Grape Expectations: Using CNN to Analyze Wine Reviews for Classification

Rohan Bhargava, Darin Moore, Griffin Stamp, Anthony Tseng

What task are we solving:

The members of our group are all regular consumers and big fans of wine but none of us are well versed in the factors that make one wine different from another or what qualities make for a good wine. We wanted to see if natural language processing could provide us some insight that we couldn't just find with Google searches.

We analyzed a wine review dataset from Kaggle that includes 130,000 wine reviews from professional sommeliers [1]. These reviews were scraped from the Wine Enthusiast website in 2017 and are typically a few sentences each, featuring detailed flavor profiles of the wine and other such commentary. The dataset also includes various attributes like wine name, variety, price, rating, country, and region, along with additional factors.

Our original idea when looking at this dataset was that we should attempt to predict a wine's rating based on its review description. Unfortunately, Wine Enthusiast only publishes reviews for wines that are rated 80 points or higher out of 100. Consequently, the ratings are all very similar to each other and only fit within that 80 - 100 range, so we did not want to explore such a limited range of ratings and predict only in that interval.

The next goal of our project was to determine a wine's region of origin given a text review. Our hypothesis was that the wine's region would influence the soil the grapes were grown in; when combined with local/specific wine-making methodologies, the resultant wine should have a distinct flavor profile. Therefore, a text description may be able to capture the wine's subtleties and essentially represent that unique flavor profile in words, allowing us to accurately determine the region. However, upon starting the project, we realized determining a region may both be difficult and not that interesting from a consumer perspective. A quick glance through our dataset revealed 1,229 different regions with some very specific, obscure ones. Ultimately, this led us to consider that predicting the type of wine may be more directly impactful and interesting since most people recognize types, not region origins of some wine anyway. This is also interesting because text reviews for different types of wines may be similar if they have similar tastes or flavors; for example, words like "dry," "bold," "robust," "nutty" may consistently occur for two different types of red wine.

As a test of our model, we first attempted to simply determine whether a particular wine was made using red or white grapes based on the review description, i.e., predicting the wine's color. To do so, we had to augment our dataset since the dataset did not contain colors for each wine review. As such, we took a grape dataset from Cal Poly, San Luis Obispo [2] that

essentially contained 20 different grape varieties and also their color. To actually now merge this with our reviews dataset, we used Python and the Pandas library to look at the grapes dataset and create a dictionary mapping of grape variety to color. With the dictionary, we were able to create a new column called “color” in our review dataset and simply dropped all rows/reviews that were for a grape variety we did not have in our grapes dataset, leaving us with approximately 41,000 reviews. Using this augmented dataset proved fruitful in predicting a wine’s color given a review. Thus, we felt confident in the model’s accuracy and moved on to predicting the wine’s specific variety. Later on, we will detail what additional work we did to augment this dataset more.

Model:

For our project, we decided to utilize a convolutional neural network (CNN). We chose to use a CNN over a recurrent neural network (RNN) because a CNN will learn meaning from repeated patterns, independent of location within text. On the other hand, an RNN will recognize patterns based only on previous context. We believe that identifying repeatable patterns of words is more important than the ordering of words within a text when determining the wine type since reviews may have similar phrases, but not necessarily have these phrases in the same location of their review.

Our model is non-static, meaning we have our vectors pre-trained with word2vec tuned specifically for wine reviews (wine2vec_code.ipynb is the Jupyter Notebook containing the code used to generate the word embeddings). With the vectors pre-trained, we get greater accuracy since the vector representations are contextually generated over all wine reviews and rely on word to word relationships instead of each words relationship with the labels.

We use a variety of kernel sizes (3, 4, 5) with 100 filters for each kernel size. We then run a two-dimensional convolutional neural network (torch.nn.Conv2d) for each kernel size. From the outputs of each neural net, we use Max Pooling to take the largest element within the different windows. Max Pooling was used instead of Average or Sum Pooling as it has better performance in practice[4]. After receiving the max element, we then run it through a single linear perceptron which is our final predictor - converting into scores for each of the classes (wine types). These scores can then have the softmax function applied to them in order to come up with the category prediction.

For our wine-type classification, we use the model presented above, but first classify the wines based on color (red or white) and then try to classify based on color. We believe that this will lead to better classification as there will be less noise for training if we know what color wine we are classifying.

Training:
In order to train our model we break up our training set into batches and use our CNN along with PyTorch's implementation of the Adam optimizer which attempts to minimize the loss. Loss is calculated using PyTorch's cross entropy loss function which is designed to work with multiclass classification.

To calculate the accuracy, we create a test set which is ten percent of the overall dataset. We then make predictions using the trained model and compare the prediction to the actual value for each element in the test set. Our testing accuracy is determined by the number of correct predictions made my the model divided by the total number of elements in the test dataset. Once the model is trained, a snapshot of it is saved for future testing/predicting.

Results:
The first predictive task we attempted to accomplish was determining the color of a wine (red or white) based on its review description. After augmenting the dataset with the additional information about wine color, we ran the basic CNN on this data. We ended up with a very high accuracy of ~98% without needing to do any hyper-parameter tuning. We believe this is due to the fact that red and white wines tend to have very different taste profiles and especially with professional reviews that is able to come out. We also wanted to test this on non-professional reviews and from our results from the demo, saw that it was able to do well on those as well. From here, we wanted to see if we could solve the tougher problem of could we narrow down the wine type as well.

Our baseline for results was to run the CNN from the starter code on the entire un-modified dataset while attempting to predict wine varieties. This model had an accuracy of 43% which we set as our goal to beat by improving the model as well as what it was being trained on.

We first decided that maybe an improvement we could make to our model was training it just on the red wine data - this could potentially allow it to analyze the more fine grain differences between red wines with no white wines as noise. We came back to our augmented review dataset (augmented with the wine color) in order to divide the data into red wine vs white. This resulted in an accuracy of around 69%. We realized some varieties in our grape dataset were misspelled and did not match those in the reviews dataset, and many varieties were not represented in the grape dataset. To deal with this, we slightly modified this .csv file by fixing grape varieties and their spelling to match what was in our reviews dataset, as well as adding the next 48 most common grape varieties and their color that we observed in our reviews dataset. This brought us to 108,000 total reviews and 68 varieties. Unfortunately, this actually dropped our accuracy down to 65%. We assume this is because introducing more varieties also

introduces more variance; some varieties only have 200 reviews, for example, and thus might be harder to predict. Additionally, some wine varieties have only subtle differences between each other.

Our next change to the model was limiting the number of categories the model had to predict from. We felt that this might make the final data more linearly separable and make it easier on the final linear predictor to classify between the categories. Towards this end, we decided to analyze only wine varieties that had more than 1,000 reviews and drop the rest; this gave us 94,000 total reviews and 24 varieties. Doing so slightly increased our accuracy back to 69%.

One of the big changes we made in order to improve our model was to switch from using non-static word embeddings to using word2vec. Initially, we considered using the pretrained Google word embeddings but we felt that those word embeddings might not capture the context and meaning that words are used within wine reviews specifically. Additionally, some of the words within wine reviews (e.x. “Tannins” or “full-bodied”) might not even appear within precomputed embeddings from news/twitter sources. From here, we trained two different word embeddings - one that included all of the words, and the other that only provided embeddings for a subset of the words. This subset was limited to words that met the threshold of appearing at least 5 times within the corpus of the review descriptions. Any words not meeting this threshold would simply be given an embedding of all zeros. We felt that this subset might remove unnecessary noise by ignoring the least common (and thus irrelevant) words. These two different word embeddings resulted in accuracies of 81% for every word being embedded and 82% for the subset embeddings. Thus, focusing on the more common words yielded slightly higher accuracy. In the future, we might want to work to tune that threshold more.

We noticed that the model tended to be more accurate when it had less categories to classify wines into. We hypothesized that wines of a particular variety might have a large spread with some wines within a category having characteristics of other types of wines. Thus as more and more wines are introduced to the model, it becomes harder to delineate between each type. To test this hypothesis, we limited the model to simply classify reviews of the two most popular red wines, Cabernet Sauvignon and Pinot Noir. This model was able to hit 93.00% accuracy - indicating that we can develop a unique flavor profile for a wine.

In the future, to improve our model, we would want to introduce non NLP features as part of the final predictor. Since the wines are categorized by grape varieties, and certain varieties of grapes might be more expensive or harder to grow, price could be another feature to add on to our prediction model. Furthermore, particular regions could also be more likely to grow certain

types of wine (e.x. Napa is famous for Cabernet Sauvignon) so we could one hot encode the most popular regions as part of the final predictor.

References

1. Kaggle Wine Reviews, https://www.kaggle.com/zynicide/wine-reviews
2. Grape Colors Dataset, https://users.soe.ucsc.edu/~eaugusti/archive/365-spring13/365-files/datasets/csv/WINE/
3. Word2Vec - Wine Vocabulary, https://www.kaggle.com/erdiolmezogullari/word2vec-wine-vocabulary
4. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, https://s3.amazonaws.com/academia.edu.documents/29882418/icann2010_maxpool.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1544644896&Signature=l09Vs02a92kfeDCtDfUDNI8ipw4%3D&response-content-disposition=inline%3B%20filename%3DEvaluation_of_pooling_operations_in_conv.pdf

Images: