"Subtle notes of [-124957e7, 0.127489, 123483e7]": Non-Binary Sentiment Classification of Amazon Fine Food Reviews with Four Vector Space Models

Bradley Goldsmith, Garrett Johnson, Zipporah Klain, Nick Ornstein

0 Introduction

Computers are becoming increasingly adept at extracting useful information from raw linguistic data, but they "understand" little about what words actually mean. Vector space models (VSMs) encode words and documents as vectors in higher-dimensional linear spaces, where each dimension corresponds to some feature of the word or text. While probabilistic models treat words as strings whose probability of occurring in the context of other strings can be calculated, VSMs encode relation information between words and documents by decomposing them into a definite number of features. The models have been the subject of much research over the past decade, and here we compare how much information about human attitudes is encoded in these vector space models by evaluating Amazon food reviews with a linear regression model.

To approach this question of "how well do VSM models encode sentiment information?", we created a classification task based on an Amazon reviews dataset. We used a linear regression model to predict Amazon star-based ratings from 1 to 5 based on review texts and to analyze the performance of various vector space models. The models we selected were doc2vec (Mikolov & Le, 2014), fastText (Bojanowski et al, 2016), tf-idf (Jones, 1972), Latent Semantic Analysis (Landauer & Dumais, 1997), and GloVe (Pennington et al, 2014). In choosing these models, we aimed to represent a vast array of common word embedding schemes used in a variety of contexts, and characterize their relative abilities to embed rating information from review texts.

We expected the tf-idf embedding to outperform plain LSA in terms of review prediction accuracy. While simple direct vocabulary counts make use of absolute word frequencies in each document, tf-idf controls for frequency of each word across documents, lending greater context to its embeddings. This additional information, encapsulating the rarity of words across the corpus, rather than relative to other words in the document, should allow for more accurate linear predictions of star ratings. Furthermore, we expected that embeddings which capture information on context and semantic relations, such as fastText and doc2vec, would outperform models that treat words as atomic or independent features, such as GloVe and LSA.

1 Related Work

The past decade has witnessed numerous increasingly complex and accurate language models, as well as a blurring of the distinction between vector space and probabilistic models. For example, BERT-based (Devlin et al. 2019) models fine-tune their word embeddings based on the downstream task for which they are being trained, and Facebook's LASER (Facebook 2019) generates sentence embeddings (as opposed to word or document embeddings) that are so comprehensive that they can identify the polarity relationship (related vs. opposite) between sentences *in different languages.* However, these new hybrid models are trained with some downstream task in mind, whereas simpler models like doc2vec (Quoc and Mikolov 2014) and fastText (Bojanowski et al. 2017) offer regular means of embedding documents in vector spaces. Turney and Pantel

(Turney and Pantel 2010) have proposed diagnostic tests for ascertaining which aspects of semantic meaning are captured in vector space models, but such diagnostics concern themselves with more complex aspects of semantic theory like semantic role labeling and relational classification. Here, we evaluate VSMs based on how much information about the sentiment of Amazon reviews is directly encoded in the model itself.

2 Models and Methods

2.1 Doc2vec

Doc2vec, or Paragraph Vector, was presented in Mikilov and Le (2014). Starting with a word2vec representation of a given document, with trained word vectors, doc2vec adds another vector, the paragraph ID. This vector remembers the “topic” of the paragraph or document and is trained alongside the word vectors in the training process. There are two versions of doc2vec: the Distributed Memory version of Paragraph Vector (PV-DM) and the Distributed Bag of Words version of Paragraph Vector (PV-DBOW). PV-DM is an extension of the word2vec continuous bag of words (CBOW) model, using all the words *plus* the paragraph vector to predict a word based on its context (surrounding words); PV-DBOW is correspondingly similar to word2vec’s skip gram, using only the paragraph vector to predict a target word. While PV-DBOW is faster, since there is no need to save the word vectors, PV-DM is more accurate, so we chose to use it here in our doc2vec model.

In this experiment, we treated each Amazon review as a paragraph, or document, with a paragraph vector created to represent the overall “topic” of each review. More specifically, to create the doc2vec embedding, we tagged each review with its star rating, built the vocabulary (for the whole corpus), trained the model (both paragraph and word vectors) by predicting holdout words, inferred a *new* paragraph vector for each Amazon review from the words composing it, and then fit a linear regression model to these vectors, predicted star ratings for the test set, and evaluated accuracy and error, as described below.

2.2 FastText

The fastText linear classifier was developed by Facebook AI researchers in 2016 by P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. fastText has several advantages as a classification model: it is fast, has performance comparable to neural network alternatives, and can compute vectors for words outside of its training vocabulary. The key to these properties is the use of subword information in word embeddings. fastText models learn word vectors as skipgrams, where each word is the sum of smaller, sub-word vectors called character n-grams. For example, the word “hello” would be marked with head and tail characters (“<hello>”) and then processed as a series of continues character n-grams of size 3 (“<he” + “hel” + “ell” + “llo” + “lo>”). In turn, document vectors (in our case, each review) are the sum of their word vectors. This approach allows fastText to capture morphological information and generalize subtle semantic connections. Because of these unique characteristics of the fastText embeddings, we believe that a classifier trained with fastText will outperform the other embeddings under examination.

The fastText API has been developed with the intention of keeping code lightweight and uncluttered. As a consequence, the available functions in the fastText python module and the nature of how embeddings are trained presented some notable hurdles when we conducted analysis. Namely, fastText produces models by taking in plain text files, training word vectors, and returning a fastText model object. This meant that both train and test data preprocessing had to be unique for creating fastText embeddings (compared with the

Table 1: GloVe Model Parameters

	Gigaword + Wikipedia 2014	Common Crawl	Common Crawl	Twitter
Corpus size	6B tokens	42B tokens	840B tokens	2B tweets
Vocabulary size	400,000	1.9M	2.2M	1.2M
Dimensions	50d, 100d, 200d, 300d	300d	300d	25d, 50d, 100d, 200d

other embeddings). Further, the model object is a multinomial logistic regression classifier with an associated matrix representing document vectors.

It was neither simple nor clear from the documentation how to extract the word vectors from the models. Because of the difficulty of the matter and the principle that we should not deviate from the intended regression type and unnecessarily hinder the performance of fastText, we decided to compare the performance of the fastText logistic regression with our other embeddings' performances using a linear regression classifier.

After the decision to use fastText's built-in classifier architecture was made, there still remained hurdles to obtaining statistical descriptions of model performance. fastText model objects only have an internal metric function for calculating precision and recall at *k*. In order to obtain accuracy and root mean square error for fastText models, we had to create our own scripts for extracting the word vectors, re-processing the testing data, and calculating the accuracy and root mean square error based on the lists of model predictions and correct labels.

2.3 GloVe

The GloVe [glʌv] model was developed at Stanford in 2014 by Jeffrey Pennington, Richard Socher, and Christpher D. Manning (Pennington *et al.* 2014). It is a global log-bilinear regression model for unsupervised word-learning, hence the name *Global Vectors*. It consists of several sets of word vectors that have been pre-trained on huge corpora in an attempt to create generalizable and adaptable vector space models. GloVe differentiates itself from other models by leveraging matrix factorization (like Singular Value Decomposition, used in LSA) and moving window-based methods (like c-bow and skip-gram) together in order to encode both global and local word occurrence statistics, and outperforms c-bow and skip-gram models on several evaluations. The authors provides ten different models on the project website, pretrained on four different corpora:

Gigaword is a static news repository and Wikipedia is the online encyclopedia, so the training data likely does not reflect the usage patterns in Amazon food reviews. The Common Crawl models are trained on data from all over the internet, so it is possible that these models better reflect Amazon review usage patterns, but not perfectly. The Twitter corpus is most similar to the Amazon review corpus in that both corpora consist primarily (or exclusively) of short observations that reflect the attitudes of a wide variety of people. Because the GloVe models are trained on data that contains more attitudes and aesthetic judgements than the other training corpora, it is anticipated that it will outperform models trained on Gigaword + Wikipedia and Common Crawl.

Document vectors are generated from the GloVe models by treating each review as a bag of words

Table 2: Model Metrics

Model	Label Frequencies	LSA (Counts)	LSA (tf-idf)	Doc2Vec	fastText	GloVe (Twitter)	GloVe (Gigaword)
Accuracy	0.423	0.309	0.375	0.361	**0.594**	0.288	0.305
RMSE	1.883	1.182	**1.120**	1.126	1.318	1.189	1.159
Recall	**0.423**	0.311	0.400	0.361	---	0.288	0.305
Precision	0.456	0.616	**0.686**	0.646	---	0.607	0.614
F1	0.439	0.413	**0.505**	0.397	---	0.309	0.281

and summing together the GloVe vectors for all of the words.

2.4 LSA

In 1997, Landauer and Dumais introduced 'Latent Semantic Analysis' as a "high dimensional associative model" for language learning. LSA uses the Singular Value Decomposition to reduce the dimensionality of document-term matrices and has proven a highly effective method of topic modeling. We generate two sets of LSA vectors for our study: one set consists of raw term count matrices generated using scikit-learn's CountVectorizer() module and the other of tf-idf matrices generated using the tfIdfVectorizer() module. tf-idf divides the raw term counts by the inverse of the number of documents in which the term appears, and in doing so assigns higher weights to words that appear less frequently on the assumption that they are more useful for differentiating documents from one another.

3 Experiments

For our dataset, we used the Amazon Fine Food Reviews dataset from Kaggle.com, which consists of about 500,000 food reviews from Amazon, spanning more than 10 years up to October 2012. From this dataset, we used the text of each review and the corresponding rating given out of five stars, training our models to predict this rating based on the text. We cleaned and lemmatized this data and created a standardized train-test split of 454,763 reviews in the training set and 113,691 in the test set.

The dataset was cleaned by removing all punctuation, converting the string to lowercase, and lemmatizing the tokens with the Lemmatizer module from the spaCy library.

To be able to compare our different embeddings effectively in our initial analysis, we standardized some variables in creating these embeddings (except for GloVe, which came pre-trained). We held our vector lengths to 50, trained for 10 epochs, and limited our vocabulary to including only words that appeared around 6000 times in the dataset, which resulted in a vocabulary of size approximately 600 (details varied slightly by model).

For analyzing the performance of each of our models (except fastText, which has a built-in classifier), we trained linear regression models from scikit-learn on the training set of the embeddings and then predicted the review ratings in the test set based on their embeddings. We evaluated these review ratings by calculating their root mean squared error with respect to their actual ratings and then also by rounding each predicted review to its nearest integer between one and five (to get a valid star rating) and computing accuracy.

4 Results

At vector size 50 and training for 10 epochs (when applicable), we obtained the following results (Table 2). As expected, fastText performed the best when comparing both accuracy (from our predicted integer star reviews) and root mean squared error (from our raw predicted ratings), though results were otherwise somewhat comparable across the board. It is worth noting that fastText's results cannot really be compared to the other embeddings' since it used its own built-in logistic regression classifier, rather than the standardized scikit-learn linear regression model we used for the rest.

The Label Frequencies column of Table 2 represents the results obtained from randomly predicting ratings based solely on the raw distribution of ratings in the training data set, serving as a baseline against which to compare our methods. Unfortunately, we found that the random predictions had *higher* accuracy than any of our models, other than fastText, although their root mean squared error was also significantly higher. We tentatively attribute this to the overrepresentation of five-star reviews in the underlying dataset.

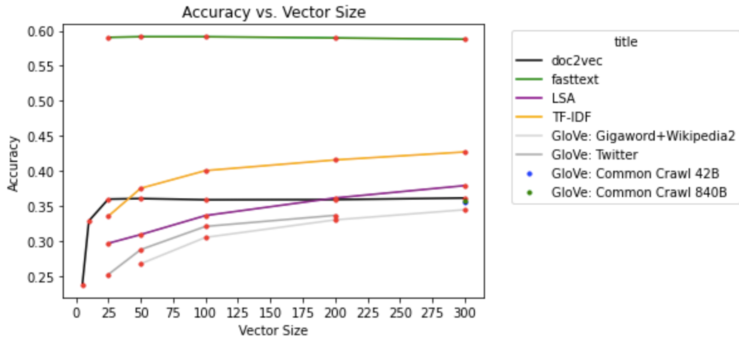
5 Additional Analysis

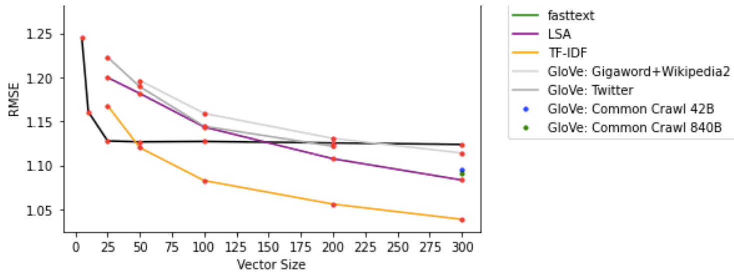
In the course of conducting our present investigation into the relative effectiveness of different word embedding techniques, a number of relevant follow-up experiments arose. These experiments are intended to develop our understanding of the different word embeddings by observing the changes that occur when certain hyper-parameters are varied, data is preprocessed differently, or when the dataset is restricted to have equal numbers of each star rating (i.e. 29769 randomly selected one-star reviews, 29769 randomly selected two-star reviews, etc.).

5.1 Vector Dimension Analysis

The first additional analysis we conducted was to examine the effect of varying the word and document vector dimensions for each embedding scheme while fixing the other hyperparameters like context, epochs, and learning rate. The context size and learning rates were fixed at the defaults set by the developers for the relevant embedding schemes and the number of epochs was set at 10. We then examined the change in the accuracy of the models as the vector dimension varied, from which we could determine the optimal dimension size for each embedding. We expected that increasing the dimensions would only improve the accuracy of models, but there would be diminishing returns after a certain point, when performance either plateaus or begins to decline. The rationale is that after a certain point, encoding more, extremely subtle information on word relationships should not significantly improve model performance on the sentiment prediction task. Our results are graphed in the figures below.

The above charts, "Accuracy vs. Vector Size"





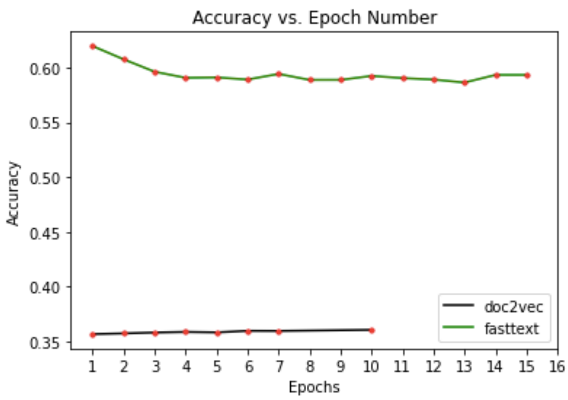
and "Root Mean Square Error vs. Vector Size," display the results of this further experiment. Results indicate that increasing vector dimensionality does improve model performance, though subsequent improvement becomes less substantial after at around 100 dimensions. fastText and doc2vec initially showed little variation as compared with LSA, tf-idf, and GloVe, but this was likely because the minimum dimension being tested (25) captured most of the variance in those embeddings. Accordingly, we have done additional analysis on doc2vec and found that indeed it does show substantially worse performance below 25 dimensions. We can reasonably expect the same to be true of fastText.

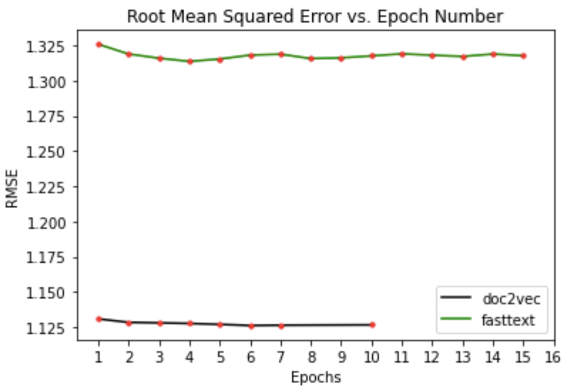
5.2 Training Duration Analysis

The second additional analysis we conducted is similar to the above study on the effects of dimensionality. Here, we varied the training epochs for doc2vec and fastText. Once again, the context size and learning rates were fixed at the defaults set by the developers for the relevant embedding schemes and now the word vector

Table 4: Filtered/Even Data

Model	Label Frequencies	LSA (Counts)	LSA (tf-idf)	Doc2Vec	GloVe (Twitter)	GloVe (Gigaword)
Accuracy	0.197	0.244	**0.280**	0.242	0.233	0.229
RMSE	1.885	1.277	**1.202**	1.295	1.292	1.295
Recall	0.197	0.244	**0.280**	0.242	0.233	0.229
Precision	0.158	0.343	**0.418**	0.334	0.316	0.325
F1	0.175	0.285	**0.335**	0.184	0.268	0.268





dimensionality was fixed at 50. We examined the change in the accuracy of the various models as they were trained on more epochs, creating a graph of the models' performance over the duration of training. The resulting graph is shown below. We predicted that, similar to above, there will be diminishing returns as the number of epochs increases. There is also a danger of overfitting the model to the training data, which will be particularly problematic for other tasks that would require models to be highly generalizable.

The above charts, "Accuracy vs. Epoch Number" and "Root Mean Square Error vs. Epoch Number," display the results of this further experiment, which was only applicable to doc2vec and fastText (as other models were either pre-trained or were trained as statistic models rather than learning models). Results indicate that increasing epoch size did not have a significant effect on the performance of our classifiers. This makes sense, because the imbalance in review classes (a majority of ratings were 5, and a larger majority were just 5 or 1) and the relatively small size of our vocabulary (just over 600 words) mean that there are not actually many associations to be learned. Performance then is mostly dependent on how the embeddings are made, rather than how long they are trained.

Table 3: GloVe Metrics

	Common Crawl 42B (300d)	Common Crawl 840B (300d)
Recall	0.357	0.358
Precision	0.644	0.649
RMSE	1.095	1.091

5.3 GloVe corpus size analysis

There are two GloVe models trained on Common Crawl corpora, but one corpus contains 42 billion tokens while the other contains 840 billion. The increase in performance is negligible (< 0.01) improvements in recall, precision, and RMSE. These results indicate that 42 billion tokens is a representative sample size of the English language and increasing the size of the training corpus beyond this point does not meaningfully improve accuracy.

5.4 Filtered Data

In our original Results section, we were disheartened to find the random classifier outperforming us significantly on accuracy, if not RMSE. We attributed this largely to the overrepresentation of five-star reviews in the dataset, meaning that a guess of five stars would be likely to be correct, regardless of the review's text. To account for this, we filtered the dataset down to a random selection of 29769 reviews of each star rating (i.e., 29769 randomly selected one-star reviews, 29769 randomly selected two-star reviews, and so on) and reran our analyses. In Table 4, it is clear that this filtering produced more expected results from the random classifier (Label Frequencies), with an accuracy of 0.197, or almost 1/5th, the expected value at chance. While our embeddings' accuracies also shrunk across the board, they now outperformed the random classifier in accuracy and continued to outperform it in root mean squared error to a similar degree.

6 Discussion

The use of a linear regression model to build predictions for each embedded review is unusual in the context of how embeddings are usually evaluated and compared (Bakarov, 2018). This method of testing would fall under Bakarov's category of 'Extrinsic Review,' and more specifically, 'Sentiment Analysis.' In using a linear regression model, which is highly simplistic in tuning weightings for each dimension to project a vector onto a 1-D ratings space, we hoped to strip away more complicated classifiers and examine the work done by each embedding.

Initial results proved somewhat surprising. Our initial hypothesis regarding the use of tf-idf versus straight counts feeding into the LSA proved correct, in that in all experiments, tf-idf outperformed the straight counts. The top performer on the initial experiment in terms of accuracy was fastText, by a wide margin (see Table 2, initial results). However, its RMSE was significantly higher than the Counts and tf-idf versions of LSA. The frequency-based random predictor exhibited similar behavior, whereby its accuracy was much surprisingly much higher than any of the trained models, but its RMSE was quite large. This suggests that fastText learned that 5-stars was by far the most popular rating, rather than encoding the meaning in each document.

The first two follow-up experiments, dimensional analysis and epoch number, confirm that classification accuracies for these embeddings can be increased via careful hyperparameter tuning. It also implies that such linear models will not necessarily be improved by increasing overhead so as to include more information during learning. In other words, there is a sweet spot for each training scenario that allows each model to be optimally accurate while remaining relatively inexpensive to train. It also indicates that semantic information is not uniformly distributed in a text; i.e. our models can be selective with what aspects of the training data are assigned high significance and actually perform better, and not worse, than if they had considered more information. Optimal performance for a low price is one of the benefits of using linear models, and we are pleased with the results of these two additional experiments.

In the fourth follow-up experiment, we controlled for the disproportionate number of fives in the dataset by using the same number of reviews from each of the five classes. This normalization negatively affected the prediction accuracy of the linear regression model. From this we can conclude that our models were achieving high accuracy just by guessing five most of the time, exploiting the preponderance of fivess in the data. Once the proportions of ratings in the data was controlled for by including even amounts of each rating, the random predictor accuracy dropped to chance. The tf-idf encoding proved most successful in this context, outperforming GloVe and doc2vec. This result can possibly be attributed to the clean simplicity of term-document matrices. The bag-of-words approach allows for the selective weighting of keywords that captures the most salient information about the sentiment of each review.

Despite being trained on gigantic corpora, the GloVe models performed worse than the LSA models, but the raw count LSA model only barely outperformed Twitter GloVe (especially at d = 100) so perhaps if we generated document embeddings using (means or some other method besides addition), the roles would be reversed.

7 Conclusion

In conclusion, fastText's built-in classifier proved more accurate than the other vector space models with the linear classifier, although its root mean square error was larger than other models, like the Counts and tf-idf implementations of LSA. Accuracy increases as the dimensionality of the vectors increases, but it does not increase linearly, leveling off toward 300 dimensions. For fastText and doc2vec, the number of training epochs has minimal bearing on each model's performance. Ultimately, we gained a variety of helpful insights into tuning the hyperparameters of vector space models like dimensionality and training epochs in order to optimize their performance on classification tasks.

For the controlled ratings frequency experiment (Table 4), the top accuracies of our models, while decisively above chance, were still not high, peaking at around 28% accuracy, and missing the correct rating by 1.2 stars on average at best. While this poor performance could be due to the highly limited vocabulary size (necessary for model implementation due to lack of GPU's available to run our code), this result likely shows that review information was not approximated well by a linear model. Future work would employ feed-forward neural networks with non-linear transfer functions as a classifier in order to evaluate the role non-linearities in these vector space models play in the encoding of sentiment in document embeddings.

References

Bakarov, Amir. 2018. "A Survey of Word Embeddings Evaluation Methods." *ArXiv:1801.09536 [Cs]*, January. http://arxiv.org/abs/1801.09536.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. "Enriching Word Vectors with Subword Information." *ArXiv:1607.04606 [Cs]*, June. http://arxiv.org/abs/1607.04606.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." *ArXiv:1810.04805 [Cs]*, May. http://arxiv.org/abs/1810.04805.

Ettinger, Allyson, Ahmed Elgohary, and Philip Resnik. 2016. "Probing for Semantic Evidence of Composition by Means of Simple Classification Tasks." In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, 134–39. Berlin, Germany: Association for Computational Linguistics. https://doi.org/10.18653/v1/W16-2524.

Faruqui, Manaal, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. "Problems With Evaluation of Word Embeddings Using Word Similarity Tasks." *ArXiv:1605.02276 [Cs]*, June. http://arxiv.org/abs/1605.02276.

Gladkova, Anna, Aleksandr Drozd, and Satoshi Matsuoka. 2016. "Analogy-Based Detection of Morphological and Semantic Relations with Word Embeddings: What Works and What Doesn't." In *Proceedings of the NAACL Student Research Workshop*, 8–15. San Diego, California: Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-2002.

"LASER Natural Language Processing Toolkit." 2019. *Facebook Engineering* (blog). January 22, 2019. https://engineering.fb.com/2019/01/22/ai-research/laser-multilingual-sentence-embeddings/.

Le, Quoc V., and Tomas Mikolov. 2014. "Distributed Representations of Sentences and Documents." *ArXiv:1405.4053 [Cs]*, May.Proceedings of the 31st International Conference on Machine Learning, in PMLR 32(2):1188-1196. http://arxiv.org/abs/1405.4053.

Maas, Andrew L, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. n.d. "Learning Word Vectors for Sentiment Analysis," 9.

Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.

Sparck Jones, Karen. 1972. “A Statistical Interpretation of Term Specificity and Its Application in Retrieval.” *Journal of Documentation* 28 (1): 11–21. https://doi.org/10.1108/eb026526.

Schnabel, Tobias, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. “Evaluation Methods for Unsupervised Word Embeddings.” In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 298–307. Lisbon, Portugal: Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1036.

Stanford Network Analysis Project. (2017, May 1). Amazon Fine Food Reviews. Kaggle. https://www.kaggle.com/snap/amazon-fine-food-reviews.

Turney, Peter D., and Patrick Pantel. 2010. “From Frequency to Meaning: Vector Space Models of Semantics.” *Journal of Artificial Intelligence Research* 37 (February): 141–88. https://doi.org/10.1613/jair.2934.

Ye, Zhe, Fang Li, and Timothy Baldwin. n.d. “Encoding Sentiment Information into Word Vectors for Sentiment Analysis,” 11.