Jordan Youner
Professor Lignos

Futurama: Who Said It? A Multi-label Classification Project:

The classification of text is a prominent task in NLP. It is the process of assigning class labels to a document using extracted features like word or character ngrams. Often it is applied to sentiment analysis (e.g classifying reviews as positive or negative), or to email spam detection. In this case, I have chosen a dataset of lines from the TV show Futurama and am classifying each with the character who said it. I chose this dataset because Futurama is one of my favorite TV shows of all time. Furthermore, I am interested to see if there are any obvious patterns in a character's dialogue and humor that a classifier could pick up on, since I like comedy and have dabbled in writing it before. I want to be clear, however, that even though I am performing a classifier task on the written dialogue of a TV show for an experiment, I explicitly **discourage** the use of generative AI or machine learning in TV or any similar writing-related industry. That all being said, I will now discuss the dataset in specific terms.

The dataset is sourced from Kaggle, and the transcripts themselves from a community wiki called The Infosphere. This specific dataset contains lines from the first 114 episodes (noted as 6 seasons but it depends on how you count things) in a .csv format where each row contains the season number, episode title, character who speaks the line, and the line itself. I ignored the first two columns of the .csv and focused on the character (label) and line (document).

Season	Episode	Character	Line
1	Space Pilot 3000	Bender	Bite my shiny metal ass!
3	The Cyber House Rules	Leela	"Well, I wouldn't mind rubbing my success in a few choice faces."
4	The Sting	Fry	"Leela, let's turn back. There's absolutely no shame in wussing out."

The size of the dataset in total is 23,813 documents. However, there are many characters who have very few or even one line, so I removed all lines not spoken by the main 7 characters: Fry, Bender, Leela, Farnsworth, Zoidberg, Amy, and Hermes. This left me with 14,439 lines, of which 3,791 were said by Fry, 3,395 by Bender, 2,998 by Leela, 1,730 by Farnsworth, 888 by Zoidberg, 863 by Amy, and 771 by Hermes. I then split the data into training, development, and test sets by using sklearn.model_selection.train_test_split() twice. The first time I split the entire dataset into train and dev/test where train was 80% and dev/test was 20%. Then I took the dev/test set and applied the split method again with 50% to data and 50% to test, ending up with a final split of 80% train, 10% dev, and 10% test. From here I began running configurations on the development set.

I ran several models on the development set. These models consisted of either multinomial naive bayes or logistic regression with different feature sets and hyperparameter values. The possible features included unigrams, bigrams, trigrams, a toggle for whether only the top 100 most frequent bigrams and trigrams were included, and a toggle for binary features or counts. For the baseline, I selected the configuration of unigrams only with binary features and no smoothing. I ran a few more experiments with 0 smoothing and then did grid searches for various hyperparameter ranges on both naive bayes and logistic regression. The scores I reported for each configuration are based on the best hyperparameter found in each range of grid searching, and they are reported in the order that I tested them. The following table has my development set results, with the baseline at the top:

Jordan Youner
Professor Lignos

Fundamentals of NLP 1
12/13/2023

Model type	Unigrams	Bigrams	Trigrams	Counts	Top 100	α / C	Accuracy (%)
Naive Bayes	True	False	False	False	False	0.0	31.86
Naive Bayes	True	True	False	False	False	0.0	20.98
Naive Bayes	True	True	False	False	True	0.0	31.93
Naive Bayes	True	True	True	False	True	0.0	31.44
Naive Bayes	True	False	False	True	False	0.4	39.68
Naive Bayes	True	False	False	False	False	0.5	39.40
Naive Bayes	False	True	False	True	False	0.8	34.00
Naive Bayes	False	True	False	False	False	0.9	34.07
Naive Bayes	False	False	True	True	False	0.8	31.30
Naive Bayes	False	False	True	False	False	0.5	31.16
Naive Bayes	True	True	True	True	True	0.9	40.02
Naive Bayes	True	True	True	True	False	0.5	36.22
Naive Bayes	True	True	False	True	False	0.7	40.10
Naive Bayes	True	True	False	True	False	0.4	37.05
Naive Bayes	True	True	True	False	True	0.6	39.96
Naive Bayes	True	True	True	False	False	0.8	36.57
Naive Bayes	True	True	False	False	True	0.7	40.17
Naive Bayes	True	True	False	False	False	0.5	37.74
Naive Bayes	True	False	False	True	False	0.4	39.68
Naive Bayes	True	False	False	False	False	0.55	39.54
Naive Bayes	False	True	False	True	False	0.55	34.00
Naive Bayes	False	True	False	False	False	0.55	34.21
Naive Bayes	False	False	True	True	False	0.8	31.30
Naive Bayes	False	False	True	False	False	0.5	31.16
Naive Bayes	True	True	True	True	True	0.85	40.10
Naive Bayes	True	True	True	True	False	0.45	36.29
Naive Bayes	True	True	False	True	True	0.75	40.24
Naive Bayes	True	True	False	True	False	0.45	37.12
Naive Bayes	True	True	True	False	True	0.75	40.03
Naive Bayes	True	True	True	False	False	0.8	36.57
Naive Bayes	True	True	False	False	True	0.55	40.17

Naive Bayes	True	True	False	False	False	0.5	37.74
Logistic Reg	True	False	False	True	False	0.5	39.75
Logistic Reg	True	False	False	True	False	0.5	40.24
Logistic Reg	False	True	False	True	False	0.5	33.93
Logistic Reg	False	True	False	False	False	0.5	34.00
Logistic Reg	False	False	True	True	False	0.5	30.75
Logistic Reg	False	False	True	False	False	0.5	30.82
Logistic Reg	True	True	True	True	True	1.5	38.92
Logistic Reg	True	True	True	True	False	0.5	38.92
Logistic Reg	True	True	False	True	True	0.5	38.85
Logistic Reg	True	True	False	True	False	0.5	38.99
Logistic Reg	True	True	True	False	True	0.5	39.40
Logistic Reg	True	True	True	False	False	0.5	39.34
Logistic Reg	True	True	False	False	True	0.5	39.68
Logistic Reg	True	True	False	False	False	0.5	40.03
Logistic Reg	False	True	False	True	False	0.2	34.63
Logistic Reg	False	True	False	False	False	0.2	34.35
Logistic Reg	False	False	True	False	False	0.3	30.81
Logistic Reg	True	True	True	False	True	0.5	39.40
Logistic Reg	True	False	False	True	False	0.55	39.82
Logistic Reg	True	False	False	False	False	0.55	40.30
Logistic Reg	False	True	False	True	False	0.15	34.35
Logistic Reg	False	True	False	False	False	0.25	34.49
Logistic Reg	False	False	True	True	False	0.25	30.68
Logistic Reg	False	False	True	False	True	0.25	30.89
Logistic Reg	True	True	True	False	True	0.45	39.34
Logistic Reg	True	True	False	False	False	0.45	40.37
Logistic Reg	True	True	False	False	True	0.35	39.61

After examining the results of the grid searches, we pick out the best 4 model configurations. They are as follows, grouped by model type:

Naive Bayes	True	True	False	False	True	0.7	40.17
Naive Bayes	True	True	False	True	True	0.75	40.24
Logistic Reg	True	False	False	False	False	0.55	40.30
Logistic Reg	True	True	False	False	False	0.45	40.37

From here I moved to the test set.

At this point I examined how each of my best 4 configurations performed on the test set. The results are as follows:

Model Type	Unigrams	Bigrams	Trigrams	Counts	Top 100	α / C	Accuracy (%)
Naive Bayes	True	True	False	False	True	0.7	40.17
Naive Bayes	True	True	False	True	True	0.75	40.24
Logistic Reg	True	False	False	False	False	0.55	40.30
Logistic Reg	True	True	False	False	False	0.45	40.37

From these results, I obtain my best model configuration, which is a logistic regression model with binary unigram and bigram features without only including the top 100 and with a C of 0.45. The following table represent a classification report from scikit-learn on the model's test set performance:

Label	Precision	Recall	F_1-Score	Support
Amy	0.47	0.07	0.12	97
Bender	0.44	0.47	0.46	339
Farnsworth	0.45	0.34	0.39	145
Fry	0.42	0.59	0.49	409
Hermes	0.50	0.12	0.19	91
Leela	0.32	0.39	0.35	277
Zoidberg	0.38	0.07	0.12	86

Accuracy			0.40	1444
Macro-average F_1	0.42	0.29	0.30	1444
Micro-average F_1	0.41	0.40	0.38	1444

The results of this experiment are notable. While the best model only outperformed the baseline by 8.51%, there is worthwhile discussion to be had. Looking at the micro and macro averages, we see a general trend where the micro-average (i.e weighted by support) is higher, indicating generally better performance on classes with higher frequency in the set. However, the reverse is true with precision. The four less frequent classes (Amy, Farnsworth, Hermes, Zoidberg) all have higher precision than recall, and other than Zoidberg, higher than the top three classes (Fry, Bender, Leela). However, since these classes have significantly lower recall, they may be a result of the model overgeneralizing on a sparse data set. This is not too surprising, especially given the results with features. Trigrams were found to typically be unhelpful, even with the top 100 feature enabled. This lines up with the idea of data sparsity, as trigrams by nature are more sparse than bigrams or unigrams, and thus benefit more from larger amounts of data. Overall, while these results were lower than expected, they do contain important insights about working with multiclass datasets specifically, and NLP more broadly.

Throughout this discussion, I have demonstrated my process of examining the performance of various model configurations for the multiclass labeling of dialogue from the TV show Futurama. The results show that logistic regression with binary unigrams and bigrams are the best configuration explored, with an 8.51% increase in accuracy over baseline. They also demonstrate things about data sparsity, like with the utility of trigrams. I found writing grid searches in an efficient manner were more challenging than expected, along with managing dataframes from the pandas package for the first time.

I have certainly gained some facility and knowledge about the aforementioned processes in this experience. Looking forward, there are a few things I would change and/or do differently with more time. I would definitely do more feature engineering, and specifically I would like to add skip-grams of at least size 2 because I think this would capture the more catchphrase-y lines on the show (e.g Hermes saying things like “Sweet/Great x of y!” where x and y rhyme). I also would definitely refactor my code to have a much larger object structure for greater ease of storing things like labels and predictions and for more efficient code reuse. Both of these are considerations and motivations for future experimentation.