Predicting Content Category

October 2016

DRAFT

What is Twitch?

Twitch is the world's leading social video platform and community for gamers, video game culture, and the creative arts. Each day, close to 10 million visitors gather to watch and talk about video games and more than 2 million streamers.

What Is The Problem?

There are hundreds of thousands of streams daily on Twitch. A large portion of these streams are community streamers, but there is also an increasingly large amount of non-community content on the platform.

Currently there is no method of detecting whether a live broadcast contains community content or non-community content. Non-community content is defined by any esports or major event which is not broadcasted by an individual streamer.

This will be a classification problem.

Potential Solution?

By training a predictive model to test on the following criterion, we can attempt to tag future broadcasts as community or non-community content to a certain degree of confidence:

- ➔ Day of the Week
- ➔ Channel
- ➔ Game
- ➔ Broadcast Title
- ➔ Average CCUs (Concurrent Users / Viewers)
- ➔ Chat Activity Velocity

I hypothesize that of the features listed above, the words in a broadcast title and average CCUs should be the most effective in predicting whether a broadcast is community content or not.

Data

- Dataset was achieved by using a SQL query of our internal data
- The dataset has 5,000 datapoints
    - Limitation here is due to the fact that we have to go through the dataset manually to tag historical broadcasts as community or non-community

Cleaning the Data

The first step was to analyze and determine how many rows / data points have nulls in them. It was determined that 63 data points have nulls in them.

Since 63 represents <2% of the total dataset, I decided to remove all those data points from the analysis.

Key Features

Recall the features in the dataset were:

- ➔ Day of the Week
- ➔ Channel
- ➔ Game
- ➔ Words in Broadcast Title
- ➔ Average CCUs (Concurrent Users / Viewers)
- ➔ Chat Activity Velocity

Of these, channel and game had to be turned into dummy variables before analysis could be conducted. A CountVectorizer would have to be applied to words in broadcast title in order to use that column as a feature.

Data Dictionary

Data	Description
time	Timestamp of the a broadcast update
day_of_week	Day of the week (0 being Sunday)
city	City of channel broadcasting
country	Country of channel broadcasting
status	Broadcast Title
game	Game being broadcasted
avg_ccu	Average concurrence of the channel broadcasting
avg_chat_activity	Average chat activity of the channel broadcasting
community	Response (1 beingc community, 0 being non-community)

The Three Approaches to the Features

Of the features available, I decided to approach them using three different ways:

- ➔ Using all features except for the words in broadcast title
- ➔ Using only words in broadcast title
- ➔ Using all features available

This is to ensure that we are not overfitting the model with features that are not necessary.

Prediction Models

The models used to evaluate the data are:

- ➔ KNN
- ➔ Logistic Regression
- ➔ Naive-Bayes
- ➔ Random Forest

These models were used on all three approaches.

Approach 1: Features Excluding Broadcast Title

By using the features from approach 1 and feeding it into the three predictive models, it was apparent that the highest achieved score was 0.943488552.

The surprising thing was that this result occurred for three different models: Logistic Regression, Naive-Bayes, and Random Forest.

Why?

Null Accuracy

To test the power of the model, I had previously calculated the null accuracy score.

Of the 4,937 data points, there were 4,658 community broadcasts, yielding a **null accuracy of 0.943488552**.

This leads me to conclude that none of the models using the features extracted from approach 1 could give me a result better than the null accuracy.

Approach 2: Broadcast Title Only

By just extracting the words from the broadcast title using a CountVectorizer() as features, I was able to achieve scores between 97 - 98%.

The highest score was using a Random Forest Classifier, which yielded a **mean score of 0.980153311.**

Approach 3: Use All Features

Lastly, I ran all the features through the four models.

Unfortunately, 0.97247660 was the highest achievable score using this method.

Summary of Results

	Approach 1	Approach 2	Approach 3
KNN	0.717237189	0.974275876	0.943488552
Logistic Regression	0.943488552	0.979748040	0.935815156
Naïve-Bayes	0.943488552	0.970617757	0.942489975
Random Forest	0.943488552	0.980153311	0.971247660

The best result was yielded from using only the CountVectorizer of the broadcast title, indicating that the words in the title is the most predictive of content type and other features seem to only be noise.

What’s Next?

- ➔ Find a better way of finding KBest: I had ran 1-10 features through the various models in approach 1 but not for approach 3 (the one that included all features) as I did not think of a processing power efficient way of doing so
- ➔ Think of more features that may be indicative of content type
- ➔ Train the model on more datapoints