CIS 530 Final Report

Wine Rating Prediction by Reviews

James Bigbee, Sae Yeon Chung, Dong Ku Im, Nayeong Kim, Jai Thirani

Abstract

Product quality prediction has been a popular field of research within natural language processing. Customer ratings is one of the most valuable measure of product quality, and companies actively analyze customer reviews for this purpose. We would like to propose a machine learning model to predict the quality of wines through customer reviews. From our dataset, reviewers have given evaluation score ranging from 80 to 100 points to each wine. This problem can be seen as a multiclass classification when we divide the range into five classes of quality (e.g. 80 - 84 being fine, and 96 - 100 being excellent). We train our model using preprocessed review texts with corresponding class label based on its review point. We compared different classification models, including baseline model implementing bag-of-words, Naive Bayes N-grams, recurrent neural network, and word vectors. After testing each model, **RNN using AllenNLP** had the best performance.

Introduction

For our project, we are attempting to give wine a score based upon a written review. Wine is often graded on a scale of 0 to 100, points are awarded based upon such criteria as color, taste, and aroma. Also, for simply being wine, a wine is given 50 points. The wines that we used were all scored in Wine Enthusiast, and unfortunately we could only find datasets that had scores above 80, which is considered a good wine. Therefore our task wasn't determining the good from the bad, but instead it was the much vaguer task of differentiating the good, the great, the outstanding, and amazing.

Although, we weren't dealing with negative reviews our task is still that of sentiment analysis, since though the range is limited we are trying to extract the opinion of the author. This task of sentiment analysis also also further complicated for us by trying to split the prediction into 4 classes not just positive or negative.

The model should take a text review and output a score of 1 through 5 representing the positivity of the review. (1 is 80 - 84, 2 is 84-88, 3 is 88-92, 4 is 92-96, 5 is 96-100)

Example 1:
(80 points)
Candied plum and red berry aromas smell like red licorice. Flavors are raisiny, with sourness and astringency. → model → 1

Example2:
(100 points)
In a stunning lineup of Cayuse Syrahs, the En Chamberlin wins by a nose. It's smooth and silky, with a tongue-bending blend of flavors that include blood and iron, umami and salt, at first overtaking the pure fruit, but adding tremendous depth and power. The endless finish unfolds into a wine with exotic spices and complex layering. -> model -> 5

The two examples above illustrate how we would like the model to act in classifying the text. The first example is one of lowest reviews and the second example is one of the highest reviews.

We believe this task was a worthy task not only for the challenge of differentiating a small range of sentiment, but also for quinques of the language used in the reviews. Unlike reviews on for example Amazon, the reviews don't use words like bad, but often act more as a description with words only having conotions of good or bad, such as sour. Therefore this task contained a few challenging points which made the task interesting and separated from a standard sentiment analysis problem.

Literature Review

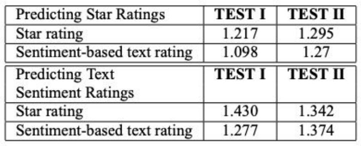
In 2017, Reddy, Ch Sarath Chandra, et al. published a paper titled "Prediction of star ratings from online reviews." Their research predicts the star rating of a product in a scale of 1-5 given the reviews for that product. As a first-step, they pre-process the review data by removing punctuations and stopwords. They also tokenize and lemmatize the text to generate root words for each review. Then, they use a 'bag of words' model which counts the occurrence of each words in the review. This model is then used as features of the classification task.

For model training, they first implemented a Multinomial Naive Bayes(NB) model as a baseline, and got a precision, recall, and fscore of each 0.63, 0.63, and 0.62. The overall accuracy (the proportion of correct prediction of star ratings) was 0.628. Then, they also tested using word bigram multinomial NB, trigram multinomial NB, bigram-tigram multinomial NB, and random forest.

The results show that while random forest took a significantly long time to train the data compared to other models, the overall accuracy of 0.698 was the highest. Also, compared to earlier researches that used Adaboost and Logistic Regression to get a f-score of 0.46 and 0.50, they have improved the results significantly.

Prior to Reddy, Ch Sarath Chandra, et al. (2017), Ganu, Gayatree, Noemie Elhadad, and Amélie Marian pioneered a way to improve rating predictions using review text content, i.e. sentiment analysis. Their research paper aims to predict customer ratings based on review text content and sentiment analysis on the text content. This research is meaningful in that it was the first paper that combines natural language processing, machine learning and collaborative filtering to harness the wealth of detailed information available in web reviews. Their test data set consist of 52,264 restaurant reviews on 5,531 restaurants from Citysearch New York. The researchers considered a review as a set of sentences, each with their own topics and sentiments, not as a unit of text. As the research was in a restaurant review scenario, they came up with topics related to the domain, which include six categories of Food, Service, Price, Ambience, Anecdotes, and Miscellaneous. These topics are what customers evaluate the restaurant on. Then, they also have an associated sentiment, which include Positive, Negative, Neutral, or Conflict. The research hypothesis was that the text of a review (as approximated by its associated topics and sentiments) is a better indicator of the sentiment of the review than the coarse star rating.

To test their hypothesis, they set up a recommendation system scenario, and explored whether textually-derived ratings are better predictors than numerical star ratings given a user's restaurant preferences. First step in testing was to manually annotate sentences in the reviews. They trained and validated their classifier for star rating using the topic and sentiment annotation with 7-fold cross validation. Their evaluation metrics was MSE values.



Predicting Star Ratings	TEST I	TEST II
Star rating	1.217	1.295
Sentiment-based text rating	1.098	1.27
Predicting Text Sentiment Ratings	**TEST I**	**TEST II**
Star rating	1.430	1.342
Sentiment-based text rating	1.277	1.374

Table 2: Prediction MSE using the restaurant average for prediction.

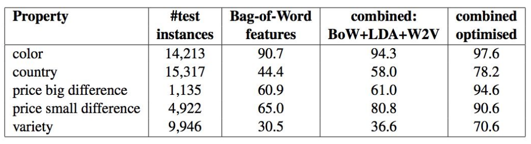
As seen in top portion of Table 2, it was found that the sentiment rating always provides better predicting accuracy of user preferences, as exhibited by lower MSE values, than the star rating. For the bottom portion of Table 2, researchers tried predicting text sentiment ratings with their model, but the performance was lower. Since the model shows good performance for ratings prediction, which is the goal of our project, we can benchmark the process this paper used, including identifying topics and sentiments of the reviews and finding correlations between sentiments and ratings.

Back to the present day, in 2016, Iris Hendrickx and Els Lefever published their research on applying NLP to wine reviews. Their research aims to automatically predict properties of wines based on reviews written by wine experts that describe the smell and flavor of the wine. The paper tries to show that wine experts have a fairly consistent language of description and that this can effectively be used to predict different properties of the wine such as its color, grape variety, country of origin, and price.

The dataset was downloaded from the website http://www.winemag.com/, owned by Wine Enthusiast Companies. The researchers gathered a total of 76,585 wine reviews which contains information about the wine such as the producer, appellation region and country, grape variety, color, alcohol percentage, price, and where to buy it. The experts who reviewed the wine rated it on a range from 80 to 100. The reviews are written by 33 different experts and have an average length of 39 words.

The dataset was preprocessed with the Stanford toolkit (Manning et al., 2014): they tokenized, PoS-tagged and lemmatized the reviews. They used both lexical as well as semantic features. Initially they used a bag-of-words (BoW) representation of the wine reviews. They lower-cased all lemmas in the review and selected only the content words (PoS tag noun, verb or adjective) that occurred at least twice in the training set. The BoW features were then combines in 2 ways - 1) 100 topics generated with Latent Dirichlet Allocation 2) 100 clusters based on word embeddings generated with Word2Vec.

The classifiers used was a LIBSVM (Chang and Lin, 2011), with the RBF kernel and optimized parameters *c* and *g* per prediction task. The parameters for SVM were optimized by means of a Grid search on a randomized subset (5,000 instances) of the training data. Table 1 presents the classification results per wine property for three system flavors: (1) feature vectors including BoW, (2) feature vectors combining BoW features, LDA and Word2Vec clusters, and (3) combined feature vectors trained with an SVM classifier with optimized hyperparameters *c* and *g*.

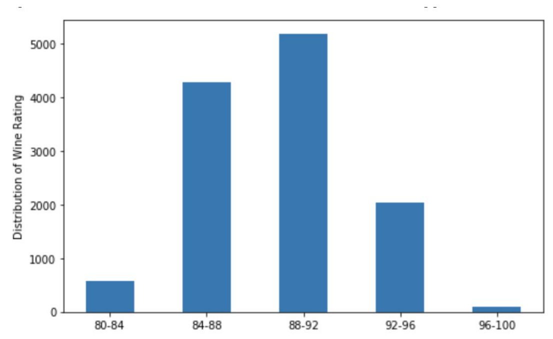


Property	#test instances	Bag-of-Word features	combined: BoW+LDA+W2V	combined optimised
color	14,213	90.7	94.3	97.6
country	15,317	44.4	58.0	78.2
price big difference	1,135	60.9	61.0	94.6
price small difference	4,922	65.0	80.8	90.6
variety	9,946	30.5	36.6	70.6

Table 1: F-scores per task for Bag-of-Word features, a combination of BoW, LDA and Word2Vec clusters, and combined & optimised LIBSVM parameters.

Experimental Design

Data: The data set in kaggle.com/zynicide/wine-reviews is split into training data, validation data, and test data, with each 80% (103972 reviews), 10% (12997 reviews), and 10% portion (12997 reviews). Each data set is a csv file that consists of the description column, rating column, and title column. Then, we split the reviews into 5 classes depending on their ratings: 80-84, 84-88, 88-92, 92-96, and 96-100. The distribution of the labels are as the below figure. As you can see, there is a great amount of 84-88 and 88-92 labels. The number of labels with 80-84 and 96-100 is very small compared to the other labels.



Number of reviews in each label

Example of data set(delimiter: '\t'):

Description Rating Title

“There's a touch of pungent apple skin at first, but the bouquet opens up over time with pretty floral tones and shadings of forest fruit and dried raspberry. For an Amarone, this wine is surprisingly bright and crisp on the close.” 87 “Giuseppe Lonardi 2007 Amarone della Valpolicella Classico”

Data Preprocessing: We preprocessed the data sets by removing stop words and punctuations and lemmatizing. Also, to perform multiclass classification, we labeled each review from 1 to 5. 1 stands for the 80-84 rating range, 2 stands for 84-88, and so on.

Evaluation Metric:
We evaluated results through accuracy and root mean squared error (RMSE).

$$\text{Accuracy} = \frac{\text{number of correct labels}}{\text{total number of data}}$$

$$\text{Root mean squared error} = \sqrt{\frac{\Sigma(predicted_i - actual_i)^2}{n}}$$

While $predicted_i$ stands for the predicted label of the ith data, and $actual_i$ stands for the actual label. Since our classes are divided as 0, 1, 2, 3, and 4, the difference was an integer value.

While the accuracy value simply counts the number of correct labels, root mean squared error would show us how far the our wrong prediction was from the correct labels as well.

Simple Baseline: Labeling all the reviews as the most frequent label (88-92) gave us a accuracy of 42.8% and a root mean squared error of 0.847.

Experimental Results

Published Baseline: The published baseline uses the following implementation. First, it creates an unigram, a simple word count dictionary, from the entire reviews. Based on the unigram word count dictionary, a dense count vector is made. The generated vector is used to build a naive bayes multi-classifier. The accuracy of the published baseline is 62.4% and root mean squared error was 0.618.

Extensions:

1) Naive Bayes with word-count and tf-idf

As the first extension, Naive Bayes is selected. As a generative model, it requires significantly large memory space and running time, so it is inevitable that a smaller version of the entire dataset is used for this extension.

Before training the model, common words associated with each label are counted. The result is the following

	Rating 1 (total: 107305)	Rating 2 (total:10 14957)	Rating 3 (total: 1498175)	Rating 4 (total: 668547)	Rating 5 (total: 32015)
Rank 1	('flavor', 3914)	('wine', 25961)	('wine', 36640)	('wine', 16943)	('wine', 934)
Rank 2	('wine', 2622)	('flavor', 25651)	('fruit', 29087)	('fruit', 11680)	('fruit', 515)
Rank 3	('fruit', 2057)	('fruit', 20146)	('flavor', 28820)	('flavor', 10063)	('flavor', 395)
Rank 4	('finish', 2031)	('aroma', 15037)	('aroma', 18430)	('black', 7407)	('tannin', 360)
Rank 5	('aroma', 2011)	('finish', 13949)	('palate', 17857)	('tannin', 6995)	('drink', 346)
Rank 6	('palate', 1534)	('acidity', 12360)	('finish', 17002)	('drink', 6897)	('black', 308)
Rank 7	('sweet', 1121)	('palate', 12192)	('tannin', 15834)	('palate', 6826)	('acidity', 278)
Rank 8	('cherry', 1084)	('cherry', 10527)	('acidity', 15565)	('cherry', 6416)	('year', 248)
Rank 9	('acidity', 865)	('drink', 9647)	('cherry', 15313)	('acidity', 5935)	('cherry', 240)
Rank 10	('note', 853)	('tannin', 9196)	('black', 14454)	('ripe', 5768)	('rich', 224)

Many words are overlapped, but there are a couple of unique words that represent their ratings. For instance, the word ‘sweet’ only appeared under rating 1, indicating that sweet wines tend to have a lower rating. The word ‘year’ and ‘rich’ only appeared under rating 5. This may be because the year the wine was made is a crucial factor to its taste.

To generate a multiclass Naive Bayes classifier, word-count vector and tf-idf are used. For both word-count and tf-idf, n-gram variations are applied; unigram, bigram, and trigram are tested. For the evaluation, accuracy is calculated, and the overall results are recorded on the following tables.

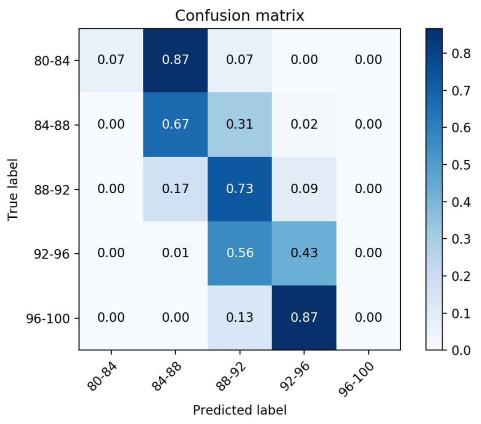
Word-count

Model	Unigram	Bigram	Trigram
Accuracy	**62.4%**	59.8%	60.5%

Tf-Idf

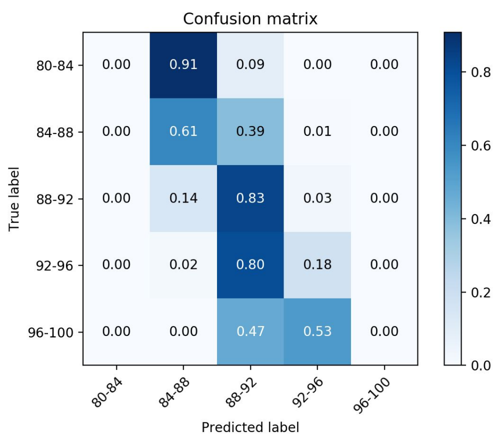
Model	Unigram	Bigram	Trigram
Accuracy	55.2%	50.1%	48.3%

According to the results, in the two cases, the unigram produces the highest accuracy. This outcome is particularly unexpected. The result implies that individual words in the dataset carry more significance than sequences of words. In other words, it means that a sequence of words do not reveal any significant correlation to the classes. One possible explanation is that the data are preprocessed. Because the documents are cleaned by excluding stopwords, lemmatizing, and tokenizing, the n-gram sequences may have lost their significance from the original text. The followings are the confusion matrices of word-count Naive Bayes results.



True \ Predicted	80-84	84-88	88-92	92-96	96-100
80-84	0.07	0.87	0.07	0.00	0.00
84-88	0.00	0.67	0.31	0.02	0.00
88-92	0.00	0.17	0.73	0.09	0.00
92-96	0.00	0.01	0.56	0.43	0.00
96-100	0.00	0.00	0.13	0.87	0.00

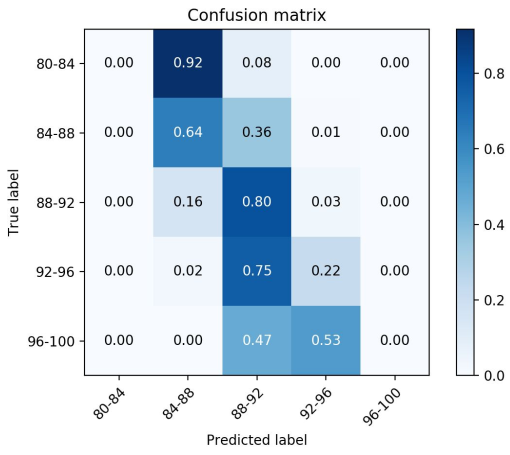
Word-count Unigram Confusion Matrix



True \ Predicted	80-84	84-88	88-92	92-96	96-100
80-84	0.00	0.91	0.09	0.00	0.00
84-88	0.00	0.61	0.39	0.01	0.00
88-92	0.00	0.14	0.83	0.03	0.00
92-96	0.00	0.02	0.80	0.18	0.00
96-100	0.00	0.00	0.47	0.53	0.00

Word-count Bigram Confusion Matrix

One thing to note from the confusion matrices is that the model performances are particularly low for the labe 80-84 and 96-100. The error is because of the un-uniformed distribution of the dataset. The following is the number of reviews per label: 80-84: 599, 84-88: 4610, 88-92: 5568, 92-96: 2131, 96-100: 89. The number of documents for 80-84 and 96-100 are significantly small compared to the other labels, especially 96-100. Therefore, the performance reveals the limitation from the dataset.

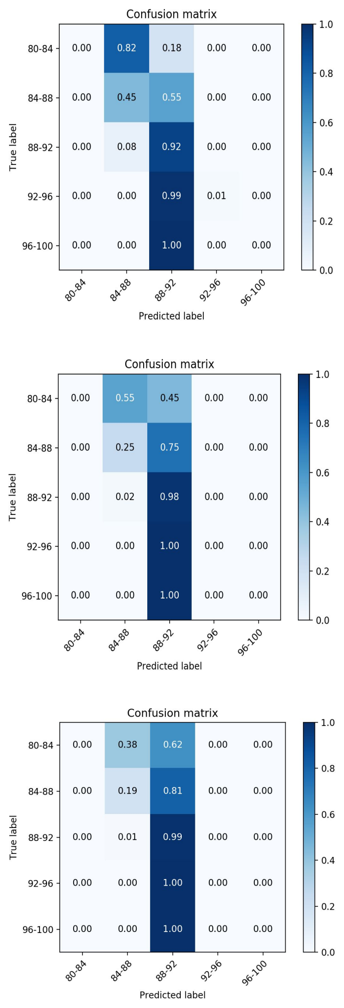


True \ Predicted	80-84	84-88	88-92	92-96	96-100
80-84	0.00	0.92	0.08	0.00	0.00
84-88	0.00	0.64	0.36	0.01	0.00
88-92	0.00	0.16	0.80	0.03	0.00
92-96	0.00	0.02	0.75	0.22	0.00
96-100	0.00	0.00	0.47	0.53	0.00

Word-count Trigram Confusion Matrix

In addition, the tables show that Tf-Idf, a popular text representation, performs worse than word-count. Tf-Idf is a text representation is often used for balancing common words. However, in this experiment, its low performance can be possibly more related to term frequency. Tf-idf also takes term frequency per document. However, our dataset is not evenly distributed. In fact, it is heavily skewed to the 88-92 rating class as mentioned above. This unbalance may have caused the Naive Bayes classifier predictions to be focused on 88-92 rating class. As the following confusion matrices represent, most of the tf-idf Naive Bayes predictions are on the 88-92 label.

Another interpretation is that the models are overfitted. Also, Tf-idf is designed to disregard common but insignificant words like stopwords. However, stopwords are excluded from the dataset during the preprocessing, which could have made the purpose of Tf-idf meaningless.



Tf-Idf Confusion Matrix (Unigram, Bigram, Trigram)

In order to verify some of the interpretations above, the same set of experiment is done including stopwords. The following tables summarize the results.

Word-count including stopwords

Model	Unigram	Bigram	Trigram
Accuracy	62.1%	60.0%	59.1%

Tf-Idf including stopwords

Model	Unigram	Bigram	Trigram
Accuracy	54.4%	49.8%	47.7%

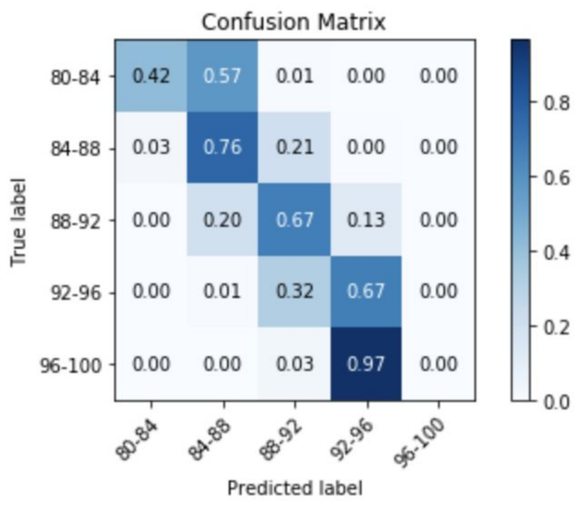
According to the accuracies, including stopwords do not increase the performance.

Consequently, the simple word-count collected as the bag of words model (unigram) performed better than tf-idf with 62.4% as its best accuracy with a root mean squared error of 0.618.

2) RNN using AllenNLP

We used a recurrent neural network library called AllenNLP. The model uses a bidirectional LSTM with 2 hidden layers. The advantage of this approach is that the word order is preserved, and the parameters are automatically chosen for the best performance of the validation data. We used the pre-processed data (without stop words or punctuations, and lemmatized). To enable the library to correctly read in our data, we created a data reader/parser. For each review, we created word-based unigram with the labels. This is the input that goes into the RNN. We also created a predictor code so the model would generate labels for each review in the test set. Then, we found the accuracy and generated the confusion matrix.

We got an accuracy of 68.8%, root mean squared error of 0.587. The confusion matrix was as the follows:



Although the accuracy value itself wasn't very high, the confusion matrix shows that for most of input, we are outputting either the correct label or the label right beneath or above it. Also, the accuracy is higher than our baseline and Naive bayes model, which indicates RNN is a better model. However, there weren't any prediction of the 96-100 label. We believe this is because we don't have enough 96-100 label reviews in our training data. However, given that most of the 96-100 labels are predicted as 92-96, we may conclude that the prediction was successful.

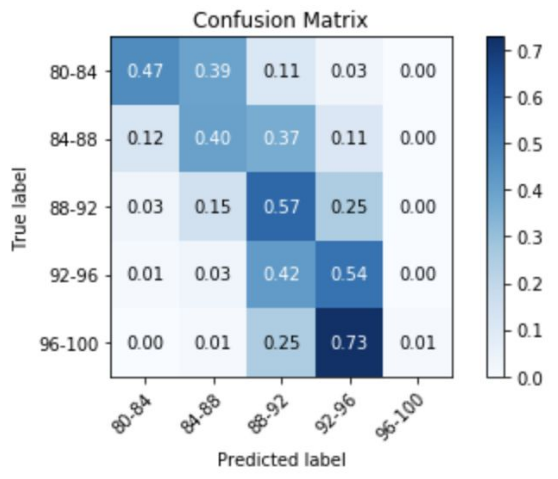
3) Word Vectors

Now, we would be considering a review as a sequence of vectors. First, we use pre-trained word embeddings to represent each word as a vector of its own, and we chose fastText for embedding technique. After embedding, we decided to experiment in different ways to represent a review using a vector.

a) Concatenation of word vectors

First, a review can be seen as a sequence of vectors of words appearing in the review. For instance, is the review says wine taste wonderful, we concatenated the vector of each of the words in order. The max length was 78, and if an review had a smaller length than that, we padded zeros. For this approach, we have appended word vectors for each review to train the classifier. We used sklearn SGDClassifier with power_t of 0.01 and eta of 0.01.

We got an accuracy of 49.6% and mean squared error of 0.848. The confusion matrix was as the follows:



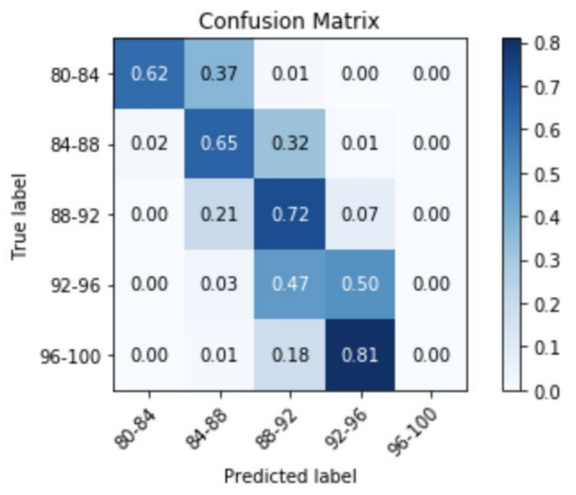
The accuracy was lower than the baseline, and the root mean squared error was worse as well. As seen in the confusion matrix, there were several labels that were off (predicted 92-96 when it was 80-84, or predicted 84-88 when it was 96-100). .

b) Average word vector

Second, we could consider a review as an average of all word vectors. We averaged all the vectors of the words in each review to represent that review. We tested SGD Classifier, Random Forest, and Logistic Regression, and the results were as follows.

Classifier	SGD Classifier	Random Forest	Logistic Regression
Accuracy	58.1%	56.4%	**61.5%**

The root mean squared error for our best model (logistic regression) was 0.606. The following figure is the confusion matrix.



	80-84	84-88	88-92	92-96	96-100
80-84	0.62	0.37	0.01	0.00	0.00
84-88	0.02	0.65	0.32	0.01	0.00
88-92	0.00	0.21	0.72	0.07	0.00
92-96	0.00	0.03	0.47	0.50	0.00
96-100	0.00	0.01	0.18	0.81	0.00

Compared to vector concatenation, averaging vectors showed a higher accuracy and a lower (better) root mean squared error. We can see that the confusion matrix has a clearer diagonal, with labels either predicted correctly, or as the one above or below. Thus, we decided to extend this model and try some experiments.

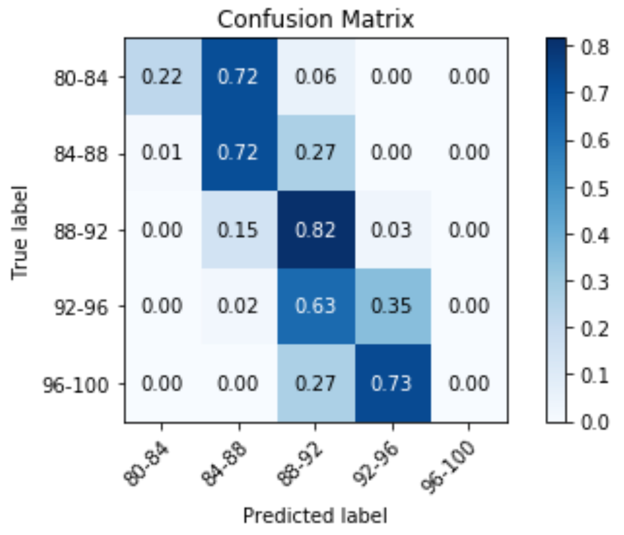
So, we tried adding several features to our model. For training the model we used the following parameters, which gave us the best result after optimizing on the dev set -

1. Num_features (Dimensionality of the resulting word vectors) = 200
2. min_word_count (Minimum word count threshold) = 3
3. Context_size (Context window length) = 8

Once we had the features for the reviews, we fitted our training dataset using Random Forest, Decision Tree, K-Mean Clustering and Linear SVM, with Random Forest giving the best accuracy.

Classifier	Random Forest	Decision Tree	K-mean Clustering	Linear SVM
Accuracy	**67.4%**	54.4%	59.3%	54.8%

As is show by these results Random Forest far outstrippted the other classification models. The root mean squared error of this model was 0.600. Using the Random Forest model we generated a confusion matrix:



	80-84	84-88	88-92	92-96	96-100
80-84	0.22	0.72	0.06	0.00	0.00
84-88	0.01	0.72	0.27	0.00	0.00
88-92	0.00	0.15	0.82	0.03	0.00
92-96	0.00	0.02	0.63	0.35	0.00
96-100	0.00	0.00	0.27	0.73	0.00

As with previous extensions when incorrect the predicted label didn't stray to far from the true label. However, also similar to other models, with the small number of 96-100 score reviews it failed to predict any as this label. The accuracy compared to other models is high but as is apparent by the confusion matrices this is mostly due to a higher likelihood of predicting the middle, which while raising the total accuracy decreases the percent of true labels for the classes that are not 88-92.

c) Inverse frequency weighted average

In this extension, we took an inverse frequency weighted average based on the hypothesis that words that appeared more frequently in the wine reviews had less classification importance. This approach would also have made sense in the case where there are stop words in the reviews. However, since we removed these prior to training the effect of the weighted averaging was less prominent.

We got an accuracy of **61.3%** with a Random Forest classifier and a root mean squared error of 0.647. Since this accuracy was less than in the case where we took a standard average, we went with that classifier.

Results

The accuracy and root mean squared error for the best models in each of our extensions is as follows. (The best model was chosen based on accuracy.)

Extension	Naive Bayes (word count & tf-idf)	Recurrent neural network	Word vectors
accuracy	67.4%	68.8%	67.4%
RMSE	0.618	0.587	0.600

Since recurrent neural network model had the highest accuracy and lowest root mean squared error, we could conclude that it was our best model. Then, it was word vector, and Naive bayes.

Error Analysis

There weren't any strong patterns between what the published baseline performed better and RNN performed better. However, for a few reviews that had long reviews, such as "plump black currant lavender extract bright purple flower touch raw beef tar show nose bottling made 20 whole cluster 7 viognier black pepper asphalt show first palate tannin softly coat mouth stand boost elderberry flavor thorough zesty acidity ensures long cellar potential drink 2017" (one of the preprocessed data), RNN did a better job predicting it was a higher label (92-96) while the published baseline predicted it as 84-88.

Also, for reviews that contained 'rich' or 'year', the published baseline predicted it as 92-96 (the closest label to the correct label 96-100). However, the RNN predicted some as 88-92 or 84-88, as well.

Conclusion

This project focuses on building a multi class classifier model that predicts a wine score based on wine descriptions. Naive Bayes, RNN, and Word Vector are implemented to build the models, and RNN had the best results. In this process, we encountered some challenges. First, the dataset from Kaggle is not uniformly distributed. There are too many positive feedback, so the models cannot be trained properly to predict a wine score. The score range is divided into the five categories, and for the last range, from 96-100, there are only a handful amount of samples. In addition, sentiment analysis is hard since all review are descriptional than emotional. Also, some wine favors were described using metaphorical references. These reviews, made by some unique reviewers, made it hard to find patterns in reviews for each labels.

Further improvements we could be possible if we had more negative feedbacks in our data set. Also, word the word vectors, if we could create a program that extract features such as acidity, taste, savor, or aroma using the description, and create a fixed length word vector with specific features on specific locations, we believe we could significantly enhance the performance.

Interesting Feature Analysis

Predicting wine ratings using different features:

We tried predicting wine ratings using review length, wine price, province, and variety.

1) Predicting wine ratings through wine review length:
accuracy: 51.5%, root mean squared error: 0.587, correlation coefficient 0.510
The data can be interpreted in the following way: the better the wine is, the more the wine reviewer has to and is willing to talk about that wine! While for mediocre wine, there's not much to talk about.

2) Predicting wine ratings through wine price:
accuracy 50.6%, root mean squared error: 0.596, correlation coefficient 0.462
The small mean squared error and the high correlation coefficient shows that there tends to be a correlation between wine ratings and wine price. Moreover, if we look at the average price of wine for each of the classes, they are each $18.3, $22.0,

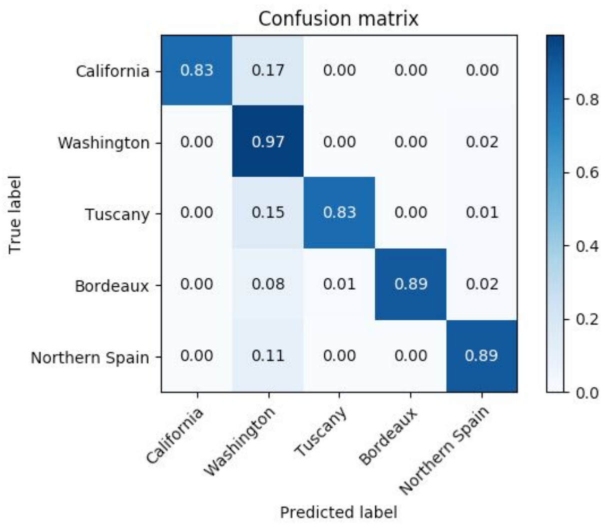
$34.3, $66.1 and $168.6. The 96-100 labelled wines had a significantly higher price. This is because it contained multiple premiere wines such as Bordeaux-style red/white blend or Chardonnay that were $800-2000.

3) Predicting wine ratings through province of wine: accuracy: 38.6%, root mean squared error: 1.121
We could conclude that province wasn't a great indicator of the ratings of wine. However, there were some provinces that were always labelled as a single label. For example, wine made of grapes from China, Jiri Valley, Atlantina, and Vale Trentino always had the rating between 80-84. Wines from Slovenia and Central Greece always had a rating of 84-88. There were also regions that produced highly rated wines such as England and Santa Cruz as well. The province that produced the best quality wine was Portugal. More than 10% of the wine produced from Portugal had the highest ratings of 96-100.

4) Predicting wine ratings through variety of wine: accuracy: 38.0%, root mean squared error: 1.179
Variety shows the type of wine. This as well wasn't a good indicator of rating. However, we were able to do some interesting analysis. While wines such as Airen or Picapollo always had the lowest ratings, Cabernet-Shiraz, an Australian red blend, always had a rating between 96-100. Muscadel and Tokay also had the highest ratings with more than a 20% chance. There were also some white wines, Moscato di Noto and Caprettone, that were always rated between 92-96.

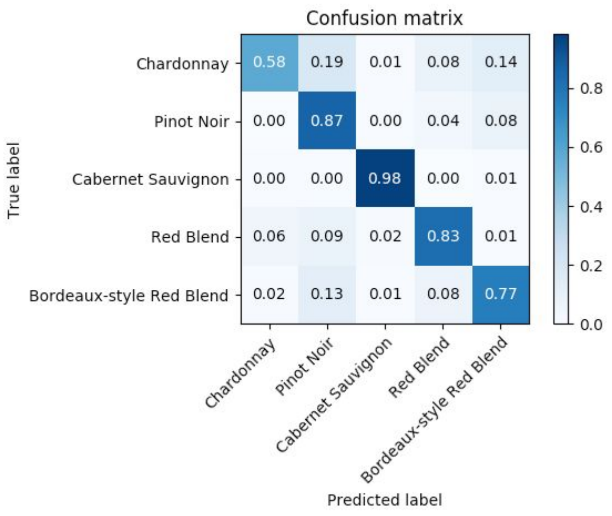
Predicting different features using wine review
We tried predicting wine review using other features including wine province and variety.

1) Predicting wine province through wine review: accuracy: 93.1%
Although predicting wine rating via its province was a difficult task, as shown in subsection (3) above with accuracy of 38.6%, we found that predicting wine variety through wine review was much more accurate. To do this task, we have configured top five provinces that produce wines, namely California, Washington, Tuscany, Bordeaux, and Northern Spain. After extracting text reviews that correspond to wines from these five provinces, we have trained a Naive Bayes N-gram model (n = 3) to classify province based on review. The accuracy turned out to be very high. One reason is that some reviews contained the province name explicitly. Another, rather subtle, reason is that each province would have signatures, and reviews for wines from the same province are more likely to contain same words. For instance, "cherry" appeared frequently in reviews for Californian wines.



2) Predicting wine variety through wine review: accuracy: 82.9%
Similar to province feature, predicting wine rating via its variety was a difficult task, as shown in subsection (4) above with accuracy of 38.0%, we found that predicting wine variety through wine review was much more accurate. To do this task, we have configured top five varieties of wines, namely Chardonnay, Pinot Noir, Cabernet Sauvignon, Red Blend, and Bordeaux-style Red Blend. After extracting text reviews that correspond to wines from these five varieties, we have trained a Naive Bayes N-gram model (n = 3) to classify variety based on review. The accuracy turned out to be high as well. First reason is that some reviews contained the variety explicitly. However, compared to province, it was less likely for reviews to contain

variety. Nevertheless, the accuracy could be high as each variety have different taste and reviewers would enjoy them with different food. For instance, many reviewers described Chardonnay wines as “nutty”, and not many wines of other varieties were not described as nutty.



Appendices

Our code to run baseline and each extension can be found in our github repository, and slides in here.

Bibliography

(1) Reddy, Ch Sarath Chandra, et al. "Prediction of star ratings from online reviews." *TENCON 2017-2017 IEEE Region 10 Conference*. IEEE, 2017.

(2) Ganu, Gayatree, Noemie Elhadad, and Amélie Marian. "Beyond the stars: improving rating predictions using review text content." *WebDB*. Vol. 9. 2009.

(3) Iris Hendrickx, Els Lefever. “Very quaffable and great fun: Applying NLP to wine reviews.” *Association for Computational Linguistics.* Vol. 2. 2016.

(4) Chai, T. and Draxler, R. R.: Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature, Geosci. Model Dev., 7, 1247-1250, https://doi.org/10.5194/gmd-7-1247-2014, 2014.