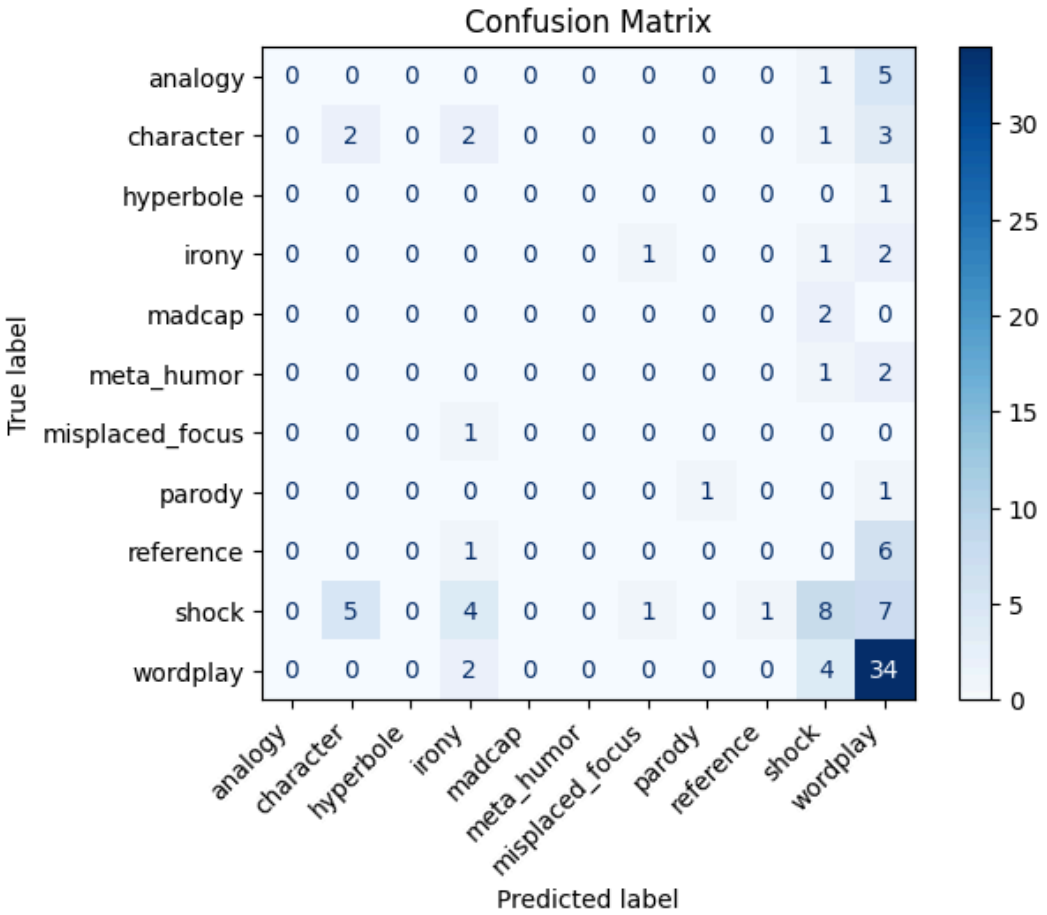
AP3 Analysis

Our model indirectly learned features that we didn't consider in our guideline, like how the frequency of a specific word can easily identify a category when we didn't take specific words into account. Though it isn't consistent, there are words that appear more frequently in more categories than others. We would reconsider how to determine a certain category by the use of a specific type of word that would clearly identify it in a category. Not only that, we found that some jokes could have been associated with multiple categories (e.g., wordplay and irony) which further complicated the guidelines on categorizing jokes.

We created a confusion matrix as shown below to see what labels our model was mislabeling and labeling properly.



Looking at the confusion matrix, the model seemed to be most confident in predicting the 'wordplay' class, with 34 instances correctly classified. This suggests that 'wordplay' jokes have distinctive features that the model can recognize. The 'analogy' class has all instances misclassified. Notably, 5 instances are misclassified as 'wordplay'. This suggests that the model lacks sufficient examples to accurately learn the 'analogy' pattern. The high number of correct predictions for 'wordplay' compared to other categories might also reflect a class imbalance in the dataset where 'wordplay' jokes are overrepresented. Classes such as 'analogy', 'hyperbole', 'madcap', 'misplaced_focus', and 'parody' do not have any true positives. There probably was not enough data for the model to learn these categories effectively. The classes with zero correct predictions are suffering from overfitting to the training data, especially given the near-perfect training accuracy and the drop in development accuracy.

Based on the weights of each class, we printed out the highest 10 features with weights.

- Top 10 Features for Class “Analogy”
    - both: 2.035
    - re: 2.227
    - common: 2.398
    - we: 2.443
    - in common: 2.488
    - in common body: 2.586
    - common body: 2.586
    - common body they: 2.586
    - snowballs: 2.798
    - babies: 3.476

- Top 10 Features for Class “Character”
    - the: 2.358
    - trump: 2.362
    - german: 2.380
    - in: 2.382
    - rabbi: 2.674
    - clip clop: 2.781
    - clip: 2.781
    - clop: 2.781
    - funny: 3.050
    - blonde: 7.080
- Top 10 Features for Class “Hyperbole”
    - stupid body she: 1.950
    - so stupid: 1.950
    - so stupid body: 1.950
    - great: 1.982
    - sorry: 2.295
    - ugly: 2.357
    - too: 2.454
    - hard: 2.502
    - started after dropping: 2.546
    - started after: 2.546
- Top 10 Features for Class “irony”

        - now body: 2.051
        - must: 2.085
        - kids: 2.165
        - deaf: 2.273
        - parrots: 2.307
        - elephant: 2.394
        - well: 2.559
        - well body: 2.580
        - wife: 2.604
        - guy: 2.766
- Top 10 Features for Class “madcap”
    - throw: 1.813
    - how hard: 1.901
    - paint: 1.905
    - cow: 2.074
    - dropped: 2.123
    - animal: 2.228
    - animal is: 2.228
    - faster: 2.260
    - than cheetah body: 2.451
    - than cheetah: 2.451
- Top 10 Features for Class “meta_humor”
    - steak and: 2.174

        - steak and cheese: 2.174
        - steak: 2.174
        - frenchman: 2.275
        - the englishman: 2.275
        - original: 2.370
        - christmas: 2.547
        - englishman: 2.729
        - jokes: 3.013
        - always: 4.239
- Top 10 Features for Class “misplaced_focus”
    - man: 1.793
    - squirrel: 1.864
    - the owl: 1.864
    - then: 1.967
    - that: 2.232
    - sent: 2.470
    - lonely: 2.637
    - woman: 2.645
    - owl: 2.796
    - my: 2.941
- Top 10 Features for Class “parody”
    - the frog: 1.815
    - kermit the frog: 1.815

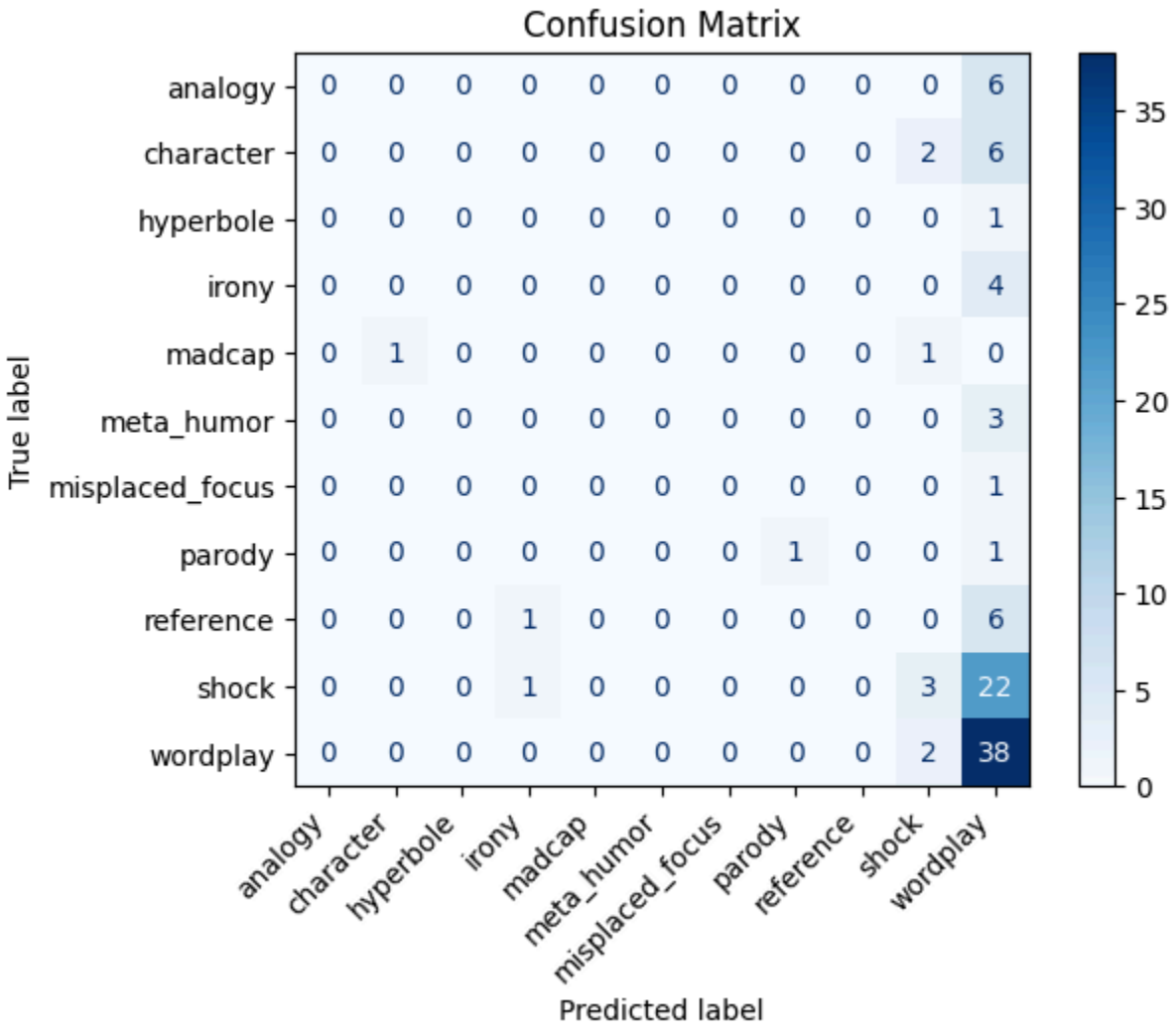
        - frog: 1.815
        - kermit: 1.815
        - kermit the: 1.815
        - smells: 1.906
        - smells like bacon: 2.061
        - smells like: 2.061
        - and smells like: 2.061
        - and smells: 2.061
- Top 10 Features for Class “reference”
        - strike: 2.442
        - audible: 2.480
        - person: 2.579
        - you: 2.593
        - speed: 2.625
        - whats: 2.654
        - just: 2.716
        - jared: 2.950
        - fresh: 3.307
        - title just: 3.812
- Top 10 Features for Class “shock”
        - meeting: 2.418
        - sex: 2.455
        - mother: 2.466

        - 11: 2.566
        - girls: 2.611
        - mouth: 2.629
        - body your: 2.712
        - on the: 3.458
        - leaves: 4.023
        - between: 6.244
- Top 10 Features for Class “wordplay”
    - because he: 1.990
    - title guess: 2.002
    - body get: 2.070
    - has: 2.096
    - they: 2.102
    - in your: 2.288
    - had: 2.436
    - dog: 2.549
    - he: 2.816
    - doctor: 3.324

As shown above, we found the top features to be sometimes completely irrelevant to the joke categories at hand, making us realize how much more complex this phenomenon we were trying to predict was than we expected. A lot of the features included common words used in English phrases such as “has”, “they”, and “he”. We figured that if we were to improve the model in the

feature, we would possibly have to get a better look at and use a database with joke related words in order to give weighted values for such words.

Further analysis we did is to use random forests and also create a confusion matrix for that model as well to see how well it fits our dataset classification task. We found that the test accuracy for best dev model was 0.420 with a 95% CI of [0.417 0.423]. The confusion matrix for the random forest model is as follows.




As you can see, it seemed to perform worse off than our logistic regression model using TF-IDF, seeming to lean even more towards classifying most of the jokes as ‘wordplay’. We figured that

the incorporation of the TF-IDF feature helped our original model in better categorizing the jokes than this random forest model.

Our model made a crucial systematic mistake when trying to categorize the jokes. We found that it relies too heavily on the training dataset as it has a very high training accuracy compared to the development accuracy. Also, the model barely has any correct predictions for labels other than wordplay. This is probably due to the fact that there were not very many examples of other labels in the 500 jokes of the adjudicated set where we got the test, dev, and training sets from.

We found bias in our model where it was unable to specifically identify a category when there is more than one category present in a joke. This makes it difficult to select a single category when there are more than one category present in a joke. We selected the most prevalent category for a joke based on the word frequency for a category. Though it wasn't bias coming from us when programming the regression model, the bias is selecting the first category that comes up without comparing the TF-IDF vectorized values with each category and which one appears more relevant rather than just selecting the highest TF-IDF values.

Our level of balance in our dataset was very one-sided. The 'wordplay' label was extremely prevalent while there were very little datapoints with other labels in our dataset. This impacts the model that we developed because it did not give enough information for our model to work off of in predicting labels other than wordplay, and therefore adhered very closely to the training set showing a very high training accuracy compared to the low development accuracy. Our dataset doesn't seem to be a very good candidate for training our model since the majority of the 500 random jokes pulled from the dataset online were wordplay jokes, which did not help our model in correctly predicting other joke labels.

Some future thoughts and recommendations that we had for improving and using this model was to consider gathering more data or using data augmentation techniques for underrepresented labels to provide more examples for the model to learn from. We also thought that revisiting the annotation guidelines for categories that are often confused to ensure they are clear and distinct would be a good idea. It would also help if there was more clarification on how to label jokes that could be under multiple categories. Another thought we had was to explore whether including more contextual features like n-grams or using word embeddings can help distinguish between similar categories. This could help in overfitting to the training dataset and have a better performance on the development dataset.