**Methodology**

In summary, our deep learning pipeline, akin to transfer learning, employs two distinct machine learning models for natural language processing and prediction. Following dataset cleaning and balancing, we enhance reviews with pseudo-labels using the 4-billion-parameter BART model as a zero-shot classifier, assuming accurate ratings in the source dataset. Subsequently, we calculate likelihoods for each review's association with pseudo-label classes, defined arbitrarily based on product sentiments and represented as numbers from 0 to 1. Finally, our neural network, utilizing both pseudo-labels and target ratings, is trained to provide robust predictions for a user's rating based solely on the detailed content.

During dataset profiling in the data cleaning and normalization phase, we identified imbalances in both the rating and comment length distributions: Over 100,000 comments received 5 stars, while others fell below 50,000 or even 20,000, indicating a rating distribution issue. Additionally, comment lengths varied significantly: with an average around 500 characters and a standard deviation of 700 characters, some comments exceeded 30,000 characters or had zero characters.

To address this, in the "balance\_data.ipynb" notebook, we filtered out entries shorter than 50 characters or longer than 2000 characters. Moreover, after shuffling, we retained only 10,000 entries per rating group, ensuring equal representation across one-to-five-star reviews. Therefore, the balanced dataset contains 50,000 entries.

With the data preprocessed, we may proceed and use the zero-shot classifier to label the comments. The model of choice is a BERT model trained on the MultiNLI dataset, which, as introduced by Samuel Bowman et al., contains a crowd-sourced collection of 433,000 sentence pairs with annotations and various genres in English. The comments, along with five possible classes ("good," "decent," "mediocre," "bad," "horrible"), are fed to the classifier, which predicts the likelihood of each comment belonging to these classes on a scale from 0 to 1. Labeling 50,000 entries sequentially on an NVIDIA Tesla V100 takes approximately 3 hours. As a result, we have four scalar values for each comment for rating prediction along with a true rating of the comment also in numbers. In other words, we have transformed comments into four numerical values and an outcome variable that is, by nature, also a number.

Utilizing the labeled dataset, we train a neural network with multiple linear layers and activation functions to capture non-linearities. To mitigate overfitting, we employ large batch sizes (4096 and 8192) and incorporate batch normalization layers after each linear layer for improved training speed and generalization. Exploring hyperparameters, we experiment with SGD and Adam optimizers, varying learning rates, both leaky and vanilla ReLU activation functions, and different layer configurations. The model with the lowest loss is chosen as the target for subsequent predictions.

**RESULTS**

In the broader context, most models exhibit an accuracy hovering around 0.71, surpassing the randomness of coin-tossing and thereby suggesting a certain level of indicative capability.

However, the noteworthy observation emerges when considering the validation loss, consistently approaching, if not occasionally surpassing, the training loss, even in the initial epochs. This, coupled with the persistent presence of relatively high losses across all models, strongly indicates that our model is grappling with overfitting.

Here are loss and accuracy graphs of one random model from our model pool that have relatively good validation loss (~0.6). From the graphs, we can clearly tell that it is overfitting.

**Analysis**