Overview

I compare two methods of random sampling, then put them to the test against 8 models for text classification. At the end of the comparisons, I choose a single classifier from the 8 to put it to the test on a holdout dataset.

Two Sampling Methods

From the EDA, I discovered that my classes in the dataset were heavily imbalanced. In order to achieve better representation of classes in my embeddings, normalization of classes was required.

To achieve this I chose two methods to randomly sample each class - random oversampling, and random undersampling. Random oversampling artificially blows up under-represented classes to match the other classes while random undersampling omits a portion from the heavily represented classes to match the others. Both methods result in the same amount of documents for training, although random oversampling may temporarily yield better validation results due to the artificial duplication of documents from the under-represented classes to match the others. This could result in the model predicting an example that it has seen before during the training loop.

Dimensionality Reduction

In the EDA, I compared the Principal Components Analysis, and UMAP to perform dimensionality reductions. Without them, the dataset is extremely sparse, despite having removed stopwords and lemmatizing during the preprocessing. I ultimately chose UMAP because it yielded better results, and because it is capable of working with both linear and nonlinear datasets which makes it more versatile.

To ensure I have the optimal parameters set for n_neighbors, and to select the model that performs best with UMAP, I run several loops over a predefined list of n_neighbor values and compare the results for each.

Optimization

The results from the previous step to reduce dimensions and to optimize parameters showed clearly that K-nearest neighbors served best as a classifier given this dataset, with higher n_neighbors value tending to yield better results. Random oversampling scored at 93% accuracy, also with an f1 - score at .930403 at 25 n_neighbors. Meanwhile, Random undersampling yielded even better results at 35 n_neighbors with an accuracy score of 94%, and an f1 - score of .943539. Plotting the ROC curves showed an almost too-good-to-be-true result, scoring an AUC at .978 and .977 respectively. This meant that there were virtually no false positive results, and that seemed very suspicious.

Testing

The two methods of sampling were put to the test on a holdout dataset. They both performed, on average, at around the same level of horridness on each iteration. I have two guesses for why this had happened. One, my model could have been terribly overfitted and the topology of embeddings of the holdout dataset could be radically different from the one the model was fitted on. This seems to be the likely case due to the model’s preference for a certain class in the predictions. Another suspect is that the same parameters for UMAP dimensionality reduction were used for the preparation of the holdout dataset, and they were inappropriate selections for the holdout dataset.

Discussion

Although at first, employing the KNN classifier yielded promising results, the predictions over the holdout dataset proved it to be a worthless model for predicting unseen datasets. Reasons for why were discussed; however, the verdict has been inconclusive. There are even more ideas for why this may have happened -- such as how the TF-IDF vectorization procedure does not capture higher level embeddings, such as context. Because text documents are so dynamic, and there are myriad ways to describe and explain the same concept which we expect our models to be able to predict, it is clear that deep learning holds a clear advantage over simple classification models like K-nearest, or random forest. I will put this hypothesis to the test in the next phase of the project.