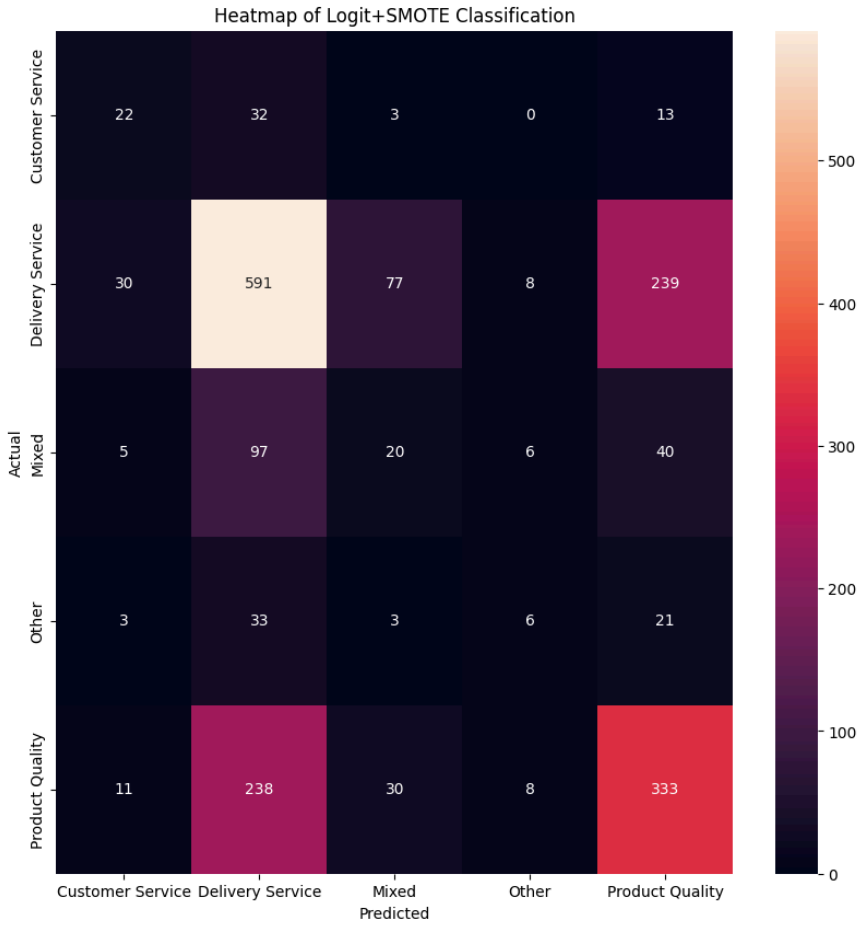
Executive Summary

We wish to classify Trustpilot reviews into a handful of desired categories. After some investigation and experimentation, the best efficiency and performance was identified with a logistic regression + SMOTE model. At present, training data is limited due to the small size of the Trustpilot dataset and the inconsistency of present categorization. Further investigation is required to determine next steps.



The Goal

Create a model to classify incoming Trustpilot reviews according to certain categories. The primary categories are Product Quality, Customer Service, and Delivery Service. In addition, the varied nature of the dataset requires the creation of a Mixed category, as well as an Other category for the rare review which has no clear connection to any primary category.

Data Exploration

The dataset originates from Hawkers Trustpilot reviews. Though it contains many fields, two are relevant to this effort: the text of the reviews, translated into English, and the tags associated with them.

These tags broadly approximate the categories into which we want to classify the reviews, with some inconsistency and roughness. Some existing tags include ‘costumer-service’ (sic), ‘product-quality’, ‘delivery-service, ‘devolucion’, and ‘serviciopost-venta.’ In addition, tags such as ‘spam’, ‘web-review’ and ‘p’ do not clearly map onto our schema, and their purposes should be examined.

The dataset provided contains 14961 rows and two columns, containing the translated english review text and the classifications respectively. Of these, 5663 rows contain actionable categories and 9298 contain blanks, used for training and unsupervised testing respectively.

Various Models/Methods

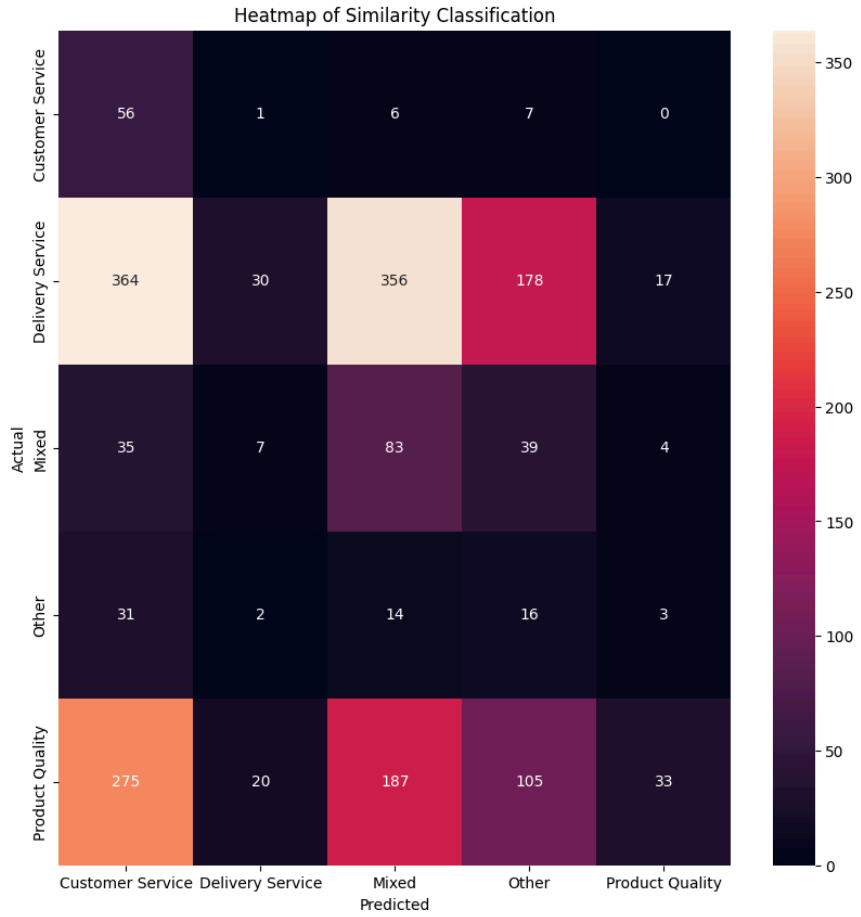
Similarity Comparison (Gensim)

The first approach to categorization involves calculating document similarity using the Gensim MatrixSimilarity module. In particular, we use a bag-of-words (BOW) approach to gather and find the associations of all the relevant words in the corpus, and can then measure the similarity of a given document (anything with multiple words, from a sentence to a whole paper) to any other document.

To classify an incoming document, we gather all our existing classified documents, called queries, and compare all of them to the incoming document. Each comparison produces a match score from -1 to 1, with higher scores indicating a closer match between the documents. We then average the scores across each query category (such as Customer Service, Product Quality, etc.) and classify the incoming statement based on the highest average score.

The computational cost of this method scales linearly as the set of classified documents increases, since incoming documents must be compared against each individually. In addition it is not possible to add new classified documents into the classified corpus dynamically: the corpora and dictionary must be retrained with the whole set each time it expands. Even at present with a small dataset, this takes several minutes. Thus, using this

method for large datasets is not recommended, and the similarity functions will be deprecated.



Time to train and classify: 4m 9s

Sidebar: Why not use k-classification?

Though an early version of this model classified according to class majority of the k-highest scores for an incoming document, we find that large imbalances in document categories make this method less reliable and prone to categorizing according to the largest categories, since the word choice within categories varies considerably.

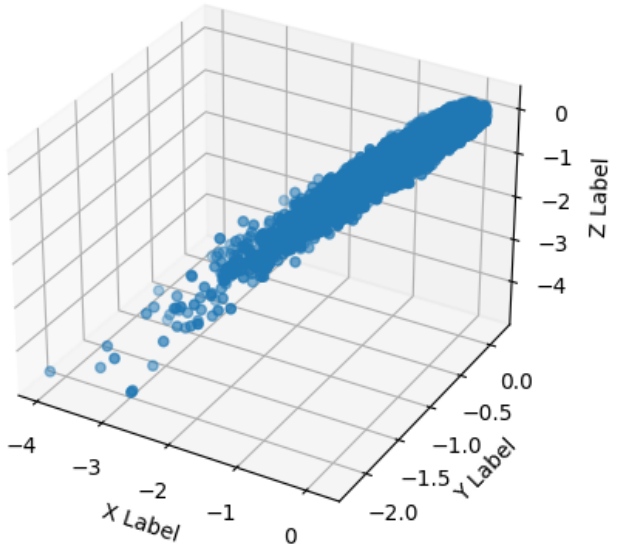
Doc2Vec (Gensim)

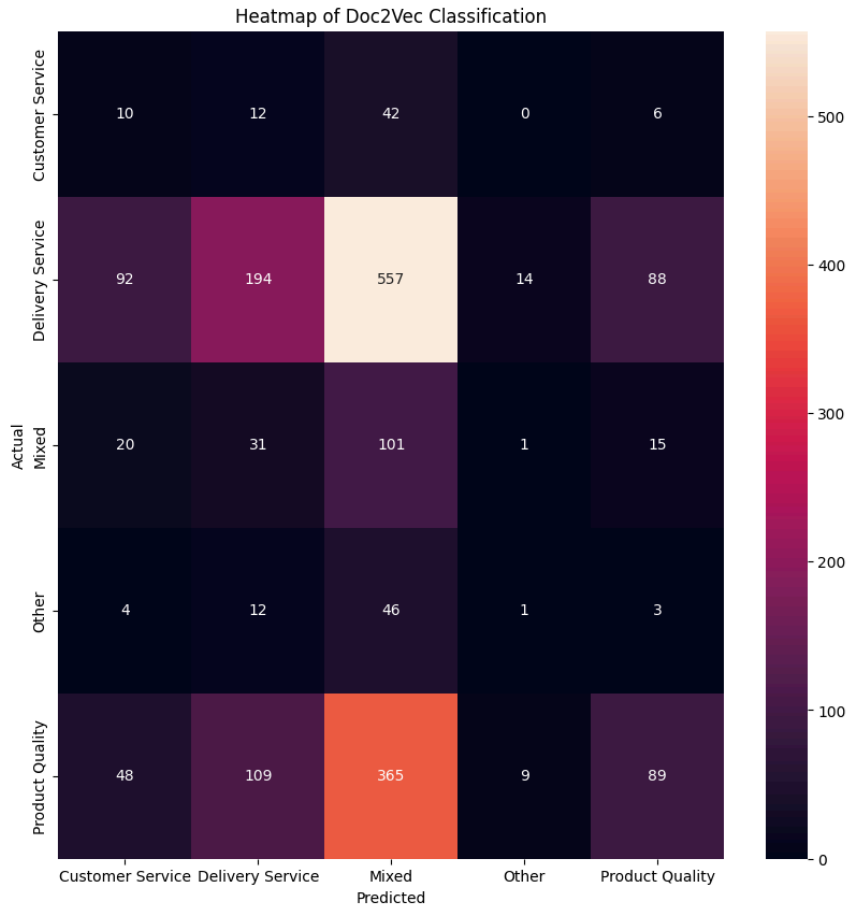
The second approach to classification involves Gensim's Doc2Vec functionality, an extension of the principles of Word2Vec algorithms (but which does not, as one might expect, run Word2Vec as a sub-component). Doc2Vec takes a corpus of documents and, taking the whole corpus into context, transforms each document into a vector in n-dimensional space (n may be set as a desired parameter). In addition, each training document is 'tagged' with one of our desired classifications (such as Customer Service, Product Quality, etc.), which is used to influence the vectorization and attempts to find clusters in n-dimensional space according to our tagged categories. Proper functioning requires a sample of trained documents which satisfies some combination of sample size and variety, where variety also increases as document length increases.

Under other circumstances, the classification can sometimes be confounded by other elements within the dataset. The present dataset, for example, which is small and composed of generally very short documents, primarily varies not based on the offered categories, but on sentiment, since positive messages tend to be very short and negative ones tend to be longer, with certain tokens being the province of the extremes. In the early stages, this approach produced an ad-hoc sentiment analysis machine despite having no exposure to the numerical ratings of each document.

As with the similarity method, new training documents cannot be added dynamically. However, classification time after training increases slower than N, and it operates much more quickly in general. Given the present size of the classification data (<25k), this is not a significant expenditure of time or compute.

In addition to the standard heatmap for visualizing performance on already-classified data, Doc2Vec performance can also be visualized, though not quantified, by simply visualizing those vectors in 2- or 3-dimensional space. For greater than 3-dimensions, visualization is accomplished using the UMAP module, which creates a lower-dimensional projection based on the principles of Principal Component Analysis (PCA).





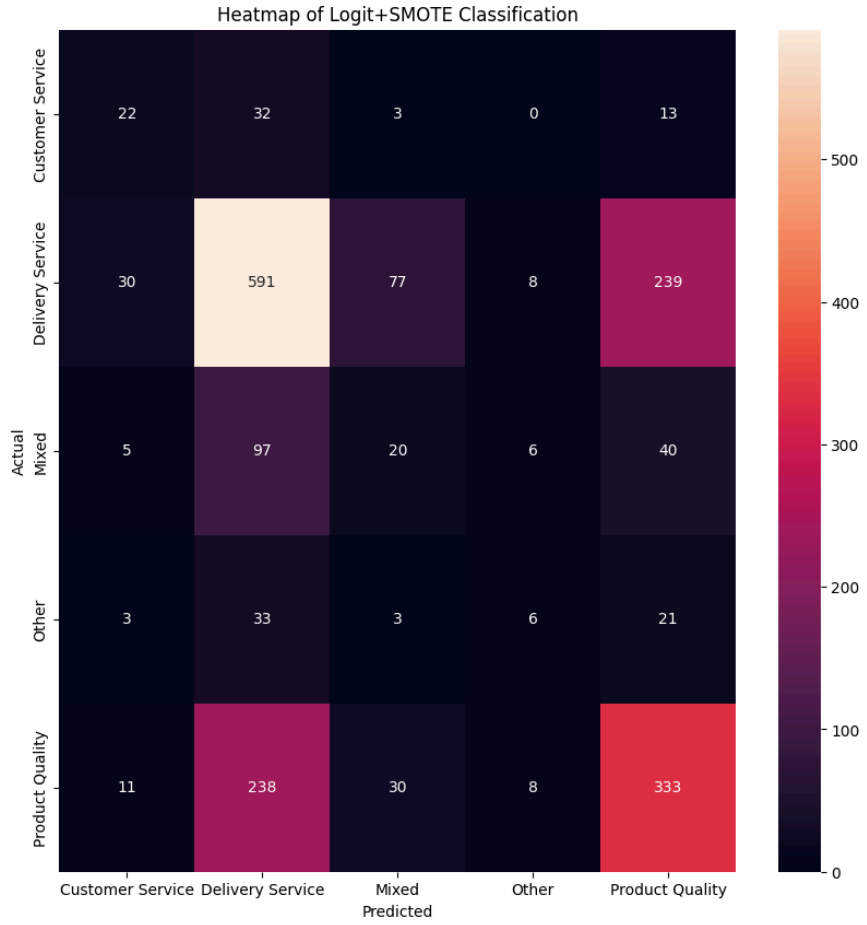
Time to train and classify: 2.07s

Logistic Regression + SMOTE (SKLEARN)

The third method uses a more traditional machine-learning approach, the logistic regression algorithm, which is commonly used for classification of quantitative data. It can be adapted to text classification using a similar approach as in the previous methods, transforming the documents into vectors using TF-IDF (Term Frequency-Inverse Document Frequency) and classifying them as quantitative data. Note that when vectorizing the data, there is the choice to include or not include the tags. Doing so can create issues with overfitting and data leaks in train-test sets, but it is not clear that doing so is disadvantageous when classifying new data.

This approach runs into the same issue as the k-classification approach of the similarity method, except that logistic regression does not provide percentage matches which can be used to create an alternative classification system. To fix the problem of class imbalance, we use the SMOTE algorithm (Synthetic Minority Oversampling TEchnique), which creates synthetic data of the less common classes. Since SMOTE is applied after vectorization, no synthetic verbal samples are created which can be interrogated. The process is entirely mathematical. Having used SMOTE to address class imbalance, we apply logistic regression as normal.

Of the three methods detailed here, logistic regression with SMOTE appears to have the best performance in classifying new data, but this is based more on subjective impressions, and it is not clear how the capabilities of different models will scale as more complete training data is provided. In addition, it is by far the most efficient, and retraining it for each incoming review, even on much larger datasets, would not be a significant cost in time or compute. Depending on the urgency of reviews, batching into small groups on a daily or hourly basis could further improve efficiency.



Time to train and classify: 613ms

Challenges

The dataset is quite small overall, with <25k Trustpilot reviews currently in the corpus. Of those, around 15k have been translated into English, and less than 6k of those have also been tagged. Because of this small size, the performance of the models is limited.

If we can translate and tag the entire 25k review set, we can expect improved performance, but our ability to evaluate that performance will be limited, and there is no guarantee that a five-fold increase in training size would yield a similar performance improvement.

The problem of class imbalance remains a challenge even with methods like SMOTE to offset it. In addition, a model is only as good as the training data it receives, and both the preexisting tags and the actual classification have some issues. A comprehensive review of Trustpilot classifications according to a formal schema could help.

Going Forward

A thorough approach to the problem of classification should interview the Customer Service team and examine how the platform is presently used, in order to determine what the current cost of manual classification is and whether further steps toward an automated classification system are merited.