Predicting Item Rating by Text Mining and NLP Analysis with Renttherunway Clothing Fit Dataset

Muchan Li
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
mul005@ucsd.edu

Yunfan Long
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
yulong@ucsd.edu

Abstract

Text Mining and NLP Techniques have demonstrated cutting-edge performance in a variety of tasks. In this paper, we investigate and employ TFIDF, SkipGram, and Continuous Bag of Words(CBOW) Model to predict item rating in each user input. In order to capture the user-item interaction, we used SVD to build our first baseline model. Since our output are one of the five numbers (2, 4, 6, 8, or 10), our task essentially becomes a classification task, and we believe logistic linear regression with softmax output would fit. Hence, we implemented our second baseline model using logistic regression. For training our model we'll use the Renttherunway Clothing Fit Dataset. The dataset includes 15 features detailing the user features such as weight, height, body_type, age, bust_size, the item features including category, "rented for" (rent purpose), and interaction features like fit, review_text, and review_date. We use auroc score, f1 score, accuracy, and mse as our evaluation metrics. In our first baseline model, we employed SVD model, which achieves a MSE of 2.3628 after fine-tuning. In the second baseline model, we implement the logitic regression and get test auroc score of 0.5397, f1 score of 0.1573, accuracy of 0.6482, and mse 2.8765. Our best final result of baseline model is the SVD model with mse of 2.3628 for test set. After custom fine-tuning, we improve the test accuracy to 2.0504 with learning rate of $1e^{-4}$ and 1r of $2.6e^{-3}$. Applying Text Mining and NLP techniques, we explored TF-IDF, SkipGram, and CBOW Model. We found that with SkipGram, we get a test auroc score of 0.8417, f1 score of 0.2877, accuracy of 0.6966, and mse of 1.9651. After we factor in the similarity for user and item, our $SkipGram_{Sim}$ model get test auroc score of 0.8451, f1 score of 0.3048, accuracy of 0.6968, and mse of 1.8856 as our best result while with TFIDF we got a similar result (mse 2.3993%) as our baseline model. This is not surprising because SkipGram generates a much more meaningful and less spatially expensive embedding, which should contain more context information and thus perform better than the TFIDF model.

1 Introduction

Clothing rental services and providers allow you to enjoy different fashion and apparels while still maintaining your budget under control. You can look like the top rock star one day, then go to a wedding in a nice and clean suit the other day, and still come back with a pocket full of hard-earned cash.

While all seems fun and cool, do you ever wonder how you get promoted the clothes and accessories you have long been craving right at your finger tips?

For giant clothing rental platforms like Rent the Runway, they leverage thousands upon thousands of customer's purchase data, profile each and every diverse consumer groups, and therefore deliver the exact match of you want in the blink of the eye. This giant systematic network is the recommender system. For this project, my team would like to crack open this system ourselves and see for ourselves how the features of customers and the clothing items interact to determine the rating of a specific clothing item from scratch.

2 Related Works

Our dataset is conveniently borrowed from Professor McAuley's lab's dataset collection. More specifically, the RentTheRunway dataset is the direct product of Misra, Wan, and McAuley's published paper *Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces* [1]. Their paper uses this same dataset to primarily address the challenge in clothing product fit prediction arising from the subtle semantics, subjectivity, and imbalanced labeling in customer feedback reviews. They identify this similar prediction task as ours to be imperative for reasons analogous to our own–solving this challenge can have tremendous impact in improving customer satisfaction and generating more profits to the sellers by reducing product returns. The solution they proposed is metric learning, which essentially aims to learn the customer, product embeddings and a subsequent distance metric such that for any transaction (i.e. customer-product interaction), the distance metric between the customer and product of one particular class is maximally different from any pair from a separate class. Their proposed solution proves to be indeed more accurate compared to conventional latent factor models as it sees a 6%-7% increase in average AUC.

In addressing this more and more recognized product size fit challenge, other researchers have devised various designs and intuitions. Of these, a few notable state-of-the-art models include the product size embedding model presented in the paper *Learning Embeddings for Product Size Recommendations* by Dogani et. al. [2] and the graph attention network presented in *Learning Outfit Compatibility with Graph Attention Network and Visual-Semantic Embedding* by Wang et. al. [4]. Both models see significant improvement in prediction accuracy (the product size embedding model claims to be as much as 40% more effective) compared to the conventional baseline models such as matrix factorization. From these background researches, we noticed that using embedding techniques to project interaction data into hidden representation spaces really helps extract the subtle semantics that would not otherwise be easily uncovered. Concretely, for the product size embedding model it learns the embedding of user purchase history and brand information and for the graph attention network it learns the embedding of text descriptors and images. Also, we noticed that it is rare to find a study that deploys this embedding approach directly on text corpus and relates textual semantics to product fitting, or in a similar sense star rating, in the case of our project's focus.

Combining this source of motivation with the skills and experiences we have on modeling texts and word embeddings, we decided to deploy a Skip Gram and a CBOW model (these two often come in pairs since their intuitions reciprocate each other) that both deal with learning word semantics. Then, we can have the textual semantics in conjunction with the numeric features per customer product interaction predict the rating one would give to a product.

[1] Misra, R., Wan, M., & McAuley, J. (2018, September). *Decomposing fit semantics for product size recommendation in metric spaces.* In Proceedings of the 12th ACM Conference on Recommender Systems (pp. 422-426).

[2] Dogani, K., Tomassetti, M., Vargas, S., Chamberlain, B. P., & De Cnudde, S. (2019, July). *"Learning Embeddings for Product Size Recommendations."* In eCOM@ SIGIR.

[3] J. McAuley. *Lecture 6-10: Latent Factor Model & Recommender System Evaluation*, lecture notes, Department of Computer Science, University of California San Diego, Nov 2022.

[4] Wang, J., Cheng, X., Wang, R., & Liu, S. (2021, July). *"Learning outfit compatibility with graph attention network and visual-semantic embedding"*. In 2021 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). IEEE.

3 Dataset

3.1 Dataset Used

Example (RentTheRunway)


```
{
  "fit": "fit",
  "user_id": "420272",
  "bust size": "34d",
  "item_id": "2260466",
  "weight": "137lbs",
  "rating": "10",
  "rented for": "vacation",
  "review_text": "An adorable romper! Belt and zipper were a little
hard to navigate in a full day of wear/bathroom use, but that's to be
expected. Wish it had pockets, but other than that-- absolutely
perfect! I got a million compliments.",
  "body type": "hourglass",
  "review_summary": "So many compliments!",
  "category": "romper",
  "height": "5' 8\"",
  "size": 14,
  "age": "28",
  "review_date": "April 20, 2016"
}
```


Figure 1: Sample Renttherunway data

The Renttherunway Clothing Fit Dataset consists of 192544 records of user reviews with 15 columns and 5 rating classes, with unbanlanced distribution per class. We perform a 80-20 train-test split to understand how well our model generalize on unseen data. The training set contain the records in random order, but training set contains different number of records from one class than another, and the test set follow the similar distribution. Table 1 below demonstrates the size of training data and testing for the different rating classes

Table 1: Data statistics

Class	Training Set	Test Set
2	99583	24948
4	42799	10589
6	8504	2193
8	2240	551
10	835	210

3.2 Dataset Statistics

First and foremost, the data frame obtained from reading in JSON formatted data file has 192544 entries, meaning there are 192544 unique customer feedbacks identified by the unique combination of user ID and item ID. The data frame has 15 columns, each storing one attribute. Below is a quick description of each attribute: "fit": whether the cloth fitted or not "$user\_id$": unique customer identifier "bust size": customer's bust size "$item\_id$": unique identifier of the rented clothing product "weight": the customer's body weight "rating": customer's star rating of this cloth renting experience "rented for": the occasion of this cloth renting "$review\_text$": the review text corpus in bulk from the customer "body type": customer's body type "$review\_summary$": short summary of the customer's review "category": the category that the rented cloth belongs to "height": the customer's height "size": the clothing product's size in its respective category "age": the customer's age "$review\_date$": the date that the current review is submitted

3.3 Data Cleaning & Imputation

Among these 15 features, we assessed and observed that 7 of these contain missing or null values, which are "bust size", "weight", "rating", "rented for", "body type", "height", and "age". To account for these, we devised a two-step approach. For step one, we would drop any customer entry that does not have a rating. We believe this is the best action to take since a customer's rating on the

```
df_raw.isna().mean()
```

```
fit                0.000000
user_id            0.000000
bust size          0.095620
item_id            0.000000
weight             0.155715
rating             0.000426
rented for         0.000052
review_text        0.000000
body type          0.076019
review_summary     0.000000
category           0.000000
height             0.003516
size               0.000000
age                0.004986
review_date        0.000000
dtype: float64
```

Figure 2: Data Missingness

renting experience is the very subject that we would like to predict using our models, which we will elaborate in the following section on “Predictive Task”. We also strip the ”lb” and convert the string type ”weight” column to numeric type , similarly, converting all string type ”height” to numeric type in inches. What’s more, we convert the bust size original in string format and has many variants (eg. ”32b”, ”33b”, ”34b”) to numeric type (eg. 32, 33, 34). For step two, we impute the rest of the missing values using different measures depending on its level of measurement (i.e. whether it is qualitative or quantitative). For qualitative missing data, such as missing “body type” or “rented for”, we imputed them with the mode of their respective columns; for quantitative data (after processing) instead, we imputed with the mean of their respective columns.

	bust_number	weight	height	size	age
count	192452.000000	192452.000000	192452.000000	192452.000000	192452.000000
mean	34.182040	136.241889	65.306056	12.245521	33.855860
std	1.660858	20.300442	2.660093	8.495457	8.040119
min	28.000000	50.000000	54.000000	0.000000	0.000000
25%	34.000000	125.000000	63.000000	8.000000	29.000000
50%	34.000000	130.000000	65.000000	12.000000	32.000000
75%	36.000000	145.000000	67.000000	16.000000	37.000000
max	48.000000	300.000000	78.000000	58.000000	117.000000

Figure 3: Statistics of User Features

To quickly get a sense of the spread of numeric data in our data frame, figure 3 above present the brief summary of order statistics for our numeric features (bust size, weight, height, size, and age) after completing the simple steps of data preprocessing as described in the paragraph above. It is interesting to note that after the series of dropping and cleaning missing values, we only lost a little fewer than 100 review records as we remained with 192452 entries. It is also interesting to see that the minimum value for age among our customers is as low as 0, meaning we probably have people renting clothes for their 0-year-old infants. The maximum values across all 5 attributes still make sense, so the dataset seems to contain little to no outliers. We then proceed with observing trends.

3.4 Exploratory Data Analysis

We observed some basic trends in our dataset. For instance, our dataset contains a disproportionately high number of max rating scores while very little negative, low ratings, as shown in figure 4 below. This is something we should certainly consider if we are going to train some type of classifier in order for it to make actually useful predictions, not just predominantly spitting out perfect scores.

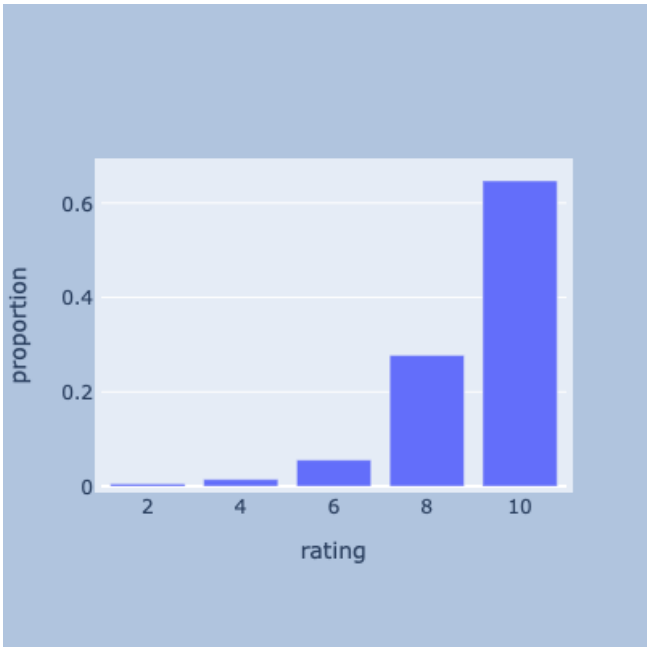

Figure 4: Rating Distribution

Among different occasions that customers rent a cloth for, “wedding”, “party”, and “formal affairs” are more prevalent, as shown in figure 5 below, which is expected.

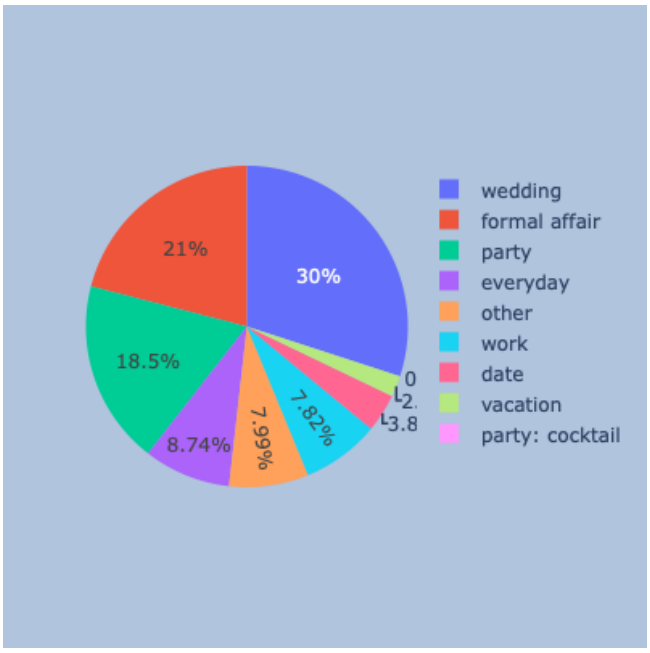

Figure 5: Rent Purporse Distribution

The distribution of the "fittingness" in figure 6 below as included in one of the most vital parts of the customer feedback is laid out as follows. Fortunately, the renting service so far seems successful as the predominant proportion of feedback indicates the clothes fit well.

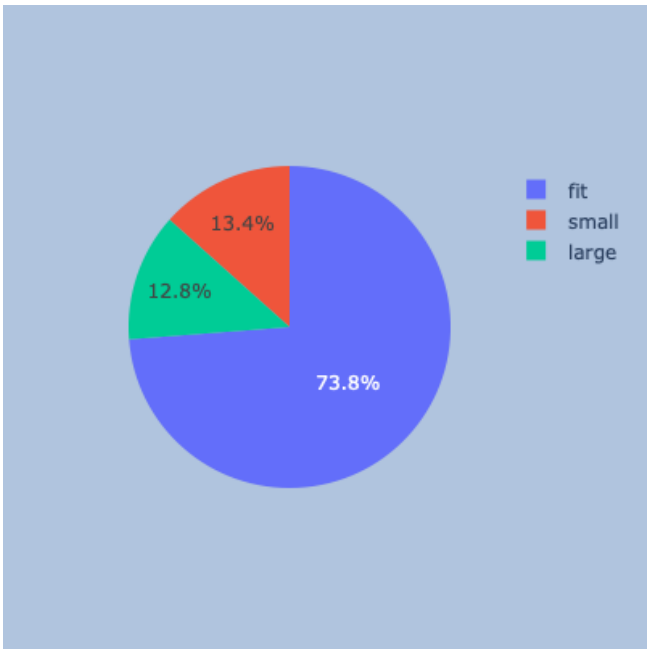

Figure 6: Fit Distribution

3.5 Interesting Text Level Findings

Since we have a piece of detailed and thoughtfully-written text review for each entry of our dataset, we decided to chain all reviews corpus together and break them back into individual words, then

assess and visualize the popularity of individual word choices and the sentiments across reviews that correspond to high rating (a perfect 10 star), average rating (a mediocre 6 star), and low rating (the lowest 2 star on the rating scale for this dataset) using word clouds.

Figure 7: 10-star rating word cloud

From the 10-star review word cloud shown in figure 7 above, we can obviously pick up some positive sentiments, such as “perfect”, “comfortable”, “compliments”, “definitely”, “loved”, etc. These are all strong evidences that indicate the customers really enjoyed the cloth.

Figure 8: 6-star rating word cloud

From the 6-star review word cloud in figure 8 above, we can still pick up a few positive sentiments, such as “great”, “fit”, “pretty”, and such. However, there is certainly less number of positive adjectives, indicating that customers no longer compliment the cloth as often as some of them do when they come across a perfect one and rate it 10 stars. But still, because of the more prevalent presence of positive sentiments as the ones observed, these clothes are overall still satisfactory, hence why they received 6 stars.

Figure 9: 2-star rating word cloud

Lastly, from the 2-star review word cloud in figure 9 above, we see quite some complaints, presumably about the size of the clothes, such as “short”, “big”, “small”, “tight”, “long”, and such. It is not hard to understand from this perspective that these clothes are not appreciated, hence they only have 2 stars.

Therefore, given the distinct sentiments and popular word representations as revealed by the word clouds from three different categories of reviews, we believe we can extract some useful insights from them and make whatever model we will eventually deploy to handle our predictive task better at differentiating different comments and assigning corresponding star ratings.

4 Predictive Task

4.1 Problem Statement

Given a new user interaction record (except the user’s rating), predict the user rating on the current rented item for every record in the Dataset.

4.2 Importance

Suppose we are able to develop a model that accurately predicts customer ratings given only the relevant review information from a customer, we can very efficiently and conveniently interpret the parameters of this model and in turn learn the subtle traits and differences that separate appreciated cloth products from the not-so-appreciated cloth products. This information can profoundly help the platform and the business learn their consumers’ preferences and improve their products, consequently generating more revenue.

4.3 Proposed Models

Baseline models:
Singular Value Decomposition (SVD)
Logistic Regression

Text Mining & NLP model:
Logistic Regression with TF-IDF
Logistic Regression with Skip Gram
Logistic Regression with Continuous Bag of Words (CBOW)

4.4 Model Evaluation

Mean Squared Error (MSE):
This metric is mainly used in assessing the performance of the baseline SVD model since it predicts continuous numeric values from the trained values drawn from the decomposed representation matrices.

F1 Score:
This metric is adopted since the majority of our models are classifiers and it is important for our classifiers to find an optimal balance in the tradeoff between precision and recall. This also addresses the imbalance in our dataset.

AUROC (or AUC):
This metric is an important alternative to F1 score since using it as evaluation pushes our classifiers to achieve the balance between true positive rate and false positive rate.

Accuracy:
This metric is used since it is a straightforward representation of how well our classifiers' predicted ratings are in proportion to the true ratings.

4.5 Data Preprocessing & Feature Engineering

On top of the preliminary missing value imputation we performed while we were having an overview of the dataset, below is the complete procedure of all feature engineering and feature selection we did in order to turn our raw data into clean data for model training.

- We extracted 2 new features "cup" and "bust number" from the original "bust size" through text manipulation.
- We one-hot-encoded all kinds of "cup" and "body_type".
- We created 12 similarity features purely from customer and item interactions.
- We created 3 features that capture the maximum, mean, and median of a particular customer's Jaccard similarity compared to all other customers that have interacted with the same clothing item by comparing the current user's item interaction history with other user's item interaction history.
- We created 3 features that capture the maximum, mean, and median of a particular clothing item's Jaccard similarity compared to all other clothing items that the current customer has interacted with by comparing the current item's user interaction history with the other item's user interaction history.
- We created 3 more features that measure and capture the same interaction as described in part a, except we switch the similarity function from Jaccard similarity to cosine similarity.
- We also created 3 more features that capture the same interaction as described in part b, except we substitute Jaccard similarity function with cosine similarity when performing the similarity calculation.
- We finally concatenate the 12 similarity features with the 2 one-hot-encoded features and the preprocessed numeric features "bust number", "weight", "height", "size", and "age" to form our training set data (For TFIDF, CBOW, and SkipGram models, we deployed additional TF-IDF feature vectorizers or word embeddings, we will explain these new additions in their respective section down below).

5 Models

5.1 SVD Baseline

5.1.1 Final Setting

Hyperparameters	Value
n_factors	100
n_epochs	30
lr_all	$1e^{-4}$
reg_all	$2.6e^{-3}$

Table 2: Final experiment setting of baseline model

5.1.2 Motivation

At first, our interaction data looks like a high dimensional sparse matrix with a lot of user-item iteractions. By using the SVD, we can perform principle component analysis (PCA) that aims to decompose a matrix (usually a set of observations) in order to find the principal axes in which the observations have the largest variance. In this way, we can find the directions in which our data are distributed, which is useful for dimensionality reduction. The two seperate representation matrix can be used to BetaU and BetaI of the latent factor model, which has proven to be very effective for dealing with interaction data.

5.1.3 Issues & Unsuccessful Attempts

When we first train the model, we tried to run 150 epochs learning rate of $5e^{-5}$. While this gives excellent training mse of 0.5324, the test accuracy suffers from overfitting. As a result, we adjust the learning rate to be $1e^{-4}$ and the model converges in 30 epochs and we get better test mse of 2.0504.

5.1.4 Results

In our first baseline model shown in table 2, we employed SVD model. We use mse as the evaluation metric because measuring the accuracy of prediction made by SVD would not make sense because SVD performs linear regression so that it's unable to map the prediction to one of the exact outputs. With default parameters of SVD, we get test mse of 2.3628 for test set. After custom fine-tuning, we improve the test accuracy to 2.0504 with learning rate of $1e^{-4}$ and 1r of $2.6e^{-3}$.

5.2 Logistic Regression Baseline

5.2.1 Final Setting

Hyperparameters	Value
max_iter	300
C	0.7
class_weight	banlanced
multi_class	multinomial

Table 3: Final experiment setting of baseline model

5.2.2 Motivation

A multinomial classification problem is very suitable for a logistic regression model with softmax output. We believe the added non-linearity of a logistic regression makes it ideal for this predictive task.

5.2.3 Issues & Unsuccessful Attempts

When we first train the model, we tried to perform one-hot encoding on every words. This makes our matrix dimension huge and generates very poor mse of 6.2352. We then decides to use Text Mining and NLP model to deal with the "review" column and focus on all the low-level features for the baseline model.

5.2.4 Results

In our second baseline model shown in table 3, we employed logistic regression model. We use all four metrics to evaluate the performance of our model. get test auroc score of 0.5397, f1 score of 0.1573, accuracy of 0.6482, and mse 2.8765. This is expected because by simply using logistic regression without extracting more meanings from the reviews or user-item interaction, it is natural to give noncompetitive results.

5.2.5 Best Result among Baseline Models

Our final setting of the hyperparameters are listed as the table 2 above. It correspond to our best final result from our custom fine-tuned SVD baseline which achieves a test mse of 2.0504.

5.3 TFIDF Model

5.3.1 Final Setting

Hyperparameters	Value
max_iter	400
C	0.62
class_weight	banlanced
multi_class	multinomial

Table 4: Final experiment setting of baseline model

5.3.2 Motivation

5.3.3 Issues & Unsuccessful Attempts

We encountered a dimension mismatch issue when building a pipeline of countVectorizer and Tfidf-Transformer. It tooks us a while to realize that the TfidfTransformer only accept pd.Series input. Therefore, we add a convert pipeline in ColumnTransformer to transform the input to pd.Series first by using squeeze() to get it to work. While the TFIDF turns out to be computationally inexpensive. It is computationally expensive. Since we generated a vocab of size 5278 and have 192452 entries in our training set We get a training set containing 153961 records. Remember that the vocabulary size is equal to the length of the TF-IDF vectors. The generated tfidf feature matrix has a size of 153961×5278, which is huge. The dimensionality curse can affect TF-IDF. In clustering, it becomes very cumbersome given the large number of records in our dataset. As a results, our model suffers from being overly complex and thus get a unsatisfatory result of get test auroc score of 0.5482, f1 score of 0.1881, accuracy of 0.6523, and mse 2.4525.

5.3.4 Results

After custom fine-tuning, we get better test auroc score of 0.7183, f1 score of 0.1881, accuracy of 0.6560, and mse 2.0504.

5.4 SkipGram

5.4.1 Final Setting

Hyperparameters	Value
max_iter	200
C	0.53
size_features	300
n_jobs	10
class_weight	banlanced
multi_class	multinomial

Table 5: Final experiment setting of baseline model

5.4.2 Motivation

As we can see above, the tfidf model doesn't work well because of the exploding dimensionality, which makes the number of parameters that need to be learned by the logistic regression unreason-

able, which can also be a potential overfit. Therefore, we decided to use SkipGram to embed the vocabs while maintaining a reasonable dimension so that learning is still effective.

5.4.3 Issues & Unsuccessful Attempts

The training was extremely slow because we didn't set the n_jobs to 10. Therefore, only 1 CPU was used to run the model and it took very long. We didn't set the n_workers earlier when training the tf_idf model either, but as we discussed, tfidf is much faster to train, therefore we overlooked the fact that Word2Vec model is much harder to train and thus need parallel computing.

5.4.4 Results

After custom fine-tuning, we get test auroc score of 0.8417, f1 score of 0.2877, accuracy of 0.6965, and mse of 1.9651.

5.5 CBOW

5.5.1 Final Setting

Hyperparameters	Value
max_iter	200
C	0.67
size_features	100
n_jobs	10
class_weight	banlanced
multi_class	multinomial

Table 6: Final experiment setting of baseline model

5.5.2 Motivation

CBOW and SkipGram are twins and they work in the exact reverse way. What's more, we believe that CBOW performs better when the vocabulary is larger. Our vocabulary has more than 30000 words and we want to give it a try.

5.5.3 Issues & Unsuccessful Attempts

The training procedure of CBOW is very easy after building up the structure for Word2Vec, the only notable thing is that we forgot to change the parameter sg in Word2Vec to 0 (meaning CBOW) but left it 1 (meaning SkipGram)

5.5.4 Results

After custom fine-tuning, we get test auroc score of 0.7888, f1 score of 0.2514, accuracy of 0.6715, and mse of 2.2414.

5.6 SkipGram + Similarity

5.6.1 Final Setting

Hyperparameters	Value
max_iter	200
C	0.57
size_features	100
n_jobs	10
class_weight	banlanced
multi_class	multinomial

Table 7: Final experiment setting of baseline model

5.6.2 Motivation

As we can see, the SVD baseline worked quite well. This makes us wonder whether capturing the user-item similarities in our dataframe would result in better accuracy in the logistic regression in SkipGram model.

5.6.3 Issues & Unsuccessful Attempts

We miscalculated the user and item similarity first. This result in the mse of $SkipGram_{Sim}$ being worse than the $SkipGram$. After we retifying it, the mse becomes better and this model becomes the best model overall.

5.6.4 Results

After custom fine-tuning, we get test auroc score of 0.8451, f1 score of 0.3048, accuracy of 0.6968, and mse of 1.8856.

5.6.5 Compare & Contrast

	Logistic Regression (base)	TFIDF	CBOW	SkipGram + Similarity	SkipGram	SVD (base)
auroc	0.5397	0.7183	0.7888	0.8451	0.8417	NaN
f1_score	0.1573	0.2147	0.2514	0.3048	0.2877	NaN
accuracy	0.6482	0.6560	0.6715	0.6968	0.6966	NaN
mse	2.8765	2.3993	2.2414	1.8856	1.9651	2.0504

Figure 10: All Model Comparison

This figure above shows the comparison of all models performance.

	SkipGram	SkipGram + Similarity	improvement
auroc	0.8417	0.8451	0.0034
f1_score	0.2877	0.3048	0.0171
accuracy	0.6966	0.6968	0.0002
mse	1.9651	1.8856	-0.0795

Figure 11: SkipGram + Similiarity vs. SkipGram

This figure above shows the comparison of SkipGram + Similiarity compare with SkipGram.

	Logistic Regression (base)	SkipGram + Similarity	improvement
auroc	0.5397	0.8451	0.3055
f1_score	0.1573	0.3048	0.1475
accuracy	0.6482	0.6968	0.0486
mse	2.8765	1.8856	-0.9909

Figure 12: SkipGram + Similiarity vs. Logistic Regression Baseline

This figure above shows the comparison of SkipGram + Similiarity compare with Logistic Regression Baseline.

	SVD (base)	SkipGram + Similarity	improvement
auroc	NaN	0.8451	NaN
f1_score	NaN	0.3048	NaN
accuracy	NaN	0.6968	NaN
mse	2.0504	1.8856	-0.1648

Figure 13: SkipGram + Similiarity vs. SVD

This figure above shows the comparison of SkipGram + Similiarity compare with SVD Baseline.

As we can see, the SkipGram + Similarity model has achieved the best score for every eveluation metrics, with test auroc score of 0.8417, f1 score of 0.2877, accuracy of 0.6965, and mse of 1.9651.

6 Results and Conclusion

The graph below presents the testset performance of all our proposed models. We cumulatively assess the performance of each by checking it against all 4 metrics, MSE, AUROC, F1 score, and accuracy, with the only exception being our baseline SVD model because it only outputs continuous rating values instead of categorically classifying them. Our most ideal logistic regression model using Skip Gram word embedding plus all 12 similarity features consistently outperformed all other models and its winning marginal is statistically significant. In particular, this model has the best test set MSE at 1.8856, best test set accuracy at 0.6968, best test set F1 score of 0.3048, best test set AUROC 0.8451.

	Logistic Regression (base)	TFIDF	CBOW	SkipGram + Similarity	SkipGram	SVD (base)
auroc	0.5397	0.7183	0.7888	0.8451	0.8417	NaN
f1_score	0.1573	0.2147	0.2514	0.3048	0.2877	NaN
accuracy	0.6482	0.6560	0.6715	0.6968	0.6966	NaN
mse	2.8765	2.3993	2.2414	1.8856	1.9651	2.0504

Figure 14: All Proposed Model Testset Results

Its outstanding performance is in alignment with our expectations as Skip Gram is particularly specialized in predicting context from individual words, and the context (whether it is overall positive, neutral, negative) of a review directly correlates to the rating the product receives. It is interesting

we did not discover this combination of Skip Gram word embedding and supplying similarity scores to be the best performing candidate from the start.

	SkipGram	SkipGram + Similarity	improvement
auroc	0.8417	0.8451	0.0034
f1_score	0.2877	0.3048	0.0171
accuracy	0.6966	0.6968	0.0002
mse	1.9651	1.8856	-0.0795

Figure 15: Skip Gram with Similarity Compared to Just By Itself

We did not discover this just from pure coincidence either. We had this intuition from observing the performance of the baseline SVD model. The SVD model essentially relies solely on customer product interactions and factorizes them into representation matrices of smaller dimensions. With its test set MSE being only second to Skip Gram based models, we realized that the interactions in our dataset may be vital to making accurate rating predictions, hence why we came up one step further with the Skip Gram plus similarity feature model. Lastly, it is crucial to tie our established model back to the thesis we started with. In the beginning, we said that our problem statement can have significant application in understanding customer preferences. For the Skip Gram plus similarity model, while it may be harder to interpret the trained parameters (in this case, the weights corresponding to each feature) of the classifier, especially the ones for the word embeddings since it resides in a hidden space that we can hardly conceptualize, since we know there are still ordinary, categorical variables, such as the one-hot-encoded body types, we can for the very least interpret the weights of these attributes and get a sense of how on category may influence the rating prediction. From the graph below, we can see that for instance, the variable "age" has a positive weight of 0.006191 associated with it, meaning the age of the customer has a positive influence on the rating he/she may give to the cloth (perhaps older people are nicer in giving out compliments). Alternatively, we see that the height of a customer has a negative weight of -0.034072 associated with it, meaning that for taller customers renting out clothes, they may more closely examine whether the cloth fits their needs, thus giving out stricter ratings. Examining weights in association with the interpretable features in such a way helps the merchants understand which group of their customers may require more attentive service, which group is more generous, etc. This information can help them maintain and even improve their products and business.

feature	weight
bust_number	0.053838
weight	0.002869
height	-0.034072
size	-0.013356
age	0.006191
cup_a	0.016090
cup_aa	0.004879
cup_b	-0.038426
cup_c	-0.007692
cup_d	0.028213
cup_d+	0.008662
cup_dd	-0.007230
cup_ddd/e	-0.018116
cup_f	-0.002192
cup_g	0.002937
cup_h	0.001954
cup_i	0.000064
cup_j	0.001559
body_type_apple	0.031831
body_type_athletic	-0.051181
body_type_full bust	0.002061
body_type_hourglass	-0.052512
body_type_pear	0.066139
body_type_petite	0.007601
body_type_straight & narrow	-0.013236
intercept	-0.009297

Figure 16: Features and Their Weights

For further work, we would really like to implement the metric learning method mentioned in the paper by Misra et. al.. Like we briefly mentioned, our dataset is imbalanced and we did set the class weights when training our classifiers to be balanced to naively offset this shortcoming. However, it would be great to have this issue mitigated before training. Moreover, given time, we would like to continually elaborate our model implementation. One of the topics we would want to explore is temporal dynamics. For this dataset, we did not include the temporal relationship because we explored its relationship with respect to other variables and deemed it not helpful. But if we were able to encode it and pass it in as one of our features, we could actually see where it has effectiveness. The other model we would like to deploy is transfer learning with some pre-trained BERT. We believe this can help us generate more meaningful word embeddings and thus uncover stronger semantics and contexts from review texts. Lastly, we would really like to incorporate our model into a recommendation system pipeline so that it not only makes predictions from a complete customer-

product interaction, it also generates interaction targets from seeing only the customer or the product, which is much more applicable and analogous to a real world recommendation system.