For our final model, we ended up going with an Ordinal Logistic Regression base classifier. Out of the box, this performed better than Logistic Regression as it made use of the nature of our labels. Ordinal Logistic Regression will take into consideration the fact that 1 and 2 have a set order and it is important to know that one rank is higher than the other, which Ordinal Logistic Regression helps us incorporate. Our final model had three features: bag of words, tf_idf scores, and Bigrams although a number of other features were tried (fully explained below). It reached a final test accuracy of 79%, which beats the baseline majority class classifier, which had an accuracy of 76%.

Since we were using ordinal logistic regression, we wanted to focus on featurizing the dataset beyond using a simple Bag of Words model. We began with basic features such as the length of the data point and the counts of different punctuation in the data point as we believed that the usage of certain punctuation like exclamation or question marks would imply a stronger personal tone, which could be associated with condescension. However, these proved not to be predictive features and we observed this empirically as our test accuracy decreased to 70%.

What we found surprising was that including NGrams as a feature worked well for us, the best one being a Bigram. This is likely because it broadens the scope of the words that we can attribute are unique to specific ranks. However, the drop when we increase this to a Trigram was a little unclear to us. It is likely because the countries and people that are referred to in the data points usually end to have two names in them, so they become helpful features.

We then added the TF-IDF count for each word as a feature and this (when used alongside BoW) increased our test accuracy from 75% to 77%. Since the TF-IDF weighs a word based on how infrequently it occurs, it makes sense how this would help the classifier as it can pay more attention to uncommon terms and by looking at the featurized training data, some of the words with the highest TF-IDF scores included “Your”, “Nude”, “Egypt”, “journalist”, etc. In particular, the word with the highest TF-IDF score, “your”, is important to highlight as, given that the chosen articles were filtered for news articles, the presence of the second-person pronoun should be minimal. News articles tend to be objective and thus, written in the third-person; unless there is a high frequency of the pronoun within quotes, it should not be very frequent among the chosen data points (articles). For the future, considering pronouns (the author’s use of first-person and second-person pronouns) could aid in identifying articles with more bias and condescension and is something that classifier made us think about adding to our guidelines.

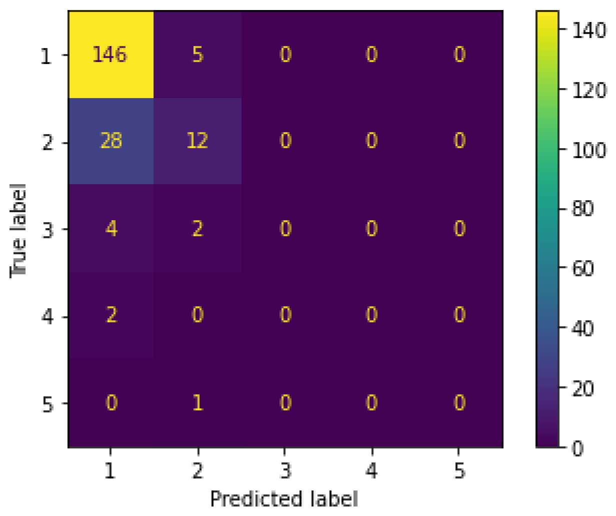
Another attempt with featurization was to try to create contextualized embeddings with BERT. We did this by using the pretrained Hugging Face model provided to us and we took an average of the word embedding for each word in that given datapoint to get one vector per data point. This turned out to not be as helpful as using features we specifically engineered, namely Bag of Words, TF-IDF scores, and Ngrams did better than BERT embeddings. Our benchmark for improvement was the majority class classifier which had an accuracy of 76%.

Some of the more complicated features we tried to include were POS and NER tags for the data points and we used the Flair external library to get these. We included these as features by tagging each datapoint and then aggregating the count of each NER / POS tag and adding it to the features dictionary. For example, a datapoint could have 5 VB's, 6 NNP's, etc. Our hypothesis behind using these features was that there would be more ADJ and ADV (Adjective and Adverb) tags in more condescending data points since they may imply tone or additional commentary being made by the writer. However, this hypothesis turned out to not be true as our empirical test accuracy dropped by 5% after the addition of these features. Possible reasons for this could be that by aggregating the counts of these tags, we were not keeping information about their relative positioning in the sentence. Furthermore, all sentences are likely to have NER tags such as Persons, Locations, etc. since we were using global news data points. Lastly, with only 600 training points and by seeing that our training accuracy was higher than the test accuracy, we were overfitting.

To combat the issue of losing information about the relative positioning of the POS tags, we created a function that combined the idea of N-grams with POS tags. Instead of creating a feature vector of the word n-grams, we did it with the POS tags and we tried a variety of n values (ranging from 2-5). The idea behind this was that if we see several adjectives right before a proper noun, it could be a more accurate indicator of tone and condescension. This did cause our accuracy to increase from the baseline 75% test accuracy (using just BoW) but was still not as useful as a word N-gram feature (reaching 79%). When used in conjunction with word n-grams, the accuracy dropped to 78% suggesting overfitting and we tried to combat this by using L1 and L2 regularization in the classifier but 78% remained the highest accuracy we achieved while including this feature.

Therefore, after trying many combinations of the above features and using regularization methods (especially L1 since we had limited data points and a lot of feature ideas), the best model remained the one that used the features of TF-IDF, Bag of Words and Bigrams, achieving a 79% test accuracy. We then investigated further into the mistakes our classifier was making and looked at the breakup of our data.

We first noted that the annotated dataset itself is extremely imbalanced. For example, across all of the annotated data, only 2 data points with a label of 5 exist; one of these data points was in the train dataset and the other was in the test dataset. Meanwhile, over 60% of the dataset was labeled as a 1. Overall, 1s comprised the majority of the dataset, with all other labels falling into the minority. It is likely due to this reason that the majority classifier performed well on the dataset and was difficult to beat in terms of accuracy; the heavy imbalance within the dataset made it inherently difficult to compensate through other model modifications (ex. class weights which we explain below)



This imbalance was reflected in the predictions. When the model was evaluated using a confusion matrix (shown above), it performed well in predicting data points with a ranking of 1; 146 true positives and 5 false positives. However, this accuracy dropped sharply for all other rankings. Out of 40 articles with true labels of 2, only 12 were accurate with a ranking of 2; the remaining 28 were incorrectly classified as 1s. Additionally, the model did not assign any data point a label higher than a 2. Thus, the remaining 9 data points that were classified as 3s, 4s, 5s, were also labeled as 1s and 2s.

This approach of mainly predicting 1s and 2s still led the classifier to still have a relatively high final test accuracy because our dataset is really imbalanced with there being a majority of these data points. To address this problem we tried upsampling and downsampling. We did this specifically by adding a class_weights parameter to the LogisticRegression model initiation and made sure to increase the percentage of 3, 4, and 5 rankings, with more of an emphasis on 4 and 5. We also made sure to downsample the 1s and 2s. Our most extreme percentages were about 30% upsampling on 4 and 5 and 10% downsampling on the 1s. In the end, this did not really make a significant difference to the model performance. This was likely due to how imbalanced our dataset was to begin with as the likelihood of encountering a 4 or a 5, even with replacement, is highly unlikely and not frequent enough for the model to learn to differentiate between 1s or 2s and 3s, 4s, and 5s. Thus, the adding the class weight penalties would be fruitless to rectify the imbalanced dataset.

It is important to note though that given the source of the data (the New York Times), conservative ratings of condescension would be preferred. Regardless though, for the future, it would be recommended to implement upsampling or downsampling into the initial dataset (before it is split into train and test data). Since the split was done for us, this was not something that could be implemented; however, adding more data points and upsampling to favor 4s and 5s.

As a result, it will likely be hard to generalize this data to other domains like another News Source because that imbalance is so specific to the NY times and the specific articles that we happened to have sampled. It will be difficult to find another domain or dataset that has this specific imbalanced bias towards a ranking of 1s and 2s. However, by annotating datasets from other news sources using a similar annotation guideline and then training the same type of Ordinal classifier on said data, we may still experience success.