Sentiment Analysis of Amazon Product Reviews

Akash Rastogi

April 25, 2017

Introduction

With the explosion of review websites, microblogging websites, social networks and similar platforms, sentiment analysis has become a prime objective for many organizations. Sentiment analysis plays an important role in receiving feedback by being a silent listener to the chatter on some subject we are interested in. It has found applications in various domains like finance (analysis of Twitter data to predict movement of stock prices), politics (analysis of Twitter data from a state to predict political support of a candidate), product quality (from product reviews), and many other areas.

In our study, we are interested in classifying the sentiment of Amazon customer reviews to better understand customer feedback for various products. We believe the underlying analysis has some interesting insights and can aid product category managers in their decision making processes. For example, our project can inform product quality managers which product categories to train on to better predict sentiment based on product reviews.

Analytic Questions

Based on our motivation, we have three analytic questions we seek to answer with this project.

1. Do certain classifiers predict sentiment of Amazon reviews more accurately than others?
2. Does the best classifier change from product type to product type?
3. Do product-specific classifiers that learn from only that product category perform better than classifiers that learn from all product categories?

Overview of Dataset

We used publicly available Amazon product review data that has been compiled by Julian McAuley from UCSD (`http://jmcauley.ucsd.edu/data/amazon/`). It contains product reviews (ratings, text, helpfulness votes, summary, etc.) and metadata (descriptions, category information, price, brand, and image features) from Amazon spanning from May 1996 to July 2014. The product reviews can be easily accessed from their website, while the metadata are accessible upon request. These datasets are categorized by the product category such as books, home and kitchen, movies and TV etc.

helpful	**summary**	**reviewerID**	**reviewerText**	**overall**	**reviewTime**	**asin**
[0,0]	Great	A2G7Y6ETCISQ4G	book.	5	07 19, 2014	60509589
[0,0]	Great!	A2G7Y6ETCISQ4G	ordered	5	07 19, 2014	28740637
[0,0]	Learning	APRTK7SCYNOHT	nostalgic..	5	07 17, 2014	28633873

Table 1: A snippet of the dataset for book reviews from the Amazon review dataset. The words from reviewText and summary were removed to have a better readability of the data.

Methodology

We use an array of algorithms, classification techniques and manually-constructed resources to help us classify sentiment of Amazon product reviews. The general approach to performing sentiment analysis is to identify the representation of the text in the vector space and map it to the labeled data. Hence, we can divide the process of sentiment analysis into two parts:

1. Creation of feature vector space to represent the text (known as text embedding)

2. Application of classification techniques to map feature space to labeled data

The baseline method for creating text embeddings is to create bag-of-words representations. Another approach is to use a word embedding technique like global vectors for word representations (GloVe) or word2vec to represent each word and subsequently to represent text. For our analysis, we are focusing on the bag-of-word representations and GloVe word embeddings. Subsequently, for mapping of the vector representations of customer reviews to the sentiment space, we used various classification techniques like naïve bayes, logistic regression and SVM (with different kernels). We have also implemented simple and deep neural networks for this task.

Definitions, Assumptions, and Data Preprocessing

We selected four product categories of Amazon product review data to conduct our analysis. Specifically, they are (i) books, (ii) electronics, (iii) movies and TV, and (iv) home and kitchen. Among the given data fields, we used `overall` field to create labels and `reviewText` field as input field. All other data fields were dropped for this analysis. To create labels, customer reviews with 5 star ratings were labeled *positive* reviews and reviews with 1 or 2 star rating were labeled *negative* reviews. We did not include the 3 and 4 star reviews to our analysis because no clear sentiment can be predicted for them based on the rating. Also, we believe that once we train our system on these extreme scenarios, we can use the same classifier to classify 3 and 4 star reviews as either *positive* or *negative* reviews.

To have a better understanding of our datasets, a distribution of the ratings from three categories is shown in Figure 1. The books category is not included because generating the graphics kept crashing our computers; however, based on random sampling the distribution is approximately the same as the other categories.

As shown above, 50-60% of the reviews received 5 stars (positive) across all the categories and 10-15% of the reviews received 1 or 2 stars (negative). This class imbalance was noted and will be discussed later. For classification in each category, we used a balanced training set with a total of 25,000 observations. The validation dataset (10,000 observations) was randomly sampled from the original dataset to preserve the distribution of the two classes.

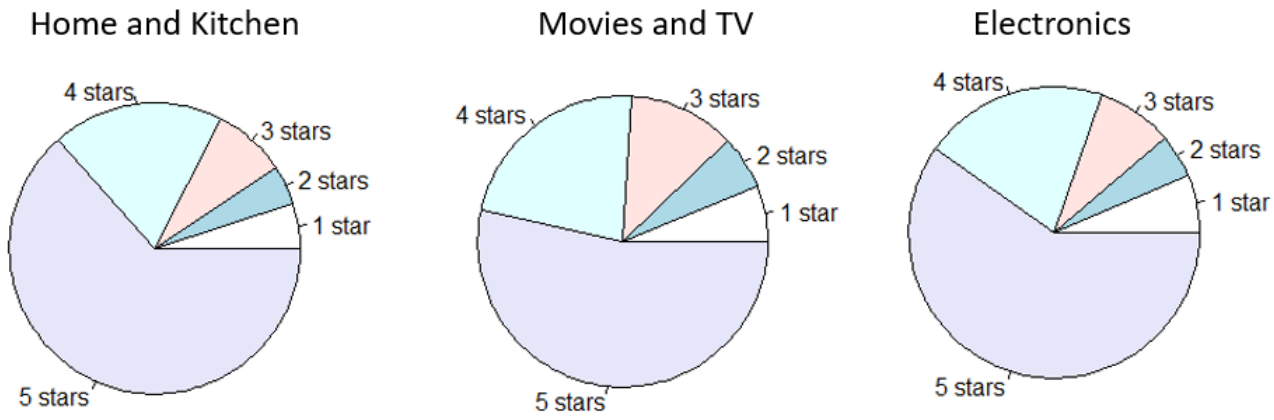


Figure 1: Distribution of review ratings across datasets. The numbers of the reviews of home and kitchen, movies and TV, and electronics are 551,682, 1,697,533, and 1,689,188 respectively.

Text Embeddings

For sentiment analysis, we first need to pre-process the texts into some vectors that can be easily analyzed. Here we used two different text embeddings: bag-of-words (BoW) and Global Vectors for Word Representations (GLoVe).

Bag of Words

This is one of the most commonly used methods for text embeddings. The central idea is to represent the text as one-hot vector encodings based on the most common words in the corpus. For this representation, we first tokenize the review and remove the punctuations and stopwords (stopwords are the commonly used words which do not contain any significance for the task at hand like "am", "was", "the"). Following this, we lemmatize the tokens i.e group together the inflected form of words so that they are treated as single word (ex. "reading" is lemmatized to "read"). After this we choose the top 200 most frequent tokens and create one-hot vector encoding for the text based on the presence of the most frequent tokens in the text. An example of this process is shown below.

Steps	**Example Review: I love books written by Dan Brown**
Tokenize the review	["I", "love", "books", "written", "by", "Dan", "Brown"]
Convert tokens to lowercase	["i", "love", "books", "written", "by", "dan", "brown"]
Remove punctuation and stopwords	["love", "books", "written", "dan", "brown"]
Lemmatizing	["love", "book", "write", "dan", "brown"]
Choose top 200 most frequent words	["love", "book"]
Convert review to one-hot vector representation of the most frequent words	[1,1,0,...]

Table 2: Data preprocessing procedures using bag-of-words.

The issue with bag of words is that we are throwing a lot of information which is not captured

in the most frequent words here. Another issue is that it doesnt consider the semantics of the words. Hence, we tried another text embedding to resolve these issues.

Global Vectors for Word Representations (GloVe)

GloVe stands for Global Vectors for Word Representation. This is the unsupervised learning algorithm which create encoding for words based on the context in which they are used.

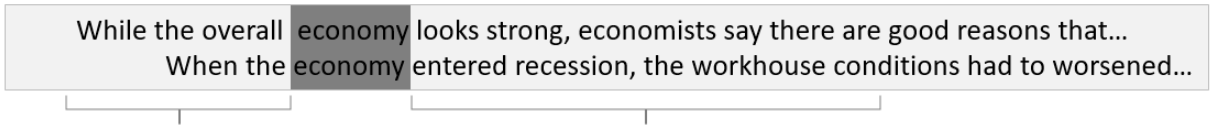


Figure 2: An example illustrating the main idea of GLoVe. The words around "economy" tell us more about "economy" than its one-hot vector representation like bag-of-words.

For our analysis, we used the pretrained GloVe vectors of 100-dimensions trained on Wikipedia 2014 and Gigaword5 (from `https://nlp.stanford.edu/projects/glove/`). After getting encoding for words, we simply sum up the vectors for all the words in the review to create text embeddings.

Exploratory Data Analysis

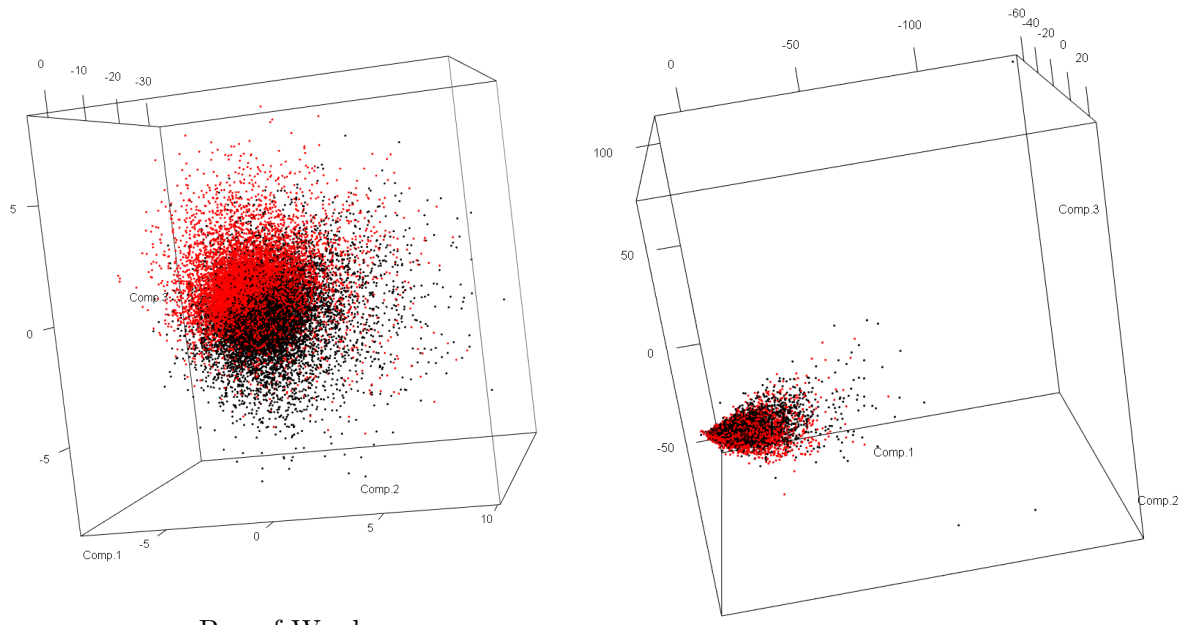
Principal Components Analysis

We first explored our data using principal components analysis (PCA). The first three principal components were used to plot the figures below. Because we are only using 200 features for the bag-of-words, we wanted to make sure that there was some separation between positive and negative reviews. Because this is purely a visual inspection, we have omitted the cumulative proportion of variance explained.

As the PCA analysis shows from Figures 3 to 6, we can identify separation for both bag-of-words and the GloVe word embeddings. The category that had least separation was electronics. This makes sense, because these are the reviews that tend to be more complex. For example, positive reviews may often contain reservations, and negative reviews might comment on a few redeeming qualities or components that the electronic had. Overall, however, there is decent separation between the classes even with just 200 features.

t-distributed Stochastic Neighbor Embedding

One drawback of using PCA with text data is that the dataset has high dimensionality. Thus, the first three components can only account for about 12% of variance for bag-of-words data and 60% of variance for GloVe data across all product categories. t-distributed stochastic neighbor embedding (t-SNE), developed by Geoffrey Hinton and Laurens van der Maaten, is a nonlinear dimensionality



Bag-of-Words

GloVe

Figure 3: PCA in 3 Dimensions with **Books** product reviews. Red points represent positive reviews and black points represent negative reviews.

reduction technique that is adept at embedding high dimensional data into a very low dimensional space, such as two or three dimensions. More formally, t-SNE computes probabilities p_{ij} that are proportional to the similarity of objects x_i and x_j as follows:

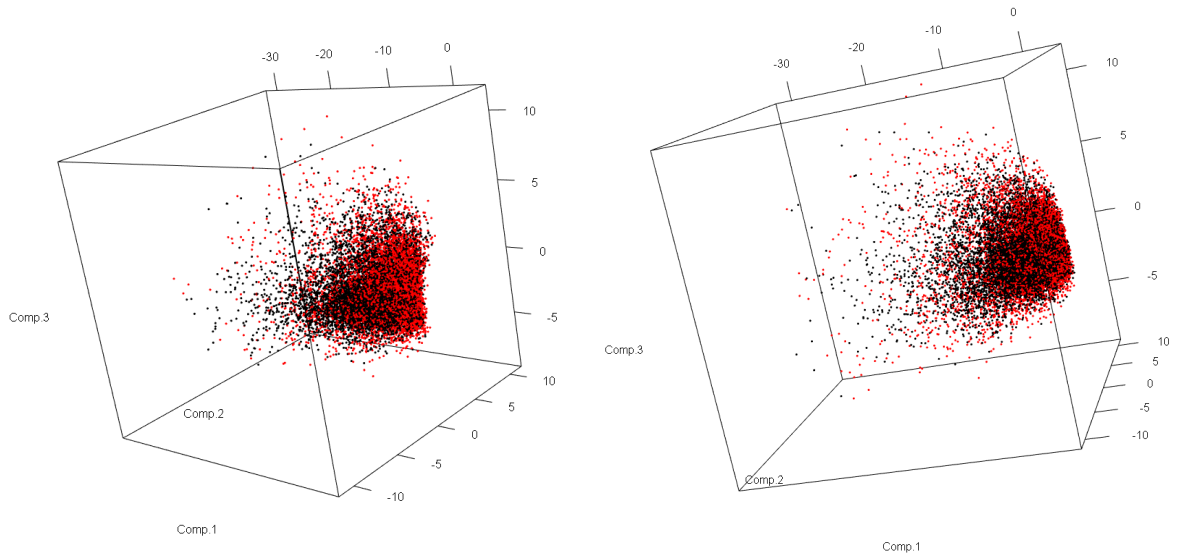
$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$

In other words, this is the conditional probability that x_i would pick x_j as its neighbor if the neighbors were picked in proportion to their probability density under a Gaussian distribution centered at x_i.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

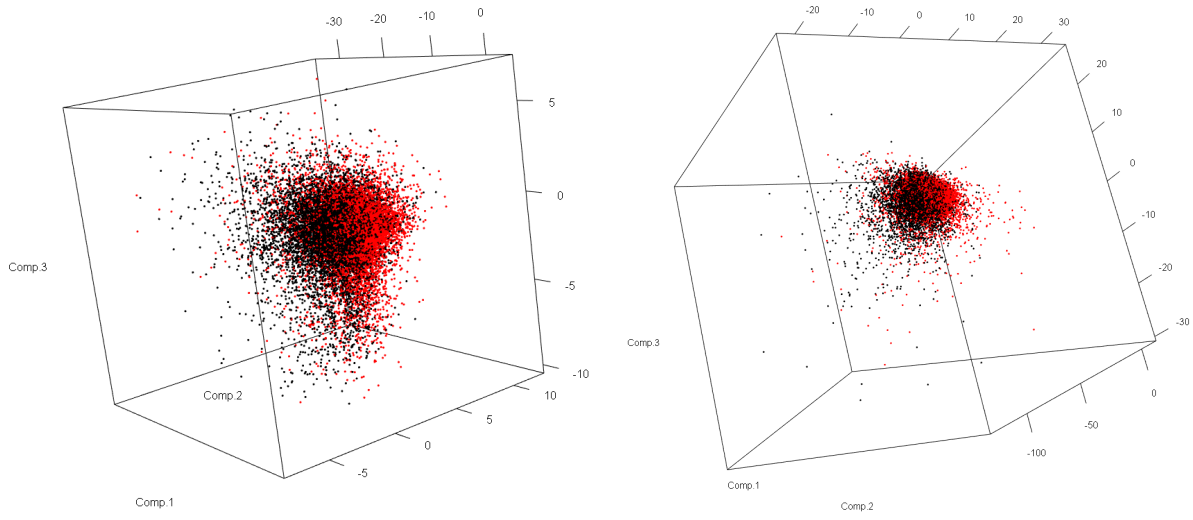
The bandwidth σ_i^2 is set according to a perplexity parameter; this bandwidth then changes according to the density of the data. t-SNE then constructs a d-dimensional map Q that reflects the similarities p_{ij} as much as possible. The Kullback-Leibler divergence is used to calculate the distance between distribution Q and distribution P. Some of the results are shown below for both bag-of-words and GloVe.

As we can tell in Figures 7 to 10, there does seem to be various groupings in our results but they do not seem to be splitting along negative and positive reviews, although some groupings have far more positive reviews (in red) than negative reviews (in black). The algorithm might be



Bag-of-Words GloVe

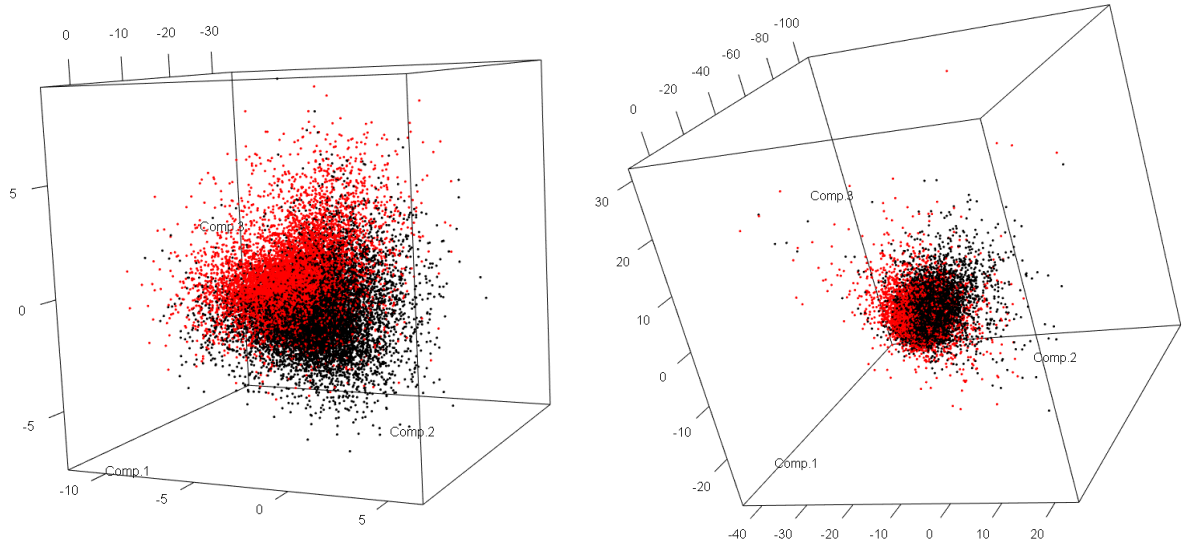
Figure 4: PCA in 3 Dimensions with **Electronics** product reviews. Red points represent positive reviews and black points represent negative reviews.



Bag-of-Words GloVe

Figure 5: PCA in 3 Dimensions with **Home & Kitchen** product reviews. Red points represent positive reviews and black points represent negative reviews.

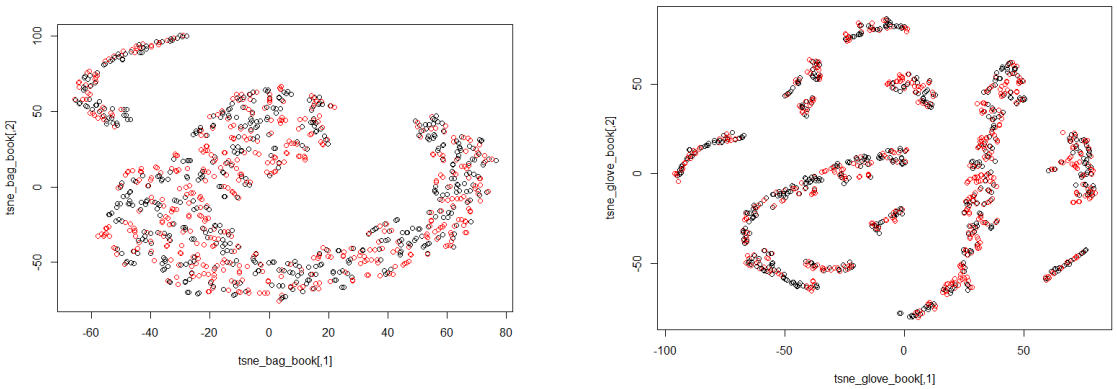
splitting along different types of negative and positive reviews. We have also used a relatively low perplexity parameter and only 1,000 reviews (randomly sampled from our training set) to create



Bag-of-Words GloVe

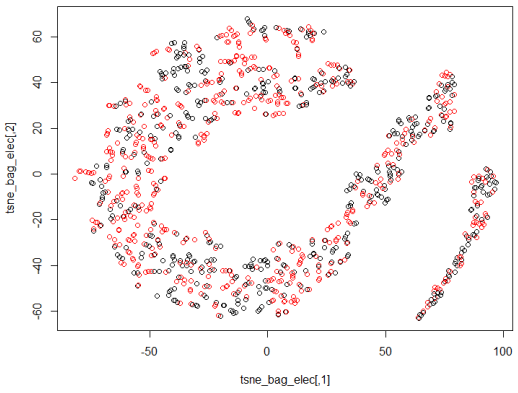
Figure 6: PCA in 3 Dimensions with **Movies & Television** product reviews. Red points represent positive reviews and black points represent negative reviews.

these graphics. The most important take-away from this exploratory data analysis is that the GloVe datasets are able to distinguish different types of Amazon product reviews more readily than the bag-of-words datasets.

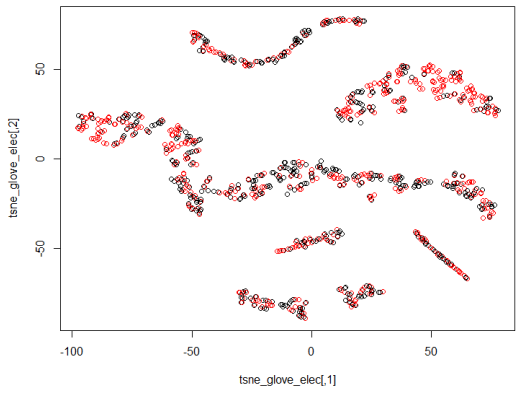


Bag-of-Words GloVe

Figure 7: t-SNE in 3 Dimensions with **Books** product reviews. Red points represent positive reviews and black points represent negative reviews.

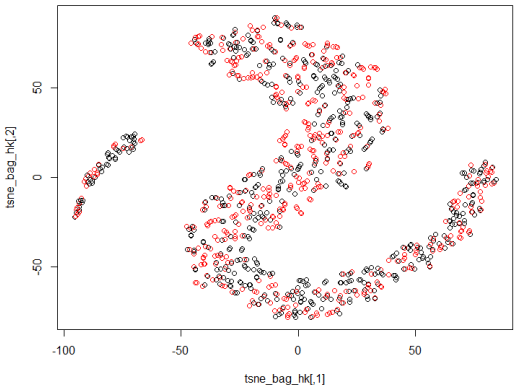


Bag-of-Words

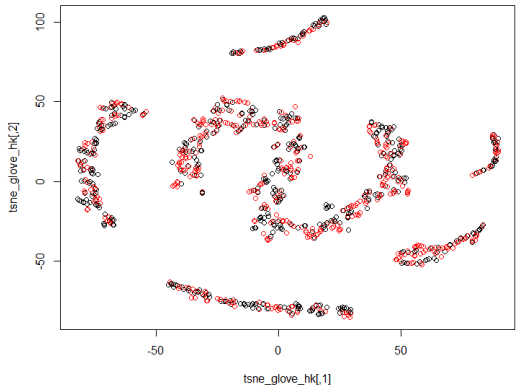


GloVe

Figure 8: t-SNE in 3 Dimensions with **Electronics** product reviews. Red points represent positive reviews and black points represent negative reviews.



Bag-of-Words

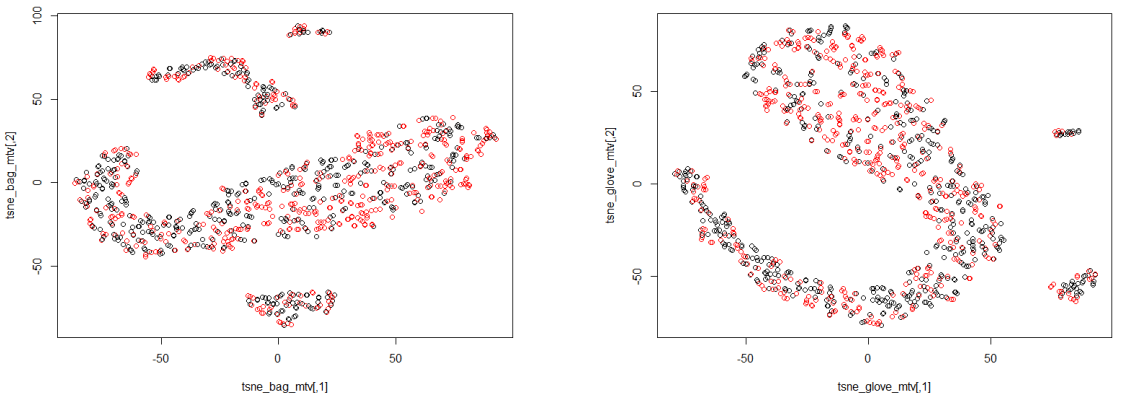


GloVe

Figure 9: t-SNE in 3 Dimensions with **Home & Kitchen** product reviews. Red points represent positive reviews and black points represent negative reviews.

Methods and Parameter Searches

The classification methods we used are logistic regression, linear discriminant analysis (LDA), naïve Bayes, Gaussian naïve Bayes, linear support vector machine (L-SVM), Gaussian-kernel support vector machine (RBF-SVM), polynomial-kernel support vector machine (Poly-SVM), and deep neural network.



Bag-of-Words GloVe

Figure 10: t-SNE in 3 Dimensions with **Movies & Television** product reviews. Red points represent positive reviews and black points represent negative reviews.

For logistic regression, L-SVM, RBF-SVM and Poly-SVM, there are adjustable parameters in models, so we searched for best parameters by doing 5-fold stratified cross-validation on the training set. We used mean test accuracies to evaluate quality of parameter settings. For logistic regression and linear SVM, we could adjust regularization strength. We did a search for inverse of regularization strength C from 10^{-2} to 10^{2} in logarithmic space.

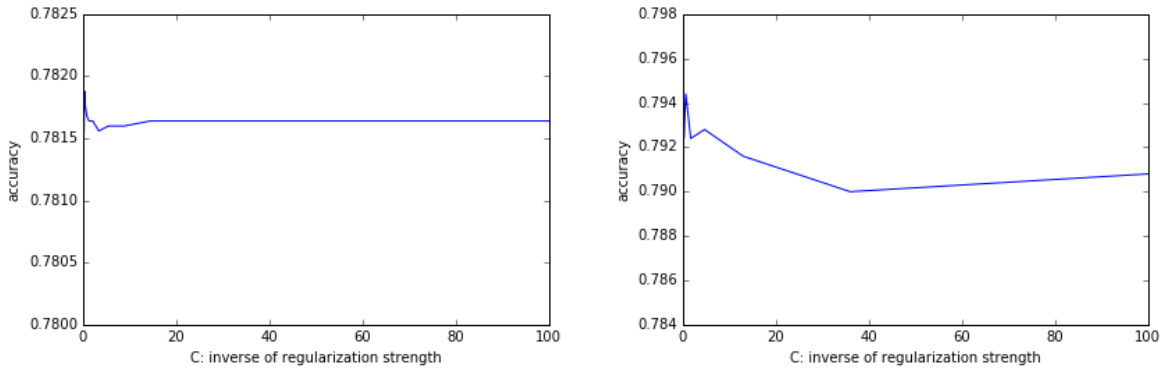


Figure 11: An example of parameter searches for logistic regression (left) and linear SVM (right). Preprocessed using bag-of-words in books product reviews.

In addition to parameter of regularization, RBF-SVM has another adjustable parameter γ, which defines how far the influence of a single training example reaches. Low values of γ means the influence range is large, while high values of γ means the influence range is small.

We did a grid search to find best combination of C and γ. Both parameters are searched in logarithmic space. C is searched in a range from a 10^{-2} to 10^{2} and γ from $10^{-3.5}$ to $10^{-0.5}$, which is determined by the results of a pre-search in a larger search space.

Except for the parameter of regularization, Poly-SVM has two more adjustable parameters, degree of polynomials d and nonhomogeneous coefficient $coef0$. The search must be done in 3 dimensions, but we only show the result in 2D with fixed best C here in the example in the figure below.

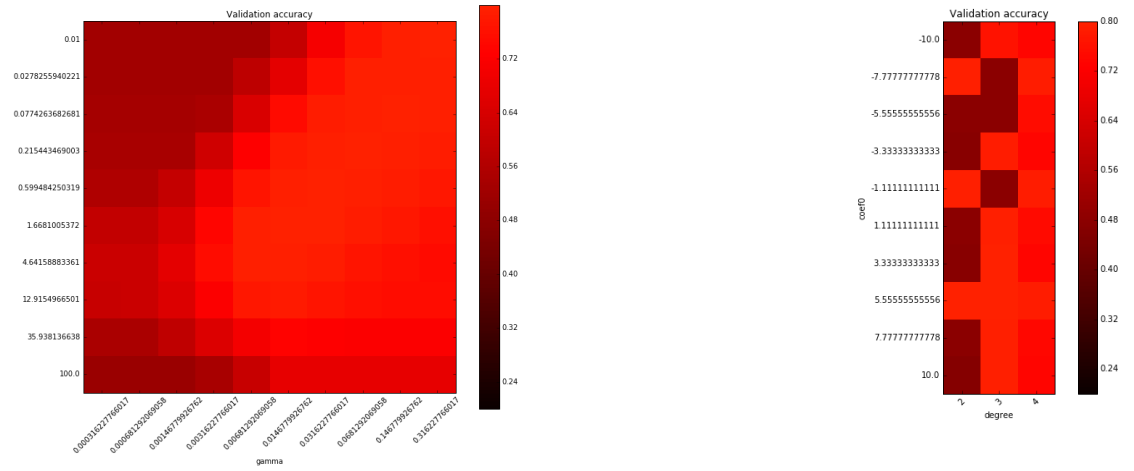


Figure 12: An example of parameter searches for Gaussian SVM and polynomial SVM. For polynomial SVM, search was done in 3D including penalty C, and the best C was fixed to show the 2D heat map. Preprocessed using bag-of-words in books product reviews.

Results and Discussion

Question 1: Do certain classifiers predict sentiment of Amazon reviews more accurately than others?

A table of the accuracies of different classifiers can be found in Tables 3 and 4.

For most methods, GloVe datasets perform better than bag-of-words, which is expected since GloVe also includes the context of the words compared to bag-of-words. The only exception is for naïve Bayes—bag-of-words datasets perform better in this case. This is also expected, because the naïve Bayes algorithm performs well with discrete data.

Question 2: Does the best classifier change from product type to product type?

The best classifier does not change much from product type to product type. In three of the categories, logistic regression and linear discriminant analysis performed best, while for Home &

Classifier	Books	Electronics	Movies & TV	Home & Kitchen
LoR (BoW)	0.784	0.772	0.796	0.779
LoR (GloVe)	0.817	0.787	0.824	0.794
LDA (BoW)	0.783	0.773	0.798	0.775
LDA (GloVe)	0.813	0.791	0.832	0.8
Poly-SVM (BoW)	0.761	0.779	0.758	0.76
Poly-SVM (GloVe)	0.813	0.786	0.804	0.811
L-SVM (BoW)	0.756	0.759	0.774	0.749
L-SVM (GloVe)	0.807	0.785	0.812	0.78

Table 3: Accuracies of the different classifiers, excluding the naïve Bayes classifiers.

Classifier	Books	Electronics	Movies & TV	Home & Kitchen
Naive Bayes (BoW)	0.743	0.749	0.737	0.778
Naive Bayes (GloVe)	0.690	0.657	0.698	0.698

Table 4: Accuracies of the naïve Bayes classifiers. The bag-of-words dataset performed better in this case.

Kitchen the polynomial SVM is the best classifier followed by linear discriminant analysis. We also noted that Movies & TV has the best accuracy among the four product categories, and electronics has the worst accuracy. But overall, the best classifier does not change from product type to product type. LDA was one of the top two classifiers in all product categories. Moreover, in all cases, training the product-specific classifiers GloVe datasets largely outperformed the bag-of-words datasets.

	Books	Electronics	Movies & TV	Home & Kitchen
Method 1 (accuracy)	LoR (0.8174)	LoR (0.791)	LDA (0.832)	Poly-SVM (0.811)
Method 2 (accuracy)	LDA (0.813)	LDA (0.787)	LoR (0.824)	LDA (0.800)

Table 5: Best two classifiers from each product category based on accuracy. All results are from GLoVe datasets.

Question 3: Do product-specific classifiers that learn from only that product category perform better than classifiers that learn from all product categories?

This was one of the main questions we wanted to answer with our project: did a product-specific sentiment classifier classify sentiment more accurately than a product-generic sentiment classifier? After answering Question 2, which showed that the top two classifiers for each product was approximately the same, we decided to create a generic classifier—that is, a classifier that would train on Amazon product reviews from all four product categories. Since LDA with GloVe datasets was largely the dominant classifier in our previous analysis, we trained an LDA classifier using all four product categories. The training size for the general classifier is the same as the training size for each of the product-specific classifiers (25,000 observations), which was created from sampling

across all four product training sets. We then applied this general classifier on different testing sets from four categories. The accuracies are shown below in Table 6.

	Books	**Electronics**	**Movies & TV**	**Home & Kitchen**
Product-Specific Classifier	0.813	0.787	0.823	0.800
General Classifier	0.84	0.814	0.805	0.851

Table 6: Accuracy of product-specific classifiers versus the general classifier. All results are from using an LDA classifier trained on GloVe datasets.

As we can see from the results, the general classifier, trained across all product reviews, outperformed the product-specific classifier except for the Movies & TV product category. We think this may be the case because reviews for movies and television products tend to employ a lexicon distinct from reviews in other product categories; thus, those peculiarities might have washed out when creating a generic classifier. Furthermore, recall from our initial PCA analysis that Movies & TV had a strong separation between the negative and positive reviews. This again hints at some of the special properties of product reviews in this category that strongly separates the positive and negative reviews; this special property may be washed out when using a generic classifier. It is also important to note that the difference between the product-specific classifier and the general classifier is not large.

But what is more interesting is that this implies that "borrowing" words that might reveal sentiment from other categories can improve classification. This may make sense in some scenarios. For example, electronics product reviewers may favor a certain set of negative words, such as, for example, $\{$"bad'', "$slow''$, "lag'', "$low - value''\}$. Book product reviewers might favor another set of negative words, such as, for example, $\{$"$tedious''$, "$boring''$, "$poor''$, "$waste''\}$. These negative words are, of course, not category-exclusive—an electronic entertainment product might be "boring", a product might be "poor"ly built, etc. Likewise, a book can be "bad" or "slow" as well. However, because these words are not favored when looking at only one product category, they do not make it as features when creating product-specific training sets. When the reviews are pooled together, however, these negative words get pooled together, so the negative word pool becomes $\{$"bad'', "$slow''$, "lag'', "$low - value''$, "$tedious''$, "$boring''$, "$poor''$, "$waste''\}$. This could plausibly explain the higher accuracy using a general product classifier.

To more formally test this, we train an LDA classifier on all categories except the home & kitchen product reviews. Again, we created a training set of 25,000 observations by sampling across the three product training sets. We then applied this classifier based on the 3 other categories on the home & kitchen testing set. The accuracies are shown in Table 7.

Our findings here seem to confirm the "borrowing" theory from before. Without even using any Home & Kitchen reviews, we were able to have a higher prediction accuracy compared to the Home & Kitchen-specific LDA classifier using a classifier trained on product reviews from Books, Electronics, and Movies & TV. This is most likely because the classifier is pooling together all the favored positive and negative words from three categories, rather than just a single category. Notice

	Home & Kitchen
Home & Kitchen-Specific Classifier	0.800
General Classifier	0.851
Classifier Based on Other 3 Product Categories	0.840

Table 7: Accuracy of the home & kitchen-specific classifier, the general classifier (trained on all four product categories), and the classifier based on the other three product categories.

that the general classifier still outperforms the classifier based on the other 3 product categories, which makes sense according to our "borrowing" theory.

Problems with Approach

Class Imbalance

As shown in Figure 1, the positive and negative reviews have an about 5:1 ratio, implying an imbalance in the two classes. In addition to train our classifiers with a balanced training set to alleviate this issue, we also used the F1-score and AUC to evaluate performance under this class imbalanced condition (shown below in Table 8 and Figure 13). We found that the classifiers with good accuracies also has good F1-score and AUC. For the sake of saving space, we only show the results for the logistic regression classifier for Books using the GloVe dataset; the results for the other top classifiers using the GloVe dataset was similar.

Logistic Regression (GloVE)	**Results**
Accuracy	0.8174
F1-score	0.8866
AUC	0.8698

Table 8: Accuracy, F1-score, and AUC of the logistic regression (GLoVe dataset) for books.

SVM Takes A Long Time to Run

Another problem is that SVM takes a very long time to run, so that we cannot feasibly gridsearch even over a modest parameter space. Computational accommodations and smarter sampling techniques need to be employed if this project is to be conducted on a larger scale. In our results above, we randomly sampled 2500 documents in our training set and used those to create our parameters. Also, we only used the 200 most tokenized words from our corpus in data preprocessing. We would expect a better performance if more features were used, but on the other hand, our PCA analysis indicates that we get decent separation even with only 200 features.

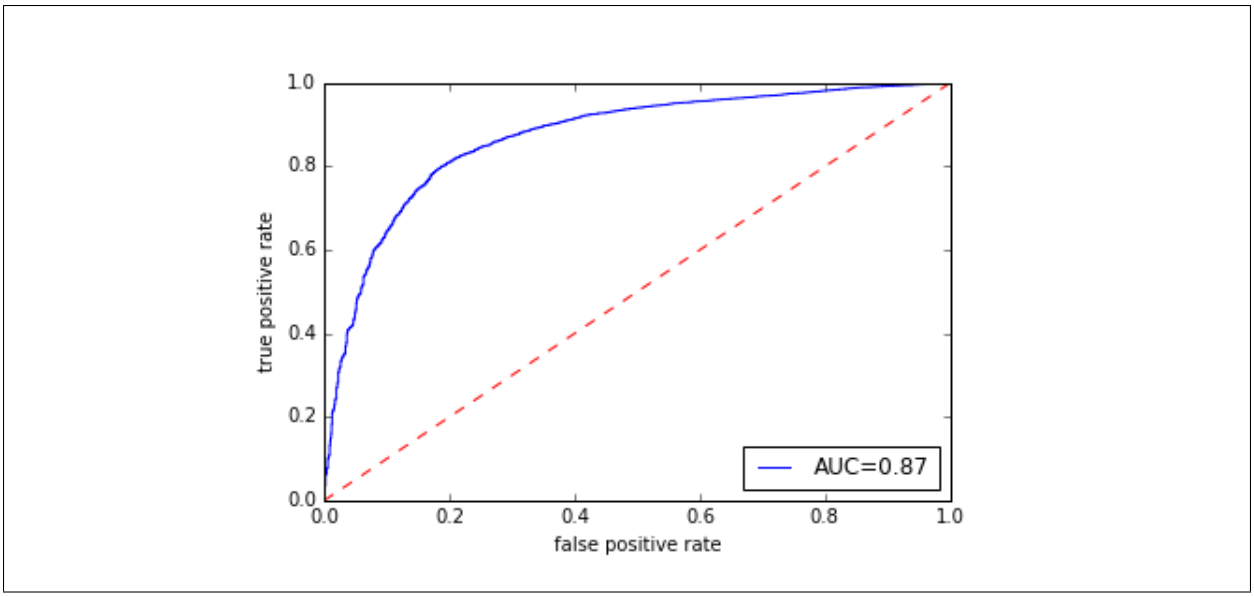


Figure 13: ROC curve of the logistic regression (GLoVe dataset) for books. AUC also shown.

Conclusion

This project was interested in classifying Amazon product reviews. There were three main questions we sought to answer.

1. Do certain classifiers predict sentiment of Amazon reviews more accurately than others?
2. Does the best classifier change from product type to product type?
3. Do product-specific classifiers that learn from only that product category perform better than classifiers that learn from all product categories?

For Question 1, we found that certain classifiers do predict sentiment of Amazon reviews more accurately than others. Moreover, we found that preprocessing the data using GloVe to obtain vector representations for words allowed the classifiers to be more accurate compared to the simple bag-of-words training sets. For Question 2, we found that the best classifiers were largely the same from product type to product type. Again, GloVe training sets provided more accurate rates of classification. In particular, linear discriminant analysis (LDA) and logistic regression seemed to do the best across categories. For Question 3, we found that the product-specific classifiers that learn only from their respective product categories do not perform better than classifiers that learn from all product categories. We argue that this makes sense because even if certain product categories favor certain types of negative or positive words, these words are not category-exclusive and therefore pooling this information allows the classifier to be more accurate. We tested this by showing that a classifier trained on Books, Electronics, and Movies & TV product reviews could more accurately classify Home & Kitchen product reviews compared to the Home & Kitchen-specific product reviews.

This project is very exciting and there are potentially many other avenues to take this project down. One example is to use more of the dataset in our analysis. The dataset includes data on how helpful or unhelpful the review is. We could potentially use this information to weight certain product reviews. Extremely popular negative or positive reviews could plausibly influence how future reviews are written, meaning that the lexicon employed in these popular reviews could trickle down to future reviews. Other ways of expanding and improving this project could include using more features (beyond 200), using larger training sets, including time-trends (such as taking into account popular words in certain time periods), and so on. Another technique we are trying to use is deep neural networks for classification; a preliminary attempt at this methodology is detailed in our final section.

Deep Neural Networks for Sentiment Classification

We tried to apply deep neural network to predict sentiments of reviews. Specifically, we used deep multi-layer neural network (MLP). MLP consists of several fully-connected layers, in which every nodes in one layer (except input layer) is connected to all nodes in the previous layer. The value of a node in hidden and output layers is the weighted sum of nodes in the previous layer followed by a transformation with an activation function. We defined a cross entropy loss for our binary classification problem and optimized by adaptive gradient descent methods. See Figure 14 for a visual definition.

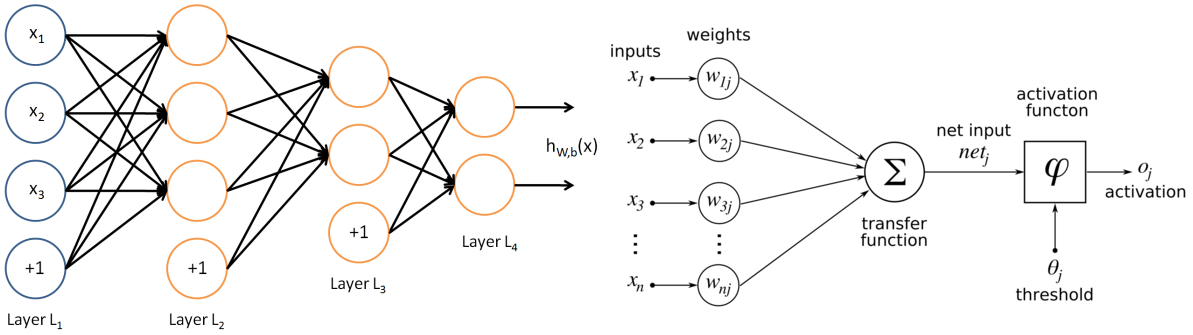


Figure 14: Diagram of structure of MLP and activation function

To improve the sentiment classification, we implemented the deep neural networks for GloVe text embeddings using tensorflow for "books" product category only. For baseline, we first trained plain vanilla neural networks without any hidden layers. We monitored the training and test errors and run the nets for 500 epochs. This improved the accuracy over the traditional techniques by approximately 4 basis points (see Figure 15).

To further improve the classification model, we implemented deep neural networks with 2 hidden layers containing 30 and 10 neurons respectively. We also added the dropout technique, with dropout parameter as 0.5, which is known to improve the accuracy of neural networks (We also tuned

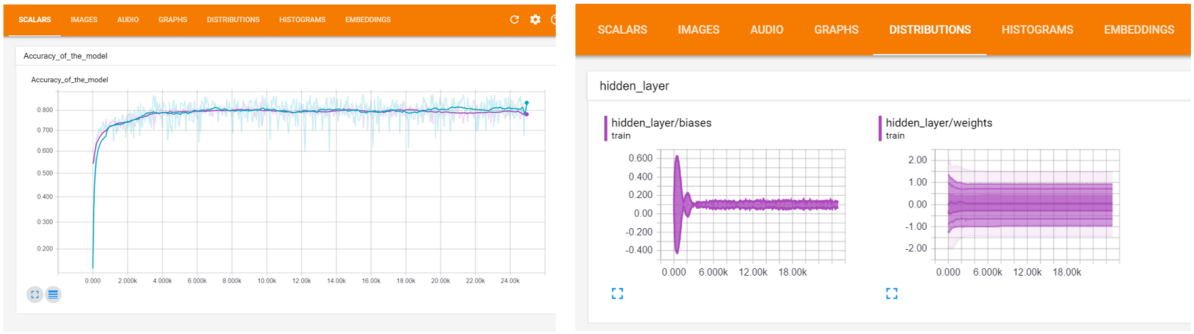


Figure 15: (Left) Shows the training and test accuracy for 500 epochs (Right) Shows the progression of weights and biases. Both figures are for GloVe text embedding for Books category.

the learning rate by implementing exponential decay of the learning rate at 0.7. This improved the accuracy over the plain vanilla neural networks by about 3 basis points (Srivastava et al.) (i.e 88.08% accuracy). The high accuracy of deep neural networks means that future work on this project would focus primarily on deep (and convolutional) neural networks (see Figures 16 and 17).

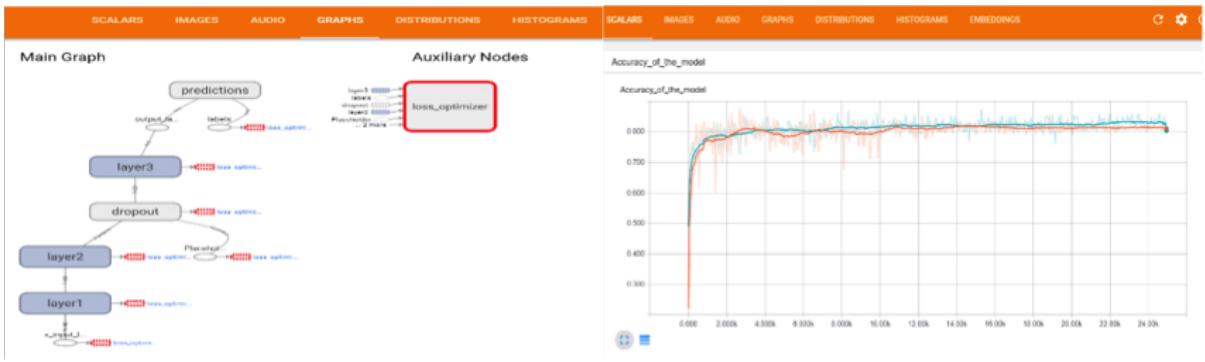


Figure 16: (Left) Shows the computational graph for the deep neural nets (Right) Shows the training and test accuracy for 500 epochs.

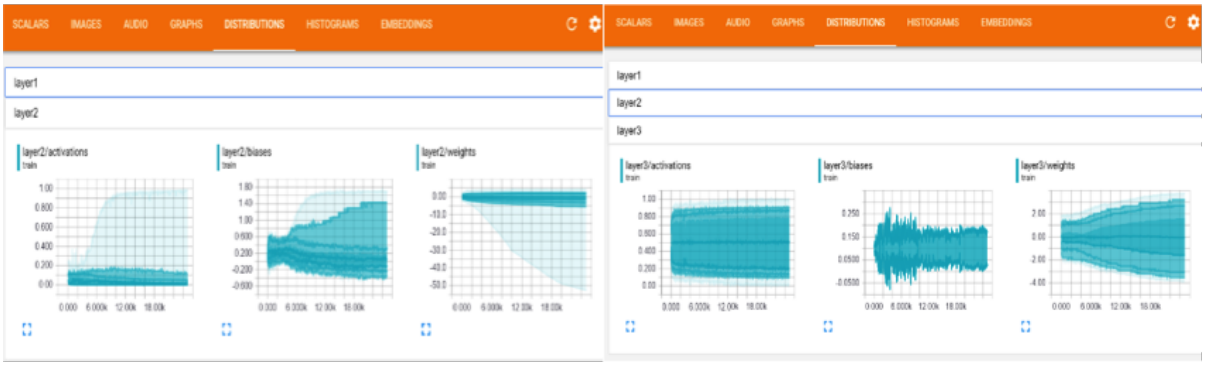


Figure 17: Shows the progression of weights and biases across for the hidden layers with time. All the figures are for GloVe text embedding for Books category.

References

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning.* New York: Springer.

McAuley, Julian. 2016. "Amazon Product Data." `http://jmcauley.ucsd.edu/data/amazon/`.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. "GloVe: Global Vectors for Word Representation."

van der Maaten, Laurens, and Geoffrey Hinton. 2008. "Visualizing Data using t-SNE." *Journal of Machine Learning Research* 9: 2579-2605.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15: 1929-1958.