Thumbs Up, Thumbs Down:

Using Natural Language Processing for Content Suggestion in the Streaming Industry

Peter Adam Aboud
Department of Computer Science
Ryerson University
Toronto, Canada
adam.aboud@ryerson.ca

I. Introduction and Motivation

It is difficult translating natural language into usable information for data analysis and software tools. This problem is notable in the content streaming industry, and for companies such as Netflix, Amazon, and Disney+. Being able to suggest content that will hold subscribers' attention is critical to maintaining an active and engaged user base. Accurately suggesting new content to users requires multiple layers of data gathering, synthesis, and analysis. This report examines the potential of using natural language processing (NLP) and machine learning to turn natural language about content into useful information for the suggestion of new content.

Whether talking to a stranger on the street or your best friend, the process of recommending a restaurant, tv show, or movie is the same. First, preferences of the other person must be identified, secondly all options must be evaluated in terms of those preferences, and finally the recommendation can be made. For example, your friend asks you for a recommendation for a new band. Knowing your friend, you already know she likes lots of different instruments and dancing, so you recommend her Toronto's own Glen Miller Orchestra. You can successfully generate this recommendation because you have subconsciously (or consciously if you are a REALLY good friend) taken stock of the things she has said about music and made some internal judgement about her preferences. Content streaming services can do something similar, looking at the user's activity they are able to draw several conclusions about what the user may prefer. Many streaming services identify relationships between their content allowing them to say: "if you liked this, try this." What if these streaming services were able to learn not just from user activity but also from their user's natural language, what users say about their service and the content they watch.

The goal of this project is to provide some of the early pathfinding required to implement a more comprehensive system for content suggestion. This "pathfinding" will consist of the development and testing of various machine learning models for effective binary classification of natural language reviews. The central idea is that by learning what language samples are positive, key words and features can be identified as positive indicators of enjoyment. And therefore, new content that shares these features can be suggested to the user. It should be noted at this point that this binary classification of positive/negative sentiment is understood as simply a first step in the development of a more comprehensive system for content suggestion. A complete description of the full system will be contained in the appendix of this report.

II. Problem Statement and Dataset

This project endeavours to showcase some effective models for the classification of sample texts as exhibiting either positive or negative sentiment. These models will be able to predict whether new input text is either positive or negative based on the constituents of the text. Services such as Netflix and Amazon may be able to apply these models to the ways in which their users review or talk about their content, and ultimately provide insight into certain customer segment content preferences.

Without access to genuine content streaming user's natural language data, this is simulated with movie reviews. This dataset was gathered from Kaggle, an online platform for all things data science. It should be noted here that the data set used for the training and development of these models is perfectly balanced, with 50% of the texts labeled positive, and 50% negative. This dataset contains reviews written by professional movie critics. As such, it is not a perfect replacement for user generated natural language. It is safe to assume that the data contains language that may not be common for every-day users. The Kaggle dataset was pared down into three sets: training (4000 samples), validation (500 samples), and test (500 samples).

III. Methods and Models

A. *Text Normalization and Pre-Processing*

The first step in natural language processing is the normalization and pre-processing of the input data. This includes:

- Tokenization – the separation of blocks of text into individual words.
- Lemmatization – the grouping together of various forms of the same word and their reduction to a single form, e.g., can't, cannot, can not.
- Stop-word filtering – the removal of words from a sample text that contain little or no relevant linguistic meaning to the sentiment of the text e.g., "the."

These are all done to create a sample text that is easier to work with, and more indicative of sentiment. The python library, Natural Language Tool Kit (NLTK), provides most of this functionality.

B. *Feature Engineering:*

Once the sample texts have been processed, a group of features is to be selected. These features (for NLP) are the words that the engineer believes to be important for determining the sentiment of the given text. It is therefore vitally important that these features are chosen carefully. For sentiment analysis, these features are the words that are believed to be uniquely indicative of either positive or negative sentiment. This group of words is referred to as the vocabulary. This examination looks at two different methods of vocabulary creation, what are termed heretofore as 'local', 'global', and 'categorical'.

1) *Local vocabulary collection.*

- After the sample texts in the training data have been pre-processed, the frequency of each word within that sample is determined. The *n* most common words are added to the vocabulary collection, not including those that are already contained in the vocabulary. This was the first method implemented, and it quickly became apparent that it has flaws. These will be discussed in the discussion section

2) *Global vocabulary collection.*

- As the name suggests this method looks at the frequency of individual words across the training set as a whole. It uses all the words that occur more than five times globally. This is a more accurate representation of words that represent the sentiment of the corpus.

3) *Categorical vocabulary*

- This method takes all the words in the corpus of sample texts and examines the occurrence rate of each word for each classification. If there is a significant divide between these values, the word or feature can be understood to be more relevant for a specific classification than mere coincidence. The motivation for this method of feature selection will be discussed in the discussion section.

C. *Feature Extraction*

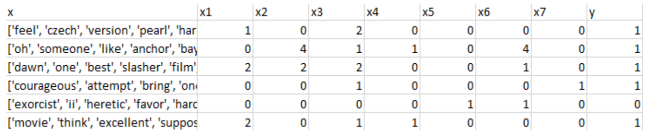
Once the vocabulary has been created, a Pandas *dataframe* is updated to include columns for each feature. Each row of the *dataframe* pertains to an input sample text, and the values in the column are updated with feature 'scores' that represent the relevance of that particular feature to the input sample. The scores are calculated in one of two ways.

1) *Term frequency (tf):*

- Simply put the score is the number of occurrences of that feature in the sample text.

2) *Term frequency inverse document frequency (TFIDF):*

- This metric is a statistical representation of the importance of a word to a document that exists in a collection. The importance of a particular word increases when it appears frequently within that document but occurs infrequently in the other documents.
  - $tf(x_j^{(i)}) = \frac{number\ of\ occurences\ of\ word\ x_j^{(i)}}{length\ of\ the\ sample\ text}$
  - $idf(x_j^{(i)}) = \log \frac{number\ of\ texts}{number\ of\ texts\ with\ at\ least\ one\ occurence}$
  - $tfidf(x_j^{(i)}) = tf(x_j^{(i)}) * idf(x_j^{(i)})$
  - Where $x_j^{(i)}$is the jth feature for the ith sample text.



x	x1	x2	x3	x4	x5	x6	x7	y
['feel', 'czech', 'version', 'pearl', 'har	1	0	2	0	0	0	0	1
['oh', 'someone', 'like', 'anchor', 'bay	0	4	1	1	0	4	0	1
['dawn', 'one', 'best', 'slasher', 'film',	2	2	2	0	0	1	0	1
['courageous', 'attempt', 'bring', 'on	0	0	1	0	0	0	1	1
['exorcist', 'ii', 'heretic', 'favor', 'harc	0	0	0	0	1	1	0	0
['movie', 'think', 'excellent', 'suppos	2	0	1	1	0	0	0	1

Fig.1 Feature extraction table using *term frequency*

Once the *dataframe* has been updated, the models can start to be trained, that is adjusting the weight or importance each feature has in predicting the sentiment.

D. *Models*

The following models are the methods by which the weights that are ascribed to each of the features were determined. These models are 'trained' with data from a table such as the one in Figure 1. Once the weights of each feature have been determined these weights can be applied to new data and a prediction can be made.

1) *Naïve Bayes:*

- This approach uses probabilities that each feature is indicative of a positive or negative sentiment given the input training data.
  - $\phi_{j|y=1} = \frac{\sum_{i=1}^{n}\left[X_j^{(i)}=1, Y^i=1\right]+1}{\sum_{i=1}^{n}[Y^{(i)}=1]+d}$
  - Where $\phi_{j|y=1}$ is the probability that the jth feature exists in a positive sample.
  - Where $[X_j^{(i)} = 1, Y^i = 1]$ represents the count of jth feature existing in a positively labeled sample.

- Where $[Y^{(i)} = 1]$ is the count of positive samples.
- Laplace smoothing was added.

- It should be noted that the above equations only represent one half of the process, the same is conducted for probabilities of features and negative samples.

- These probabilities are then used to predict the classification of new data. This is accomplished by finding the product of multiplying the number of times each feature takes place in the sample by its sentiment probability. If the positive sentiment probability for each sample is greater than the negative, it can be predicted to be a positive text.
  - $P(X_1, X_2, \dots, X_d|Y) = P(X_1|Y)P(X_2|Y) \dots P(X_d|Y)$

2) Logistic Regression:

- This model defines a cost function and iteratively seeks to find the local minimum by a process called gradient descent.
  - $j(\theta) = y^{(i)} - \frac{1}{I+e^{-\theta^T x^{(i)}}}$
  - Where θ are the list of feature weights, and $x^{(i)}$ are the feature values for that sample.
- Weights are iteratively updated with the following:
  - $\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \frac{1}{1+e^{-\theta^T x^{(i)}}}\right) x_j^{(i)}$

3) Support Vector Classification (SVC):

The third model implemented was a Support Vector Machine. A non-probabilistic binary linear classifier, this model creates a representation of the examples as points in space. Such a rendering allows for a determination of linear separations between classifications. New examples are mapped into the same space, and their position relative to the linear separation determines their classification. This model can utilize data distributions that are not linearly separable by using what is called a kernel trick. This is accomplished by mapping the data to higher dimensions.

E. Validation and Testing

To make definitive statements about the efficacy of these models, a rigorous validation method was implemented. K-fold validation was carried out for each model where $k = 10$. This process involves splitting up the validation data set into *k* piles. The model is trained on *k-1* piles, and the remaining pile is used as a test set. This is repeated *k* times, such that each pile is used as a test set. The mean accuracy for the *k* iterations is the final output value of the cross-validation, refer to figure 2.

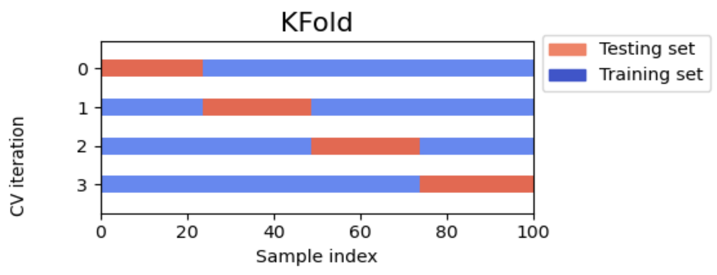


Figure 2. K-Fold validation visualization

IV. Results and Discussion

The investigation for a suitable model began with a Naïve Bayes implementation. This was constructed from scratch by consulting course material from Dr. Nariman Farsad's *CPS803 Machine Learning* course taken at Ryerson University. This model uses only the Pandas and Numpy libraries. This model returned reasonable results on the training set of 4000 examples, with a peak accuracy of ~80%, see Figure 3.

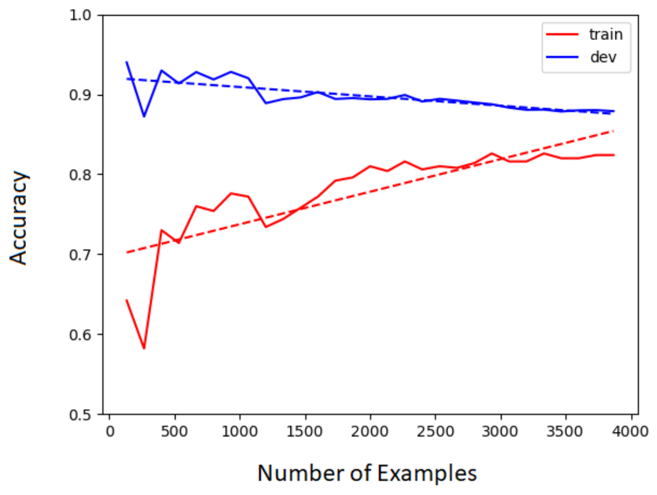


Figure 3. Bias-Variance graph for Naïve Bayes model, with TF feature extraction and local vocabulary selection.

Figure 3 demonstrates the model's steady improvement in accuracy as the training set increases. "Train" represents the accuracy of the model being trained on x-examples and tested on the test set. "Dev" represents the accuracy of the model being tested on itself. The decrease in dev accuracy is a result of the vocabulary creation method. The 15 most common words are taken from each sample, but only if they are not already represented in the vocabulary. As a result, there is no linear positive correlation between training set size and vocabulary size. This is an issue, so a new feature selection method was implemented to try and overcome this.

To improve performance, a global vocabulary creation was implemented to broaden the applicability of each individual feature. This prevents the vocabulary from unrealistically weighting words/features that are only present in a few individual samples, and from succumbing to the issue with Figure 3. Also switching to a TFIDF feature extraction method more accurately represents the importance of a particular feature within a corpus. This resulted in a moderate improvement, with a final accuracy of ~84%, see Figure 4.

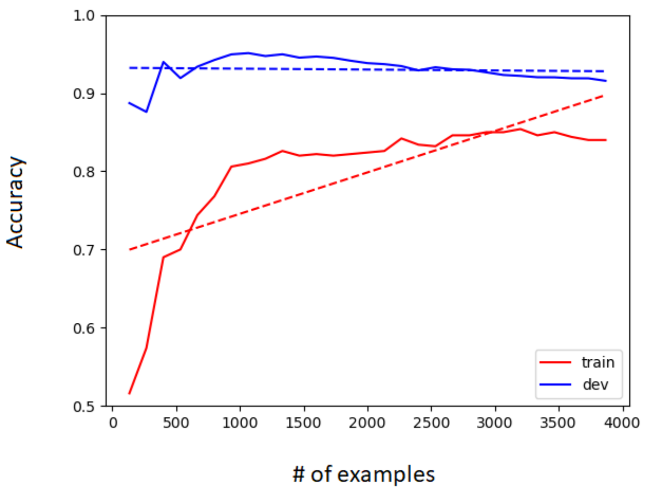


Figure 4. Bias-Variance graph for Naïve Bayes model, with TFIDF feature extraction and global vocabulary selection.

Accuracy is not the only, or the best, way of evaluating a model. It tells the creator nothing about how the model is failing. To better understand the performance of the model it is important to construct what is called a *confusion matrix*. Each row represents the instances in a predicted class while each column represents the instances in an actual class, refer to figure 5.

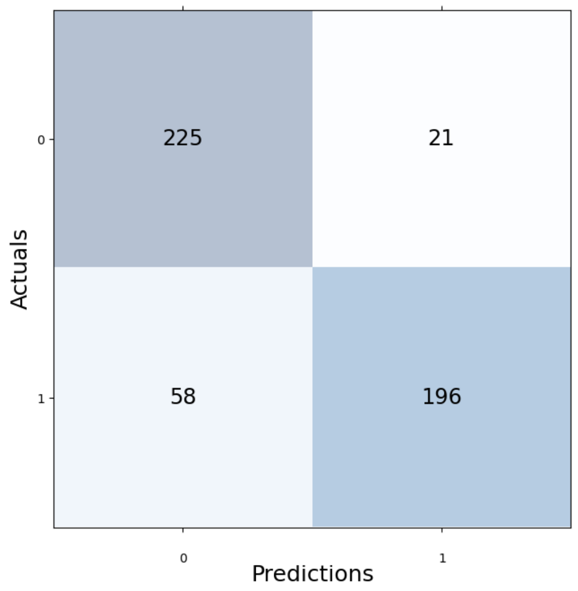


Figure 5. Confusion matrix for Naïve Bayes model, with TFIDF feature extraction and global vocabulary selection.

Figure 5 shows that the model is making mistakes when it comes to predicting negative reviews. It is more likely to mislabel a negative review as positive than vice versa. This does not have any meaningful implications for what is needed to use this model for, however a more balanced error distribution would be something to strive for.

With some early success via a probabilistic model, the investigation turned to non-probabilistic methods for classification, i.e., Logistic Regression and Support Vector Classification. The logistic regression model was built by the author, while the SVC model was implemented using *Sklearn*. These models, and the probabilistic model were all validated via k-folding the validation set. Every combination of feature extraction, feature selection, and model was explored, and the results are captured in Figure 6.

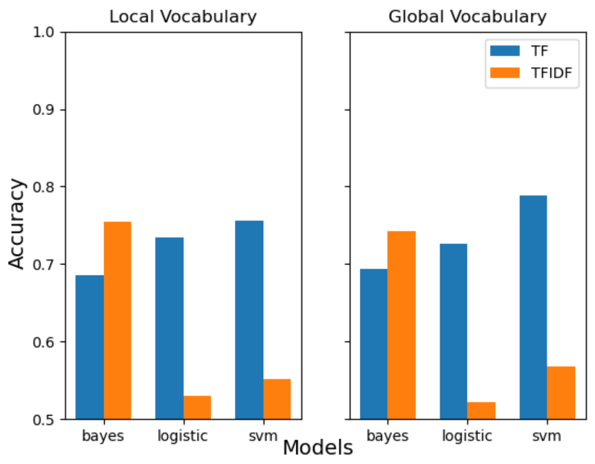


Figure 6. K-fold validated accuracy rates for the models explored.

A keen eye will notice that the values in the above figure are significantly lower than those purported earlier in this report. This is because the validation set, which the cross-validation was carried out on, is a tenth of the size of the normal training set. The cross-validation scores above are simply a confirmation that the models maintain their performance across a variety of training and test sets, i.e., they are not being over fit. Furthermore, switching from TF to TFIDF seems to drastically reduce the performance of the non-probabilistic models (LR and SVM). The exact cause for this reduction was not discovered by the time this report was authored. This will be an ongoing line of inquiry for the duration of the project.

Figure 6 shows that the most effective model is SVC with global vocabulary and TF feature extraction. On the original training set this model configuration attains accuracy scores of ~88% and is the best product this investigation provided to date.

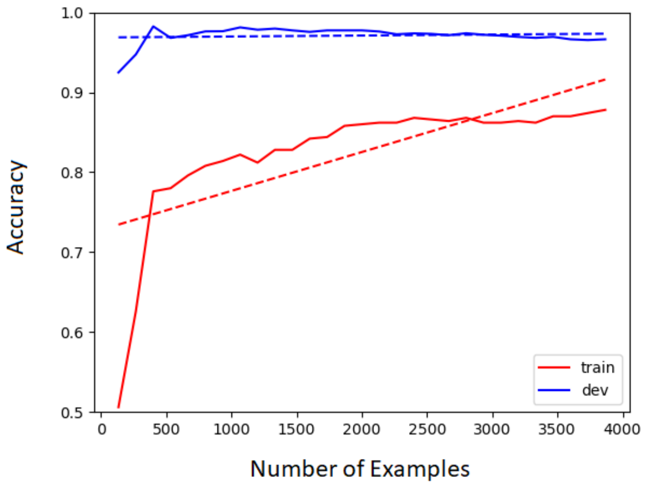


Figure 7. Bias-Variance graph for SVC model, with TF feature extraction and global vocabulary selection.

All these charts and graphs about accuracy show that the models are functioning, and that they are a relatively effective means of classifying sentiment of a given sample text. However, they also indicate which words and features are most indicative of sentiment. Recall that training a model means finding the 'weight' of a given feature. This can be understood to mean the relative importance of that feature or word for the specific classification. Therefore, the features with the highest weight can be understood to be the most important for the model's classification process. This can be most easily visualized via word clouds, see Figure 8.

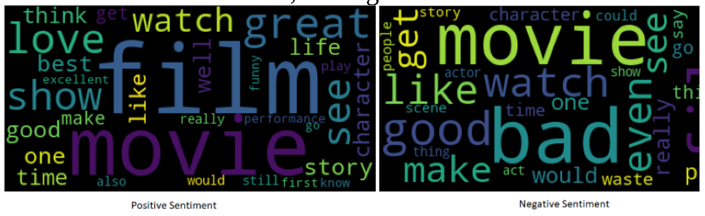


Figure 8. Word cloud representation of the most impactful features for classifying sentiment.

The above figure alerted the investigation to an interesting phenomenon. Looking carefully at the word clouds indicates that words that one would expect to indicate negativity or positivity are represented accordingly, however the word clouds also share features. How is it that 'film', 'movie', 'watch', and 'good' can be indicators of both positive and negative sentiment. This highlights the importance of comprehensive feature selection. To correct this, another feature selection method was implemented by building on the existing 'global' method. By examining the occurrence of each feature in the positive and negative samples, that feature can be correctly categorized, hence the term 'categorical vocabulary selection.' For example, the word "movie" may occur 250 times in positive samples and 280 times in negative samples. The occurrences of this word are so close in value, that it cannot be associate with positivity or negativity and therefore it should be left out from the vocabulary. For contrast, "great" shows up in positive samples 600 times, and in negative samples 300 times. The difference is so great that is fair to assume that this word can be categorized as positive, and therefore should be added to our vocabulary. This investigation determined that a 30% difference between occurrence rates was sufficient in filtering out the features that were previously contributing to both sentiment classifications. After implementing this feature selection method, model accuracies improved across the board, and the word cloud visualizations generated are much more in line with expectations, see Figure 9 and 10.

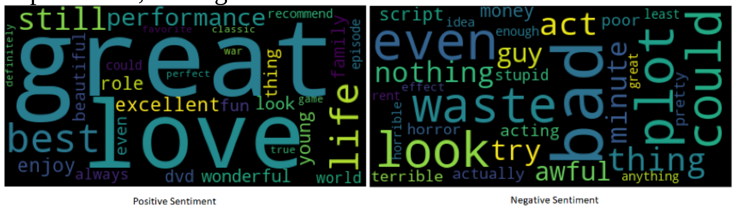


Figure 9. Word cloud representation of the most impactful features using 'categorical vocabulary selection'

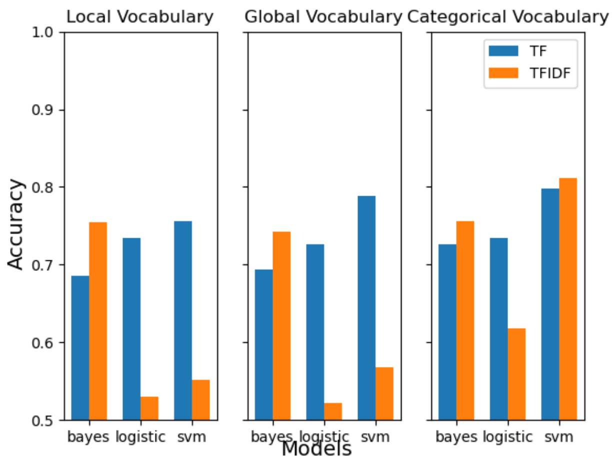


Figure. 10 Validated accuracy scores with categorical feature selection

V. Conclusions and Implications

This investigation sought to find a suitable method for the classification of natural language reviews based on sentiment. As demonstrated by the previous section this investigation can conclude that it is vital to correctly balance model parameters, feature selection methods, and feature extraction variants. This report tried to convey the thought process of this investigation, as one that is constantly self-reflective and critical of the mechanisms which are producing ongoing results. This development methodology can be visualized in Figure 11.

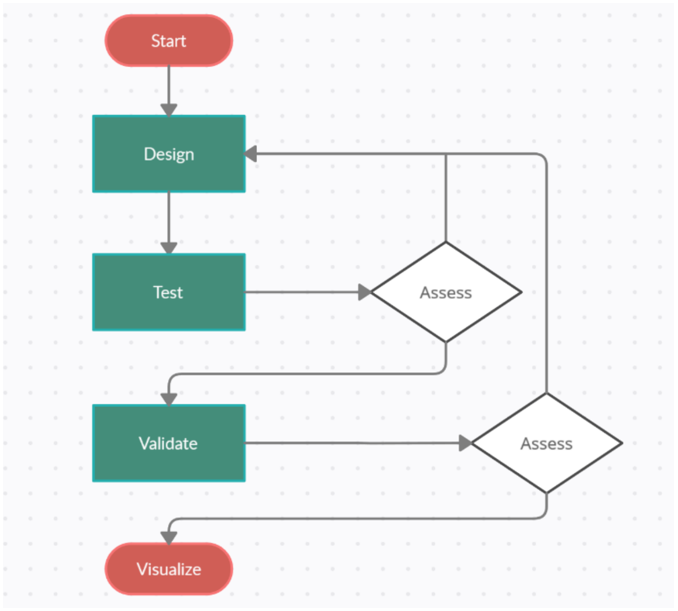


Figure 11. Development methodology for NLP model construction

This is a potentially unending cycle of redesign, test, and evaluation. However, this project is not just about finding the best way to model sentiment classification, but to act as proof of concept for a wider content suggestion system. Recall in the introduction the example of your friend who wants a new music recommendation. The process by which you made your selection looked something like the following:

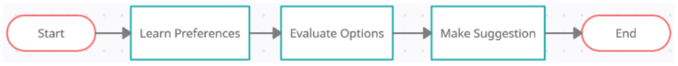


Figure 12. Diagram showing the flow of making a recommendation or suggestion.

If the goal is to be able to make good recommendations to users of content streaming services, then the same process must be followed. In creating an effective model for positive and negative classification of text the first stage is complete. The model learns exactly what words are used to convey positive feeling towards certain content, as visualized by the word clouds (Figure 9). The second and third phases represent future work required for this project. First, a word2vec implementation will help expand what is known about the user groups preferences. Synonyms, and related concepts can be considered as indicative of the same sentiment and therefore can also be included. Word2vec will be able to identify these other features. Secondly, new content will be 'evaluated' by analyzing descriptive text of each title and attributing a score that reflects how similar it is to the features that word2vec and the models have identified. Third, it is important to determine the score threshold that needs to be exceeded for making a 'confident' suggestion, and finally forward results and make recommendations.

A simplistic example might proceed as follows:

- Aggregate tweets or reviews from Canadian young adults.
- NLP model shows that features: 'intense', 'action', and "beautiful" are most important when determining a sample is positive.
- Word2Vec expands our feature list to include: 'chaos', 'fighting', 'explosions', 'hot', and more.
- New content descriptions are scored, and those that exceed threshold value are forwarded as suggestions, including:
  - Fantasy Island (2020) – "When the owner and operator of a luxurious island invites a collection of guests to live out their most elaborate fantasies in relative seclusion, chaos quickly descends."

Clearly more work remains in implementing the entire system. It is the hope of the author that a proof of concept has been demonstrated. Namely that natural language samples from users may be used as fuel for new content suggestion. The effective implementation of this system or an analogous one would represent another tool for content streaming services to keep their users engaged on their platform.

VI. References

[1] https://github.com/OneHandKlap/thumbs_up

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

[3] Farsad, N. (2020, September 26). *Logistic Regression*. Lecture presented in Ontario, Toronto.

[4] Farsad, N. (2020, October 9). *Naïve Bayes*. Lecture presented in Ontario, Toronto.

[5] Farsad, N. (2020, October 16). *Support Vector Classification*. Lecture presented in Ontario, Toronto.

[6] Arlot, Sylvain; Celisse, Alain (2010). *"A survey of cross-validation procedures for model selection"*. Statistics Surveys. **4**: 40–79. *arXiv:0907.4728. doi:10.1214/09-SS054.*

[7] Amatriain X., Pujol J.M., Oliver N. (2009) I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. In: Houben GJ., McCalla G., Pianesi F., Zancanaro M. (eds) User Modeling, Adaptation, and Personalization. UMAP 2009. Lecture Notes in Computer Science, vol 5535. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02247-0_24

[8] https://numpy.org/

[9] https://pandas.pydata.org/

[10] https://scikit-learn.org/stable/index.html