Automatic Scoring: Let Support Vector Machines Taste Wine

Max Moons, *Radboud University, IEEE*
& Tom Janssen Groesbeek, *Radboud University, IEEE*

***Abstract*—Wine experts are able to better describe smells and flavors than novices as and their descriptions are more consistent with other wine experts. We have conducted several experiments in order to automatically predict the score of the wine based on a review written by a wine expert. Several known Natural Language Processing methods are applied to turn wine reviews into feature vectors, which are in turn fed to a Support Vector Machine. The SVM functions as a classifier used to predict the score of a wine review. Additionally, we compared the performance of the SVM while using both predefined word embeddings and custom word embeddings. The experimental results suggest that a setup that combined a bag-of-words model with a Latent Dirichlet Allocation and a Word2Vec model made the classifier perform worse in predicting the wine ratings compared to when only the bag-of-words model is used.**

I. Introduction

Recent study shows that descriptions of smells and flavors by experts differ from those by novices. Wine experts tend to provide more source-based terms and the agreement on smell and flavor is more consistent for wine experts than it is between novices [2]. One of the reasons suggested by the study was that wine experts are trained to use language in a specific way as they engage in more 'wine talk'. This would suggest that in order to be able to better describe smells or flavors, one also has to train to verbalize the actual experiences.

Verbalizing and sharing experiencesare facilitated greatly by the rise of social media. People can now seek their information on, among other things, domain specific blogs our pages dedicated to sharing product reviews [1]. Next to textual messages varying from very short-sighted and uninformative to descriptive and near poetry like reports, many review pages enable satisfaction scoring, which allows them to sort based on the accompanying score or simply provide a short and visual summary of the message.

With the knowledge that expert descriptions differ linguistically from novices, we are interested in what connections there exist between the specific language usage and provided satisfaction score. We will be performing several Natural Language Processing (NLP) methods on a large wine expert review dataset in order to investigate the predictive power of certain language usage on the final satisfaction score. The main goal of this paper is to train a classifier to predict the score of a wine.

Training automated systems to predict wine properties could be useful in the long run for similar work on automatic metadata prediction. To be able to provide satisfaction scores to texts in a consistent and accurate manner, might benefit the development of recommender systems based on metadata and user-based filtering [5]. This motivated us to study the predictive power of expert level reviews.

The remainder of this paper is structured as follows. First we will discuss related work to be able to better formulate our research question, then we will explain the data used, our methods and report our results. The last section of this paper includes a discussion on the results, our own conclusions and some ideas for future work.

Max Moons, s4256417.
email: Max_Moons@hotmail.com
Tom Janssen Groesbeek, s4229738.
email: tomjg@hotmail.nl

II. Background

Our experiment is very similar to the work of Hendrickx et al.[5]; their dataset also originates from Winemag.com and they used the same models to process the reviews. One important difference is that we aim to predict the score, while Hendrickx et al. predicted the price (2 categorical divisions), country and the color of the wine. Aside from that, we implemented the same NLP methods.

One these NLP methods is the Bag-of-Words (BoW) model. This is a representation method for object categorization, which is also known as the Vector Space Model for Automatic Indexing [9]. Each document gets represented as a vector with the amount of identified relevant terms as length. Each of the terms gets assigned either a 0 or a 1, depending on whether the term is present in the document. The more similar two analysed documents are, the closer they will be in the vector space that contains all the analysed documents. One benefit of this model is that it is not computationally difficult to implement, but a disadvantage is that any information about word order is lost.

Aside from the BoW representation, we took care to train our own word vectors by making use of the Word2Vec tool. This tool was initially developed by Tomas Mikolov et al. at Google [7] and it provides state-of-the-art word embeddings. The W2V implementation first constructs a vocabulary from the training data and learns a vector representation of words. The Word2Vec tool can be used to capture strong regularities in the vector space, provided the training set is sufficiently large and the right parameters have been chosen. The Word2Vec could use either the Skip-gram (SG) or the Continuous Bag-of-Words (CBOW) architecture. The CBOW architecture is generally faster, but the SG architecture performs better at infrequent words. The vector size of the output vectors

can also be chosen, in general a bigger dimension is better, but this is not always the case. So the best dimension should be determined by testing different configurations on the data set, although this is very time demanding. The context window size determines the amount of neighbouring words of the target word that are processed, where a value around 5 is usually chosen for the CBOW architecture, while a value of around 10 is usually chosen for the SG architecture. The training algorithm that is used is either the hierarchical softmax (HS) algorithm or the negative sampling (NS) algorithm; HS performs better for infrequent words, while NS performs better for frequent words and with low-dimensional vectors.

Another NLP model used in this project is Latent Dirichlet Allocation (LDA), which is a generative probabilistic model for collections of discrete data like text corpora [8]. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modelled as an infinite mixture over an underlying set of topic probabilities. Each topic is characterized by a distribution over words. The relevant parameters for LDA include the number of topics, the minimum coverage and the amount of passes over the data set. The optimal parameters depend on the data set that has been used. The minimum coverage determines the minimum threshold to assign a topic to a text and the amount of passes determines how often the entire dataset should be processed.

Support Vector Machines (SVMs) are a technique used for data classification [11]. There are different kernels that can be used with SVMs and in general the RBF kernel is a reasonable first choice (see figure for a visual represenatation from Scikit[1]). This kernel maps samples into a higher dimensional space so the SVM can cope with a non-linear relation between class labels and attributes. The RBF kernel has relatively few hyperparameters (C and γ), while for example to polynomial kernel has more hyperparameters. The best values for C and γ are different for every application, so they should be determined for every problem. One way to do this is a grid search, in which different values for parameters are tried and evaluated. After a few iterations the values that yield the best accuracy are chosen.

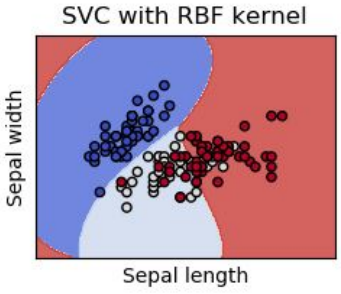

Fig. 1
A plot from the iris dataset

[1] http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html

In a related study dedicated to automated text evaluation, Long Short-Term Memory networks were used to represent the meaning of texts [3] rather than SVMs. Alikaniotis et al. introduced a model that learns the extent to which specific words contribute to the texts score and used this model to automatically rate essays written by Grade 7 to Grade 10 students, all essays consist of 150-550 words each. Given the excellent results their best network achieved and the comparable goal of the model, we initially considered of using an LSTM-based network to rate the wine reviews, but eventually we decided to reproduce the experiment of Hendrickx et al. method-wise.

III. Project

Aside from simply trying to develop an automatic review scoring system, we will also attempt to answer a few related research questions. Our first research question: **is it possible to train a SVM to be able to predict wine scores by analyzing wine expert reviews accurately?** Given the results of Hendrickx et al.[5] we hypothesize that SVMs can achieve an F1-score of at least 0.70.

Next we would also like to investigate what influence custom word embeddings have on the performance of a classifier in comparison to predefined word embeddings like GloVe. Therefore, our second research question: **will custom word embeddings trained on domain specific data outperform predefined word embeddings on a classifying task?** Given the very specific context of this domain we hypothesize that setups that employ a custom Word2Vec model will outperform setups that use a pre-trained GloVe model

The next couple of sections will go into more detail on how we will try to answer these research questions. First we will discuss what data we worked with and the various preprocessing steps we took to clean and prepare it. Next we will discuss some of the features we extracted from the data and with what NLP methods we acquired those features. At the end of this section we will discuss the way we defined our SVM and how we tested its performance.

A. The Data

The dataset we used originates from winemag.com[2], which is a website for wine enthusiasts. A total of 129,971 wine reviews are in the dataset and every entry contains the following data: country, review, designation, score, price, province, region, wine taster, wine taster Twitter ID, wine name, grape variety and the winery. Every wine has been tasted blindly by professionals. The only actual data we will use from the entries are the score (to verify the classification) and the review. The following paragraph is one of the 129,971 reviews from the dataset:

"Aromas include tropical fruit, broom, brimstone and dried herb. The palate isnt overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity".

[2] https://www.kaggle.com/zynicide/wine-reviews/data

B. Preprocessing

Our project focusses on predicting a wine's score and our setup closely follows the experiments of Hendrickx et al. Therefore we tried to perform similar preprocessing steps on our data as in their work. This means that we had to consider the part-of-speech tags for every word in the dataset. Before we started pos-tagging the reviews, we decided to tokenize every text. This way we could get rid of tokens with a length of 1 and tokens containing digits. Then we started with the pos-tagging; we tried to do this as soon soon as possible so the grammatical structure would be preserved for the pos-tagging. After pos-tagging, we removed stop-words and made sure to make all tokens lowercase.

The next step was to filter the reviews on content words by making use of their pos-tag. According to Hendrickx et al., content words are nouns, verbs and adjectives. We used the Standford NLTK, which had the following content word labels: JJ (adjective, ordinal), JJR (adjective, comparative), JJS (adjective, superlative), NN (noun, common), NNP (noun, proper), NNS (noun, common, plural), VB (verb), VBD (verb, past tense), VBG (verb, present participle), VBN (verb, past participle), VBP (verb, present tense), VBZ (verb, present tense, 3rd person singular). We determined for every token in the review whether its pos-tag was one of these specific tags; if not it would be discarded. We also filtered out the content words with only 1 or 2 occurrences in the entire dataset. We decided on these numbers as this reduced the final content word count, which meant less memory expensive feature vectors and faster training time.

After filtering on content words we had to make sure to turn the actual scores into one of six labels. The labels we worked with are listed in in figure 2; this categorization originates from Winemag.com. There are actually 7 labels, but winemag.com does not publish reviews of wines that score below 80.

points	
Classic	98-100
Superb	94-97
Excellent	90-93
Very Good	87-89
Good	83-86
Acceptable	80-82
Unacceptable	below 80

Fig. 2
The winemag score labels

The last preprocessing step we took before splitting the data into training and test set, was to compute the Bag-of-Words corpus of the entire dataset. We intentionally did this on the entire dataset to prevent the risk of missing out on certain content words that might otherwise only occur in the test set. This way we ensure that BoW feature vectors in both the test and training set have the same dimensionality.

The dataset has all its entries with a grape variety that has been reviewed less than 200 times [5] removed and it has been split into a train set (80%) and a test set (20%). The training and test set both have the same ratio of grape varieties and the same ratio of scores (figure 3). The exact numbers with regard to preprocessing are listed in table I.

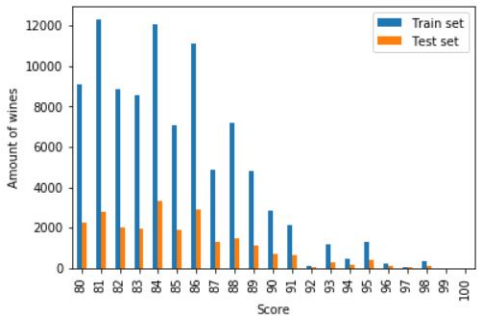

Fig. 3
A histogram of the scores of the train and test set

TABLE I
The dataset before and after filtering

Total reviews	129,971
Varieties >=200 reviews	118,623
Total varieties	708
Filtered varieties	62
Length train set	94,614
Length test set	23,649
Content Words	42,726
Content Words (>2 occurrences)	18,026
Shortest review (content words)	1
Longest review (content words)	73
Average amount of content words	24

C. Design

Our first two experimental setups are similar to those of Hendrickx et al.; in one experment we used a BoW representation of the wine reviews as only input for the SVM to predict the labels and in the second one we extend these BoW representations by making use of Latent Dirichlet Allocation (LDA) and Word2Vec [5]. As extension of this work we decided to include two more experiments. One in which we only extend the BoW representation with LDA and one in which we extend the BoW representation again with LDA but this time instead of defining our own word embeddings with Word2Vec, we make use of the predefined word embeddings by GloVe. GloVe is an unsupervised learning algorithm that can be used to obtain vector representations for words[3].

[3] `https://nlp.stanford.edu/projects/glove/`

C.1 Extracting Features

In order to create BoW feature vectors we make use of the content words. We filtered the entire dataset on content words that had more than 2 occurrences and used the unique content word count as number of dimensions for the feature vector. After this we took the content word counts of each review as input for the feature vector. This means that the number of mentions of a content word in the review is the value that that is set to the corresponding index of that content word.

To compute the feature vectors based on the LDA topic distribution, we trained an LDA on the entire corpus with number of topics set to 100 and setting the threshold to assign a topic to a text to 1%. This means that when a topic covers at least 1% of a text it gets assigned to it. The feature vector based on LDA is the topic probability distribution of a text; the probability for every topic that has been identified is an element of the vector. We combine the LDA feature with the BoW features to create one big feature vector.

Next, we used the Gensim Word2Vec implementation [10] to train our own word embeddings based on the entire dataset. We trained them with a context size of 8 and we set the word dimensionality to 200 features. After training we used KMeans clustering with number of topics set to 100 to cluster the word embeddings. These clusters where then used to create binary feature vectors. The vector thus had a dimension of 100 and if a certain word in the text was present in a cluster it got the value '1' in the feature vector and '0' otherwise. Similar work was done for the GloVe feature vectors, but this time we did not have to train our own word embeddings. We used the 200-dimensional GloVe word embeddings to maintain the same dimensionality as our own word embeddings, then we create the same feature vectors again based on KMeans clusters.

We made sure to use the same parameters for LDA and Word2Vec as in the work by Hendrickx et al. [5]. However, in their work not all parameter settings for both LDA and Word2Vec were described. LDA has as additional parameter the number of passes that it should perform over the corpus. We decided to let LDA pass the corpus only once to reduce training time. Similar, Word2Vec has the 'sg' parameter that defines the training algorithm to be used. There are two options, namely skip-gram or CBOW. We decided to go for the skip-gram training algorithm as it is would work better with low frequent words[7]. Since our dataset consists of wine reviews filled with different adjectives to describe the wine, we expected our texts to contain many low frequent words.

C.2 SVM and Grid search

For our project we decided to work with the python library sklearn that provides an implementation of SVM. One of the parameters that needs to be set for the SVM is the kernel function that it should use. The kernel is a similarity function and with sklearn we could pick either a linear, polynomial, rbf or sigmoid kernel; we used the rbf kernel [5].

Additional parameters for the SVM with RBF kernel include the penalty parameter C and the kernel coefficient γ. We computed the optimal settings for these two parameters by performing grid search on a random sample of 5000 subjects of the training set. This resulted in the optimal settings C = 5 and $\gamma = 0.02$.

D. Evaluation

To evaluate the performance of our classifier we used the precision, recall and F1-score metrics. Precision is the measure used to specify how many of the predicted classes are actually correct and recall is the measure to specify how many of the total number of actual labels are predicted correctly. These two measures are combined in the F1-score, where F1 stands for a special case of the F measure. With the F1 measure β is set to 1. If this is not set to 1 it can be used to emphasize either precision or recall score. Thus by setting it to 1, both scores are equally important for the final F score [6]. The F1-score is simply measured by comparing the predicted labels per review to the actual labels.

IV. Results

The scores for recall, precision and the F1-scores are listed in table IV. Figures 5 and 6 respectively show the class confusion matrix and the normalized class confusion matrix of the BoW experiment

TABLE II
The results of the experiments

	Recall	Precision	F1
BoW	0.495	0.591	0.526
BoW+LDA	0.491	0.591	0.523
BoW+LDA+GloVe	0.467	0.587	0.502
BoW+LDA+W2V_NS	0.467	0.580	0.500
BoW+LDA+W2V_HS	0.469	0.577	0.501
BoW (Vanilla)	0.204	0.162	0.161
BoW (2occ)	0.495	0.591	0.526

V. Summary

The results are quite disappointing considering we used the same parameters for the BoW, LDA and W2V (as far as these were reported) as Hendrickx et al. [5] did and we also performed a grid search to determine the optimal C and γ parameters. The lowest F-score Hendrickx et al. got was for the variety prediction task, which was 70.6 with optimised parameters, while our highest F1-score was 0.526 for the experiment with just the BoW model. The difference in F1-scores between the default BoW configuration (BoW Vanilla) and the BoW optimised C and γ parameters, show that the parameters set in the BoW model cause a huge increase in performance. The performance does not seem to increase by including LDA, W2V or Glove, but

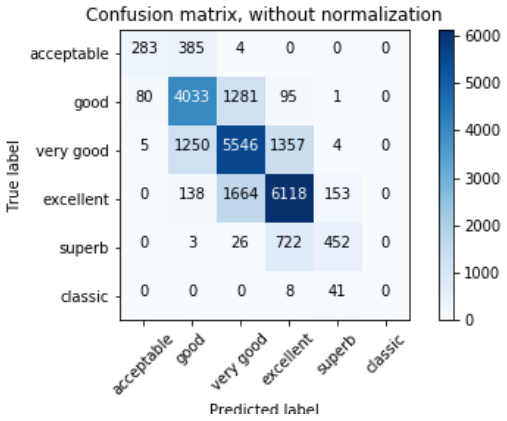

Fig. 4
A confusion matrix of the BoW experiment

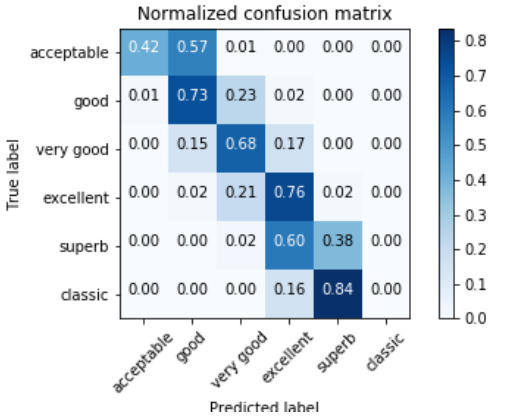

Fig. 5
A normalized confusion matrix of the BoW experiment

the performance of the classifier even drops after including these features, while we expected an increase given the results of Hendrickx et al.

We also performed some additional experiments to check for possible ways of improving our overall performance. One additional experiment was an experiment with just the BoW features, but this time we included content words with just two occurrences. The performance of this BoW experiment (BoW 2occ) obtained identical results to the Bow experiment where we excluded content words with two or less occurrences. Furthermore, we decided to also experiment with a different parameter setting for Word2Vec. The default setting of the gensim Word2Vec implementation made use of negative sampling. In an additional experiment we adjusted this parameter such that hierarchical softmax was used instead of negative sampling. This parameter change resulted in a slightly higher F1-score. This was also in contrary with what we expected, as the hierarchical softmax algorithm was said to perform better with infrequent words [7].

One reason we can think of is that the entire dataset was not that balanced in scores. When looking at figure 3 it can be noticed that there are many reviews with a score between 80 and 90 and not so many over 90. When we then look at figure , we notice that indeed our classifier labels most reviews as either good, very good, or excellent. It also misclassifies other scores as either one of these three classes most of the time.

Aside from the unbalanced dataset, we did have some trouble trying to reproduce the work of Hendrickx et al. Some parts of their work were not that specific on the precise implementation of either BoW, LDA and W2V feature vectors. While the results of our SVM classifier is promising, its performance is not as good as the performance of the classifier of Hendrickx et al..

VI. Conclusions

From our results we can conclude that adding more features to a BoW representation does not improve predictions of scores based on reviews. It would even seem that adding LDA, W2V or GloVe features decrease performance on such a task. The performances are, however, very reasonable with F1-scores of around 0.5 for the main experiments (BoW, LDA, W2V, and GloVe), but not as high as we hypothesized (at least 0.70, based on the results of Hendrickx et al.) . The F1-score lays between 0 and 1 with the best value at 1 where it would have perfect precision and recall. We think the disappointing performance is due to the unbalanced data used for training and testing. This explanation is supported by the normalized class confusion matrix (figure 5), which shows that the classifier performs much better on the overrepresented classes (good, very good and excellent) and much worse on the underrepresented classes (acceptable, superb and classic).

Our second hypothesis is rejected as well; custom Word2Vec word embeddings do not outperform predefined GloVe word embeddings used in this classifying task. When we incorporated the GloVe word embeddings we got an F1-score of 0.502, while we got a very similar (but lower) F1-score of 0.500 when we made use of self-trained Word2Vec word embeddings.

Still it is questionable how reliable these results are. There is a lot of room left for improvement and if we could have done something differently, we would most certainly test with different parameter settings or other ways of constructing the feature vectors. We would also take the time to train with more (content) words, reviews and longer training times. Especially LDA could have been trained with multiple passes over the corpus, which would probably make some difference as 1 pass is likely to be insufficient. Finally, we would most definitely have taken the time to deal with the unbalanced data; probably by including some sort of weighting or by collecting more reviews with higher scores.

VII. Future Work

As we are left with some questionable F1-scores for our experiments, we do believe that there is plenty of future work. First of all, we would encourage others to perform additional similar experiments but with a data set where

every class has a similar amount of entries to see what effect this has on the overal performance.

We would also advice to investigate the effect of including wine reviews with less than 200 reviews. The reason why we removed those entries is because we believed certain words would occur more often in reviews of certain varieties, therefore we assumed that varieties that have not been reviewed often would contain more rare words. This assumption has not been tested, so this could be an idea for future research.

It would also be wise to experiment with more content words as we decided to exclude those with only 2 or less occurrences. We did not test what effect these low frequent content words would have on the performance of W2V and LDA. Furthermore, we believe that there is still some work to be done in comparing the performance difference between using custom word embeddings or predefined ones. Maybe combining these embeddings with a Neural Network would provide even better performances compared to using a SVM.

References

[1] A. M. Kaplan and M. Haelein, Users of the world, unite! The challenges and opportunities of Social Media, *Elsevier*, 2009.

[2] I. Croijmans and A. Majid, "Not All Flavor Expertise is Equal: The Language of Wine and Coffee Experts," *PLoS ONE*, 2016.

[3] D. Alikaniotis and H. Yannakoudakis and M. Rei, Automatic Text Scoring Using Neural Networks, *Proceedings of the 54th Annual meeting of the Association for Computational Linguistics*, 2016.

[4] S. Lev-Ari, How the Size of Our Social Network Influences Our Semantic Skills, *Cognitive Science*, 2015.

[5] I. Hendrickx and E. Lefever and I. Croijmans and A. Majid and A. van den Bosch, Very quaffable and great fun: Applying NLP to wine reviews, *Proceedings of the 54th Annual meeting of the Association for Computational Linguistics*, 2016.

[6] ChengXiang Zhai and Sean Massung, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*, ACM, 2016.

[7] Word2Vec: Tool for computing continuous distributed representations of words, *Google Code Archive*, `https://code.google.com/archive/p/word2vec`, posted July 30, 2013, accessed June 23, 2018.

[8] Blei, D. M., Ng, A. Y., and Jordan, M. I., "Latent dirichlet allocation" *Journal of machine Learning research*, 993-1022, January 2003.

[9] Salton, G., Wong, A., and Yang, C. S., "A vector space model for automatic indexing", *Communications of the ACM*, 18(11), 613-620, November 1975.

[10] Rehurek, R., and Sojka, P., "Software framework for topic modelling with large corpora." *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. 2010.

[11] Hsu, C. W., Chang, C. C. and Lin, C. J., "A practical guide to support vector classification", Technical report, Department of Computer Science, National Taiwan University, 2003.

Appendix

A. Author Contribution

TABLE III
Author Contribution: Report

Report	Tom	Max
Abstract	X	
Intro	X	
Background		X
Project (Data)		X
Project (Preprocessing)	X	X
Project (Design)	X	
Project (Evaluation)	X	
Results		X
Summary	X	X
Conclusion	X	X
Future Work	X	X
Graphs & Tables		X

TABLE IV
Author Contribution: Code

Code	Tom	Max
Preprocessing	X	X
Train & Test Split		X
BoW	X	
LDA	X	
W2V		X
GloVe		X
SVM	X	
Grid Search	X	
Evaluation	X	
Data Visualization		X
Experiments	X	X

B. Code

Our implementations of the different experiments can be found on our public Github repository: `https://tomjg14.github.io/artificial-sommelier/`.