Instructor: Julian McAuley

FA22 CSE 158

29 November 2022

Assignment 2 : Amazon reviews Rating Prediction Report

Data

We used one of the dataset from the collection of datasets produced by Professor Julina McAuley's lab. The dataset consists of video game reviews from the Steam platform. It contains 7,793,069 reviews in total, data from 2,567,538 users and 15,474 items. The data can be downloaded from this link:

https://jmcauley.ucsd.edu/data/amazon_v2/categoryFiles/Video_Games.json.gz

While there are complete raw review datasets, these datasets are too large for the purpose of this educational project. Therefore, we used the smaller version of the raw dataset — the 5-core dataset, which is the subset of the data in which all users and items have at least 5 reviews.

Data Cleaning

After downloading the json file, we represent the data with a Pandas DataFrame, where each row represents a single review. It has the following columns: overall, verified, reviewTime, reviewerID, asin, reviewerName, review Text, summary, unixReviewTime, vote, style, and image.

	overall	verified	reviewTime	reviewerID	asin	reviewerName	reviewText	summary	unixReviewTime	vote	style	image
0	5.0	True	10 17, 2015	A1HP7NVNPFMA4N	0700026657	Ambrosia075	This game is a bit hard to get the hang of, bu...	but when you do it's great.	1445040000	NaN	NaN	NaN
1	4.0	False	07 27, 2015	A1JGAP0185YJI6	0700026657	travis	I played it a while but it was alright. The st...	But in spite of that it was fun, I liked it	1437955200	NaN	NaN	NaN
2	3.0	True	02 23, 2015	A1YJWEXHQBWK2B	0700026657	Vincent G. Mezera	ok game.	Three Stars	1424649600	NaN	NaN	NaN
3	2.0	True	02 20, 2015	A2204E1TH211HT	0700026657	Grandma KR	found the game a bit too complicated, not what...	Two Stars	1424390400	NaN	NaN	NaN
4	5.0	True	12 25, 2014	A2RF5B5H74JLPE	0700026657	jon	great game, I love it and have played it since...	love this game	1419465600	NaN	NaN	NaN

Then, we converted the entries in reviewTime to pandas datetime type in order to conduct exploratory data analysis conveniently. We eventually kept only overall and reviewText columns, dropping all other columns irrelevant to our model and prediction after exploratory data analysis.

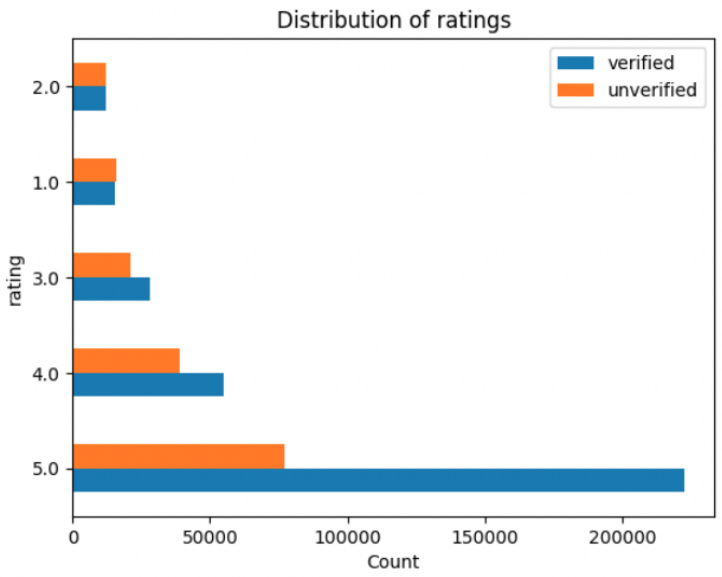
Data Preprocessing

We shuffled the dataframe, then splitting the shuffled data frame to two parts containing ⅕ and ⅘ of the rows, respectively. Then we sampled 20,000 rows from the first ⅕ part to form a testing set, and sampled 100,000 rows from the second ⅘ to form a training set. Our feature function, which takes in a string and returns a string, converts all the characters to lowercase, strips all trailing whitespace and removes all the punctuation from the review text. We applied the feature function to all the review texts in both the training and testing set.

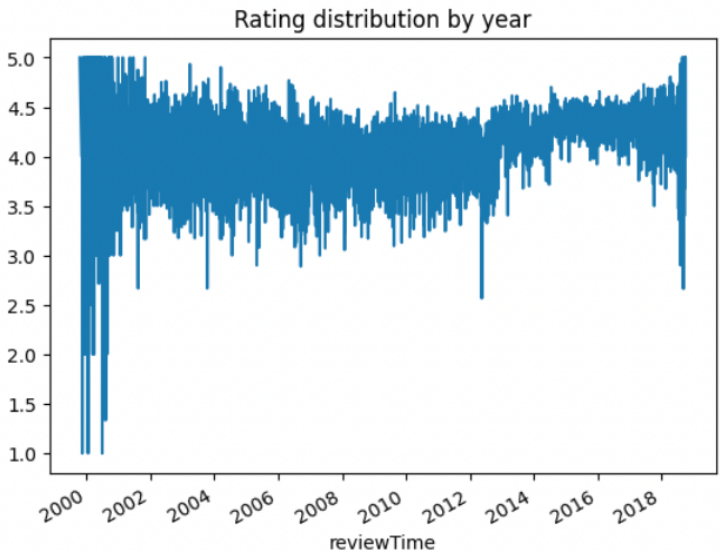
Exploratory Data Analysis

After calculating the group rating mean of verified and unverified reviews, we find that verified reviews have higher rating mean than unverified reviews. Verified reviews have a mean rating of 3.91, while unverified reviews have a mean rating of 4.37. Noticing the interesting difference between the means, we want to know what is the main reason that leads to this difference through plotting the distributions of ratings of verified reviews as well as the distribution of ratings of unverified reviews. We generated the following plot:

From the plot, we can find that verified and unverified reviews have similar amounts of 1 and 2 star ratings, while verified reviews tend to have slightly more 3 and 4 star ratings and have significantly more 5 star ratings than unverified reviews. Thus, the difference of rating means primarily comes from the discrepancy in the amount of 5 stars ratings.



Besides verification, we also speculate that rating might also be related to review time. The following plot represents the ranges of rating in different years:



We can observe that during 1999~2001, the earliest several years in the dataset, the review ratings tend to have higher variance, which means there are lots of high ratings as well as low ratings. After that, the ratings seem to stabilize within a range, presenting lower variance, until 2018, when there suddenly appear more high and low ratings again, but the range is not as wide as the ones in 1999~2001. Then, we calculated the mean rating each year and made a plot:

There does not exist an explicit linear correlation between review time and average rating. However, the annual average rating does fluctuate a lot. The average rating formed a decreasing trend starting from 1999, reached the bottom in around 2006 and started to rebounce, until 2015, when the annual mean rating seemed to decrease again.

Since our goal is to predict the rating of a given review with a decent accuracy, the part of a review that gives the most useful and deterministic information is neither verification nor review time, since our exploratory data analysis does not demonstrate a reasonable and reliable relationship between these two factors and the review ratings. Therefore, we turned to review text instead, which is the most informative part of the data about the rating. We are going to perform sentiment analysis on the review text to predict the review rating in our project.

Predictive Task

As described above, we would work on predicting Amazon Reviews ratings on video games from corresponding review text. We did this mainly for two reasons. Firstly, the lack of ItemId makes it difficult to perform models that involve the interaction between UserID and ItemID. Both Collaboratively filtering, and latent factor model needs sample data of User and Item. Secondly, we think review text could be a better inference from which we could predict the ratings. There are many words that feature a 5-star review while many other words could feature the 1-star reviews. So basically we could rely on only reviewing text data ("reviewText" column of our dataframe) of each user to perform ratings predictions with many basic to advanced NLP techniques.

As for the metrics and baseline of our project. We mainly have evaluative metrics for ratings prediction: Category Accuracy and MSE(RMSE). We have two metrics because there are two ways to interpret the ratings prediction. In one way we can treat it as a numerical regression prediction task, upon which we can use MSE/RMSE for evaluation. On the other hand, ratings, in some sense, could represent the sentiment or category of the review, where a 5 could mean highly recommend and 1 means not recommend at all. While MSE/RMSE is taken for reference, later in the analysis of result, we would mainly use accuracy as metrics because most models conducted in this project are classifications models instead of regression models. For the baseline model, we used simple BoW with ridge regression. We process the data by counting the occurrence of each word to form a BoW vector, and use that vector to perform regression. Then we found the regression may not work that well on such prediction tasks and we switched to classification prediction. We use TF-based and another TF-IDF-based model with Logistic regression and SVM models for rating classification prediction. We preprocess the data both by removing stop words and use Sklearn TF/TFIDF vectorizer and input the transformed vector to Logistic Regression

models. Eventually, we use a pre-trained Word2Vec embedding model to encode each review text, and input the encoded text to a self-constructed LSTM model. After 50 epochs of training, the classification accuracy achieved the highest of all, around 65%.

Baseline Regression Model: Count-based BoW model with Ridge Regression

Firstly, at the start, we use the ridge regression method to predict rating mainly MSE and RMSE wise. Our assumption will be that it will have a good performance on MSE and RMSE, but may be behind on accuracy calculation, since regression will produce float prediction results which compare to our rating are all whole numbers. We choose a count-based BOW model to pre-process the data, that's enough for an easy opening. By counting how many times each word appears in all of the reviewText, we sort the words' frequency in reverse order and take the top 1000 words to turn each arbitrary reviewText in the train dataset into fixed-length vectors. We then use the processed reviewText vectors and rating score in the train dataset to fit a Ridge regression model. Finally we use the unseen reviewText vectors in the test dataset to fit our final model.

We firstly remove any space and punctuation in reviewText.

```
def feature(t):
    t = str(t)
    t = t.lower().strip()
    t = [c for c in t if not (c in punct)]
    t = ''.join(t)
    return t
```

Then we count the word with the top 1000 frequency among all words appearing in reviewText.

```
def wordCountF(data):
    wordCount = defaultdict(int)
    for d in data:
        for w in d.split():
            wordCount[w] += 1
    return wordCount
```

```
reviewTaken = df['reviewText'].apply(feature)
countReview = wordCountF(reviewTaken)
counts = [(countReview[w], w) for w in countReview]
counts.sort()
counts.reverse()
words = [x[1] for x in counts[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

Finally we turn each reviewText in the training dataset into a fixed length vector by comparing the frequency of each word appearing in reviewText also appears in our pre-process 1000 word bag.

```
def feature2(datum):
    finalR = []
    for d in datum:
        feat = [0]*len(wordSet)
        for w in d.split():
            if w in words:
                feat[wordId[w]] += 1
                flag = 1
        feat.append(1) #offset
        finalR.append(feat)

    return finalR
```

```
review_trainS = feature2(reviewTakenTrain)
review_testS = feature2(reviewTakenTest)
```

We use the processed train dataset and the rating score to fit the Ridge regression model then and predict with the test dataset.

```
clf = linear_model.Ridge(alpha=1.0, copy_X=True, fit_intercept=False, normalize=False,
    tol=0.001) # MSE + 1.0 l2
clf.fit(review_trainS, ratings_train)
theta = clf.coef_
predictions = clf.predict(review_testS)
```

✓ 21.4s

As you can see, Our RMSE finally reached 1.15 and our MSE reached 1.33, meanwhile our category accuracy is pretty low since we have to round float prediction results to compare with our actual all whole number rating dataset.

```
print("RMSE for ratings prediction using BOW is {}".format(RMSE(predictions,ratings_test.values)))
print("MSE for ratings prediction using BOW is {}".format(MSE(predictions,ratings_test.values)))
print("Accuracy of rating predictions using BOW is {}".format(np.mean(predictions.round()==ratings_test.values)))
```

[180] ✓ 0.1s Python

```
RMSE for ratings prediction using BOW is 1.1572781871950861
MSE for ratings prediction using BOW is 1.339292802557545
Accuracy of rating predictions using BOW is 0.303
```

From there we will take the result of MSE and RMSE of the count-based BOW model and Ridge regression. We will then switch on the classification prediction to increase our category accuracy.

Baseline Classification Model: TF(IDF)-based model with Logistic Regression/SVM

We picked logistic regression and support vector machine models with balanced class weights as two baseline classification models. Because these two are well established and widely used models for classification tasks.

TF based model with Logistic Regression and SVM:

We first used a simple term-frequency data encoding scheme to process the dataset. Because this is similar to the method we have used in previous assignments. The performance of logistic regression and SVM is verified from previous results, which functions as robust baselines for the comparison against our proposed model. With stop words removed and all the review text encoded into numerical values based on term frequency, our logistic regression model reached an accuracy of 0.595 and our SVM model reached an accuracy of 0.628 on the test set.

TF-IDF based model with Logistic Regression and SVM:

We then added IDF (inverse document frequency) to the TF data processing pipeline to better incorporate terms in different video game reviews. Because video games have different types and playstyles, the terms and other words which contain important information regarding the rating are likely to vary for different games. This causes the simple TF approach to ignore many words that convey important information about the rating because they may not appear frequently under the entire word space. The IDF method can effectively extract the words that are information-rich but lacking in overall frequency of the word pool. With TF-IDF, our logistic regression model reached an accuracy of 0.581 and our SVM model reached an accuracy of 0.634 on the test set.

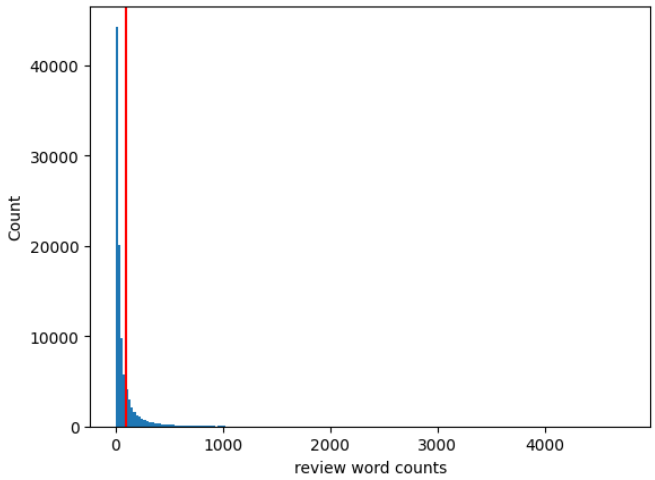
Here, the decrease of performance for linear regression model from TF to TF-IDF is likely because the data is harder to separate with linear functions. The complexity of input features has increased as the

inputs now have the encoding of low-frequency but heavily weighted words. Thus the data has become less linearly separable, resulting in the poor performance of linear regression model.

Proposed Classification Model: LSTM with Word2Vec embedding model

Finally, we used a more advanced LSTM based on Word2Vec for ratings classification. We think this is a viable method because LSTM, compared with other basic models and RNN, is able to capture long range word semantic dependency and make more accurate predictions on the meaning of sentences. We use a pre-trained Word2Vec embedding model (link: https://tfhub.dev/google/Wiki-words-250/2) from tensorflow hub. Our proposed model is based on Word2Vec because we regard it can get more sense of what the word "means" compared with baseline TF-IDF vectors which just represent how important a word is to each review text.

The embedding model has 250 dimensions, and due to limited computation power, we restrict each review text to just encode the first 200 words because most word lengths of review text are within 100 words. And it would be a waste of computation if we take the maximum length of all trained reviews. So in general, each encoded review text would be a (100, 250) dimensional vector. Also it should be noted here that we only sample 100000 review texts from shuffled training dataset for training. Again this is due to limited computation power as LSTM's time complexity is really high, and that 100000 reviews are already enough for LSTM to learn the patterns.



Also, for the ease of training, we one-hot encoded each rating. For example, a rating of 5 is converted to [0,0,0,0,1]. Our LSTM model is quite simple and intuitive. It contains a LSTM layer, a dropout layer, and two fully connected layers, one with Relu activation and the final output with softmax activation and dimension of 5. Since we regard this prediction task mainly as a Classification task to classify review text, we use categorical cross entropy as our loss function along with adam optimizer and accuracy as metrics. On one hand this means that our classification accuracy could be higher, on the other hand, without aiming to minimize the MSE/RMSE, they could be even larger than the baseline model.

```
from tensorflow.keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential

model = Sequential()
model.add(LSTM(32))
model.add(Dropout(0.2))
model.add(Dense(20, activation='relu'))
model.add(Dense(5, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
history=model.fit(trainX, trainY,epochs=50)
```

After 50 epochs of shallow training, the training accuracy could achieve around 70% while test accuracy could achieve 67% on sampled shuffled test data. However, as we expected before, the model didn't perform well on MSE and RMSE, as the model mainly aims to optimize accuracy instead of minimize MSE/RMSE.

```
RMSE for ratings prediction using LSTM is 1.1396051947933548
MSE for ratings prediction using LSTM is 1.2987
Accuracy of rating predictions using LSTM is 0.6727
```

Literatures Review

Our Dataset are Amazon reviews Video Games dataset (2018) (link:https://nijianmo.github.io/amazon/index.html), and updated version of Professor Julian McAuley's Amazon Product Data (https://jmcauley.ucsd.edu/data/amazon/).

There are many similar studies that we look at for reference in the papers by Jianmo Ni et al. This paper works and reviews data from Yelp and Amazon Clothing, and it utilized models including BoW-XGboost, CNN, LSTM, BERT for

classifying reviews as being good or bad. In our project, we make a slight modification to the idea and predict the ratings as categories of the extent to which users think the video game is good or bad (from 1-5). Also, due to limited computation power, we use some much simpler models from lectures and by original design for prediction.

In my opinion, BERT might be a really state-of-art model used for text inference and classification. This is because BERT is very versatile, could be pre-trained and fitted into many distinct applications just by modifying and fine-tuning the last output layer (Jacob Devin, et al, 2018). Also, since BERT incorporates an attention mechanism, it could be much better at capturing and highlighting the keywords that could determine the attitude or sentiments of review text. For example, it might be able to capture words like “excellent”, “great game” etc to make more accurate predictions. However, while LSTM could somewhat “understand” the words’ meaning especially after embedding, it might not be acute enough compared with the much more advanced BERT.

In another paper about detecting offensive language in Social networks (Ashiwini Kumar, et al, 2021), it used annotated tweets as a dataset to classify whether a sentence is offensive. In their results, LSTM and BERT achieve better results than most other Machine Learning models (Ashiwini Kumar, et al, 2021), which is similar to our conclusion that LSTM could work better than other basic Machine Learning classification models.

Results and Conclusions

Our proposed LSTM with Word2Vec embedding model reached an accuracy of 0.6727. This model outperformed all other baselines in prediction accuracy on the test set as shown in the table below. The Word2Vec feature representation worked well with LSTM and outperformed other feature representations like BOW, TF and TF-IDF. Our model is 6.2% better in prediction accuracy on the test set compared to SVM with TF-IDF, the best-performing baseline model, which is significant enough (>5%) to draw the conclusion that our model is better than SVM with TF-IDF.

Model Type	LSTM + Word2vec	Logistic Regression + TF	Logistic Regression + TF-IDF	SVM + TF	SVM + TF-IDF	Ridge + BOW
Accuracy	**0.6727**	0.595	0.581	0.628	0.634	0.303

The TF-IDF representation (for SVM) and Word2Vec representation worked well and the TF and BOW representations did not work well comparatively. For the Word2Vec processing, we used the first 100 most-frequent words in the training dataset for encoding. For our proposed model, we first used an LSTM layer with 32 as the output dimension to extract information from the Word2Vec encoded input data, and then we used a dropout layer with rate of 0.2 to prevent overfitting and add robustness to the model. Then we added two linear layers with a dimension of 5 and softmax as the activation function for the last linear layer in order to make categorical predictions. We then trained our model for 50 epochs, which is the suitable number of epochs for our model to converge but not overfit after multiple trials with different numbers of epochs. Our model was able to achieve significant result improvement thanks to the Word2Vec's ability of mapping the target word and its context words (words around the target word) into one vector space and LSTM's ability of capturing long range word semantic dependency. With these two models combined our proposed model is able to utilize both short range and long range word semantic information to better comprehend the review data for categorical rating predicting. Other baseline methods such as SVM and linear regression with TF and TF-IDF failed to reach such accuracy because they all take the processed review data as a whole and treat each word independently. This prevents the models from capturing the sublinear relationship among words, which hinders the models from making as-good predictions.

Reference:

Ni, J., Li, J., & McAuley, J. (2019, November). **Justifying recommendations using distantly-labeled reviews and fine-grained aspects.** In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 188-197).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). **Bert: Pre-training of deep bidirectional transformers for language understanding**. *arXiv preprint arXiv:1810.04805*.

Kumar, A., Tyagi, V., & Das, S. (2021, December). **Detection of Offensive Language in Social Networks Using LSTM and BERT Model**. In 2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA) (pp. 546-548). IEEE