Article Theme Classification Based on Titles

Introduction

Numerous articles are being posted on the internet on a daily basis. For a platform that facilitates members sharing their ideas through writing and reading, creating an index system for information of such a high volume can be a challenging task. Although many approaches can be taken to design such systems, there are two general families, keyword-based and theme-based.

Keyword-based systems give authors the freedom to label their own work. By specifying keywords or #tags, the authors allow readers to search for information that aligns with their interest through author-generated labels. Albeit very flexible and dynamic, a keyword-based indexing system can be very costly to implement and maintain. Because keywords may be submitted by the author in a much more arbitrary way, it embeds uncertainty in the process of searching and retrieving information. For instance, a word may have various cases and derivations, and different words can also be semantically similar. In order for the readers' search to hit the target accurately and comprehensively, the index system has to be able to understand different cases and variations of the same word, as well as establish semantic similarity.

On the other hand, a theme-based system only allows articles to be labeled with predefined topics. Instead of dealing with a potentially infinite number of keywords, a theme-based indexing system can be more organized, and help readers quickly narrow down the scope of their search. Such advantages make it an ideal complementary system to keywords.

This project aims to explore different machine learning approaches that can automate the task of classifying articles into different themes according to their titles. The final classification algorithm should be able to suggest possible themes based on titles. The semi-automatic algorithm should suggest the most likely categories for the publisher/creator to choose from and help promote a consistent framework that facilitates a more robust and efficient indexing and searching system.

Data Source

The data source comes from Kaggle (Medium Post Titles — Medium Post Titles, Subtitles, Categories, https://www.kaggle.com/nulldata/medium-post-titles). It contains the titles of 126,095 articles posted on Medium (https://medium.com). These articles belong to and are labeled with 93 different themes spanning various STEM areas, arts and social sciences.

Data Cleaning

The texts in the titles contain punctuation marks, which were not needed in the analysis. Therefore, all punctuation marks were removed. Table 1 shows the examples of the texts before and after cleaning.

CATEGORY	TITLE	TITLE (CLEANED)
WORK	"21 Conversations" - A fun (and easy) game for...	21 conversations a fun and easy game for team...
SPIRITUALITY	"Biblical Porn" at Mars Hill	biblical porn at mars hill
LGBTQIA	"CISGENDER?! Is That A Disease?!"	cisgender is that a disease

EQUALITY	"Call me Nat Love" :Black Cowboys and the Fron...	call me nat love black cowboys and the frontie...
ARTIFICIAL-INTELLIGENCE	"Can I Train my Model on Your Computer?"	can i train my model on your compute

Table 1. Comparison between texts before and after cleaning

Exploratory Analysis

In this dataset, different categories do not occur equally -- the frequency of different categories are illustrated in Figure 1. This indicates that when holding part of the data as a test set, the split should be stratified.

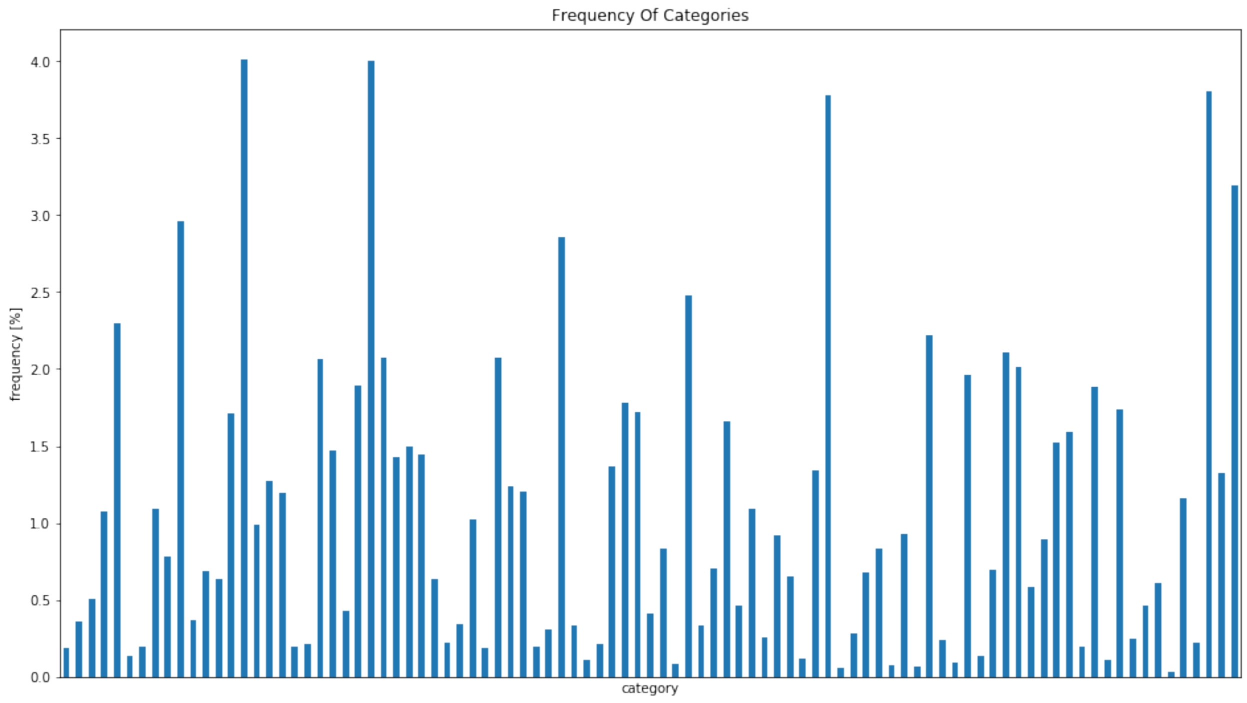

Figure 1. Occurrence of categories

I first used the bag-of-words model to extract features from the texts. Using TF-IDF, for each title, a vector representation was created. For exploratory purposes, the vocabulary was constrained to unigrams and bigrams that appear more than twice but less than 50% of times throughout the dataset. Common English stop words – (prescribed by *scikit-learn*) were also ignored.

Averaging the vectors within a category, I was able to generate a mean vector representation for each theme. Since each element in the vector indicates a categorical mean TF-IDF score for a word, the mean vectors can be visualized as wordcloud charts.

The generated wordclouds shows that the vector representations were able to capture the most prominent features shared by titles within the same category. The high-score words picked by the model all seemed reasonable given the corresponding themes (Figure 2).

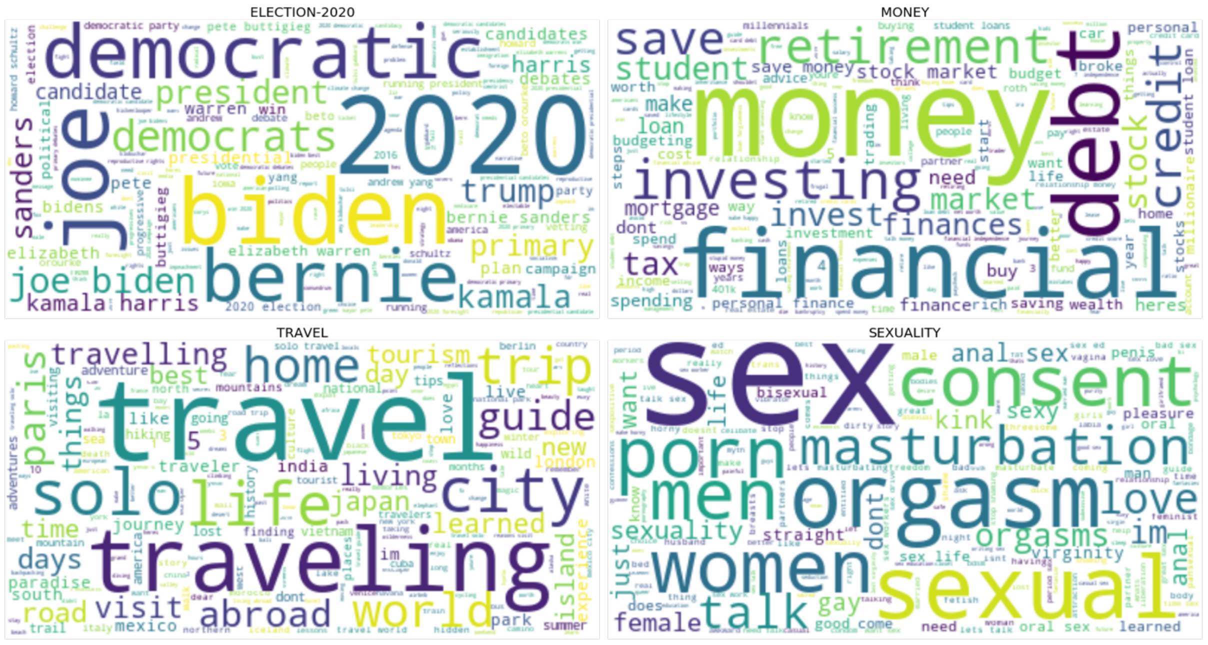

Figure 2. Wordcloud visualization of some categories based on mean TF-IDF vector representation

Cosine similarities were computed between the mean vector representations of each theme (Figure 3). By ordering the cos-similarity matrix based on hierarchical linkage, I was able to group together categories with intuitive relations. For instance, *artificial-intelegence*, *data-science* and *machine-learning* are grouped together; *ux*, *design* and *visual-design* are grouped together. Thus, when I assigned each category an integer label according to their order lined up by the hierarchical clustering based on cosine similarity, conceptually related themes got neighboring labels.

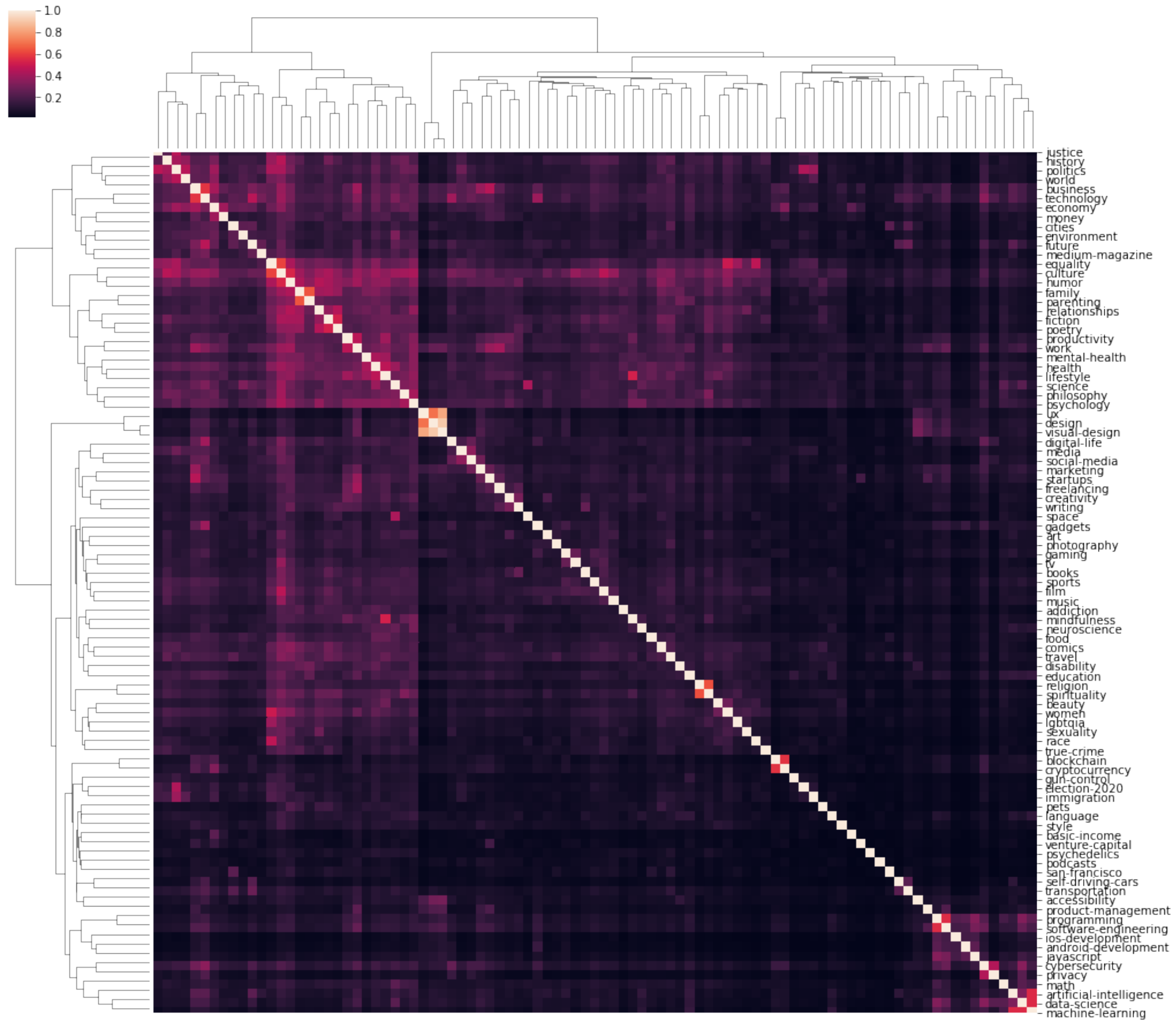

Figure 3. Cosine similarity matrix

Naïve Bayes Classifier

I built a naïve Bayes classifier attempting to assign the correct themes for the titles. 20% of the data were held out as the test set while 80% of the data were used during the training of the model.

The training set was first used to build a TF-IDF vectorization model. The complete data set were then transformed into the TF-IDF space. I used the multinomial naïve Bayes model for the classification task. The choosing of hyperparameters were completed through a 5-fold cross validation search, which determined the range of n-grams, the threshold and cut-off frequency for a word to be included in the vocabulary and the pseudo-count for additive smoothing.

The best choice of hyperparameters generated an accuracy of about 35% on the test set, while the model achieved a multiclass precision of 46% on average. Figure 4 shows the macro-average of multiclass precision and recall. To achieve an average precision more than 90%, the decision threshold should be no less than 0.975. At that level, the average recall is 0.08%.

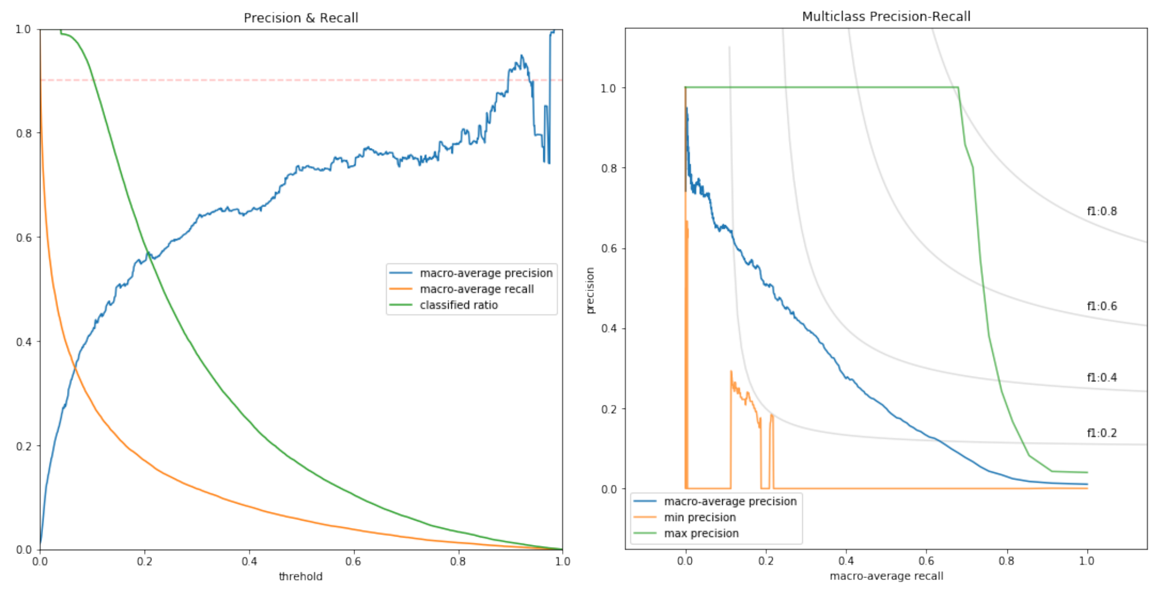

Figure 4. Precision-recall of naïve Bayes model

The confusion matrix generated from the test data (Figure 5.) points out some major pairs of misclassification, partly because those involved categories are tightly related -- *venture-capital* misclassified as *business*, *basic-income* misclassified as *economy*, *election-2020* misclassified as *politics*, *lgbtqia, race* and *women* misclassified as *equality*…

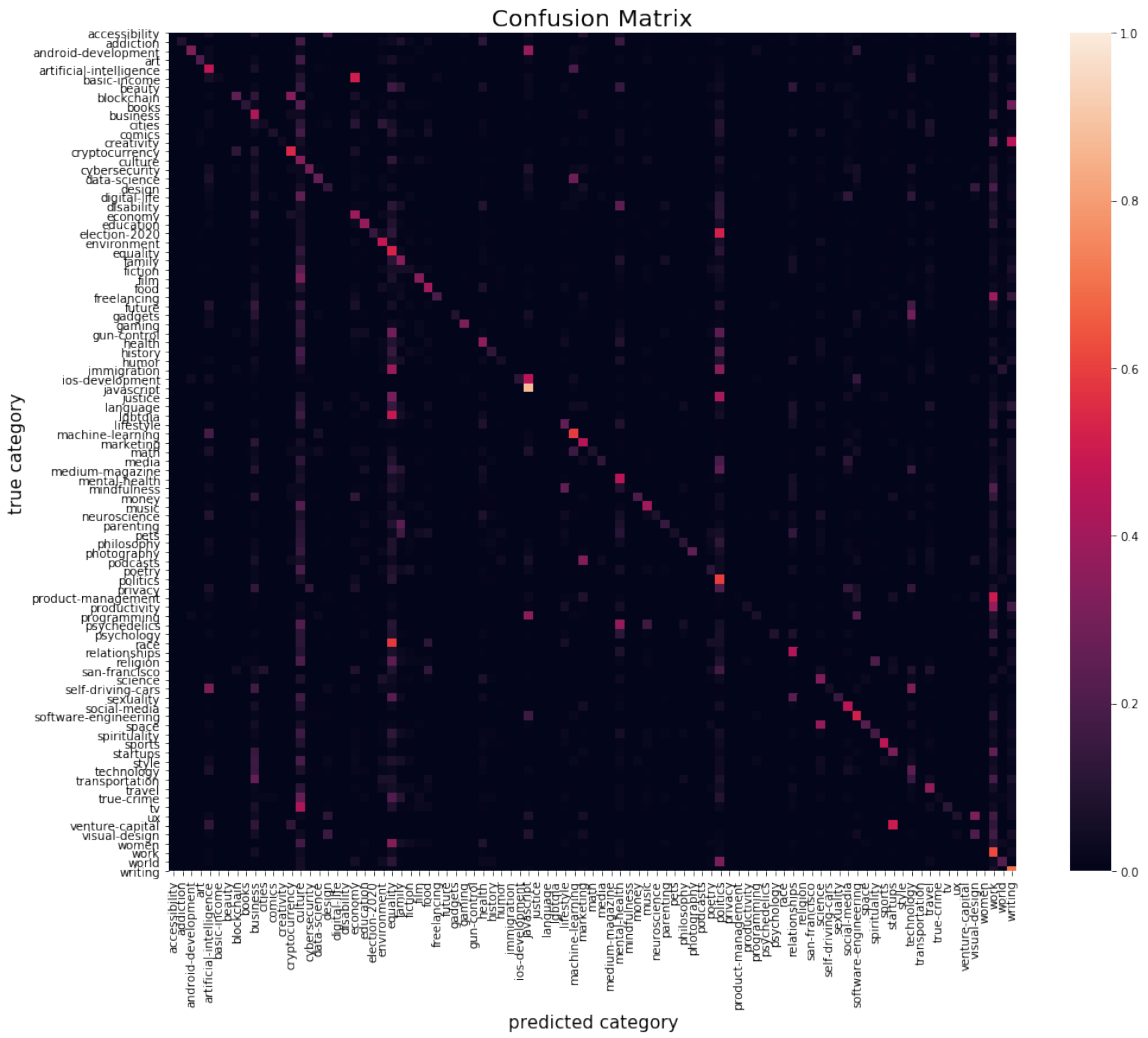

Figure 5. Confusion matrix of naïve Bayes model

Titles in many categories were misclassified into *culture*. Among them, *tv* and *film* caused major confusions (Figure 5). When I took a closer look at words with high TF-IDF scores in the real and predicted *culture* class (Figure 6), as well as the original *film* and *tv* (Figure 7), I saw that both words 'movie' and 'tv' are included in the wordcloud of the true *culture* class. A lot of other major features are also shared by either *film* or *tv* with *culture*, and made major contribution in the misclassified *culture* class, for instance, 'hollywood', 'game', 'thrones', 'black', 'netflix' and so on.

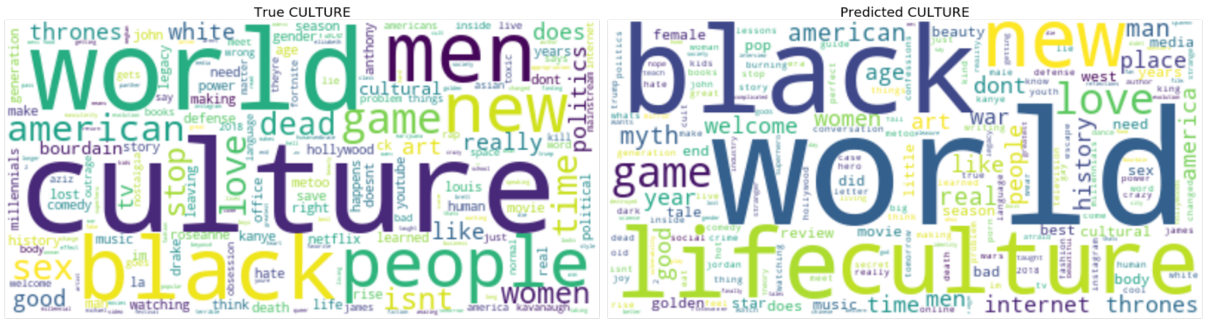

Figure 6, Wordcloud of true *culture* (left) and misclassified *culture* class by naïve Bayes classifier

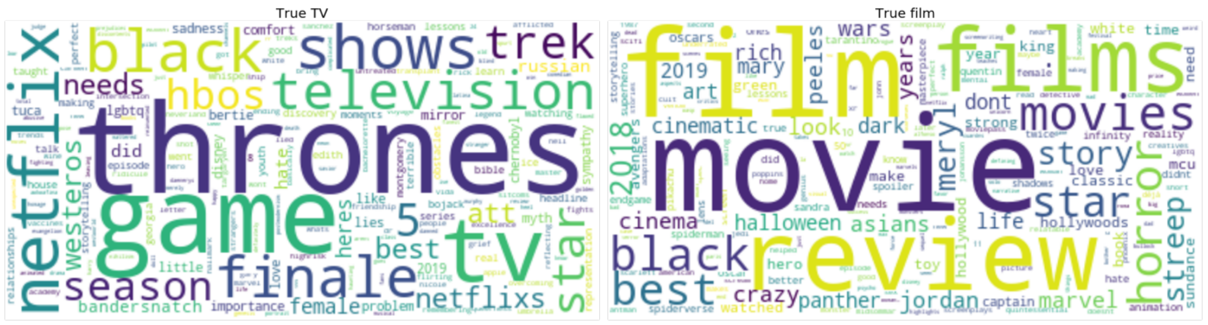

Figure 7, Wordcloud of true *tv* (left) and *film* (right) class

Multilayer Perceptron Network

I built multilayer perceptron networks for the classification task with various architectures. The networks took input of TF-IDF vector representations as well. However, due to limited computing resources, only unigrams were used because n-grams of higher orders would significantly increase the size of vocabulary, i.e. the dimension of the vectors. Words that appeared less than twice or occurred in more than 10% of the documents in the training set were rejected; this is consistent with the best parameters for the naïve Bayes classifier.

Given the available computing power, the structure of 2 hidden layers of 256 units in each layer was the most cost-effective choice without compromising the performance. Such network was able to achieve an accuracy of 41% on the test set, and a multiclass average precision of 45%. Compared to the naïve Bayes model, the multilayer perceptron network had better F1 scores (Figure 8). However, for identical precision, a higher decision threshold is required compared to the naïve Bayes classifier (Figure 9). To achieve an average precision of 90%, the decision threshold should be more than 0.991. The average recall is 0.5% at that threshold. The increase of accuracy of the model can be mainly attributed to better recall compared to the naïve Bayes classifier. However, there is a trade-off observed between recall and precision.

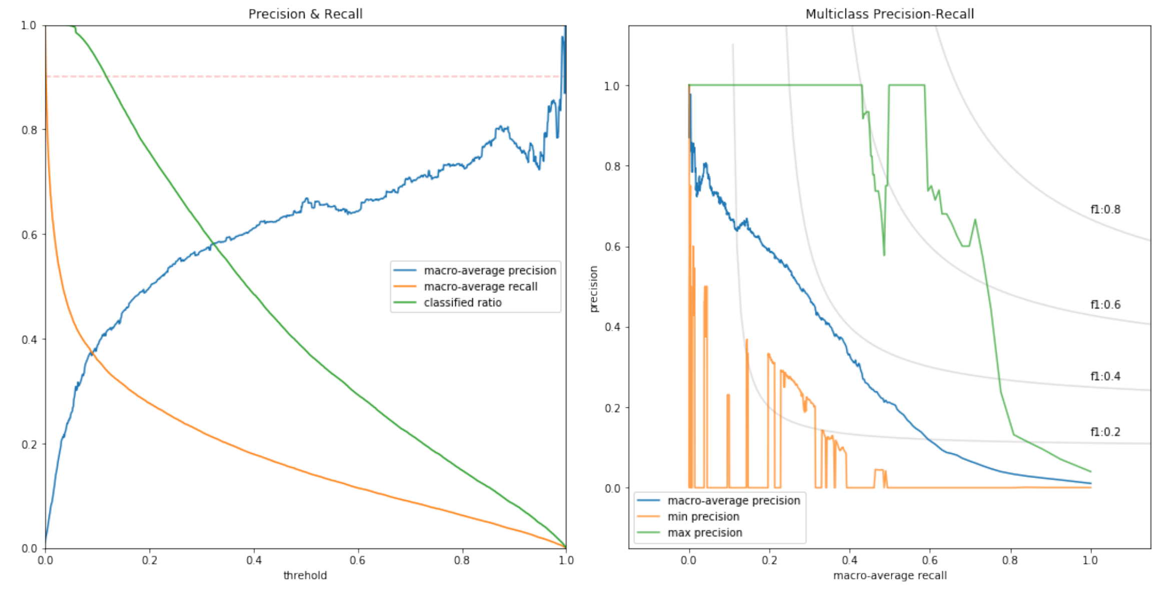

Figure 8. Precision-recall of multilayer perceptron network

The confusion matrix (Figure 9) presents some reasonable misclassification pairs, *privacy* misclassified as *technology*, *basic-income* misclassified as *economy*, *election-2020* and *immigration* misclassified as *politics*, *lgbtqia* and *race* misclassified as *equality*, *venture-capital* misclassified as *startup*, *space* misclassified as *science*...

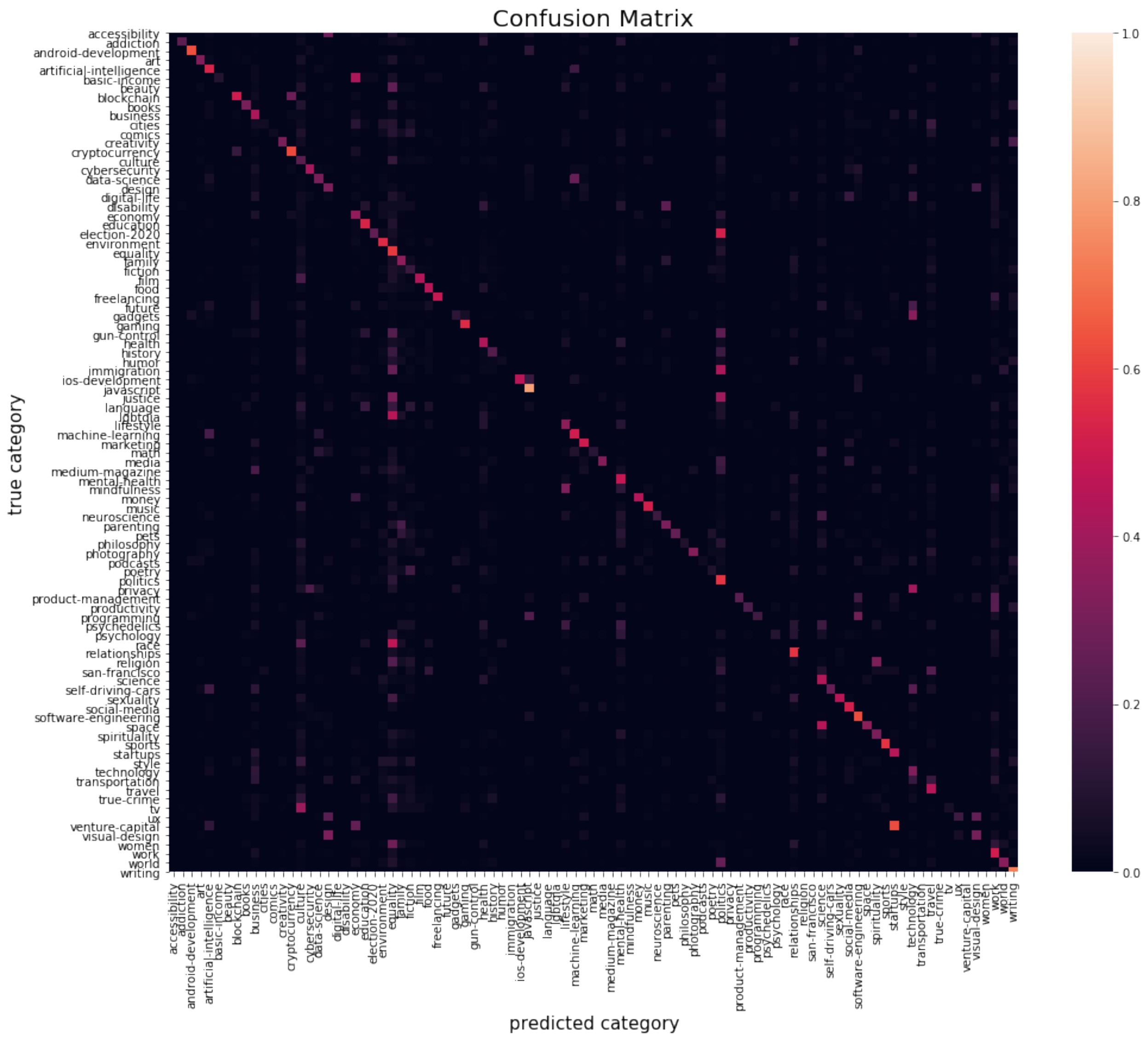

Figure 9. Confusion matrix of multilayer perc

Recurrent Network

I experimented on recurrent neural networks of different configurations. Vector representations generated by bag-of-words models are not suitable as input to a recurrent network as bag-of-words representations are not sequentially dynamic. I used the continuous bag-of-words model to generate word2vec embeddings. The continuous bag-of-words method trains a network that can predict the target word given the surrounding contextual words. Word2vec projects each word into a high-dimension vector space where contextually close words correspond to vectors located spatially close.

Individual words, which occur more than (including) twice in the training set, were mapped to 128-dimension vectors as the algorithm utilized the 6 words around it in the text as the context. 128 dimensions and 6-word-wide context were chosen after experimenting with various configurations. Higher dimensions and more lengthy context did not improve the performance of the classifiers. The word2vec model was trained on the training set solely. I then converted the titles in both the training and test sets to sequences of vectors using the word2vec model. Novel words that were not included in the vocabulary formed during training were mapped to a vector with all elements equal to 0. As optimization algorithms for the neural network require input sequences to be of uniform length, all sequences were pre-padded with 0-vectores.

After testing multiple architectural configurations, a network with two recurrent layers followed by two perceptron layers demonstrated best performance. In each layer, there were 256 units. The network achieved an accuracy of 22%, and an average multiclass precision of 22%. Consistent with its worse accuracy compared to the naïve Bayes classifier, the model presented an inferior precision-recall curve – f1 score less than 0.2 (Figure 10). To reach a 90% average multiclass precision, the minimum threshold should be 0.984; the corresponding average multiclass recall is 0.6%. The confusion matrix (Figure 11) demonstrates similar misclassification patterns to that of the naïve Bayes model and the multilayer perceptron network.

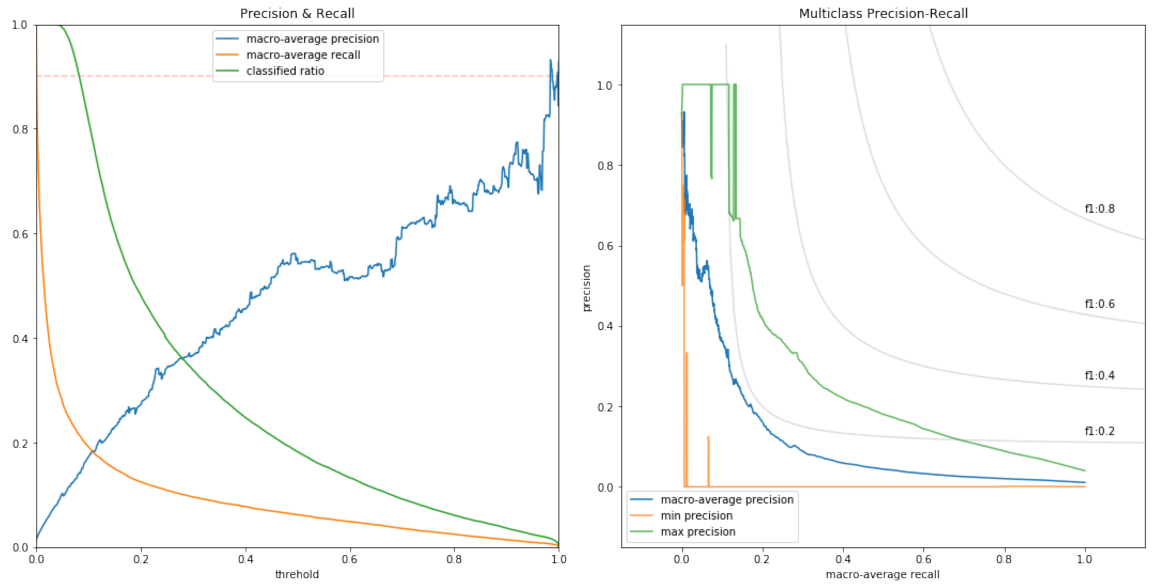

Figure 10. Precision-recall of recurrent network

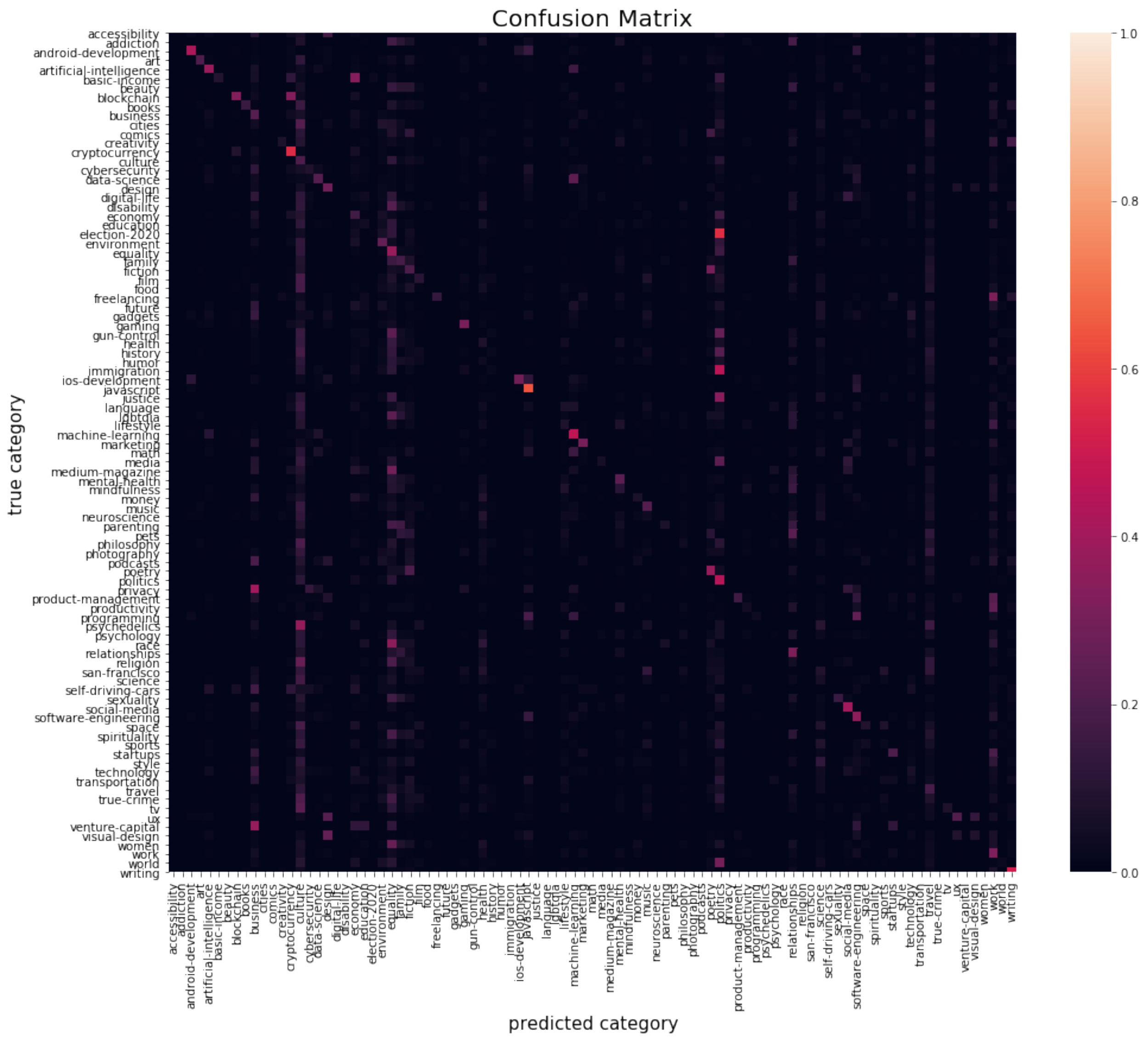
Figure 11. Precision-recall of multilayer perceptron network

Long Short-Term Memory Network

The long short-term memory network also takes input of continuous bag-of-words embeddings. For the recurrent network, the word2vec model converted each word to a 128-dimension vectors, and increasing the dimensionality of the vector representation did not improve the recurrent network's performance. However, increasing the dimensionality of the input was able to boost the accuracy of the LSTM network. Thus, for the LSTM networks, each word was mapped a 512-dimension vector.

After varying the configuration of the LSTM network, it was observed that with the limited computational resources I have access to, a network with a layer of 512 LSTM units followed by a 512-perceptron layer gave the most cost-effective results. It achieved about 29% accuracy on the test set, and 26% average multiclass precision.

Compared to the recurrent network, the LSTM classifier performed better in terms of accuracy and its precision-recall curve reached better f1-score level, although not as good as the multilayer perceptron network or the naïve Bayes classifier. However, the increased accuracy is reflected more in the recall while the precision is significantly inferior to the other three models (Figure 12). The confusion matrix (Figure 13) shows that the LSTM network suffered very similar misclassification error to that of the other three models.

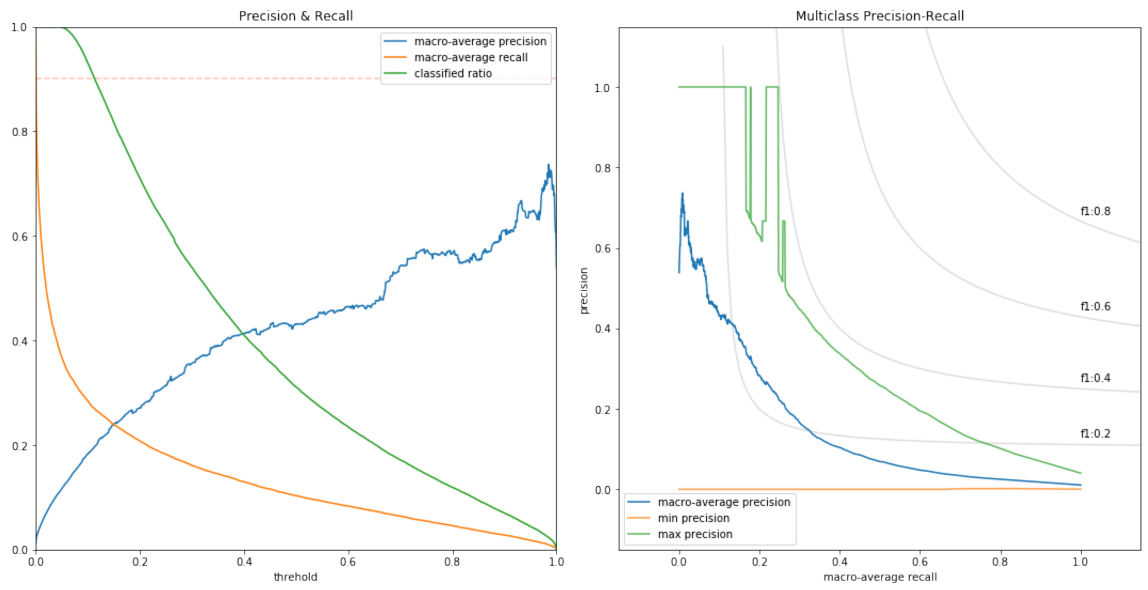

Figure 12. Precision-recall of LSTM network

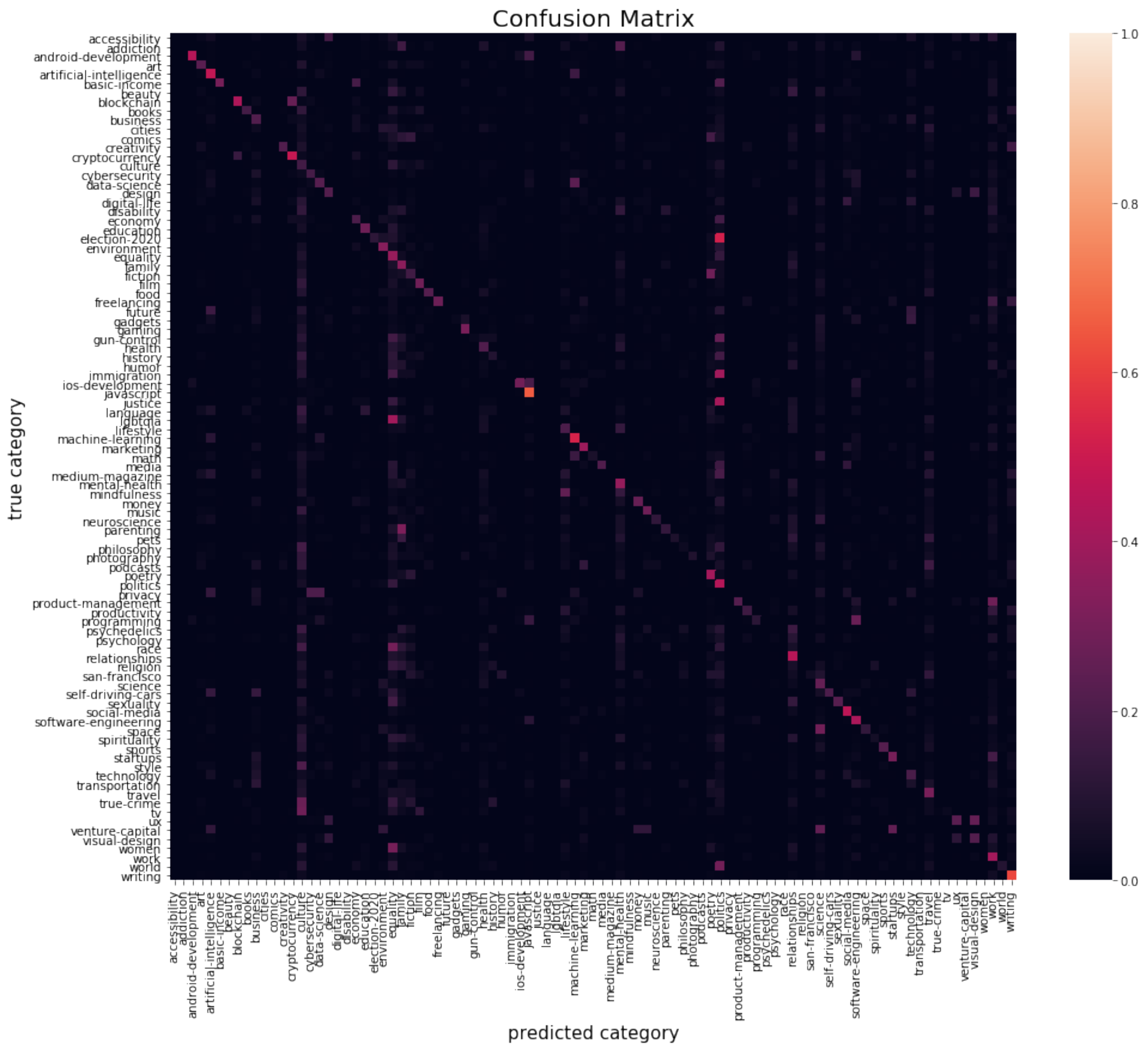

Figure 13. Confusion matrix of LSTM network

Optimal Model

Regarding the accuracy, the naïve Bayes and multilayer perceptron network models perform significantly better than the recurrent network and the LSTM network. Similarly, at the same probability threshold, the naïve Bayes and multilayer perceptron network classifiers show significantly better precision (Figure 14).

The reason behind the inferior performance of the recurrent network and the LSTM network might be two folds – the type of vectorization method they require and the mismatch between the characteristics of the data and the specialty of the networks. Both the naïve Bayes classifier and the multilayer perceptron network take input of bag-of-words vector representations. These vectors are very sparse and static, and the models are ignorant of the words' meaning, which can be ideal for short texts. On the other hand, the recurrent network and the LSTM network accept input from continuous bag-of-words embeddings. The model was trained to be aware of the words' meaning based on context, and the vector representations are dynamic and sequential. On the surface, the recurrent network and the LSTM network may be expected to take advantage of the more sophisticated input as it is more meaningful and dynamic. However, in this case, the classification is based on only titles, which tend to be not lengthy, thus maybe providing insufficient context to build a strong word2vec model.

The recurrent network and the LSTM network process sequential data, and the input sequence should be long enough, especially for the LSTM network, for the network to learn any meaningful dynamic patterns from the data. However, as titles are usually very concise, a lot of them may not be suitable for recurrent and LSTM networks.

Between the naïve Bayes model and the multilayer perceptron network, the naïve Bayes model demonstrates better precision, but the multilayer perceptron network excels in terms of recall and the number of titles that got classified (Figure 14).

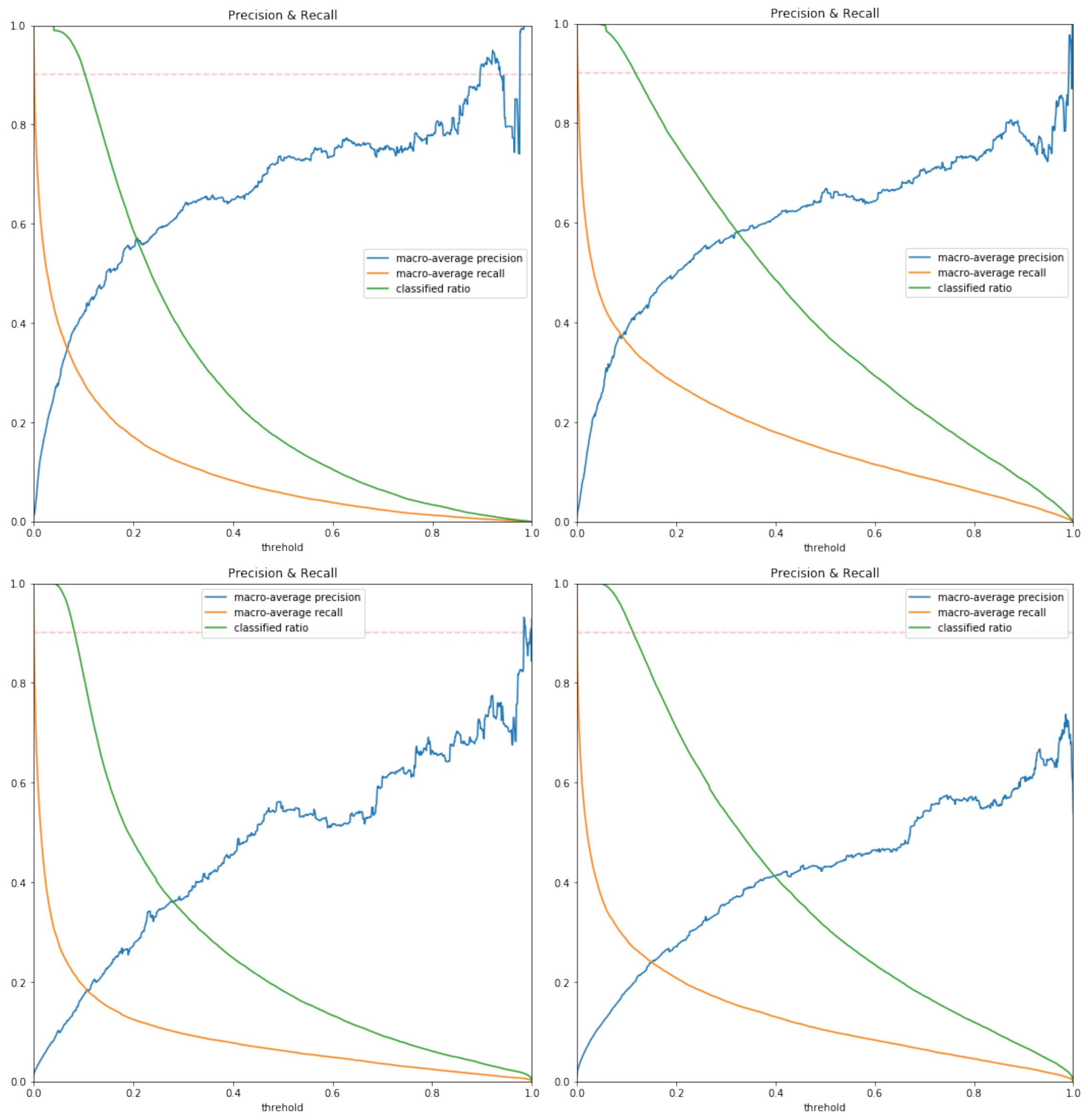

Figure 14. Comparing precision and recall of different models – naïve Bayes (upper left), multilayer perceptron network (upper right), recurrent network (lower left) and LSTM network (lower right)

If a recommendation system is to be built, where given a probability threshold for the chosen classifier, all themes with predicted probability higher than the threshold should be suggested as possible choice for the user to choose from, a correct suggestion rate can be calculated for the proportion of documents whose suggested themes include the real theme. In this sense, for a very low threshold, the correct suggestion rate would only be slightly higher for the naïve Bayes classifier (although the recommendations would also include a lot of inaccurate and redundant themes). On the other hand, and more importantly, for higher and more reasonable working thresholds, the correct suggestion rate is significantly higher for the multilayer perceptron network (Figure 15). This suggests the multilayer perceptron network may be a better model to use when building such recommendation systems.

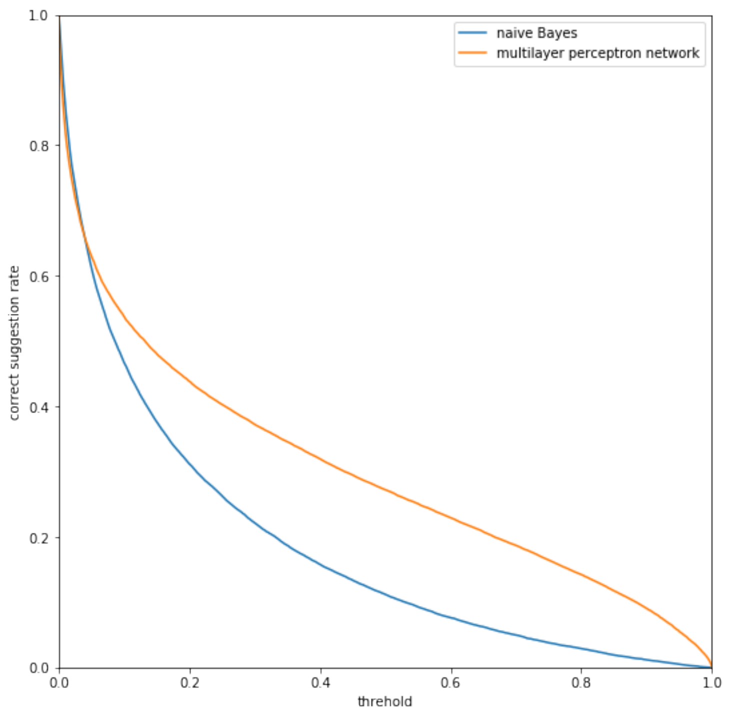

Figure 15. Correct suggestion rate of naïve Bayes model and multilayer perceptron network

Discussion

One of the major reasons why the classifiers struggled to improve performance is that the original data contain too many categories, which makes the problem extremely challenging because some of the categories share similar connotation. In order to test the full potential of the machine learning algorithms, I reduced the number of categories by aggregating similar themes. After grouping categories, there are 52 themes remaining. The naïve Bayess classifier's accuracy increases from 35% to 44%; the multilayer perceptron network's accuracy increases from 40% to 49%. Both classifiers show better performance in making better recommendations, allowing higher thresholds to be used for acceptable correct suggestion rates (Figure 16).

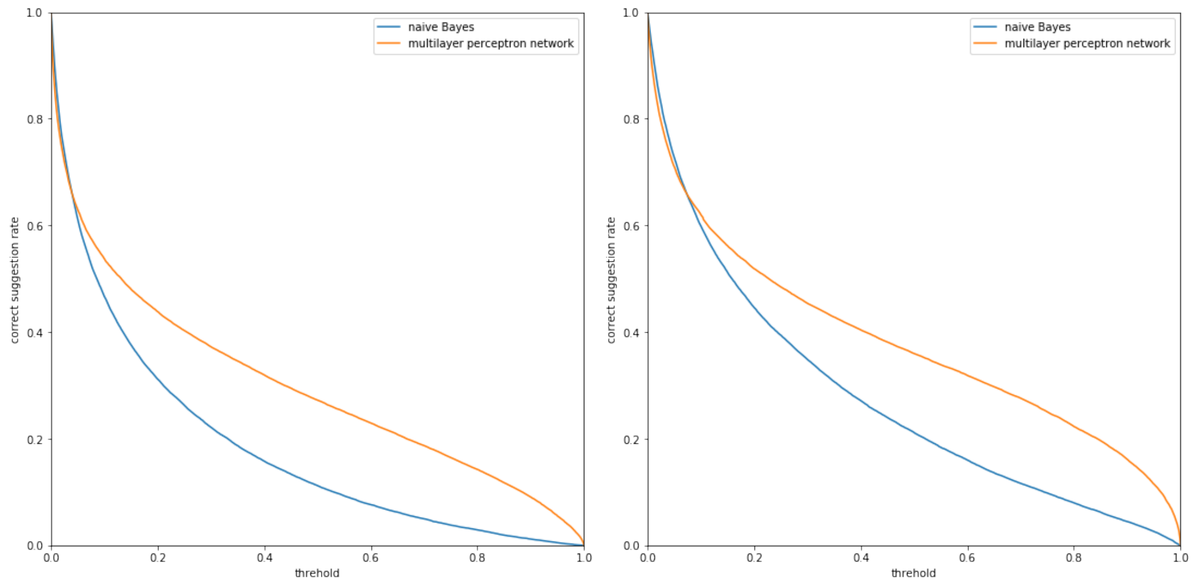

Figure 16. Correct suggestion rate of naïve Bayes model and multilayer perceptron network. Left – 93 themes, right – 52 themes.

None of the above analyses involved pre-trained models. However, it is possible that models trained on a large variety of corpus could boost the performance of the classifiers. Thus, instead of directly applying TF-IDF method on the original texts, I used WorfNet lemmatization model to tokenize the titles before TF-IDF. This process reduces words to their base forms regardless of the time, case, number and sex in the original text. This step does not further increase the performance of the multilayer perceptron network.

Conclusion

This project demonstrates multilayer perceptron network is a feasible model to build a recommendation system that suggest themes for articles on an online publishing platform. Given a finite number of all possible themes, the multilayer perceptron network can present the most likely themes for the author to choose. As this approach classifies content based only on the titles, it can also be applied to other content-sharing platforms, such as videos and images.