CMPS 242 - Machine Learning (Final Report)
Analysis of Rating Dimensions Using Review Text

Bharath Nagesh *bnagesh@ucsc.edu*
Madhu Shivashankaraiah *mshivash@ucsc.edu*
Numra Bathool *nbathool@ucsc.edu*
Tanay Parekhji *tparekhj@ucsc.edu*

Git repository link : `https://github.com/madhusedu/topic_modeling.git`

Abstract

Modern day systems take in both ratings and review texts from the user but generally consider only the weighted overall ratings for performing different tasks, such as predictions and assessing the overall quality of a business. This according to us is not the optimal method. Ratings are highly subjective and what might be considered as favorable by one could be unfavorable for another. After a detailed observation of the review texts on IMDb, Zomato and Yelp we noticed that users implicitly provide information on their likes and dislikes. We plan to implement a system which takes in not only the rating but also the review text in order to form an opinion. We consider all the reviews of the users and then use data mining to summarize them. We also classify some reviews belonging to the same topic. form an opinion on the preferences of the users, then finally use those opinions in order to improve the overall rating quality for the business and perform better user predictions.

1. Problem Statement

Along with the expansion of the internet, a number of services such as e-Commerce and Review websites have emerged. A lot of people prefer using these services because of many factors, some of them being :

- **Reliability :** These services provide excellent delivery of product/information and then back it up with a good level of customer care. Also, the ratings provided on these sites aren't just a number that is just floated to hoodwink the customer into buying the product/using the service making them very reliable
- **Ease of use :** All these services have a clean web/app interface which makes it very convenient for the people to use them.

The effect of the above factors can be seen in the growth of the industries, as indicated in Figure. 1 and Figure 2.

The general method used to increase the reliability of a product/business/service is by collecting ratings reviews from the people who have used them. This facilitates future customers to form a better opinion of the product when deciding to whether or not use the product.

After a thorough observation of ratings and reviews provided on websites such as IMDb, Yelp, Amazon we observed that The overall rating for any product is formed as a weighted average of the ratings provided by all the users. This method completely disregards the review text provided by many of the users. The users generally provide important information which can be used to improve the overall rating of the product. For an example of this situation, we can consider reviews provided for a restaurant. An user may provide a rating of 3 stars for the restaurant and then provide a review which makes the restaurant actually worth 2.5 or maybe even 4 stars.

This problem that is presented above provided us with the opportunity to develop a topic modeled system which uses Latent Dirichlet Allocation technique in order to understand the reviews provided by the user. After doing so we attempt to develop an application which fine tunes the overall quality of ratings for any given product/business.

2. Algorithm Formulation

In order to perform all the operations we first started off by extracting the required text from the review data file. After this, the extracted review text were processed into a format

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

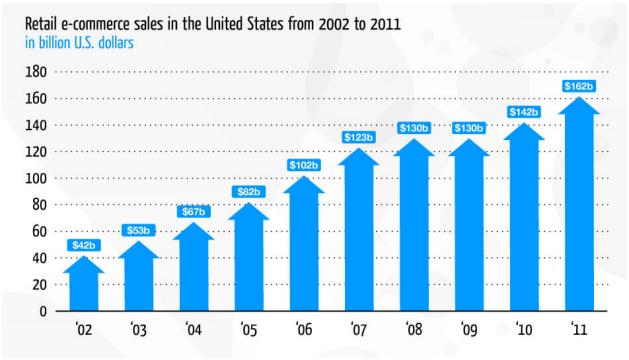

Figure 1. Retail sales in the USA between 2002 and 2011.

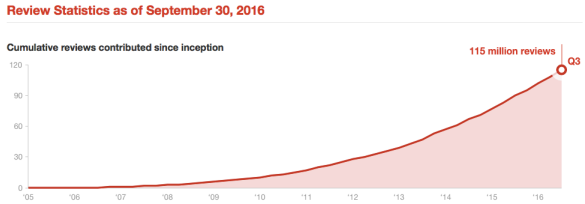

Figure 2. Yelp Review Statistics

which can then be use within the code and identification numbers were assigned to every unique words. This list of unique words had to exclude words such as "a", "the" and so on. This was done to avoid wrong analyses of the extracted review text.

For our algorithm we decided to specify the number of topics that we would be using to group the set of documents into upfront (denoted by K). For our cases we defined 2 as the number of topics.

How are we going to implement the idea?

The Yelp Data Set consists of many categories of user reviews based on Business, Review, User, Check-in and Tip. In our project, we mainly focus on the ?Review dataset? which consists of user reviews. We also used ?Business dataset? to select only restaurant reviews to avoid training the model on unnecessary data. In a broad sense we perform the following:

- **Preprocessing reviews by:** ů Converting all reviews to lower case
  ů Tokenizing the reviews
  ů Removing StopWords
  ů Stemming the words to its root

- Sentence Labelling wherein we implement topic modelling using the LDA algorithm and then classifying each subtopic into groups such as ?Price? ?Time of the day? ?Food? ?Service? etc... Sentiment Analysis by using Bag-of- words to detect frequently occurring words, and grouping them in sentimental zones and granting each category a sentimental score.

- Category Rating Prediction by combining the two implementations mentioned above after which we are able to successfully generate an average rating for every category.

For the first part of algorithm we randomly assigned words to one the 2 topics that we have initialized. Along with generating this topic assignment list, we created two matrices which are :

- **Word-topic matrix :** in other words we created a count of the words being assigned to each of the 2 topics.

- **Document-topic matrix :** This is the matrix with the distribution of the topic assignment list.

In every document d, we observe every word w. We then chose a topic t with the probability equal to $p(w \mid t) \times p(t \mid d)$, denoted by the following mathematical notations:

$$p(z_i = j|z_{-i}, w_i, d_i) = \frac{C^{WT}_{w_ij} + \beta}{\sum^{W} w = 1 C^{WT}_{wj} + W\beta} \times \frac{C^{DT}_{d_ij} + \alpha}{\sum^{T}_{t=1} C^{DT}_{d_it} + T\alpha}$$

$$\theta^j i = \frac{C^{WT} ij + \beta}{\sum^{K}_{K=1} C^{WT}_{Kj} + W\beta}$$

$$\phi^j d = \frac{C^{DT} dk + \alpha}{\sum^{T}_{t=1} C^{DT}_{dk} + T\alpha}$$

where,

$p(z_i = j)$: Probability that word i is assigned to topic j.

z_{-i} : Represents topic assignments of every other word.

w_i : Word ID of the i_{th} word.

d_i : Document containing the i_{th} word.

C^{WT} : Word-topic matrix.

$\sum^{W}_{w=1} C^{WT}_{wj}$: Total number of words in each topic.

C^{DT} : Document-topic matrix.

$\sum^{T}_{t=1} C^{DT}_{d_it}$: Total number of words in document i.

η : Parameter to set the topic distribution for the words.

α : Parameter to set the topic distribution for the documents.

W : Total number of words in the set of documents.

T : Number of topics.

After drawing the new topic we update the topic assignment list with newly sampled topic for the word *w* following which the word-topic and document-topic count matrices are incremented with the new sampled topic for word *w*.

After we complete learning the topics for *r* iterations iterations, we use the count matrices to obtain the word-topic distribution and document-topic distribution.

It is assumed that each document is a mixture of all topics, thus after computing the probability that each document belongs to each topic, we use this information to see which topic each of the documents belong to and possible words that are associated with each topic. After forming an opinion on the topic-document correlation, we attempted to tune the parameters.

Since the starting point sampling point of dataset is chosen randomly, thus it makes sense to discard the first few iterations. Due to the fact that they most likely do not reflect the properties of distribution accurately. And another parameter is the number of iterations omitted during the training. This prevents the correlation between samples while iterating.

3. Feature Engineering

In order to perform the function of topic modeling we needed a dataset, for this we used the Yelp user review?s dataset which was provided along with the other datasets in the Yelp dataset. The dataset was found to be very large in order to process and obtain results quickly. Due to hardware limitations and high processing power required to run a huge sample of data we started running the code with variable sample size. We started with 500 reviews and then increased the sample size to 1000 reviews. Consequently we started running the code with increasing sample sizes and found that after 20,000 reviews the processing time taken is high and the results does not vary much with the sample size above this limit. We performed our final Computation on 50000 reviews. The output result we have uploaded and shared in github was for this sample size. We also tried to perform this computation for a dataset with more samples but the processing power required to perform this was high and due to our hardware limitations our systems were not able to perform this computations. Therefore we had make this our upper limit of sample size. This is the data we used to train our model . we have also tested the model using a sample size of 5000 reviews and we observed that the result of training data and test data were comparable. This en-sured that our results was obtained efficiently on a large enough dataset required to generalize our algorithm.

Are features adequate and reasonable? What pre processing did you do?

The primary feature that was required to perform the desired task was review text and ratings . using the reviews we found the probalbity of a review belonging to a topic which is important in many contexts and then by finding the weighted sum of probabilities we were able to give an appropriate ratings for the review and we complared the rating with the rating given by the user and there was a 65% match between our result and th e ratings given by the user. This shows that there are a good percentage of rating which match the reviews. We thought that ratings and reviews were a great features as they are the major features that will be used by the users and by finding a correlation.

- Word-topic matrix
- Document-topic matrix
- The various probabilities of occurrences

However we could not use the review text as given within the dataset. We had to process the data so that the necessary functions could be applied to it in order to obtain the results.

In order to obtain and process the data we first had to convert the JSON file into a useable CSV file. Once this was done we sub-sampled the dataset and incremented the sample size. We made a list of reviews which is converted to list of words. We then pass this list of words through stop words library which will remove the unimportant words from the list of words in the review. After passing it through stop words we then pass to stemming library which converts all the words into their root words which will be used to compare words and count similar words as one word. We then tokenize the list of stemmed words for further processing. The tokenized words are then compared against affin list which consists of all the positive and negative words with their respective sentiment score

We created specific methods within our code to implement the tasks which are explained in sections 4 and 5.

4. Evaluation

We compare our LDA model that was constructed from scratch with the LDA model obtained using genism. It was observed that, given all the parameters were the same, the perplexity calculated for the genism-LDA was found to be -4280.5691 and that for our LDA algorithm is -2678.1889.

Since we know that Perplexity is a measure of how easy a probability distribution is to predict, from the above we can establish that the genism model is much easier to predict than the one we constructed (as it has a lower perplexity value).

The reason for this is because we convert our corpus to bagofwords in the model constructed using genism. Having a bag of words gives us accurate results and hence a lower perplexity.

The results have been included within the lda.pdf file in the report folder of the repository.

5. Results and Observations

After recognizing the topics from the LDA model, we were left with words that are relevant to what would help derive a rating from a review. Each word was found to have a certain probability of being present in each topic. Next, we found the probabilities of a review belonging to a topic. The topic with the highest probability was assigned to the review. We also found the list of words that have the highest chance of being in a topic i.e. the top words in each topic.

From a readymade database of weights assigned to most common words occurring in the dataset (that help decide the sentiment behind the review), we calculated the weighted probability of each word occurring in a topic finally obtained a score for a review based on its probability of being associated with a particular topic. This score helped us assign a rating to the review as per a fixed set of ranges for the score. We then compared this derived rating to the rating provided by a reviewer. It was found that 65% of the ratings obtained from the reviews matched the ratings given by the user. This doesn?t necessarily mean that we were right and the reviewers were giving inconsistent reviews for ratings. It could simply mean, we weren?t finding the best range for classifying reviews. Interestingly, the very low and very high ratings given by users were consistent with their reviews and more often than not, the ratings we found matched the ratings given by the user. However, when a user gave a more average rating, we often perceived their review to not match with the rating our algorithm calculated.

An observation we made was that using more data increases the accuracy of the result. As we used more reviews, the reviews-to-rating normalization became more efficient. We also noticed that, as we were identifying topic belongingness of words by individual words, context was not recognized. People are less likely to use passive speech during reviews. For example, ?the pizza was horrible? appears more often than ?the pizza was not the best?. Hence, due to the sheer size of the data, the occurrences of passive negative words may not make a huge difference in the larger picture.

The primary challenge that we faced during the project was understanding details of various algorithm and picking the most feasible one that we could implement well and that would help us receive some interesting results. Along the

way, we faced a steep learning curve as we didn?t have much experience with these algorithms.

We obtained top words in each topic as follows :

Top 5 words in **topic 0** are ['fresh', 'like', 'popular', 'free', 'disgusting']
Top 5 words in **topic 1** are ['better', 'win', 'saved', 'worst', 'disappointed']
Top 5 words in **topic 2** are ['trouble', 'pay', 'great', 'support', 'miss']
Top 5 words in **topic 3** are ['good', 'bad', 'pleasant', 'worthy', 'mistaken']
Top 5 words in **topic 4** are ['best', 'awards', 'failure', 'sadly', 'awesome']

6. Challenges

During the course of the project we encountered a few significant challenges which we eventually were able to over come. Some of the challenges that we encountered were.

- Understanding the LDA algorithm along with the parameters and the equations necessary for processing of the data
- Once we got a hold on the topic we had to arrive at a sub-sampled dataset which was large enough to get a good result

7. Learning objectives achieved

Over the course of the project over the last few weeks we achieved a good proficiency in many areas. The highlights of our learning include :

- We gained a good understanding of the Latent Dirichlet Allocation algorithm. Along with the algorithm itself we were able to understand it's functioning and the pro's and cons associated with it.

  We were also able to get a good grip on it's implementation.
- It helped us discover and understand new packages available in python. Packages such as gensim and pyLDAvis.

8. Deliverables

We believe that 8 weeks was a short period for us to work on the project and that more could be done provided we had more time available. A few of the ideas that we had are :

- To create an API which could then be used to generate ratings based on the review text.
- To use the code against a larger dataset with more review texts.
- To use the code against different datasets. By this we mean that we intend to use the code against dataset obtained from other sites such as Amazon and eBay. This would help verify cross-platform functionality.

References

[1] T. L. Griffith, M. Steyvers. *Finding Scientific Topics* PNAS, vol. 10, 6 April - 2004.

[2] Ke Zhai, Jordan Boyd-Graber. *Online Latent Dirichlet Allocation with Infinite Vocabulary* In proceedings, *0th International Conference on Machine Learning*, Atlanta,GA, 2013 JMLR : WCP volume 28. Copyright 2013 by the author

[3] D .M. Blei, A. Y. Ng, M. I. Jordan. *Latent Dirichlet Allocation* Journal of Machine Learning Research. January, 2003

[4] Blei, D. M., Griffiths, T. L., Jordan, M. I., Tenenbaum, J. B. *Hierarchical topic models and the nested Chinese restaurant process* In Advances in Neural Information Processing Systems 16. Cambridge, MA, USA: MIT Press. 2004

[5] Erosheva, E. A. *Grade of membership and latent structure models with applications to disability survey data.* Unpublished doctoral dissertation, Department of Statistics, Carnegie Mellon University.

[6] Hofmann, T. *Probabilistic Latent Semantic Analysis.* In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence.

[7] Zhuang.L, Jing.F, Zhu.X. *Movie Review Mining and Summarization.* CIKM '06, Arlington, VA, USA, 2006.

[8] Hu.M, Liu.B *Mining and Summarizing Customer Reviews.* KDD '04, Seattle, WA, USA. 2004

[9] McAuley.J, Leskovec.J *Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text* In Proceedings of the 7th ACM conference on Recommender systems, Hong Kong, China. 2013