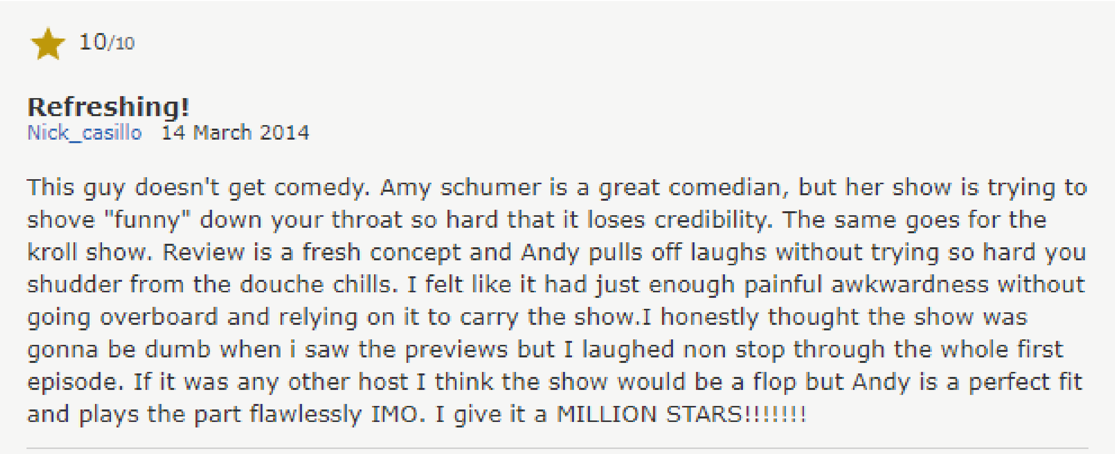
## Application results

### Sentiment Analysis of IMDB Movie Reviews

An application needs to be tested and evaluated even it has been successfully built. The goal for this process is to find if the product meets the business goal, to have the sentiment analysis in real-time, and to have the ability to predict the next word in aid of writing a review for users. Four reviews were sampled from the IMDB as the out of sample data to see if the application is meeting the business requirement. Two positive reviews and two negative reviews were used.

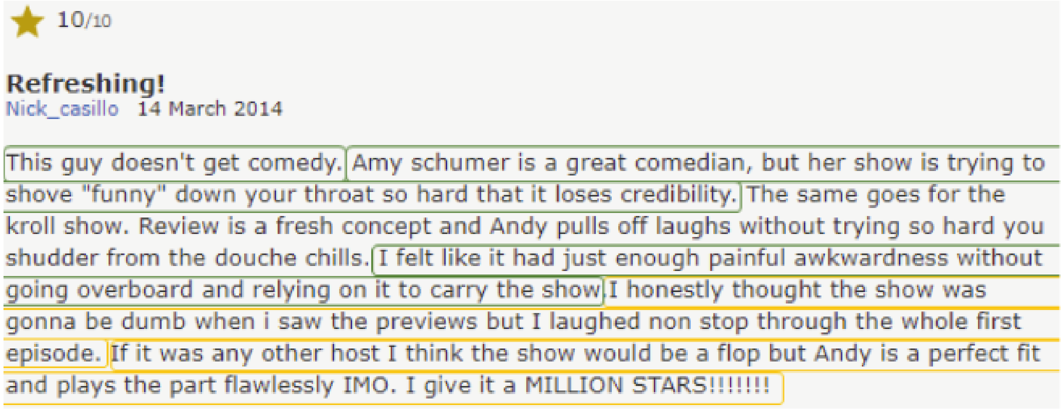
**Table 14**. Application test for out of *sample data from IMDB*

| # | Review | Predicted |
|---|---|---|
| 1 | <br>★ 1/10<br>**Mediocre POS**<br>eoqvlpfj-42239 21 July 2015<br>This show is a weak reboot of the Australian show. It might have been better if Daly didn't play it as a fake character and actually performed some of the acts depicted, since fake reality TV is already a tiresome trope, it might have been a bit more entertaining if they used real people instead of extras posing as members of the public. It would have been better to really see him messed up on cocaine, or release a sex tape, the show is so weak its the only thing that can save it. Megan Stevenson is an annoying generic bubble-head, who essentially has no role on the show, other than to state the obvious to Daly, while auditioning for her next piece of sheet on the Fox News and Friends leg flesh stool. | Negative Review ☹ |
| 2 | <br>★ 1/10<br>**NOT original in the least. Poorly done ripoff.**<br>WhoDoIThinkIAm 15 August 2015<br>This sad, unfunny snore-fest is NOT original or refreshing as so many people here are claiming it is.<br>With just a tiny effort to know what they are talking about before posting their review, ironic I know, they would have easily found out that this "remake" is really a direct "ripoff" of a brilliant Australian show from just a few years ago.<br>It's all over YouTube and has was so controversial in the first place it was nearly banned from TV in Australia.<br>This show doesn't even come up with its own topics to "review". The first episode of this dreck uses the same topics of its much funnier and, indeed, original counterpart. How is that original? Andy Daly is boring. His forced "acting" style is so awkward I couldn't help but feel embarrassed for him. His "lovely assistant" is just not necessary to the show and is clearly only there to be the token "pretty girl" all shows in the US seem to require.<br>All the extras suck, as well. It's as if they all went to the same attended the School for Wooden Actors and even failed at that.<br>Since the ideas are a ripoff of the original it's pathetic that the writing of this show is so poorly and lazily done. They had great source material to draw from and they couldn't even to anywhere near a quality job.<br>Just skip this mess and watch the real original and refreshing Review with Myles Barlow on youtube. | Negative Review ☹ |

| | | |
|---|---|---|
| 3 | <br>10/10<br>**Outstanding Comedy!**<br>shiva777-9-153849 27 July 2014<br>I was a bit skeptical about the concept behind this show. What saves it from banality is just how creative and edgy each episode is. The viewer has NO idea what is going to happen next. There is no formula and the tension is often ratcheded up to excruciating levels. There are tons of laughs here and the back stories are woven in expertly. So many comedies are played very hammy with lots of stereotypes. This is a very refreshing new form of comedy where the backdrop is more realistic with only some of the characters being over the top. If you're a fan of Louie, The Office or Curb Your Enthusiasm you will likely really love this show. It's fresh and Andy Daly plays the role of the hapless reporter to perfection. I hope that we see more hilarious comedies coming. Great stuff! | Positive Review 😊 |
| 4 | <br>10/10<br>**Refreshing!**<br>Nick_casillo 14 March 2014<br>This guy doesn't get comedy. Amy schumer is a great comedian, but her show is trying to shove "funny" down your throat so hard that it loses credibility. The same goes for the kroll show. Review is a fresh concept and Andy pulls off laughs without trying so hard you shudder from the douche chills. I felt like it had just enough painful awkwardness without going overboard and relying on it to carry the show.I honestly thought the show was gonna be dumb when i saw the previews but I laughed non stop through the whole first episode. If it was any other host I think the show would be a flop but Andy is a perfect fit and plays the part flawlessly IMO. I give it a MILLION STARS!!!!!!! | Negative Review ☹ |

From Table 14 shown below, the application successfully predicted two negative reviews, item 1 and item 2, as Negative Review. For the two positive reviews, one, item 3, was correctly predicted. However, the other review, item 4, was predicted as a Negative Review. This kind of error is a false negative. To further investigate this case, examine Figure 18 closely, there are three strong negative sentences at the beginning of the red boxes. There are few positive sentences at the end of the review as shown in the yellow boxes. It can be observed that some negative words are within the yellow boxes, such as “dumb” or “flop.” So, it might be ambiguous to the model. And, by having an ambiguous situation existing in the review, the application predicted this review as a negative review although its ground truth is positive review. This can be one of the issues in the application.

**Figure 18**. False negative breakdown



10/10

**Refreshing!**

Nick_casillo 14 March 2014

This guy doesn't get comedy. Amy schumer is a great comedian, but her show is trying to shove "funny" down your throat so hard that it loses credibility. The same goes for the kroll show. Review is a fresh concept and Andy pulls off laughs without trying so hard you shudder from the douche chills. I felt like it had just enough painful awkwardness without going overboard and relying on it to carry the show I honestly thought the show was gonna be dumb when i saw the previews but I laughed non stop through the whole first episode. If it was any other host I think the show would be a flop but Andy is a perfect fit and plays the part flawlessly IMO. I give it a MILLION STARS!!!!!!!

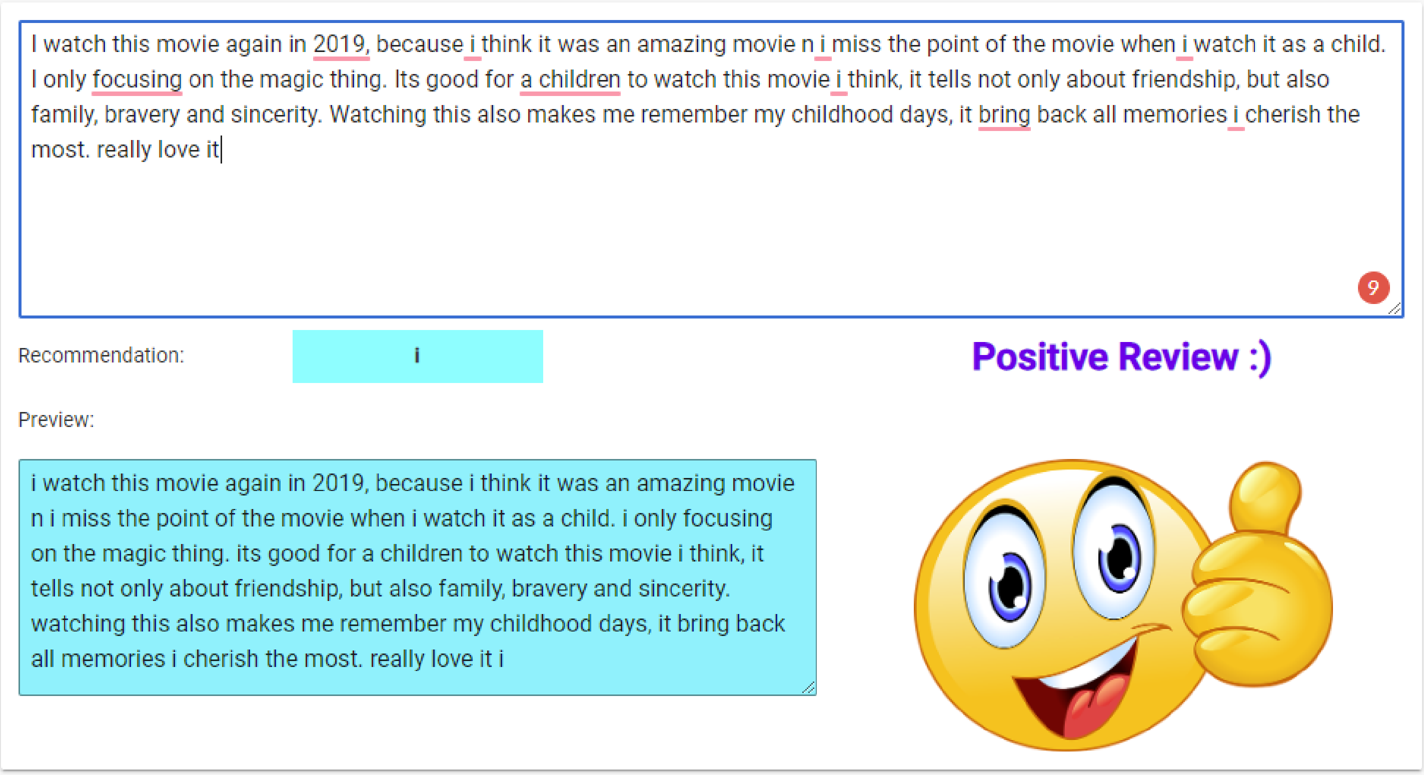
### Next Word Prediction for Movie Reviews

Several phrases, sentences were tested for the next word prediction. However, there is an interesting result in the application. That is to predict the next word for a sentence such as “I like to watch harry….”. Usually, people will expect the next word to be "potter" because of the famous sequel movies and it is the name of the main character. And surprisingly, by looking at Table 15 below, the more complex model such as the trigram model did not predict the desired word that people usually expect. Only, the bigram model can predict the word that people wanted which is “potter”. Because the language model is designed with the back-off approach. It starts to predict the next word from the complex model to the simplest model. In this case, the application cannot correctly predict the next word that people subconsciously expected. And, by looking at this case, it is obvious to conclude that complex model does not always work the best.

**Table 15.** *Application predicts the next word for the sentence, 'I like to watch harry …'*

| Models | Predicted Word |
|---|---|
| **Five-grams model** | ‘macy’ |
| **Four-grams model** | ‘macy’ |
| **Trigrams model** | ‘macy’ |
| **Bigrams model** | ‘potter’ |

**Figure 19.** *Screenshot of the application*



I watch this movie again in 2019, because i think it was an amazing movie n i miss the point of the movie when i watch it as a child. I only focusing on the magic thing. Its good for a children to watch this movie i think, it tells not only about friendship, but also family, bravery and sincerity. Watching this also makes me remember my childhood days, it bring back all memories i cherish the most. really love it

Recommendation: i

Positive Review :)

Preview:

i watch this movie again in 2019, because i think it was an amazing movie n i miss the point of the movie when i watch it as a child. i only focusing on the magic thing. its good for a children to watch this movie i think, it tells not only about friendship, but also family, bravery and sincerity. watching this also makes me remember my childhood days, it bring back all memories i cherish the most. really love it i

## Conclusion

A textual movie review tells customers, film producers, and other stakeholders about the strong and weak points of the movie, and more in-depth analysis, in general, could categorize significant numbers of movie reviews in sentiment categories like positive or negative. This type of analysis could help customers decide if the movie is worth their money. A summary of all reviews for a movie can help customers make this decision by not wasting their time reading all reviews.

Sentiment analysis of a movie review can rate how positive or negative a movie review is and, hence, it is overall rating. Therefore, the process of understanding if a review is positive or negative can be automated using classifiers algorithms and develop appropriate models to predict the review sentiment category using previously labeled data set like The Internet Movie Database (IMDb).

Words level sentiment classification provides an overall opinion of the document on a single example like movie reviews. This analysis presents an overview of the related work of sentiment analysis at the word level, mainly the approach of classifier models, as machine learning was considered as dominance at this level. The text classifiers and the text representation used (words) in general, identify the sentiment expression on the movie review data set that was selected for the analysis. The analysis used two types of classifiers and compared their performance metrics setting on the different parameters. The results reached an overall performance of almost 90 percent of accuracy. For non-technical readers, this means that they can use the model to classify movies reviews. However, it is also required to consider some limitations and the possibility of classifying a good movie as negative. This tradeoff will be significant or not depending on the context and the importance of the decision. For example, 90 percent accuracy could be noncompliance for a movie business stakeholder.  Whatever the context, the simple method always is the most recommended, particularly for sentiment analysis using "big" text data. In the marketing context, almost all businesses understand the importance of reviews for product design and marketing, yet only a few uses the right tools to analyze its impact. For costumers, this analysis will help them make better buying decisions or identify fake reviews on products or services.

Social related review mining is a challenging task because reviews are written with mixed real-life review data and ironic words. Existing sentiment analysis models can be improved further with more semantic and commonsense knowledge. Those models will require sophisticated approaches and update datasets to be used as a realizable tool for review categorization, particularly for the classification of positive and negative reviews.

## Reference

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

Shi H., Liu Y. (2011) Naïve Bayes vs. Support Vector Machine: Resilience to Missing Data. In: Deng H., Miao D., Lei J., Wang F.L. (eds) Artificial Intelligence and Computational Intelligence. AICI 2011. Lecture Notes in Computer Science, vol 7003. Springer, Berlin, Heidelberg

Essays, UK. (November 2018). The Importance Of Online Reviews. Retrieved from https://www.ukessays.com/essays/film-studies/the-importance-of-online-reviews-film-studies-essay.php?vref=1

Lakshmi Devi B., Varaswathi Bai V., Ramasubbareddy S., Govinda K. (2020) Sentiment Analysis on Movie Reviews. In: Venkata Krishna P., Obaidat M. (eds) Emerging Research in Data Engineering Systems and Computer Communications. Advances in Intelligent Systems and Computing, vol 1054. Springer, Singapore

Github repository for CSV file and notebook:
https://github.com/TheClub4/IMDB_Sentiment_Analysis

https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/kernels

Tun, Wai. (2016). Sentiment Classification of Movie Review Comments using Naive Bayesian Model. 10.13140/RG.2.2.26270.13122.

https://towardsdatascience.com/imdb-reviews-or-8143fe57c825

https://ecs.utdallas.edu/research/researchlabs/msp-lab/publications/Burmania_2016_2.pdf

https://medium.com/towards-artificial-intelligence/naive-bayes-support-vector-machine-svm-art-of-state-results-hands-on-guide-using-fast-ai-13b5d9bea3b2

https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

https://www.google.com/url?sa=i&url=https%3A%2F%2Farxiv.org%2Fpdf%2F1612.01556&psig=AOvVaw1DH7Hk60_86B-uASfQ_5ll&ust=1589773903663000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCLCU-MH_uekCFQAAAAAdAAAAABAE

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

Rizvi, Mohd Sanad Zaki. (2019, August 18). A comprehensive guide to build your own language model in Python! Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-language-model-nlp-python-code/