Predicting Wine Quality using Text Reviews

Mandy Gu Follow
Feb 12 · 9 min read

As someone who doesn't drink a lot of wine, I usually rely on the 100 point Wine Scale and the purple "Staff Recommended" sticker to pick something off the shelf

I will be using this Kaggle data set which contains wine reviews, the rating of the wine (measured in points) and other relevant information scraped from WineEnthusiasts.

The data set contains two files scraped on separate days. I will be using the file **winemag-data_first150k.csv** as the training set and **winemag-data-130k-v2.csv** (scrapes from a later date) as the

testing est.

Here is the repository for the code.

. . .

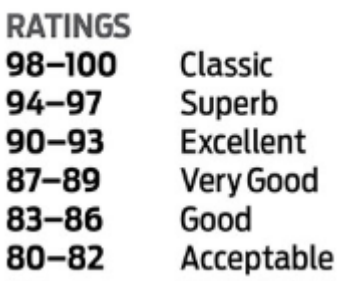
Objective: Train a Machine Learning model which predicts Wine Quality based on Text Review

WineEnthusiasts use a points scale ranging from 1 to 100 to rate their wines (1 being the worst, 100 being the best). Unfortunately, the website only posts positive reviews, which is why the range of the scores in the data set only range from 80 to 100.

This means that our data set is not representative of the problem we are trying to tackle. *Any models built using this data would only be applicable to well received wines.*

Gather some domain knowledge

Before we start with our analysis, we should try to gather some domain knowledge. After exploring the website and reading some other wine related resources, I found a nifty classification system based on the points system.



To an end user (i.e. wine shopper), the points are only as important as the information they convey. If we frame this as a classification problem with the above breakdown, we can retain the necessary information on wine quality while reducing the dimensionality of the problem.

Decision: I will frame this as a Sentiment Analysis problem where the review will be used to determine whether the wine is [a] Classic, Superb, Excellent, Very Good, Good or Acceptable

Exploratory Data Analysis

The next step is getting to know our data a little better. This can give us important insights to tackle the problem better. In addition to the review and rating, there are other relevant information such as the wine price, variety (grape type) and location the wine was produced.

	country	description	designation	points	price	province	region_1	region_2	variety	winery
0	US	This tremendous 100% varietal wine hails from ...	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz
1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110.0	Northern Spain	Toro	NaN	Tinta de Toro	Bodega Carmen Rodríguez
2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sauvignon Blanc	Macauley
3	US	This spent 20 months in 30% new French oak, an...	Reserve	96	65.0	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi
4	France	This is the top wine from La Bégude, named aft...	La Brûlade	95	66.0	Provence	Bandol	NaN	Provence red blend	Domaine de la Bégude

Preview of our data

If we wanted to build a more comprehensive model, we can also include these features as predictors to determine the quality of the wine. To combine text descriptions with other predictive features, we can either create an ensemble model (with the text classifier being one of them) or a hierarchical model where the results from the text classifier will be used as one predictive variable.

For the purpose here, we will only be exploring the relationship between the review and wine score.

Examining Data Integrity at a High Level

There are no missing entries from the points and description column. As we have noticed before, the wine points are shifted to the right. From my personal experience, the prices are also shifted to the right.

```
(              points          price
 count  150930.000000  137235.000000
 mean       87.888418      33.131482
 std         3.222392      36.322536
 min        80.000000       4.000000
 25%        86.000000      16.000000
 50%        88.000000      24.000000
 75%        90.000000      40.000000
 max       100.000000    2300.000000,)
```

Output from data.describe()

```
RangeIndex: 150930 entries, 0 to 150929
Data columns (total 11 columns):
Unnamed: 0     150930 non-null int64
country        150925 non-null object
description    150930 non-null object
designation    105195 non-null object
points         150930 non-null int64
price          137235 non-null float64
province       150925 non-null object
region_1       125870 non-null object
region_2       60953 non-null object
variety        150930 non-null object
winery         150930 non-null object
dtypes: float64(1), int64(2), object(8)
memory usage: 12.7+ MB
```

Output from data.info()

Examining the Textual Data

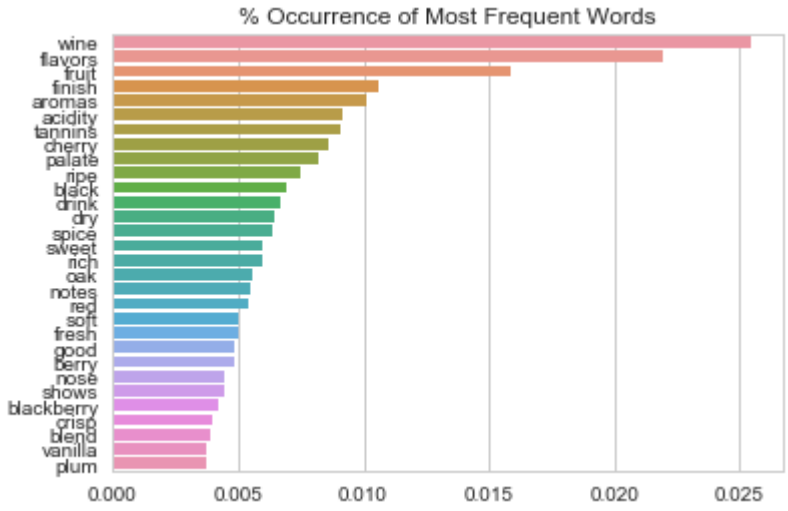
The reviews look very "clean". There doesn't seem to be any grammatical nor spelling errors and all reviews are written in very concise language.

Sample review:

```
This tremendous 100% varietal wine hails from Oakville and
was aged over three years in oak. Juicy red-cherry fruit and
a compelling hint of caramel greet the palate, framed by
elegant, fine tannins and a subtle minty tone in the
background. Balanced and rewarding from start to finish, it
has years ahead of it to develop further nuance. Enjoy
2022–2030.
```

Some of the reviews require wine knowledge to fully understand. In the above example, *tannin* is a textual element that makes wine taste dry.

We can also take a look at how frequently the most common vocabulary terms occur.



the most commonly occurring word is "wine" which occurs just over 0.025% of the time

Preparing for Classification

Previously, we identified our classes as the following:

We can map reviews by their points to one of our classes.

Unfortunately, our data is not very balanced.

There are no reviews in Class 4 (94–97 points — Superb) and most of the reviews are concentrated in Classes 1,2 and 3. Unbalanced classes are problematic, but can be addressed by sampling the larger classes or setting class weights. The total absence of one class, however, is much more concerning.

Decision: I will be combining Class 4 and 5 together as the

new Class 4. Any reviews in the range or 94–100 points will fall under this class.

Should the Textual Data be cleaned?

We also have the option of cleaning/normalizing our wine reviews. The biggest factor affecting this decision is the type of learning algorithm we want to use. If we choose to embed each review as its own vector and pass the input into an one-to-one classifier, then we should spend a great deal more time normalizing the text. On the other hand, processing the text sequentially as a representation of many vectors makes normalizing less important.

Processing the text sequentially (where typically each word is represented by its own vector and word relationships are learned) also helps address ***word-sense disambiguation*** (the same word possessing multiple meanings) and identify ***synonyms***. Since the reviews share very similar context with very specific lingo, I am not very concerned about word-sense disambiguation or synonyms. But considering the generally positive sentiment of the reviews, I am concerned that an one-to-one classifier might not be able to pick up the subtle differences between neighbouring classes.

Decision: I will be using a Recurrent Neural Network to predict the classes by passing each review as a sequence of vectors. I will also keep text descriptions in their original form.

It's hard to tell if this choice is better than using an embedding technique such as TF-IDF and passing that into an one-to-one classifier. Maybe in the future I will try that too and compare the

results.

Embedding the Text

The natural choice for embedding each word as its own vector is to use neural based embedding techniques such as word2vec, GloVe or fastText. We also have the option of training our own embedding model or to use pre-trained vectors. Since the text we are working with is nothing out of the ordinary, using pre-trained word vectors might help us understand language better.

Decision: we will use pre-trained word embeddings.

But which family of embeddings shall we use? I will eliminate fastText right away as it builds word vectors by summing vectors of character n-gram level. Since the type of text we are working with is not likely to contain out of vocabulary words (no misspellings, unusual slang or abbreviations), it will not benefit from this granularity.

Even so, it's often hard to identify which technique will work best. Since I find some of the wine descriptions to be analogous in nature, I will opt for GloVe in hopes that learning from co-occurrence structures will help better understand semantic comparisons.

Decision: we will use pre-trained GloVe vectors

Several pre-trained GloVe word vectors can be downloaded here. I will be using the Common Crawl with 840B tokens as it contains the largest vocabulary and is also case sensitive. Each word vector from glove.840B.300d has 300 coordinates.

Before loading the pre-trained embeddings, we should define some fixed parameters.

Num_classes: self explanatory — this is the number of classes we are working with

embedding_dim: this is the dimensions of the word vectors

epochs: number of forward and backward passes through all of the training examples

batch_size: the number of training examples in each pass

max_len: the maximum length (in words) which will be considered in a text description. Any descriptions with more than 100 words will be truncated. Any descriptions with less than 100 words will be padded to meet this length.

class_weights: we observed earlier that our classes are not very balanced. Classes with higher weights attached to them (class 0 and class 4) will have a higher impact on the learning algorithm. **Each instance of class 0 is treated as 7 instances.**

Note on choosing max_len: capturing too few words results in lost information, but capturing too many results in issues with data sparsity. Ideally, we want to choose a value which captures the entirety of most text descriptions without introducing too many zero sequences.

. . .

Loading in the required libraries:

We also have to one-hot encode the target class.

Train-Validation Split

Even though we already have a designated test set, it can still be a good idea to divide the training set into training and validation for the purpose of parameter tuning.

Tokenizing Inputs and Preparing the Embedding Matrix

I will be using Keras's text_to_sequences method to preserve the word sequence of the text. Each word will be mapped to its vector from the pre-trained word vectors. Any descriptions under 100 (max_len) words will be padded and any descriptions over 100 (max_len) words will be truncated so that the input is the same length.

Words found in the text that do not appear in the vocabulary of the pre-trained word vectors are initialized to the zero vector.

Note: If there are many words outside of the vocabulary, then it might be worthwhile to devise a more intelligent way of initializing these words

Training the Classifier

Since the text descriptions are generally short, I will be using GRU units as opposed to LSTM units. With shorter descriptions, we have less need for a memory unit and can benefit from GRU's more efficient learning algorithm.

I will also be utilizing early stopping, which (in this case) uses validation accuracy to determine whether we should continue training the network. When validation accuracy continuously drops over several epochs, early stopping will kick in and end the training process. It will also save the optimal weights as a "checkpoint" (in this case as model.h5) and rewrite the weights if accuracy improves. With early stopping, we can let the network train for many epochs without worrying too much about overfitting.

The patience parameter is a threshold used to determine if/when to prematurely end training. Patience = 3 indicates that an early stop will be initiated if there are no improvements to validation loss after 3 epochs.

The structure of the Recurrent Neural Network is quite simple. The network has one bidirectional GRU layer with 50 units followed by a pooling, dense and dropout layer. Bidirectional means that the network will learn the text sequences in their original order as well as the reverse order in which the words appear.

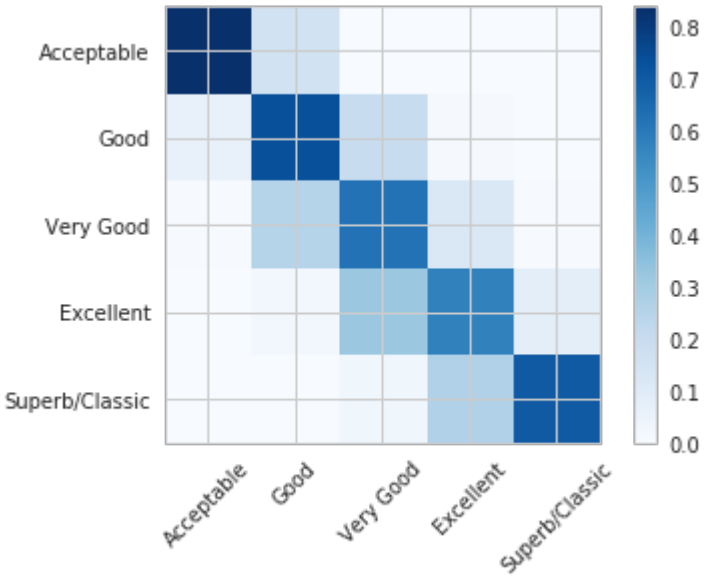
The classifier is also optimizing on accuracy. Accuracy does not differentiate between incorrect predictions the way a human might. To a human, predicting a Class 0 as a Class 4 will be worse than predicting a Class 0 to be a Class 1. To the network, it makes no

difference. For future implementations, it might be worthwhile to devise a metric which reflects this relationship.

It's time to assess our model — let's bring out the test set!

Accuracy over Test Set: 64%

Let's take a look at the confusion matrix. In this matrix, the values are measured as percentages to reflect the unbalanced nature of our data.



Not a bad start!

. . .

We should keep in mind that since all of wine reviews are very positive, this classifier will only be applicable to *well received wines*. It would be interesting to try this with a different data set in the future.

Cheers!

Let’s end with a wine picture