Classy Classification: Classifying and Generating Expert Wine Review

Frederick Robson
Department of Computer Science
Stanford University
Stanford, CA 94305
frobson@stanford.edu

Loren Amdahl-Culleton
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305
lkac@stanford.edu

Abstract

We use an LSTM to classify wines from expert reviews of the wines according to their geographic location, grape variety, price and quality. We find that using an LSTM outperforms a baseline of Naive Bayes, albeit not significantly. Using the same dataset of reviews, we also use an attribute to sequence model to create alternative reviews for the same wines. We find that we are able to build a language model that creates reviews which contain accurate series of words, but that struggle with grammatical, and syntactic sense.

1 Introduction

For millennia, wine has been one of the most popular drinks around the world. As long as there has been wine, individuals have spent time talking about wine. In 149BC, Pliny the Elder talked at great length about his favorite wine, noting that ”There is now no wine known that ranks higher than the Falernian.” While today, the ability to effectively taste wines and describe the flavor of wines has become highly valued. In 2003, a wine buyer for a supermarket chain in the UK insured her olfactory systems ability to taste wine for £10mm [1], while in 2008, a vineyard owner from the Bordeaux region insured his nose for $8mm [2].

As a result of the long history of the language around wine and the value that has been assigned to the ability to taste wine, we were interested in the insights that recent deep learning techniques can provide around the topic of wine. Firstly, we were interested in whether the language that experts use to describe wines is sufficient to effectively classify the wines. Is the way that experts talk about wine systematic enough for an accurate deep learning classifier? Secondly, we were interested in whether we could use a language model to automatically generate a review of a wine given some simple facts about the wine. In other words, could we develop an expert wine reviewer of our own?

2 Data

Our dataset is a list of 129,971 reviews of different wines by professional wine tasters at Wine Mag, one of the preeminent wine magazines.[3] Each review was tweeted by one of Wine Mag’s to Wine Mag’s twitter account and so was limited to 280 characters. Some examples of the reviews are as follows:

[1]https://www.telegraph.co.uk/finance/2870834/Wine-taster-is-insured-for-10m.html
[2]https://www.seattletimes.com/entertainment/winemakers-nose-insured-for-8m/
[3]Our dataset can be found at https://www.kaggle.com/zynicide/wine-reviews

"Delicate aromas recall white flower and citrus. The palate offers passion fruit, lime and white peach with a hint of mineral alongside bright acidity."
Baglio di Pianetto 2007 Ficiligno White (Sicilia)

"Blackberry and raspberry aromas show a typical Navarran whiff of green herbs and, in this case, horseradish. In the mouth, this is fairly full bodied, with tomatoey acidity. Spicy, herbal flavors complement dark plum fruit, while the finish is fresh but grabby."
Tandem 2011 Ars In Vitro Tempranillo-Merlot (Navarra)

Beyond the review, each entry in the dataset provides further information on characteristics of the wine. Specifically, it provides information on:

1. Title: The name of the wine
2. Variety: The types of grapes used in the wine, for example Pinot Noir.
3. Country: The country the wine was produced in, for example France.
4. Province: The region within the country that the wine was produced in. The specificity of the provinces ranged widely. For example, California and Bordeaux are listed as provinces, even though California is thousands of times larger than Bordeaux.
5. Region 1 and 2: Gives more specific information about the location of the wine. For example, region 1 might be Alexander Valley, and region 2 might be Sonoma
6. Price: The cost of the wine at the time of review
7. Points: The rating the wine received from the expert wine taster. The rating ranges from 80 to 100
8. Title: The title of the wine
9. Taster Name: The name of the reviewer
10. Taster Handle: The twitter handle of the reviewer

The following table summarizes the data for each of these different categories:

Table 1: Dataset

Property	Country	Province	Region 1	Region 2	Variety	Winery	Title	Taster Name	Taster Handle	Price	Points
Type	Discrete	Discrete	Discrete	Discrete	Discrete	Discrete	Discrete	Discrete	Discrete	Continous	Continous
# Unique Values	43	425	1229	17	707	16757	118840	19	15	9386	21
# Unknown Values	63	63	21247	79460	1	0	0	26224	31213	8996	0

2.1 Data used in Models

From this dataset, we decided to focus on only five of these categories. We firstly discarded Taster Name and Taster Handle because we were not interested in trying to predict who wrote the review. Second, we discarded Region 1, Winery and Title. Although each of these categories were of interest, we decided that they would simply be too difficult to predict. For each of these categories, there are thousands and thousands of possible classes and often each class only has one member. Therefore, any model would have a difficult task classifying a review into any of these categories. Finally, although Region 2 had a reasonable number of classes for a classification model, we decided that the data was too sparse.

Therefore, we were left with five categories to predict for each review: Country, Province, Variety, Points and Price. Of these categories, Country, Province and Variety were discrete, while Points and Price were continuous. To ensure that our outputs (and specifically our loss functions) for each model were comparable, we decided to convert Points and Price into discrete distributions. Converting Points into a discrete distribution was simple. Each wine is given a score between 80 and 100, and therefore we created 21 discrete classes for each score. Converting Price into a discrete distribution was a little more complex because there were 9386 different price points in the dataset. In the end, we ran the K-Means clustering algorithm to place each price into one of ten classes.

Finally, we split our dataset into training, development and test in the ratio 70 : 10 : 20.

Table 2: Discretized Dataset

∼	Country	Province	Variety	Price	Points
# Classes	43	425	707	10	21
Mean Class Size	3021.12	305.67	183.83	12097	6189.1
Median Class Size	86	12	6	1417.5	3758
Max Class Size	54504	36247	13272	52645	1207
Min Class Size	1	1	1	7	19

3 Previous Work

3.1 Previous Review Classification Work

A number of excellent papers have been published around classifying online product reviews. Typically, these papers have focused on classifying the sentiment of the product review. In a famous unsupervised example, *Turney et al (2002)* used Pointwise Mutual Information (PMI) and Information Retrieval (IR) to classify reviews as either positive or negative. In their model, if the average semantic orientation of words in a piece of text (as trained on a large corpus) is positive then they classify the sentiment as positive.

More recently, research has focused on using deep learning as a way to classify natural language. *Conneau et al (2017)* used an extremely deep convolutional neural net to classify a number of different public datasets. Inspired by how deeper neural nets have advanced machine vision, they created a 29 layer deep neural net that outperformed the state of the art n-gram and TF-IDF results, as well as other neural net approached, on almost all of the datasets that they examined. Of particular interest to us was their performance on a set of Amazon product reviews, where they significantly outperformed all other models. Although, their results were interesting, it would have been impossible for us to even approach a 29 layer neural net with our limited time and computing resources.

Finally, there has even been research on classifying wine reviews. *Hendrix et al (2016)* also attempted to classify wines based on their reviews using an extremely similar dataset. They too used a dataset of Wine Mag reviews, but they used an earlier version, as a result, their dataset only has 76,585 entries, while our dataset has 129,971. They attempted to classify the wines along only four dimensions.

1. Color: Red, White, Rose
2. Country: USA, France, South Africa etc.
3. Price: They used two different variants of price categorization: *Large Difference* where cheap wines cost less than \$10 and expensive wines cost more than \$100, and *Small Difference* where cheap wines cost less than \$15 and expensive wines cost more than \$50. Any wine that was neither cheap nor expensive was discarded from their test dataset for price.
4. Variety: They only considered wines that were produced from a single grape, so a 70% Merlot, 30% Pinot Noir mix was excluded. Additionally, they manually matched different words for the same grapes. For example, they matched *Pinot Grigio* (Italian) with *Pinot Gris* (French).

For each of these categories they trained a unique model. Their model used a combination of Latent Dirichlet Allocation, Word2Vec embeddings, and Bag-of-Word features. Their F-Scores are shown below:

3.2 Generative

In our generative model, we use a variation of the *Attribute to Sequence* model outlined in *Dong et al*. In their paper, they generate reviews of products on Amazon given attributes of the product (i.e. User, Product Name, Rating, etc.). Their approach is to encode these attributes into a vector using a multilayer perceptron, the result of which is fed into an LSTM as its first hidden state. This LSTM,

Table 3: *Hendrix et al* Results

Property	# Test Instances	Bag of Word Features	Combined BoW +LDA+W2V	Combined Optimized
Color	14,213	90.7	94.3	97.6
Country	15,317	44.4	58	78.2
Price (Large Difference)	1,135	60.9	61	94.6
Price (Small Difference)	4,922	65	80.8	90.6
Variety	9,946	30.5	36.6	70.6

given some initial input word, is then used to generate a review, word by word, conditioned on these initial states. This encoder-decoder architecture is extremely interesting for the latent spaces it can produce. Particularly, work by Bowman, et. al. (2016) and Bhargava, et. al. (2017) gave us initial inspiration to examine auto-encoders which in turn led us to explore generative tasks more broadly. Eventually, we arrived at the language modeling task outlined in Section 5 of this paper.

4 Classification Models

4.1 Baseline

To create a baseline for our other models, we used a multinomial Naive Bayes. To compare our models against this baseline, we decided to use three metrics: accuracy, F1 weighted, and F1 macro. F1 weighted is the mean F1 score for each class, weighted for the size of each class, while F1 Macro is the unweighted average. We decided to show F1 scores as well because of the imbalance of our dataset. As can be seen from the data summary 2 table above, each of our output categories is extremely skewed. The most common categories have thousands of entries, while the least common have only one entry.

For our Naive Bayes model, we used Laplace smoothing. To decide on the best smoothing constant α, we tested different values of α on our development data set. Having decided on the best α, we then ran that Naive Bayes on our test set. The results are shown below.

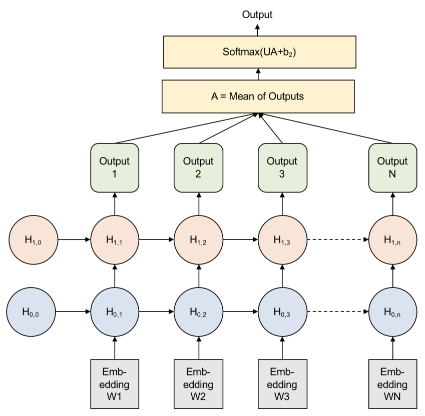
Table 4: Naive Bayes Baseline Results

	Country			Province			Variety			Points			Price		
	Acc	F1(W)	F1(M)	Acc	F1(W)	F1(M)	Acc	F1(W)	F1(M)	Acc	F1(W)	F1(M)	Acc	F1(W)	F1(M)
Dev															
$\alpha = 1$	0.808	0.835	0.187	0.568	0.658	0.036	0.451	0.534	0.026	0.230	0.245	0.110	0.552	0.564	0.226
$\alpha = 2$	0.776	0.826	0.141	0.501	0.609	0.024	0.395	0.501	0.015	0.216	0.255	0.082	0.554	0.579	0.203
$\alpha = 3$	0.751	0.817	0.114	0.442	0.557	0.018	0.363	0.480	0.011	0.209	0.265	0.066	0.558	0.589	0.197
$\alpha = 4$	0.729	0.807	0.094	0.396	0.521	0.013	0.345	0.464	0.009	0.202	0.270	0.055	0.555	0.592	0.189
$\alpha = 5$	0.714	0.797	0.084	0.364	0.493	0.010	0.332	0.453	0.008	0.197	0.273	0.049	0.553	0.597	0.182
$\alpha = 10$	0.663	0.754	0.067	0.299	0.442	0.004	0.292	0.405	0.007	0.190	0.276	0.039	0.534	0.609	0.148
Test															
	0.807	0.832	0.177	0.568	0.657	0.031	0.448	0.530	0.021	0.185	0.268	0.039	0.468	0.557	0.108

Interestingly, our baseline is able to outperform *Hendrix et al's* for some categories. We attribute the out-performance to the difference in the datasets as our dataset is significantly larger than *Hendrix et al's*.

4.2 Unique LSTM for Each Category

Our initial approach was to train an different LSTM for each category. Each LSTM had the following structure:



For each token in the review, we found the appropriate GLoVe embedding, replacing any token not in the GLoVe embedding with an unknown token. We then fed each word into a two layer LSTM neural network to create an output vector of size *num classes* for each token in the input. We then summed each of these output vectors, put them through another neural net layer, and finally took the softmax cross entropy loss of the final output.

As noted in our data summary section, there were often missing labels in our dataset. Our solution for missing data was simple - we simply excluded any review that was missing the label for the LSTM's category from the train, development and test dataset.

The model also depended on a number of hyperparameters: Hidden Size, Hidden Layers, Learning Rate. We found that two hidden layers was always more effective than one hidden layer, and that training three hidden layers took an impractically long time. Therefore, all our results used two hidden layers. For the other hyperparameters, we found that different values were more effective for different outputs. The learning rate and hidden size are noted at the bottom of the results section below.

4.3 Results

Table 5: Unique LSTM Models

∼	Country	Province	Variety	Points	Price	Average
Accuracy						
Train	0.949	0.783	0.654	0.504	0.571	0.692
Dev	0.825	0.600	0.534	0.274	0.496	0.546
Test	0.817	0.597	0.529	0.270	0.483	0.539
F1 (W)						
Train	0.952	0.810	0.686	0.512	0.585	0.709
Dev	0.833	0.628	0.563	0.279	0.509	0.562
Test	0.821	0.625	0.558	0.276	0.496	0.555
F1 (M)						
Train	0.427	0.163	0.094	0.357	0.245	0.257
Dev	0.271	0.082	0.081	0.178	0.222	0.167
Test	0.258	0.070	0.064	0.183	0.202	0.155
Hyperparameters						
Learning Rate	0.001	0.001	0.001	0.001	0.05	∼
Hidden Size	200	200	200	100	100	∼

4.4 Combined LSTM

Our second approach was to train a single LSTM that could simultaneously predict each of the five categories. The layout of the LSTM was similar to the LSTMs used in the previous section. The main differences came within the output layer. Rather than a vector of length *num classes*$_i$, our output was now a matrix with shape $(5, 707)$. The matrix had one column for each category, and 707 rows for the category with the largest number of classes (variety). We needed our approach to be vectorized but we also needed to account for the fact that different attributes had different numbers of classes and thus we created a mask of size (num_attributes (5) x num_classes_max (707)) which we applied to our output prior to the softmax. This effectively forces the softmax-CE to only consider the non-zeroed values for each attribute row and thus only predict on the given attribute's valid class range.

Another challenge with this model was in accounting for missing labels. Although each category is only missing a small proportion of its data, if we excluded any review that was missing one or more pieces of data, then the size of our training, dev and test set would dramatically shrink. Therefore, to keep track of which labels were missing during the loss operation, we simply added another "0" class to our num_classes_max. Any label that was encoded with a zero we knew was missing and thus applied applied our ground truth label vector as a sort of boolean mask on the output loss vector from the softmax-CrossEntropy function. Any loss from an attribute that had a corresponding ground truth label of "0" was simply set to zero itself.

As far as the parameters of our model are concerned, we began tuning the combined model using similar hidden size and learning rate as the unique LSTMs. However, we realized that our model was not able to over-fit even when running for long epochs. Increasing the hidden size to 300 solved the issue almost immediately.

4.5 Results

Table 6: Combined LSTM Model

∼	Country	Province	Variety	Points	Price	Average
Accuracy						
Train	0.956	0.776	0.725	0.384	0.629	0.694
Dev	0.804	0.590	0.527	0.264	0.531	0.543
Test	0.806	0.586	0.513	0.260	0.529	0.539
F1 (W)						
Train	0.958	0.801	0.751	0.395	0.675	0.716
Dev	0.809	0.616	0.550	0.273	0.577	0.565
Test	0.810	0.612	0.536	0.268	0.574	0.560
F1 (M)						
Train	0.468	0.206	0.177	0.269	0.243	0.273
Dev	0.261	0.088	0.090	0.168	0.181	0.158
Test	0.262	0.082	0.076	0.170	0.167	0.152

4.6 Analysis of LSTM Models

As shown in the results above, both of our models outperformed baseline Naive Bayes, albeit only slightly. It is interesting to try to explain why our models only outperformed the baseline slightly. Our hypothesis is that the format of the data meant that an LSTM was not able to add a significant amount of value over Naive Bayes. Consider the following example review:

> "Baked plum, molasses, balsamic vinegar and cheesy oak aromas feed into a palate that's braced by a bolt of acidity. A compact set of saucy red-berry and plum flavors features tobacco and peppery accents, while the finish is mildly green in flavor, with respectable weight and balance.."
> *Felix Lavaque 2010 Felix Malbec (Cafayate)*

The majority of the review is made up of a list of nouns or adjectives that describe the flavor of the wine. The key advantage of an LSTM over Naive Bayes for classification tasks is that LSTM models are implicitly able to capture the importance of order and dependency in a sequence of words. However, for a list of nouns and adjectives, the order and dependency of the words is unimportant. The meaning of ”Baked plum, molasses, balsamic vinegar...” is almost identical to ”Molasses, balsamic vinegar, baked plum” (perhaps the only difference is that the most important flavor might come first). Our LSTM is only able to outperform the Naive Bayes model in the other parts of the sentence where the word ordering and dependencies are important. Because the majority of our descriptions have a similar format to this example - where large chunks of the sentence are lists of nouns or adjectives - our LSTM is only able to slightly outperform the baseline.

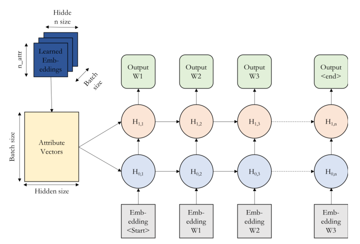
An interesting insight about the results is that *points* was consistently the most difficult category to predict. Across each of the models and each of the evaluation metrics, *points* was consistently the worst performer. Our initial hypothesis was that the reviewers were normalizing their scores for the price of the wine. That is, they might prefer a $500 wine to a $20 wine, but will rate the $20 wine more highly as the $20 wine is better value for money. However, after reviewing the dataset, this hypothesis seemed unlikely because the correlation between *points* and *price* was 0.41. Additionally, if the reviewers were normalizing for the price, then hopefully our combined LSTM model would have been able to capture this fact. As a result, we came to the conclusion that either the reviewers deliberately attempt to use neutral language to describe the wines, or their reviews are extremely subjective.

Finally, we also observed that the performance of our combined LSTM was nearly equivalent to the combined performance of each of the individual LSTMs. The average test accuracy across all categories for the unique LSTMs was 0.543, equal to the average test accuracy across all the categories for the combined LSTM. What was remarkable about this result was the difference in the number of parameters. The unique LSTMs had a summed hidden size of 800, while the combined LSTM only had a hidden size of 300. Because we were able to achieve similar results with a much smaller hidden size in the combined LSTM, we concluded that our model was able to learn about one category from other categories. Intuitively, this thought makes sense. That a wine is from Italy also implies that the wine is not from California; that a wine is from France implies that it is more expensive than a wine from Germany. An interesting way to test this hypothesis would be to run Principal Component Analysis or another similar algorithm on the dataset, and see how much of the variance can be captured in two or three dimensions.

5 Generative Model

For the second part of our project, we constructed a generative model. The model took the attributes of the wine (the country, the province, the variety and the price) and outputted an “expert” wine review of our own.

5.1 Model



Our model takes inspiration from the Attribute to Sequence work done by Dong et. al. in generating Amazon Wine reviews. While their model encoded attributes using a multi-layer perceptron, we took a slightly simpler approach. Much like the word embeddings, we initialized one large embedding matrix for all of our attributes across all categories and thus indexed into the matrix to retrieve a corresponding attribute embedding. These attribute embedding vectors (of same dimension as our hidden states) were then averaged before being set as our initial hidden state at train time. Our aim here was to condition our wine review language model one the characteristics of the wine itself.

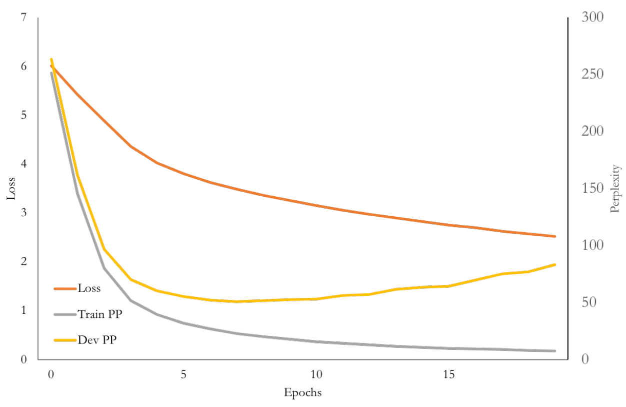
Apart from the attribute embeddings, the main block of our model is a **two-layer LSTM**. Here, we use sequence tagging to train our language model. That is, given a word in a wine review, its corresponding label is the next word in the sentence. Following our **softmax-CE** loss, we set up our model to update both our attribute embeddings and our pretrained-glove embeddings. We do this in the hopes that it bias our word vectors to produce syntax specific to the somewhat esoteric characteristics of wine reviews.

Following training, our model is ready to generate reviews. Here, instead of feeding in an entire sentence to our RNN, we only give a pre-defined <START> token which was present at training and the embedded attributes of the given wine for which we wish to generate a review. Then, after each forward pass we take the output word predicted and feed its embedding in as our next input while feeding the output hidden state as the new input hidden state. We do this until reaching an end token or until meeting some threshold length.

5.2 Results

Table 7: Language Model Results

	Accuracy	Perplexity
Train	0.376	22.868
Dev	0.338	50.704
Test	0.335	51.851



Example Output:

> ”hint Not balance Grand excellent Despite edge leads point in power popular quite quality on feels easy gardens touch Quite natural welcome shop calls sweet interest lime suggests pops effort turn until acids focused . identified few note passing proportion power all substantial streak blackberry welcome de maximum promise out any hit mineral includes white dried make because nonetheless Balanced.”
> *Italy, Tuscany, Sangiovese*

5.3 Analysis

We evaluated our language model using both one-to-one accuracy and Perplexity.

We were pleased to see that our model was able to effectively train on the dataset. Over time the accuracy and perplexity dropped substantially, achieving an extremely respectable 0.335 test accuracy and 51.851 perplexity. If given more time, we believe we could improve these results through further hyperparamater tuning. In particular, we noticed that the development perplexity gradually rose after the fifth epoch while the training perplexity continued to fall. This result suggests that our model was over-fitting and could potentially be solved by either reducing the learning rate or reducing the size of the hidden layers.

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

At a more qualitative level, although the majority of the example review is gibberish, there are pockets of text that are fairly naturalistic. The phrases ”substantial streak of blackberry” and ”nonetheless balanced” could conceivably have come from real wine reviews.

For a final (and even more unscientific evaluation) we compared our output reviews to our personal reviews of actual wines that we tasted. In particular, we purchased a Grifone Tuscan Sangiovese (2016) and used our model to output the example output. We were pleasantly surprised to find that many of the adjectives in the example review matched our own perception of the wine, finding that the Sangiovese had a ”substantial blackberry streak.”

6 Conclusions and Future Work

For our classification model, both our unique and combined LSTM's outperformed the Naive Bayes Classifier baseline and we believe that further hyperparameter tuning could see that performance gap increase due to signs of over-fitting in some of the categories. For our generative model, we saw a respectable accuracy and perplexity in our results, as well as correct characteristics, even if the output was not coherent.

We believe that there is significant potential for further work on both approaches. Both models could be enhanced by adding more attributes to our model. Some attributes such as the vintage would be especially valuable. For our generative model, we would like to explore different input features as well as perform attention over the attributes at each time-step. Finally, it would be interesting to create a generative model that relied on the chemical composition of the wine, rather than attributes such as price or country. The chemical composition could provide a much more detailed and richer dataset to create accurate reviews with.

References

[1] Hendrickx, Iris & Lefever, Els & Croijmans, Ilja & Majid, Asifa & Van den Bosch, Antal. (2016). Very quaffable and great fun: Applying NLP to wine reviews. 306-312. 10.18653/v1/P16-2050

[2] Dong, L., Huang, S., Wei, F., Lapata, M., Zhou, M., & Xu, K. (2017). Learning to Generate Product Reviews from Attributes. In Proc. EACL17, pp. 623632

[3] Boya Yu, Jiaxu Zhou, Yi Zhang, Yunong Cao (2017). Identifying Restaurant Features via Sentiment Analysis on Yelp Reviews (https://arxiv.org/pdf/1709.08698.pdf)

[4] Conneau, A., Schwenk, H., Le Cun, Y., & Barrault, L. (2017). Very Deep Convolutional Networks for Text Classification. Retrieved from https://arxiv.org/pdf/1606.01781.pdf

[5] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., Bengio, S., & Brain, G. (2017). Generating Sentences from a Continuous Space. Retrieved from https://arxiv.org/pdf/1511.06349.pdf

[6] Bhargava, D., Vega, G., & Sheffer, B. (2017). Grounded Learning of Color Semantics with Autoencoders, 17. Retrieved from http://web.stanford.edu/class/cs224n/reports/2762012.pdf

[7] Pennington, J., Socher, R., & Manning, C. D. (n.d.). GloVe: Global Vectors for Word Representation, 15321543. Retrieved from http://www.aclweb.org/anthology