Restaurant Recommendation System Based on Yelp Data:
W266 Final Report

Aaron Yuen, Alexander Herring, Charlene Chen

Abstract

Recommendation systems play an important role in driving continual engagement to sites like Amazon, Netflix, and Yelp. However, these algorithms rarely explore the nuances in review text by users, and instead rely on traditional features like review ratings and engagement signals. In this paper, we explore the use of a recurrent neural network (RNN) with LSTM units to generate star rating predictions and recommendations for Yelp restaurants, primarily leveraging Yelp restaurant review data. Additionally, as an experimental approach, we explore a recommendation system using collaborative filtering and LDA topic modeling. We detail how our RNN with LSTM units model dramatically outperforms a Gaussian Naive Bayes baseline given that RNNs can determine relationships between words. We also explore the kinds of systematic errors generated by the model through a thorough error analysis. An interesting finding from our error analysis showed that our RNN with LSTM units model performed poorly in review text with sections that had polarizing sentiments. We also analyze error examples from our experimental recommendation system that uses collaborative filtering and LDA topic modeling to emphasize the importance of topic pool quality in generating good recommendations.

1. Introduction

In e-commerce businesses, such as Amazon, Yelp, TripAdvisor, and Netflix, recommendation systems play an important role in driving sales, attracting new users, as well as maintaining existing users. Users with similar behaviors in writing reviews would likely have similar tastes of products, such as restaurants, hotels, and movies, that can be leveraged for providing better recommendations of restaurant to users. Recommendation systems started to emerge in the 1990's, focusing on star rating structure [2]. Although the most popular approach when building recommendation system is to predict *ratings*, an alternative way can be to predict relative preference *ranking* of items [11]. In general, according to how the recommendations are made, there are three approaches: a *content-based* method that makes recommendations based on items that users preferred in the past; a *collaborative* method that makes recommendations based on users that have shown similar tastes in the past; additionally, a *hybrid* method that is the combination of both [2]. There is a tradition of leveraging star rating as the primary feature to generate restaurant recommendations for a given user [1]. This paper explores text reviews as the primary input for a recommendation system.

This project aims to primarily use language-based features such as review text to understand the user's preference and then generate personalized restaurant recommendations to specific users. In this project, we train a recurrent neural network (RNN) with LSTM units to generate a recommendation system of ratings for Yelp restaurants [7][8]. We also explore an experimental topic modeling approach using collaborative filtering with LDA topic modeling for generating recommendations of rankings without using neural networks or word embeddings [5][6].

2. Background

Nowadays, web and online products penetrate into almost all aspects of our lives, from watching videos and listening to music, to purchasing products and choosing restaurants or hotels. Meanwhile, users are presented more and more options on what to watch, listen, purchase, or where to visit. Users continue to benefit from automatic recommendation systems that proactively suggest content for them based on their interests. For ecommerce and online firms, recommendation systems also play very important roles in generating revenue, retaining current users, obtaining new users, and even finding business opportunities.

In general, the input of a recommendation system is the data of users and their reviews. These inputs often involve star ratings, text reviews, other non-text based features such as usefulness and funny counts of the review, and idiosyncratic attributes of users and specific items or places being reviewed. The output of the system can be predicted star ratings or preference ranking of items for specific user.

Deep learning structures have provided revolutionary advances in attacking natural language processing problems. The application of deep learning in recommendation systems with text-based features shows great potential via a better understanding of item characteristics and user preferences. In particular, Convolution Neural Networks (CNN) are often used for text-based feature extraction, while RNNs are powerful for dealing with the with the temporal dynamics of ratings and sequential features in recommender systems [14]. In this project, we make use of an RNN structure with LSTM units to train our primary model. A benefit of using LSTM units is that they help solve the vanishing gradient and long memory problem of other RNNs. They do this by adding

special units with memory cells and a set of gates that allow them to learn longer-term dependencies. [13]

Likewise, collaborative filtering systems are one of the most used approaches for building recommender systems. They try to predict the recommendation of items for one user based on the items rated by other users [2]. However, most collaborative filtering systems use non-text based features to train the model, such as star rating. To leverage user reviews and better capture the nuances in user preferences, this paper explores an experimental collaborative filtering model with text-based features [5]. Additionally, in order to generate text-based features, we use a Latent Dirichlet Allocation (LDA) method to create useful topic distributions from reviews [6].

3. Methods

3.1 Dataset and Data Processing

We trained and evaluated our models by using the publicly available Yelp academic dataset. It includes a database of businesses, users, check-ins, tips and photos. Most importantly, the business reviews include the full text written by users, alongside star ratings (integer values from 1 to 5), and can be traced back to individual users and businesses.

The raw Yelp dataset consists of 2.6M business reviews, 51k businesses, and 367k reviews. For pre-processing, non-restaurant businesses were filtered out, as we want to scope to a specific domain. In particular, through earlier exploratory data analyses, reviews on non-restaurant businesses tend to cover a wide spectrum of topics specific to the business domain. And, given there were >100 unique business categories (with "restaurant" having the highest number of reviews), we feel a more generalized recommendation system that recommends all businesses would not perform as well.

Users with five or less reviews were also filtered out, since users with little review text will likely lead to low quality or meaningless restaurant recommendations. As a result of the filtering, the final dataset consists of 1.5M reviews for the training set, and 373k reviews for the test set.

3.2 Gaussian Naive Bayes (Baseline)

Sentiment analysis is widely used in predicting user preferences and building recommendation systems [3]. In sentiment analysis based recommendation systems, machine learning algorithms like Naive Bayes or Support Vector Machine (SVM) can be trained to predict the overall opinion of products across many users [4]. This project uses a Gaussian Naive Bayes classifier as a baseline to predict the sentiment (integer star ratings from 1-5) for a restaurant to determine both whether or not to recommend this restaurant to a user, and how strongly to recommend a restaurant to a user. We decided to predict the numeric star rating for restaurants instead of using a binary recommend / not recommend prediction as a binary prediction system would not capture the strength of a recommendation. It also would limit visibility into where the model is making specific errors, such as on training data for restaurants rated at 4 stars versus 5 stars. In general, a rating of 1 star could be seen as "strongly do not recommend," 2 stars as "do not recommend", 3 as "neutral", 4 as "recommend," and 5 as "strongly recommend."

Our baseline Gaussian Naïve Bayes model serves to form a basic reference point against which to score our more advanced models, including our RNN with LSTM units model and our collaborative filtering recommendation system with LDA topic modeling. As a Naïve Bayes model, the baseline relies on a Bag-of-Words assumption, and does not consider relationships between words. We chose a Gaussian Naive Bayes approach over other kinds of Naive Bayes models, such as Multinomial Naive Bayes, primarily because it allows us to natively use pre-trained word vectors from the Stanford Global Vectors for Word Representation (GloVe) project [12]. These word vectors include positive and negative values, which will not work with Multinomial Naive Bayes models without normalization, but will work with Gaussian Naive Bayes models. Multinomial Naive Bayes models require the values of features to be constrained between 0 and 1, whereas Gaussian Naive Bayes models work well with a wider array of continuous values, including negative values. As we will discuss more in our RNN with LSTM units model discussion, we used GloVe word vectors based on a 6 billion token and 300-dimension vector dataset.

The methodology for running our Gaussian Naive Bayes baseline is somewhat simplistic, especially when compared with our RNN with LSTM units and our collaborative filtering recommendation system with LDA topic modeling. After performing tokenization and processing, and splitting our data into training and test datasets as described in section 3.1, "Dataset and Data Processing," we apply the GloVe word embeddings to the training and test datasets. We then train a Gaussian Naive Bayes model from scikit-learn on the training data and generate our star rating sentiment predictions on the test data. Overall, our Gaussian Naïve Bayes baseline did not perform as well as our RNN with LSTM units model. We will discuss the performance of this model further in the Evaluation Summary section (section 4.1) of this paper.

3.3 Recurrent Neural Network Model with LSTM Units (RNN with LSTM)

Recurrent Neural Networks (RNNs) are excellent at solving a wide variety of language tasks. Unlike Bag-of-Words models, such as Naïve Bayes, RNNs can determine relationships between words. However, RNN models are known to have problems dealing with long-range dependencies, such as text samples that are over 50 words long [7]. Long short-term memory (LSTM) based RNN networks help to alleviate this problem [8]. This makes them useful for the Yelp dataset as there is a proportionally small, yet still sizeable number of reviews over 300 words long (6.8%). As such we decided to build an RNN model with LSTM units.

We designed our model using Keras on top of a TensorFlow backend [17]. We utilized pre-trained GloVe word vectors to form our embedding layer. We specifically used the "Wikipedia 2014 + Gigaword 5" dataset that includes 6 billion tokens, a vocab of 400 thousand, and 300 dimensions. As we will discuss in our Experiments section (4.5.1), the number of dimensions used in the word vectors had a sizeable impact on our model accuracy and loss scores. This is why we elected to utilize 300 dimensions over the other 6 billion token GloVe datasets that included 50, 100, and 200 dimensions. For tokenization, processing, and splitting our data into training and test subsets we used a very similar process to the Naive Bayes baseline to allow for easier comparability.

When deciding other parameters for our primary RNN with LSTM units model, we worked to balance the processing time for our models versus the effect of the parameters on model accuracy and loss scores. Furthermore, since iterations of the models take a very long time to run (our primary RNN with LSTM units model took over 21 hours on a Google Cloud Compute instance with 8 processors and 64GB of RAM), we relied on recommendations found from academic papers and literature for some of the initial parameters [10][17]. We decided to use a maximum sequence length of 400 due to the small number of reviews over this length (5.6%) Furthermore, as the Yelp dataset show bias towards higher ratings (67% of ratings were 4 or 5 stars), we balanced the dataset by down-sampling the training data to use the number of instances in the least populated class (this ended up being the 2-stars class that comprised 9.7% of all ratings). We additionally decided to use 100 memory neurons over smaller numbers, such as 50, to produce higher levels of accuracy without adding too much to our run time. This also influenced our decision to utilize a single LSTM layer. We found the number of epochs to have a pronounced effect on model accuracy (see Experiments in section 4.5.1), and capped it at 30. However, to limit run time we decided to have our model stop training if its accuracy failed to improve for 2 epochs in a row. In the end, our primary model tended to run for between 8-10 epochs.

Overall, our RNN with LSTM units model vastly outperformed our Gaussian Naïve Bayes baseline. We will discuss the performance of this model further in the Evaluation Summary section of this paper (4.1).

3.4 Collaborative Filtering (CF) Using Topic Modeling

Our experimental recommendation system is inspired by the method of collaborative filtering on topic modeling [5] where the topic distribution is generated by the LDA (Latent Dirichlet Allocation) method [6]. Collaborative filtering has been a classic way to generate recommendation systems [1]. Although it has some limitations, such as handling sparse datasets, and only being effective when preferences for a restaurant already exists in a dataset, it is a reliable method to work with when using topic modeling methods, such as LDA. What is interesting is that this approach does not leverage word embeddings nor deep neural networks. Hence this approach is considered experimental in the context of the W266 course.

For this approach, we derived topics with LDA trained from pre-processed Yelp restaurant review text. Stopwords and punctuations were removed before lemmatization. 100 topics were used in the final LDA model using the pre-processed text after experimenting with various number of topics. The LDA algorithm can be implemented as described in this paper [6], but we decided to leverage LDA from Scikit-learn instead.

The review-topic distribution is then derived from the trained LDA model. The review-topic distribution is essentially a matrix where for a given review, the question of "what is the topic?" is broken-down into a 100-topic probability vector. The appendix section details some examples of what the review-topic distribution looks like for some sample reviews (section 6.3). We then used this matrix to aggregate to a user-topic matrix so the distribution for a given user represents what topics the user is most likely interested in.

The user-topic matrix was a key component used for user-similarity in collaborative filtering. We leveraged this to find the k-nearest neighbors and use the highly-rated restaurants from these neighbors to be used as candidates for restaurant recommendations. Lastly, a recommendation ranking score is computed based on the similarity score to the nearest neighbors as well as the review rating of the restaurants reviewed by the nearest neighbors to rank the top "n" restaurants that we would recommend to the user.

4. Results and Discussion

4.1 Evaluation Summary

For comparing between the final models, accuracy, F1, and correlation metrics were used. For the Gaussian NB baseline and RNN-LSTM models, this is essentially a sentiment analysis-based evaluation. The accuracy and F1 scores rely on the predicted and actual ratings to be exactly correct. Even if the predicted and actual ratings are only one star apart, this is considered a "wrong prediction". The correlation metric helps capture how correlated the predicted and actual ratings are, so even if a model has very low accuracy, the scores may be well correlated if the scores were mostly one rating off (e.g. predicted 4 stars instead of 5 stars).

The primary RNN with LSTM units model dramatically outperformed our Gaussian Naive Bayes baseline in accuracy, F1 score and correlation. The RNN with LSTM model achieved an accuracy score of 62.7%, an F1 Score of 0.63, and a correlation of predicted test labels vs. actual test labels of 0.53. This compares to the Gaussian Naive Bayes baseline of 35.1% accuracy, an F1 Score of 0.36, and a correlation of 0.18. In addition, our collaborative filtering recommendation system with LDA topic modeling achieved an accuracy score of 37.0% and F1 Score of 0.43, which only slightly beat the baseline.

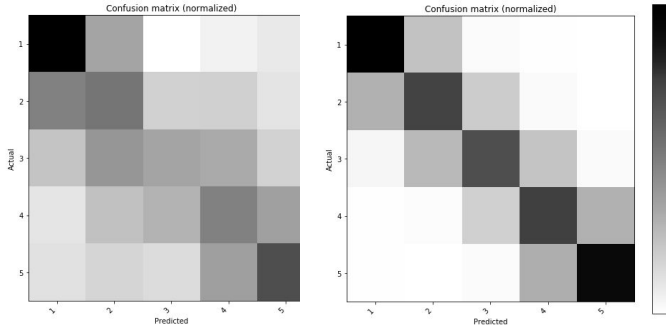
However, given the inherent structure of the CF model based on LDA, it is important to note that these scores are not a direct apples-to-apples comparison given that an "accurate" label requires the CF recommendation system to have recommended a restaurant for which the user has already given a review. As a result, the accuracy score can be considered as a lower-bound since some "inaccurate" labels could potentially be "accurate" if the recommendation system recommended the restaurant that a user has already reviewed.

4.1.1 Model Summary Table

Model	Accuracy	F1 Score	Correlation
Gaussian NB (Baseline)	35.1%	0.36	0.18
RNN-LSTM	62.7%	0.63	0.53
Collaborative Filtering w/ LDA Topic Model	37.0%**	0.43	0.39

** Not a direct comparison given that an "accurate" label requires the CF recommendation system to have recommended a restaurant that the user has already given a review on.

4.1.2 Confusion Matrices



Gaussian NB (Baseline) **RNN-LSTM**

4.2 Gaussian Naive Bayes (Baseline)

As seen in the Evaluation Summary (4.3), the Gaussian Naïve Bayes model did not perform very well in comparison to the RNN with LSTM units model and the collaborative filtering recommendation system with LDA topic modeling. When examining a confusion matrix of the actual vs. predicted values (shown in section 4.1.2 and enlarged in section 6.4), there was a great deal of spread. While the majority of the star ratings categories from 1-5 were most likely to predicted correctly (>50%), particular the extreme 1-star and 5-star ratings, the predictions were frequently incorrect. The 3-star rating was actually more likely to incorrectly be predicted as a 2-star rating (~30%) than correctly as a 3-star rating (~25%). There was also a notable chance (~10-15%) that the most extreme star ratings (1-star and 5-star, respectively) would be predicted as the opposite most extreme star rating (5-star and 1-star, respectively), creating a spread of ± 4 stars and flipping the sentiment of a review.

4.3 Recurrent Neural Network Model with LSTM Units

As discussed above in the Evaluation Summary (4.3), the RNN with LSTM units model dramatically outperformed both the Gaussian Naive Bayes baseline

and the collaborative filtering recommendation system with LDA topic modeling on accuracy, F1 score, and correlation. This is not surprising as RNNs can determine relationships between words, which can in turn be used to improve model accuracy and loss measurements.

When examining specific predictions from the model, we observed that any particular star rating from 1-5 was most likely to be predicted correctly vs. incorrectly. However, there was ~30-45% chance that the prediction of a test star rating was ± 1 stars above or below the actual star rating (with the prediction for 3-stars being closer to ~30% and the predictions for the extreme 1-star or 5-stars being closer to ~45%). In Example 1 in Appendix 6.2.1, the review received a predicted rating of 5 stars, but actually was given 4 stars. In reading the review text, it would be difficult for a human to judge whether this review text should be 4 or 5 stars. As a result, accuracy could have been higher if our model had treated 4 and 5 stars the same, since both ratings essentially would yield a "recommend".

Similarly, reviews with a spread of ± 2 stars often used more positive or negative language than the review seem to warrant. This spread was very unlikely to be seen for actual 1-star or 5-star reviews (~0%), was ~5% for 2-star and 4-star reviews, and ~10% for 3-star reviews. In Example 2 in Appendix 6.2.2, the review received a predicted rating of 5 stars, but actually was given 3 stars. The review used extremely positive sentiment phrases, such as "this gem of a dive bar & grill" and "crazy delicious" yet nonetheless was given 3 stars.

Reviews with a spread of more than ± 3 stars (less than 5% across 1, 2, 4, and 5 star ratings) had sections that read as if they had the opposite sentiment of the overall review score (e.g. the reviewer seemed angry or hostile during sections of a positive review). In Example 3 in Appendix 6.2.3, the review received a predicted rating of 2 stars, but actually was given 5 stars. The reviewer wrote early in the review the highly negative sentiment phrase "For anyone trying to compare this buffet to one that is double or triple the price, don't. Just get out your fat wallet and move over the strip", though later wrote the positive sentiment phrases "decent value with good, everyday food" and "Palace Station wins". The predicted review score seemed to weigh the longer, negative sentiment phrase much more highly than the shorter, positive sentiment phrases, generating a prediction that had the opposite sentiment of the actual star rating given.

Reviews with a spread of ± 4 stars were extremely uncommon (almost 0% across 1-star and 5-star ratings), but had sections that seemed to contradict the review score, and often contained sarcasm. They also tended to be longer than less confused reviews. This particular aspect likely occurred because RNN models tend to perform poorly with longer sequences. Though LSTM units do help to alleviate this problem, they do not eliminate it. In Example 4 in Appendix 6.2.4, the review received a predicted rating of 1 stars, but actually was given 5 stars. For this example the model was completely off, though this can be reasoned by the many mixes of positive and negative sentiments in the review text. Phrases like "Beware. Stay away" and "don't eat any!" were mostly sarcastic from the reviewer, referring only to diabetics. For most of the remainder of the review, the sentiment is positive, but it contained phrases like "Will never buy that grocery store crap again" which the model picked up as negative sentiment. As a result, the model sees the review text as strongly negative, even though the reviewer rates the restaurant highly positively with 5 stars.

4.4 Collaborative Filtering with LDA Topic Modeling

The collaborative filtering w/ LDA did not perform as well as the RNN with LSTM units even though the accuracy was a lower-bound calculation. From the error analysis performed on the final model (detailed examples in the appendix section), it was crucial to have meaningful, well-represented topics in the LDA model in order for the recommendation system to perform well. While the final model did have some well-formed, meaningful topics like "tikka masala" and "vietnamese pho" that represent well in Indian and Vietnamese-type reviews respectively, the LDA model also produced some malformed, meaningless topics like "uniform worn twinsburg" and "swartz boyfriend bit". From deeper analysis, about 23% of reviews had topic distributions where these malformed topics had >25% probabilities. These reviews and users that had high probability in these malformed topics led to a suboptimal nearest neighbor performance.

Additionally, a review of specific examples in the error analysis shows that the topic pool seems to have missed topics there were essentially in some reviews. For example, a review that is specific to Moroccan restaurants may have a poor review-topic distribution because the topic pool doesn't seem to have Moroccan food topics. Thus, the LDA model will only pick up restaurant-specific topics when generating the review-topic distribution for that review. The appendix (6.3) shows an example of a review text that referred to subjects that was not well represented in the topic pool, and hence the review-topic distribution was essentially meaningless.

4.5 Experiments

4.5.1 Experiments for Recurrent Neural Network Model with LSTM Units

Before determining the final parameters for the primary RNN with LSTM units model, we performed several variations. Our final selection generated the accuracy (62.7%), F1 (0.63), and correlation (0.53) scores noted in the Evaluation Summary (4.3). Due to the long training time for the full dataset (over 21 hours), we primarily experimented on a much smaller 10k reviews dataset. The number of dimensions used in the word vectors had a sizeable impact on our model accuracy. This is why we elected to utilize 300 dimensions over the other 6 billion token GloVe datasets that included 50, 100, and 200 dimensions. When we used too few dimensions, such as 50, our accuracy scores were often under 30%. Furthermore, while our RNN with LSTM units model is designed to run a variable number of epochs and stop once the accuracy has not improved for two epochs, capping the number of epochs too early dramatically lowered the accuracy. On some runs of the model with parameters otherwise similar to the final model the accuracy would be ~42% after one epoch and ~48% after two epochs.

4.5.2 Experiments for Collaborative Filtering Using Topic Modeling

The Evaluation Summary (4.3) details the accuracy (37.0%), F1 (0.43), and correlation (0.39) scores for the final LDA model, but some variants were experimented, such as varying number of topics, number of words per topic, and user-topic aggregation. That said, like the RNN with LSTM units model, most of the variations were experimented on the smaller 10k reviews datasets and not the final dataset. This is because training the LDA model as well as generating the user-topic distribution matrix is very time and computationally expensive on the final dataset. As a result, we varied parameters using the smaller dataset to directionally decide on the parameters used for the final dataset.

Varying the number of topics greatly affected the quality of the individual topics. We started with 50 topics, which tend to have less malformed topics since only the most common topics were surfaced, but also many nuances in the reviews were not captured due to the limited set of generic topics. Experimenting with high numbers of topics (e.g. 1,000 was tested in the most extreme case), we saw very similar topics like “kalbi short ribs” and “korean short ribs”. As a result, user-similarity did not perform as well since these two topic probabilities are independent even though they are semantically similar. The variants with high number of topics also led to more malformed, meaningless topics.

The number of words per topic was also experimented upon. For the number of words per topic, three was most ideal since one and two-word topics were too generic and did not capture the nuances of the review text, such as “tossed” and “fried”. Four or more words per topics also led to very specific topics, which as a result did not cover a wide variety of subjects especially with a limited pool of topics, such as “chicken eggplant italian style” and “swedish crepes lingonberry butter”.

5. Conclusion

5.1 Next Steps

Given more time, for the RNN with LSTM units model we would like to experiment using different pre-trained word vectors, such as Word2Vec, Doc2Vec, or even create our own. Additionally, we would like to experiment with a block list of words that should not factor into the LDA model. This can be done in the pre-processing layer where any word that matches in the block list will get filtered out before feeding into the LDA model for training. This may help reduce noise in certain topics.

Likewise, we would like to continue tuning hyper-parameters. The RNN with LSTM units and LDA models took 21 and 12 hours to train respectively with the final training dataset. Given the long training time, we were unable to fully hyper-tune the model parameters as we had hoped. We tuned some parameters using the 10k dataset but the parameters that were optimal for the 10k dataset may not have been optimal for the full training set.

5.2 Summary

The primary RNN with LSTM units model dramatically outperformed our Gaussian Naive Bayes baseline in accuracy, F1 score and correlation, while the LDA model only slightly outperformed the Gaussian Naive Bayes baseline using the same metrics. The aspect of RNNs being able to determine relationships between words helped with the performance, but through error analysis we found that the model did not perform well in some review text where there were polarizing sentiments. Additionally, the topic pool in the LDA model contained some meaningless topics that led to degraded performance in generating useful user-topic distributions for nearest neighbor search.

6. Appendix

6.1 Detailed F1 Scores by Category / Star Rating

6.1.1 Gaussian Naive Bayes

Stars	F1 Score	Precision	Recall
1	0.27	0.58	0.37
2	0.17	0.33	0.23
3	0.26	0.23	0.24
4	0.43	0.30	0.35
5	0.53	0.42	0.47
Weighted-Average	**0.39**	**0.35**	**0.36**

6.1.2 RNN with LSTM

Stars	F1 Score	Precision	Recall
1	0.74	0.72	0.76
2	0.57	0.57	0.58
3	0.57	0.57	0.56
4	0.54	0.55	0.53
5	0.70	0.71	0.70
Weighted-Average	**0.63**	**0.63**	**0.63**

6.2 Error analysis - Recurrent Neural Network Model with LSTM Units

The following error analysis focuses on specific examples where the prediction was ± 1, 2, 3 and 4 stars off from the actual stars, each sharing some insights into some systemic issues found in the model.

6.2.1 RNN with LSTM Example 1

Most errors in predicted star ratings were within ± 1 stars of the actual ratings:

Predicted stars: 5, Actual stars: 4

In this example, the prediction was not very far off from the actual prediction. From reading the review text, it is also difficult for a human to judge whether this review text should be 4 or 5 stars. As a result, accuracy could have been higher if we treat 4 and 5 stars the same, since both ratings essentially would yield a “recommend”.

> I can't believe all the negative reviews. I've been eating at Bubbas' for years and the food is always top notch. Best Eastern, NC bbq in Charlotte. Get the large plate and include the Brunswick stew. Sweet tea is strong and tasty,

6.2.2 RNN with LSTM Example 2

Reviews with a spread of more than ± 2 stars often used more positive or negative language than the review seem to warrant:

Predicted stars: 5, Actual stars: 3

In this example, the review contained certain phrases like “this gem of a dive bar & grill” and “crazy delicious” that led the LSTM model to believe that the review was highly positive. That said, the user gave a review of only 3 stars despite of the highly positive phrases used.

> I had been to this plaza 100s of times in my life and never noticed **this gem of a dive bar & grill**. Good service, food & prices. The Mediterranean Dip is **crazy delicious**. I wish it was socially acceptable to order it and just eat it with a spoon!

6.2.3 RNN with LSTM Example 3

Reviews with a spread of more than ± 3 stars had sections that read as if they had the opposite sentiment of the overall review score (e.g. the reviewer seemed angry or hostile during sections of a positive review)

Predicted stars: 2, Actual stars: 5

This example shows where the LSTM model may not perform as well. The review text starts with a very negative sentiment, but is later counteracted with very positive sentiment. However, the LSTM model weighted the negative sentiment highly, likely given the longer phrase compared to the positive-sentiment phrases. As a result, even though the user gave a 5-star review, the model predicted it as 2-stars.

> **For anyone trying to compare this buffet to one that is double or triple the price, don't. Just get out your fat wallet and move over the strip**. For others of us who want

decent value with good, everyday food, this is the place. I'm not overly picky and I like non complicated food that I can identify with, I guess you'd call it comfort food. I've been to all those expensive buffets and I still say for the value Palace Station wins.

6.2.4 RNN with LSTM Example 4

Reviews with a spread of ± 4 stars also had sections that seemed to contradict the review score, and often had sarcasm. They also tended to be longer.

Predicted stars: 1, Actual stars: 5

Lastly, in this example where the LSTM model was completely off, this can be reasoned by the many mixes of positive and negative sentiments in the review text. Phrases like "Beware. Stay away" and "don't eat any!" was mostly sarcastic from the reviewer, referring only to diabetics. For most of the remainder of the review, the sentiment is positive, but it contained phrases like "Will never buy that grocery store crap again" which the model picked up as negative sentiment. As a result, the model sees the review text as negative, even though the reviewer rates the restaurant as 5-stars.

Diabetics: **Beware. Stay away**. Buy some for me and **don't eat any!** ;)
OMG, **soo good**, probably the **best butter tarts that I've had** AND I'm not even a sweets person! Definitely sweet though. Wash down with unsweetened coffee/tea. About $2.50-3ea. 2 tarts were like $6 'n change.

I sorta 'forgot' about the tarts (Plain, Pecan) for a coupla days after SoupFest - long story - but when I ate them, they tasted fresh still. An indication of the amount of sugar perhaps?

Pastry tasted like a butter cookie. **Will never buy that grocery store crap again** as it can't even compare! Mmm...drool...

Tip: A few spots around back of the building, in front of skatepark.

6.3 Error analysis - Collaborative Filtering (CF) with LDA Topic Modeling

Below are some sample topics derived from the from the final LDA model (100 topics in total):

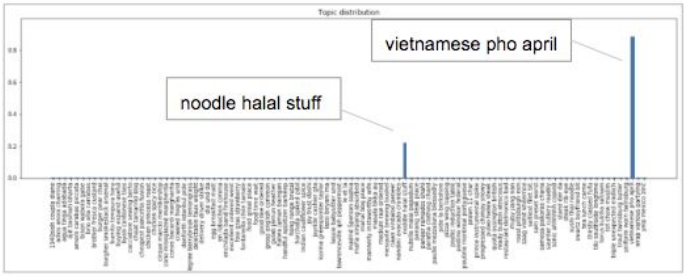
Good topics	Okay topics	Bad topics
Topic #62: masala tikka	Topic #5: food great place	Topic #72: uniform worn twinsburg
Topic #76: vietnamese pho	Topic #63: excellent ordered went	Topic #89: swartz boyfriend bit
Topic #95: steak rib eye		
Topic #96: naan vindaloo paneer		

The good topics like "tikka masala", and "vietnamese pho" well represents many reviews for Indian and Vietnamese restaurants respectively. Other good topics like "rib eye steak" also helps capture reviews where there were mentions of steak, likely from steakhouse or American restaurants.

Interestingly, there were also some topics that are not necessarily specific to a food or food category, but gives hints at the sentiment of the review, like "food great place" and "excellent ordered went". However, there were also many malformed or meaningless topics in the topic pool in the final model. From deeper analysis, about 23% of reviews had topic distributions where these malformed topics had >25% probabilities.

Example of a good review-topic distribution:

Pho King **Vietnamese** Cuisine has a really strange menu... aside from the usual **Vietnamese Pho**, Bun or whatever... it is the first **Vietnamese** restaurants I been to offering Mala (numbing spicy) lamb noodle soup?! And there are quite a few lamb dishes on the menu too… But coming here as a first timer, it is hard for me not to stay with my usual choice at a **Pho** joint? I will leave those lamb dishes for the more adventurous type!\n\nSpecial House **Pho** Small ($5.75)
The broth was nicely seasoned and with a hint of herbs and anise. Beef Meat was plentiful, the tendon was especially tende. **Noodle** was the fresh type. Size wise the portion is more like a medium in a lot of places!!

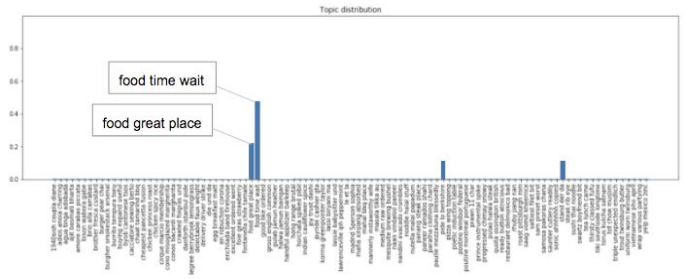


This review had many mentions of "vietnamese pho", and the topic pool also happens to have "vietnamese pho april" as a topic. As a result, the topic-review distribution tends to capture this review well with "vietnamese pho april" having the highest probability. The review-topic distribution also picked up "noodle halal stuff" since there were mentions of noodle in the review. This, however, accounted as a small probability.

Example of a bad review-topic distribution:

i have eaten here a couple **times** and i don't think i was unhappy about anything other than their prices. Its kind of expensive to me but it taste **great**. They have one of the best **turkey burgers** I've had in a long **time** and their **banana**

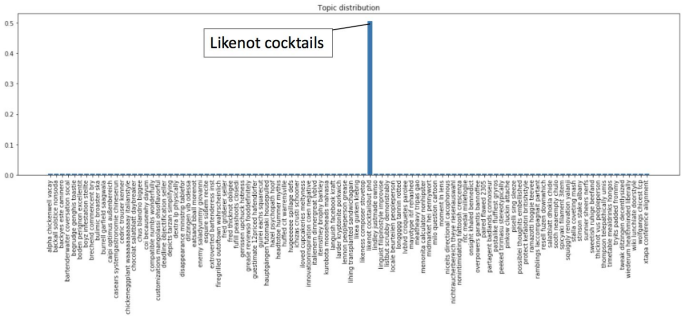
pudding is pretty tasty. The **food** is made to order and their cashiers are very friendly. Its nice and clean, plus its located in a nice shopping mall.



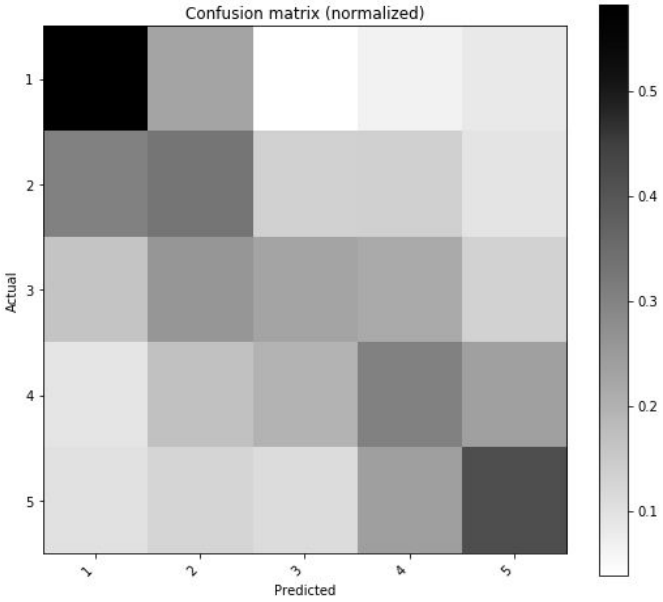
The review-topic distribution for this review text had high probabilities for “food time wait” and “food great place”. This is likely because “times”, “great”, and “food” were mentioned in the review, but the distribution does not well represent the review. As well, in the topic pool, there were no topics like “turkey” or “banana pudding“. If topics like these were present in the topic pool, then the review-topic distribution may have represented the review with higher probabilities for those topics.

Example of a bad review-topic distribution due to missing topics in the topic pool:

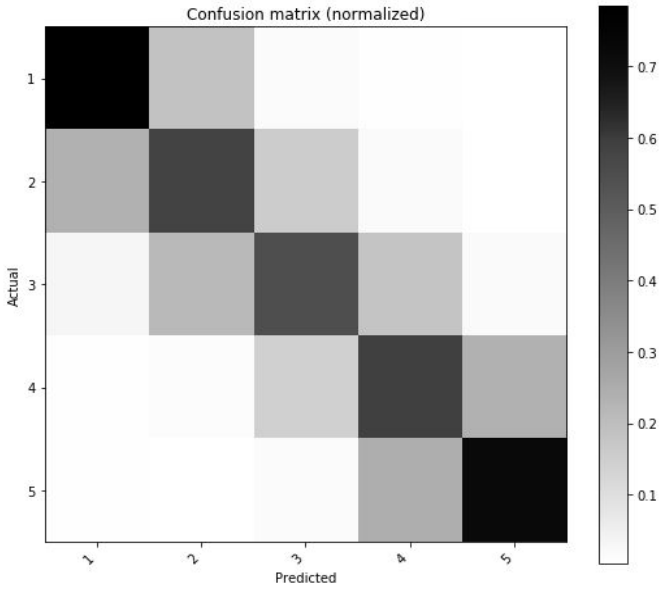
When you say your a **vegetarian** don't recomend the **potato soup** with **Bacon**. Then don't mess around with me about the refund, with no mea culpa.



Unfortunately, in the final model, topics relating to “vegetarian” or relating to specific foods like “potato soup” and “bacon” were not present. As a result, the LDA model essentially assigned a random probability to a topic in the pool and the resulting review-topic distribution is meaningless.



Gaussian NB (Baseline)



RNN-LSTM

6.4 Confusion Matrices

7. References

1. Ansari, A., Essegaier, S., & Kohli, R. (2000). Internet Recommendation Systems. *Journal Of Marketing Research (JMR)*, *37*(3), 363.
2. ADOMAVICIUS, G., & TUZHILIN, A. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE Transactions On Knowledge And Data Engineering*, (6), 734.
3. Li, C., Guanliang, C., & Feng, W. (2015). Recommender systems based on user reviews: the state of the art. *User Modeling And User-Adapted Interaction*, *25*(2), 99-154. doi:10.1007/s11257-015-9155-5
4. Pang, B., Lee, L., Vaithyanathan, S.(2002). Thumbs up? Sentiment classification using machine learning techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, July 2002, pp. 79-86. Association for Computational Linguistics.
5. Jobin Wilson, Santanu Chaudhury, Brejesh Lall (2014). Improving Collaborative Filtering based Recommenders using Topic Modelling. *Cornell University Library.*
6. David M. Blei, Andrew Y. Ng, Michael I. Jordan (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research 3 3 (2003) 993-1022.*
7. Attention and Memory in Deep Learning and NLP. (2016, January 3). Retrieved from http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/
8. Denny Britz (2016). A Recurrent Neural Network Based Recommendation System (Stanford Liu and Singh paper from Slack). *WILDML, Artificial Intelligence, Deep Learning, and NLP*.
9. Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *Cornell University Library.*
10. Recurrent Neural Network (RNN) – Part 4: Attentional Interfaces. (2017, August 18). Retrieved from https://theneuralperspective.com/2016/11/20/recurrent-neural-network-rnn-part-4-attentional-interfaces/
11. R. Jin, L. Si, and C. Zhai (2003). Preference-Based Graphic Models for Collaborative Filtering. *Proc. 19th* Conf. Uncertainty in Artificial Intelligence, AUI 2003.
12. https://nlp.stanford.edu/projects/glove/ - Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
13. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
14. S. Zhang, L. Yao, A. Sun (2017). Deep Learning based Recommender System: A Survey and New Perspectives. *ACM J. Comput. Cult. Heritl. 1- 35*
15. *S. Khusro, Z. Ali, I. Ullah (2016). Recommender Systems: Issues, Challenges, and Research Opportunities. ResearchGate. Retrieved from https://www.researchgate.net/publication/294860665_Recommender_Systems_Issues_Challenges_and_Research_Opportunities*
16. *Chien, Y,-H, Edward I, George (1999). A Bayesian Model for Collaborative Filter*ing. *Technical Report.* Statistics Department, University of Texas at Austin,
17. Daniel Voigt Godoy, Yelp Dataset Challenge, (2017), GitHub repository, https://github.com/dvgodoy/YelpDatasetChallenge