Review based Recommendation System

ARIF SARFARAZ WAGHBAKRIWALA and AANAND DHANDAPANI, Northeastern University, USA

A set of reviews from Amazon's data set have been analysed here to construct a recommendation system. In general, product recommendation system work on popularity based on ratings. This has many shortcomings. In our approach we try to construct a system that takes into account, the popularity based on review, rating and time. Several Models on different data-sets with multiple configurations were trained and a model with better evaluation metrics was chosen as the "`better`" `recommendation system`.

Additional Key Words and Phrases: Data-sets, Recommendation System, Unsupervised Sentimental Analysis, Word2Vec, Clustering

1 INTRODUCTION

With billions of products on Amazon and hundreds to thousands of reviews on average for each, it becomes confusing to choose a product. To add an extra layer of confusion, now a days, Businesses and sellers get people to write positive reviews and give better star ratings. Thus, it becomes hard for an average person to choose between products. This is where a review-based recommendation system would strive to aid in the struggle a lot.

1.1 Motivation

Generic Popularity based recommendation will not be able to solve the issue discussed above. It works by recommending items viewed and purchased by most people and are rated high. It is not a personalized recommendation. And it does not necessarily mean that the product is good. Reason being, a product that had been in the market for a long time will have obviously been bought more than a product that was recently released. However, a product that is currently "`trending`", not popular, should be recommended by the recommendation system.

1.2 Contribution

1.3 Arif Sarfaraz Waghbakriwala

Finding relevant Dataset, Data Preprocessing, Data Modelling, Hyper-Parameter Tuning, Visualizations, Evaluating the Model using Precision, Recall and F1 score.

1.4 Aanand Dhandapani

Finding relevant Dataset, Data Preprocessing, Data Modelling, Hyper-Parameter Tuning, Evaluating the Model using Confusion Matrix and MAP (Mean Average Precision)

2 RELATED WORKS

2.1 What has been done?

Amazon's current Algorithm is based on Item-item Collaborative Filtering[1]. This is a derivative of the generic User-User Collaborative filtering. Amazon had to improve upon the User-user filtering because it had many issues. For example in cases when a large number product had fewer ratings it became Computationally costly and it's adaptability to user profile changes was bad.

Authors' address: Arif Sarfaraz Waghbakriwala, waghbakriwala.a@northeastern.edu; Aanand Dhandapani, dhandapani.aa@northeastern.edu, Northeastern University, P.O. Box 1212, Boston, Massachusetts, USA, 02115.

Item-item Collaborative filter was invented by Amazon to resolve these above mentioned issues. It takes into account the rating distributions per item and not per user. With the number of users being many times than items, each item will obviously have more ratings than individual users. Hence, an average on item's rating generally doesn't change much. Hence, this leads to more stable rating system for the model. And therefore the model doesn't have to be rebuilt for smaller changes in user profile. When users consume and rate a particular item, the item's that are similar to it are added to the user's recommendation system or pool.

2.2 What needs to be done?

Amazon's recommendation system is impressive in that it takes into effect the user experience of a product and links related products together based on rating distribution. However the focus of this project is to build upon it. Even though ratings are a good numerical representation of a user's experience, it lacks on two instances. Firstly, at describing how bad/good a product was verbally. With the lowest rating possible being 1 and the highest rating being 5. a recommendation system based on Ratings would always think that product is still viable or the best product to date respectively. However an analysis on the reviews themselves would give a better intuition on how good/bad that product was. Secondly, Time. Amazon's recommended system was revised based on the fact that the number of ratings is significantly greater than number of users. Now, Let's assume a product P by brand B was released in year 2000 had an average rating of 4.5 with over 200K people rating the product. Now, Assume that a product P' was brought to the market by B in the year 2019. Then, if the product is indeed an improved version of P and is currently trending, then the system should start recommending P' instead of P as it's top recommendation. However, That is not the case. Hence, a recommendation system, that takes into account, reviews, ratings and time would be a better system than the existing one.

3 BACKGROUND INFORMATION

3.1 Word Embeddings

Word embeddings is a technique wherein individual words are converted into a a vector, which in a way is a numerical representation of the word. Each word is then linked to one vector, this vector is then learnt by the use of methods such as Skip-Gram or CBOW (Continuous Bag Of Words). These vectors try to understand the word's various aspects with regard to the overall context. These aspects include the semantic context, semantic relationship definitions, etc. Word embeddings are an integral part of solving problems in NLP. They depict how humans understand language to a machine.

3.2 Word2Vec

Word2vec[2] is a tool that makes use of Word Embeddings. It takes the input, which are a list of words and gives word vectors as result. As a first step, it creates a vocabulary from the training set and then learns vector representation of the provided words. The resulting word vector is then used as features for many NLP and ML applications.

Since the word2vec uses Cosine Similarity to calculate distance. A simple way to analyze the learnt word vectors is to find corresponding words that are similar. The distance feature serves this purpose.

In word2Vec, there are two main algorithms that are used to learn word vectors. They are continuous bag-of-words (CBOW) and continuous skip-gram (SG). By default, the CBOW is used. But both of the algorithms learn the representation of a word. And these representations are useful for prediction of other words in the sentence.

3.3 CBOW

Continuous Bag of words (CBOW)[2] works by trying to predict a word based on the words that are present surrounding it. By the use of windows, the CBOW tries to understand the context by weighing words that surround the target word. In essence, we try to understand why words co-occur and thereby understanding their context.

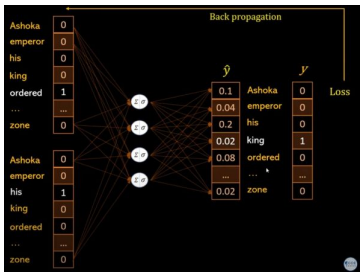

Fig. 1. Word2Vec using CBOW - Architecture[1]

4 PROPOSED APPROACH

The approach, put forward in this project along with formal terminologies, can be elucidated in multiple steps.

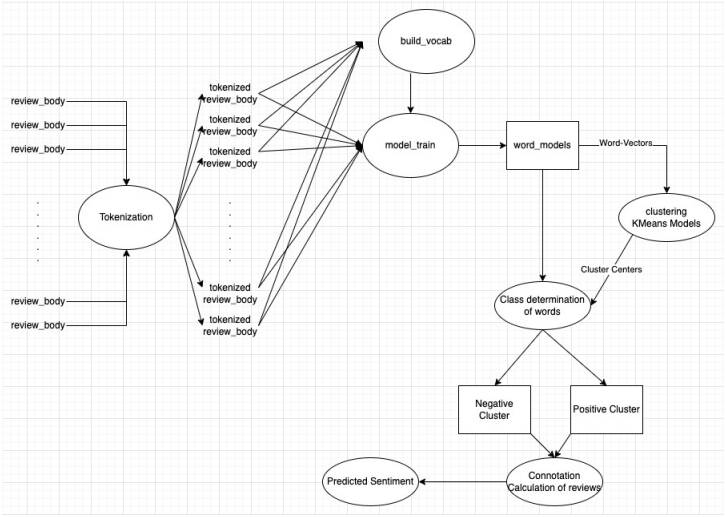

Fig. 2. Proposed Methodology Flow

The first and the most common step is to pre-process the Data-sets. This involved feature selection that was hand-curated, handling missing values, which in this case was not that much of a hassle as the proportion of missing values was infinitesimal, so just dropping the missing values was enough. Then, selecting observations of unique products that are adequate enough for the model to learn unbiased parameters and dropping those products for which there are not enough number of observations (the threshold set here is 100). Moreover, Feature Engineering of variables such as 'ratio_votes' based on helpful total_votes, 'True_value' based on 'star_rating' and 'predicted_value' based on 'score'

(connotation score).

The second stage of the approach comprises of tokenizing the review_body where all of the reviews had to go through a function that cleans the text, converts them to standard ASCII values using 'unidecode()' function, turns all the text into lower-case, gets rid of numbers, removes stop words using gensim.utils[5] package's simple pre-process function and eliminates words that have no semantic meaning based on the corpus available in "nltk library"[5].

Third and the most important step, that is the model training, where the models were trained on individual data-sets of the three chosen for-trial-purposes, which can then be used for generic implementation. To train each model, first a basic model object from gensim.models[4] is created using the "Word2Vec()" function, where certain parameters are to be specified such as learning rates, window size, number of neurons in hidden layers, minimum frequency of the words, thread-workers and many more. Then, the tokenized reviews are passed to the "build_vocab()" function that is callable by model object that was initialized in the first step. This returns a corpus of unique words, set of words across all the reviews in that data-set. Further, the model is trained by passing the processed reviews (tokenized reviews) to the train() function, that is again callable by the model object that was initialized in the first step, which now also contains the vocabulary of the reviews of the data-set. Post training, the model-object consists of the word-embedding, 300 component vectors (1x300) for each of the word in model's vocabulary, which is numerical representation of each word.

In the fourth step, the vector-representation of words or the word-embedding are passed to the 'KMeans' model from sklearn.cluster[6], which segregates them into two clusters and determines the centroid of positive and negative cluster.

In the fifth step, class of each word is determined in the user-built function 'class_determination', passing the centroids produced by the 'KMeans' and 'Word2Vec' model itself. Distance of each word from both the centroids is calculated using the in-built function 'wv.similar_by_vector', wherein cluster point (the positive and negative centroids) is passed and topn value is gives values in increasing order of number of words and their distances to cluster center specified. Based on the distance of each word to both the cluster centroids, the word is assigned to a class that it is closest to.

Once the connotation (class) of the words are determined, that is whether it emits a negative or positive sentiment (represented by -1 or 1), the score of each review is a total summation of negative and positive words, where positive score indicates a good review, bad review otherwise. The results are contrasted with ratings given by the user to that product, with the notion that low ratings will never go with a good review, meaning if the products were rated less indicates the users didn't like a certain product, hence the generalization is that that product will be given a negative review.

In the Final step, a query from the user is taken as input, and based on the category queried by the user, the data-frame containing the pre-calculated scores is picked. The chosen data-frame is scanned for the most recent and top-n (n being the number of reviews for a unique product) scores grouped by unique products, which is averaged. The final output is the top-k (k being the number of unique products) products based on the averages.

5 EXPERIMENTS

window-size: number of words used for predicting the current word (n words on both sides).
hidden layer size: number of neurons in hidden layer, that determines the dimensions of word-embeddings.

Hyper-parameter Tuning: There are two main parameters that played an important role in affecting the performance of the trained models, mainly window size and number of neurons in hidden layer. Smaller window size (<=4) with 150 to 200 neurons in hidden layer and Larger window size (>=7) with same number of neurons evinced low performance. After trying multiple combinations, the final set of parameters contained window size of 10 and 300 neurons in the hidden layer showed the best results.
One such scenario of contrast is presented below (window-size of 10 and 300 neurons VS window-size of 6 and 300 neurons)

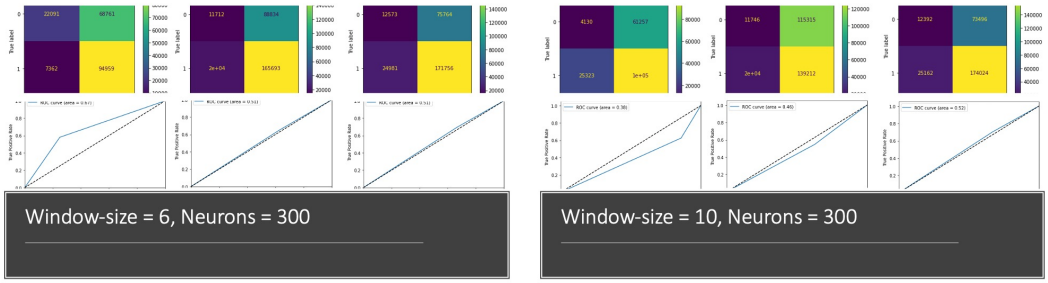

Fig. 3. Confusion Matrix ROC Curve - Model Comparison

There was an instability observed in Word2Vec, that upon training model on same blend of parameters, slightly different results were obtained.
Data-set Description: The dataset was downloaded from the official site, https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt . This has datasets in the form of TSV files for 46 different variables. The dataset includes columns like, marketplace, customer_id, review_id, product_id, product_parent, product_title, product_category, star_rating, helpful_votes, total_votes, vine, verified_purchase, review_headline, review_body and review_date. The feature we are concentrating more on are star_rating, votes and review_body. The reviews in data-set are dated back to 2010's till 2020, which are only used for strengthening the approach of recommending based on the unsupervised sentiment of reviews that can be implemented for generic data-sets with recent reviews.

Results :

The end-result for the project was the calculated connotation scores based on the sentiment of words derived from their contextual surroundings. If the connotation score is positive and the star-rating is above the set threshold (which is 3), then the prediction of connotation score is set to true, indicating the product was rated high and liked by the user, therefore giving positive feedback. If the results are in opposite direction, 'low rating with positive connotation scores' or 'high rating with negative connotation scores', then the prediction is false. The precision, recall, f1 score and MAP for the data-sets taken into consideration are as follows

The below figure of metrics depicts the difference of the same set of parameter combination as mentioned in Hyper-parameter Tuning. The model with window size 6 and 300 neurons has better precision, recall and F1 score than the

	Precision	Recall	F1 Score	Map
Jewelry Data-Set	Positive : 0.928 Negative : 0.243	Positive : 0.580 Negative : 0.750	Positive : 0.714 Negative : 0.367	0.667
Musical Instrument Data Set	Positive : 0.893 Negative : 0.116	Positive : 0.651 Negative : 0.372	Positive : 0.753 Negative : 0.177	0.666
Watches	Positive : 0.873 Negative : 0.143	Positive : 0.694 Negative : 0.335	Positive : 0.773 Negative : 0.200	0.667

Window-size=6, Neurons=300

	Precision	Recall	F1 Score	Map
Jewelry Data-Set	Positive : 0.801 Negative : 0.063	Positive : 0.626 Negative : 0.140	Positive : 0.703 Negative : 0.087	0.6667
Musical Instrument Data Set	Positive : 0.876 Negative : 0.093	Positive : 0.547 Negative : 0.373	Positive : 0.673 Negative : 0.148	0.6656
Watches	Positive : 0.874 Negative : 0.144	Positive : 0.703 Negative : 0.330	Positive : 0.770 Negative : 0.201	0.6667

Window-size=10, Neurons=300

Fig. 4. Evaluation Metrics

model with window size 10 and 300 neurons. Whereas, the Mean Absolute Percentage Error for both the Combinations are almost same for both the combinations.

6 FUTURE WORKS AND CONCLUSION

The system can be extended to find review anomalies in any generic reviews-dataset. E-commerce websites are strict with regards to reviews that were falsely generated. However, people have a workaround of leaving an honest (positive) review if the company gives the product for free. This recommendations system can be enhanced to make the recommendations less biased, irrespective of outliers and to analyse connotation scores which can be used to detect reviews that are, on average, very high.

In conclusion, the system works pretty descent in how it gives recommendations. It is not biased towards ratings, rather is an accumulation of Ratings, reviews and time. With further improvements to the model in-terms of hyper-parameter tuning and finding weights of certain hyper-parameter through Neural-Networks, can improve the efficiency of this model.

7 REFERENCES

[1]: Word2Vec : https://patents.google.com/patent/US6266649

[2]: Putra Fissabil Muhammad, Retno Kusumaningrum, Adi Wibowo. 2021.Sentiment Analysis Using Word2vec And Long Short-Term Memory (LSTM) For Indonesian Hotel Reviews, Procedia Computer Science. Volume 179. Pages 728-735. ISSN 1877-0509

[3]: Official Documentation for Gensim Package- https://radimrehurek.com/gensim/models/word2vec.html

[4]: Official Documentation for Gensim.models Package https://radimrehurek.com/gensim/models/word2vec.html

[5] : Documentation for nltk.corpus Package- https://www.nltk.org/howto/corpus.html

[6] : Documentation: KMeans - https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html