Binary Classification of Anime Genre from Anime Review Text

Erick Amaro Hernandez, Felix Peng

Introduction

Sometimes it can be important to identify whether a blob of text falls into a certain category. For example: Instagram warns you if your comment sounds a little mean, sites want to block NSFW text posts, and email services try to flag down spam mail. For this assignment, we tackle a more light hearted instance of this problem – Can we predict the genres of a show given a MyAnimeList review of said show?

MyAnimeList (here onwards referred to as MAL) is a comprehensive anime database website, akin to what IMDB is to movies. The site allows users to create accounts, track watched shows, and give numerical or text based ratings to different shows.

We are curious on the strategies we can take to make models that can predict whether a review is talking about an anime labeled with a certain genre, such as the selection of vocabulary, embeddings and vocabulary size.

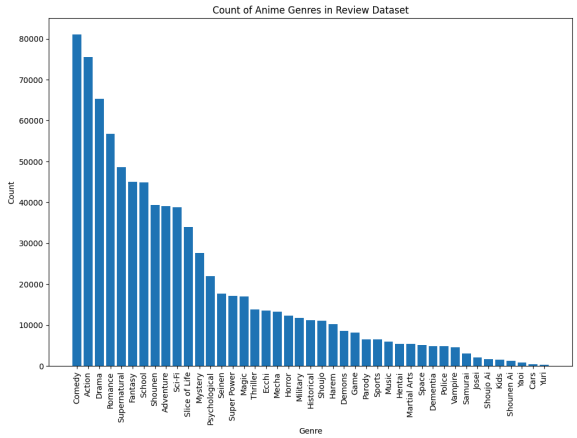
Dataset

This dataset was found on Kaggle, and was built by scraping the top anime on MyAnimeList. The dataset consists of 192,112 unique reviews, spanning 43 different genres. An anime can be composed of multiple genres.

Distribution of Reviews

We counted the number of reviews by increasing its count if a review reviewed an anime whose genre labels contained that genre.

The different reviews span 43 different genres, with the most frequent being Comedy, Action, Drama, Romance, Supernatural, and Fantasy. We'll be focusing our attention on broad categories such as "Romance" where we might be able to find common vocabulary in it, as well as some more narrow genres such as "Mecha" (anime focused on robots or mechanical suits, often piloted by humans). We use the word "narrow" to describe genres where the vocabulary used in it's reviews might be significantly different than that used in reviews for other genres (e.g. NSFW terms appearing more in reviews for NSFW anime, more robot-focused words in mecha reviews).



Cleaning Data

Before any review text data was touched, we cleaned the data. Cleaning the data involved the following steps in order: Removing punctuation, turning all words lowercase, removing any formatting that might have been accidentally scraped like newline (\n) characters, removing stopwords, and finally using NLTK's PorterStemmer to stem words.

Using PorterStemmer comes with some quirks such as “story” being stemmed to “stori” as it isn’t 100% accurate, but at least stems consistently and the original word can be easily recovered when seeing a stemmed word.

What words are used for reviews?

As discussed before, when comparing categories, we hypothesized that different categories would have different sets of words associated with them. The following visualizations of words in certain categories show that this is indeed true.

We compiled the top 500 unique words used in reviews for each genre, such as the top 500 words of reviews for “Shoujo” (genre of anime targeted primarily towards young female audiences) and the top 500 words of reviews for “Horror”, and made word clouds of those words exclusive to each genre.

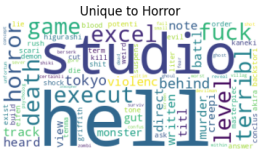
In the “Mecha vs. Sports Anime” clouds, we can see that the names of different sports and terminology can be seen exclusive to Sports’ vocabulary, while more mature words like “death”, “kill”, and words like “neon”, “genesis” which could hint at reviews often comparing a Mecha anime to one of the most famous Mecha animes “Neon Genesis Evangelion”.

In “Horror vs. Shoujo” the same could be seen with more mature themes being exclusive to Horror, and more lighthearted words being exclusive to Shoujo. The same “Evangelion” effect happens where “Sailor”, “Moon”, one of the most popular Shoujo animes, is exclusive to Shoujo’s vocabulary.

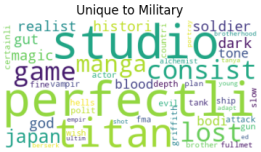
In contrast, when the vocabulary of two similar genres like in the “Military vs Mecha” word clouds are compared, the clouds are a lot less dense than the previous examples indicating they share more vocabulary. It still shows though some differences like how the military has darker topics like “blood” or other forms of weaponry like “ship” and “tank”.

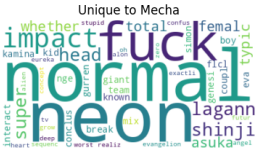
Mecha vs. Sports Anime

Horror vs. Shoujo Anime



Military vs. Mecha Anime





TF-IDF of Most Popular Genres

Another metric similar to word frequency is TF-IDF, which as discussed in lecture measures more the relevancy of a word in a document (in our case, a review). In this part of the dataset exploration, we seek to discover what are the most important words when discussing a genre.

This will be accomplished as follows, using “Comedy” as an example. We will filter the dataset to reviews of animes labeled as “Comedy”, and calculate the 500 most commonly used words in such reviews. This will

be used as our word dictionary, which we can then use to find TF-IDF scores for each word for each review, resulting in a TF-IDF embedding for each review under “Comedy”.

After the embeddings are calculated, we can average the TF-IDF score for each word in the embedding across all embeddings, leaving us with the average semantic importance of a word in a review in the set of reviews for “Comedy”.

When looking at the importance of words in the “Mecha” category, we “Gundam”, “Evangelion”, and “Mecha” were all in the top 25 words in terms of importance. For “Sport”, uncommon words like “ping”, “pong”, “box”, “max”, “skate” were in the top 10. For horror, “vampire”, “monster”, and “ghoul” were in the top 10. When ranking words like this, we see that words that may not crack the top 100 get shot all the way to the top when ranking them by their average importance in a review.

Although we will not be covering it in this paper due to the time it would take to compute, it sparked another curiosity in us on a third way of creating our dictionaries for these embeddings, and that would be by picking those with the highest average importance in terms of TF-IDF scores.

Predictive Task

Is it Genre X?

The predictive task we’ll be tackling with this dataset is predicting what genre someone is talking about when given a random review. As multi-class classification was not discussed in this class (predicting multiple categories per prediction), we will not be predicting all the possible genres in the dataset the review could be, but rather building a different Logistic Regression model for each genre that will be predicting whether or not a review is talking about the genre that model will be trained to recognize. We aim to see how we can optimize this sort of problem.

Features

Since we’re using Natural Language data for features, we will be making embeddings of each review in order to make a prediction on them, this could be through TF-IDF embeddings or Bag of Words, which we’ll be comparing performance of each strategy in the following section

Hyperparameters

Our first Hyperparameter was the embedding method, whether reviews were embedded with a TF-IDF strategy or Bag of Words strategy

The second hyperparameter was how large we wanted to build our dictionary, which simply uses the top N words with the highest frequency in the dataset

The third hyperparameter is what subset of the dataset we’ll be looking for “the top N words”. For instance, if we were training a model to predict whether a review is for an “Action” anime, our model pipeline allowed us to only use the top N most frequent words in reviews for Action anime. This idea comes from our analysis of the dataset above, where there could be better clues from the vocabulary that is exclusive to certain genres. In summary, this is a switch we can flip to build the dictionary on a specific subset of the data.

Training

The data used to train a model to predict a genre was constructed from all the reviews for a genre, as well as an equal amount of reviews that were not reviewing the genre, giving us a 50/50 split for positive and negative reviews. Since no genre made up over 50% of the

reviews, we were able to make models for each genre exactly the same as stated.

In training the model, we used a 70/30 ratio for our training and test data split, keeping a 50/50 ratio of positive and negative examples on both sides of the split. This process of building the dataset and splitting the data allowed us to have completely balanced training, prediction, and assessment on whether or not we're doing better than a crude baseline of just guessing True or False all the time.

In order to effectively compare the results of different models with different combinations of hyperparameters a pipeline function was created so that a model's training data could be collected, split, and put into a model with a single function call, with the arguments being the different hyperparameters discussed above.

Performance Assessment

To assess the performance of the different models we will be training, we will just be using the accuracy of the models. We will be building models for each of the 43 genres in the dataset, with models of each genre using dictionary sizes of 100, 200, and 500, embedding reviews with TF-IDF or Bag of Words, and tweaking the subset of words the model was trained on.

Model

In order to optimize this textual model, we made 12 models resulting from the different possible combinations of each of the hyperparameters discussed in the section above, made for each of the 43 different categories resulting in 516 different models which took about an hour to complete the pipeline for all of them . The only issue in building these models was Python peaking at 10GB of RAM usage at one point, which might have been a problem were this run on a smaller system. After searching through all these combinations of hyperparameters, we could do some analysis.

The results of all these models were put into a csv. Then we inputted them into a Pandas dataframe where we were able to see what hyperparameters tanked accuracy, what genres couldn't be improved on even with other hyperparameter tuning, and whether or not the embedding function really did anything of value.

First, for each genre, we found what model out of the 12 got the highest accuracy, then we counted how many times each combination of hyperparameters showed up in this list of highest performing genre predictors. Thus this decides the model we will rank as the most effective when trying to predict the genre with the tools we gave ourselves. It was mostly a landslide victory for two particular models.

Thirty genres were most accurately labeled by a model using a 500 word dictionary, with the words being chosen as the top 500 words used in reviews of the genre we were predicting rather the most common words in the dataset, and using Bag of Words for embedding. In second was nearly the same model which most accurately predicted ten genres; it kept the same hyperparameters as the winner, but used TF-IDF embeddings instead.

Literature

In finding Literature related to the problem of categorizing given pieces of texts, we looked for mainly literature regarding spam-email and NSFW-text classification. We ended up finding 2 on spam-filtering and one not on NSFW-classification but on guilt-classification: predicting whether or not a

user is expressing guilt given the text content of a forum post.

In the first paper by Dada et.al, they go into the state-of-the-art methods used in spam email prediction as well as previously used approaches such as previous-likeness which is described as using a training set of spam emails and training something to classify spam if it looks like the training data, like the machine learning methods we talked about in class. Its mentioned that further models make use of the user's own email set (focusing on a specific subset like we did!), as well as another layer like Google's vast knowledge of domain reputation to aid in the classification, Neural Networks were also stated to be part of GMail's spam detection.

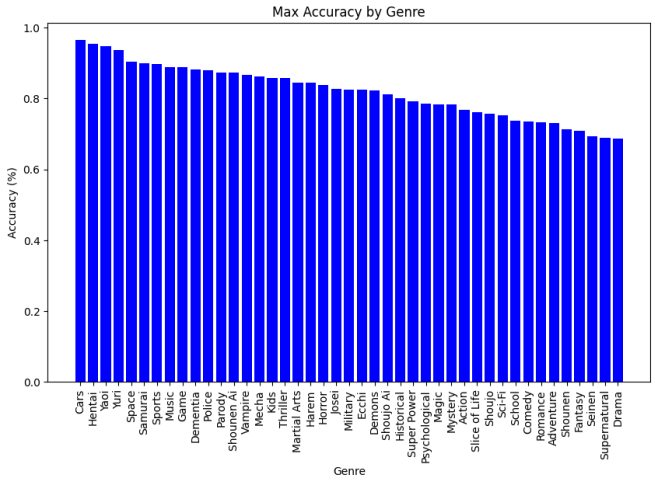
The second paper by Kontsewaya et. al also focuses on their results of spam filtering using different models, of which they use popular Machine learning approaches like Naive Bayes, K-Nearest Neighbors, SVM, Logistic regression, Decision tree, and Random forest using a Bag of Words approach on a dataset of spam-not spam emails. We were surprised to find out that Logistic Regression came out on top over all the other methods, followed Extremely closely by a Naive Bayes model. Given that spam emails we would consider a "Narrow" genre of text, we're not too surprised that Logistic Regression with Bag of Words did this well similar to our study

The third paper on guilt-classification by Meque et. al is extremely similar to our research, uses a depressing dataset which which combines datasets of a venting forum where users can categorize what they're venting about, and a topic-labeled corpus of suicide notes. All the dataset had multiple emotion labels like our many genre labels, but they focused on only classifying as contains/does not contain guilt. Their results were also similar to ours where they compared TF-IDF features vs. Bag of Words features, where TF-IDF won but by very little. They also used many machine learning approaches for their models, and the model that did the best was the Multinomial Bayes. Since the emotion of "guilt" is extremely broad, literally a complex human emotion, their Logistic Regression models came in second with a similar score to what our "broad" genres performed.

Results

Max Accuracy By Genre

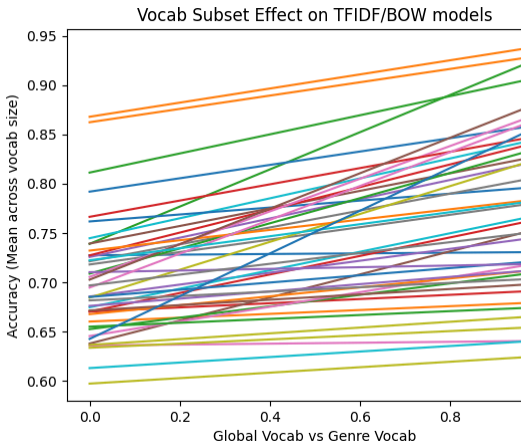
The first thing we checked when our 500-model training session was over was what was the max accuracy we got for genres across the 12 models we made for each genre, which is visualized in the bar graph below.



Our hypothesis of "narrow" genres being easy to classify seems to be correct. If you refer to the distribution of reviews we have for each genre, you can see that although "Drama", "Action", "Comedy", and "Romance" are the top 4 most popular and "broad" genres, they were the hardest to classify despite their abundance of data available compared to other genres, scoring about 73% accuracy on average for these popular genres.

For the top performing genres, we have an outstanding 96.5% accuracy rate for the “Cars" genre, then the second highest which we’ll refer to as the NSFW genre, third and fourth highest being anime genres that focus on LGBT themes; all of which scored more than 93% accuracy. The rest of the top half of performers were more “Narrow” genres like “Samurai”, “Police”, and “Mecha”, which scored greater than 80%

Dictionary-building Methods



In order to measure the effectiveness of using the top 500 words in the entire dataset vs using only the top 500 words in the subset of reviews for the genre, we grouped a genre’s models by whether or not they were using global or subset training data, and averaged across the rest of the hyperparameters (the data was nearly identical if we focused on only one embedding method). The first thing that immediately stood out was that every model benefitted from this change in dictionary vocabulary.

Since every genre benefitted, we checked out which models benefitted the most and on average how much did a model benefit from using these subsets of vocabulary. The category that benefitted the most from this switch to genre-subset vocabulary was actually the “Vampire” category, gaining 21% in comparison to it’s global vocabulary counterpart, while other more “narrow” categories like those discussed in the results before this section gained about +18% accuracy.

The models that gained the least amount of accuracy were broad, and actually had the most reviews out of any others in the dataset such as Drama and Comedy, only gaining about 1% accuracy.

On average, a model gained about 7.5% accuracy when using genre-specific vocabulary.

Embedding Methods and Dictionary Size

Earlier we stated that almost every Bag of Words model outperformed their TF-IDF model. The effect was extremely miniscule, with only 8 genres getting an accuracy boost of over 1%. This is surprising because TF-IDF is supposed to get deeper in meaning when it comes to embedding a piece of text, but was outperformed by a model that just took into account term frequency. In fact, some categories like “Slice of Life” and “Adventure” lost accuracy, but not a significant amount at only about -0.3%.

Also mentioned earlier was how a dictionary size of 500 beat out all the lower sizes in almost every model. Surprisingly, the only model whose accuracy maximized at 200 was the “Cars” genre model which beat out every other model in our suite of models, which we can’t find a good reason as to why.

Works Cited

Meque, A.G.M., Hussain, N., Sidorov, G. *et al.* Machine learning-based guilt detection in text. *Sci Rep* 13, 11441 (2023). https://doi.org/10.1038/s41598-023-38171-0

Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems", Heliyon, Volume 5, Issue 6, 2019, E01802, ISSN 2405-8440, https://doi.org/10.1016/j.heliyon.2019.e01802.

Yuliya Kontsewaya, Evgeniy Antonov, Alexey Artamonov, "Evaluating the Effectiveness of Machine Learning Methods for Spam Detection", Procedia Computer Science, Volume 190, 2021, Pages 479-486, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.06.056.