CS 410 Text Information System
Final Report

Sentiment Analysis on Tripadvisor Hotel Review

Group Name: Team Experts

Group members:
Zhiyan Jiang (zjiang2, Captain)
Xin Peng (xinp2)
Shan Shan (ss163)

Dec.3, 2022

1. Introduction

Opinion mining is an important component of text mining. As part of opinion mining, sentiment analysis is a special case of text categorization regarding the degree of emotional variation contained in the opinions. The problem is defined as using an opinionated text object as input, and output a sentiment tag or label.

Sentiment analysis can be performed at three levels with different complex levels. The first level is polarity analysis. Each review is splitted into words, which are given a sentiment score value. Then an overall polarity value is calculated for this review. This level is simple because it only engages review context rather than any ratings. The relative sentiment score for each word in a review has been predetermined and not context-specific. From Natural Language Processing (NLP) perspective, this level only includes part of speech (POS) tagging.

The second level is overall rating prediction. This level is complicated because it is essentially supervised learning (multiclassfication) and involves overall rating. The categories are typically more than that in the polarity analysis. In addition, skewed (i.e., unbalanced) training data distribution imposes obstacles for analyzing the minority categories.

The third level is latent aspect rating analysis (LARA). This level is much more complicated because latent aspects must be created and rated. Take the hotel review for example, the latent aspects can be “rate”, “amenities”, “services”, among others. This would require a significantly larger and more decent dataset to cover all the aspects. Moreover, the latent aspects (i.e., topics) and corresponding word distribution must be handled during the analysis.

In this project, the authors perform sentiment analysis on a specific dataset of hotel reviews through data cleaning, polarity analysis and overall rating prediction. The goal is to practice the data cleaning skills, sentiment analysis procedures, as well as interpretation skills of data, and have a better understanding of text mining and analytics.

This project includes data cleaning, polarity analysis, and overall rating prediction. The last part was implemented using two individual approaches - logistic regression and Long-short term memory. The predicted results are also interpreted and discussed. The code is written in Python, and major libraries include sklearn, nltk, and numpy.

Contributions:

Zhiyan Jiang: Performed polarity analysis and visualized the quantity for all reviews and each rating. Performed overall prediction using logistic regression, visualized and evaluated prediction results, and interpreted the results.

Xin Peng: Investigated Google Colab implementation and availability for project use purpose and implemented Google Colab mount in for data. Implemented LSTM related model execution and scenario testing and debugging.

Shan Shan: Performed data analysis, data cleaning to change words to lowercase, remove special characters and stop words, stemming and lemmatization, remove most frequent words, word cloud. Worked on the LSTM model but was not able to get desired results.

1.1 Instruction of using the code

1. The code is located at below GoogleColab link:
https://colab.research.google.com/drive/1PX5up5arRzdt3UbX5EDBnpVX1io3z4gU?usp=sharing

2. Download the dataset at below link:
https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews
3. Make sure to have a google drive account, using the one of @illinois.edu should work fine.
4. Save the dataset to the google drive folder. Make sure to create a folder called “projdata”, then save the dataset in that folder, make sure the dataset name is “tripadvisor_hotel_reviews.csv”.
5. Simply click through each step and you should get the result.

2. Methodology

2.1 Dataset and platform

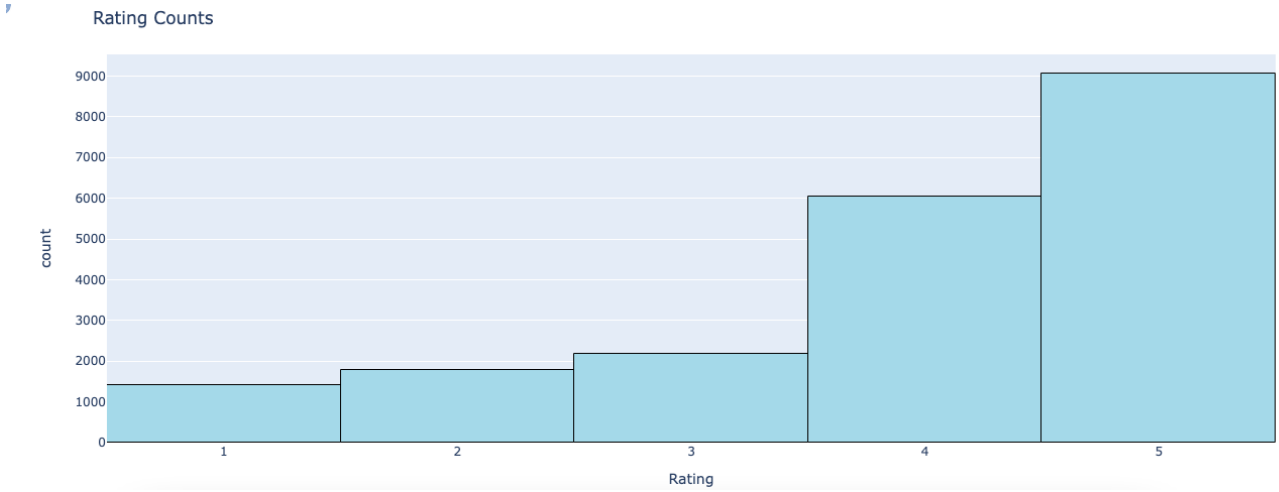
The dataset we selected is the TripAdvisor Hotel Reviews, which is located at https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews. The reason we chose this dataset is that hotel reviews can be important for travelers to understand which hotels are good or bad, and makes it easier to better plan the upcoming trips. By performing sentiment analysis of the dataset, it can help us understand why reviewers feel happy or unhappy about the hotel or what makes a hotel good or bad.
The dataset has over 20000 rows of reviews and ratings from various reviewers. There are 2 columns in the dataset, the first column is the review comments, the second column is the rating from 1 to 5, which can be considered as labels for the sentiment analysis.
The platform we used is the google colab, which makes it easier to use the imported libraries and collaborate on the coding. To use the code, the user can simply click through each step to get the output of each cell.

2.2 Data Analysis and Cleaning (Shan)

2.2.1 Data Analysis

Firstly, data analysis is performed to see how the reviews and ratings are distributed. The results show that there are 1421 rating 1 reviews; 1793 rating 2 reviews; 2184 rating 3 reviews; 6039 rating 4 reviews and 9054 rating 5 reviews. The analysis shows that the majority of people are giving 4 and 5 ratings, which means there are more “positive” reviews.

Table 1. Rating Distribution



2.2.2 Change to lowercase

Based on a quick glance through the dataset, the majority of the review text are clean, however to ensure the model performed later can produce better accuracy, there are several data cleaning and preprocessing steps being performed. First, we changed the review text to all lower cases, this is to ensure the analysis model will treat all words equally.

2.2.3 Remove Special Character

Several reviews have some special characters in them, so we decided to remove them and only keep the numbers and letters. Some cases where special characters are removed are shown below.

```
nice rooms not 4* experience hotel monaco seattle good hotel n't 4* level.positives large bathroom mediterranean
nice rooms not 4 experience hotel monaco seattle good hotel nt 4 levelpositives large bathroom mediterranean suit
```

Figure 1. Remove Special Character Example 1

```
n car 38. location central downtown street w. it__ç_é_ quick walk points tourist business, yes spa
car 38 location central downtown street w it quick walk points tourist business yes space needle b
```

Figure 2. Remove Special Character Example 2

2.2.4 Remove Stopwords

We also removed stop words such as “the”, “a”, “an”, etc, this is to ensure we keep only the most important and meaningful information. The library we used here is the Natural Language Toolkit (NLTK), and the package is nltk.corpus.stopwords.words

2.2.5 Stemming

Next, we performed stemming of the review text. The words are usually in different formats, the purpose of stemming is to convert words in different formats to the same common base form. Stemming is the process of removing the last few characters of the words. The library we used here is the Natural Language Toolkit (NLTK). And the package is nltk.porter.PorterStemmer.

Some cases where stemming changes the results are shown below.
The first example shows that “location” is changed to “locat”, “friendly” is changed “friendli”, “apartment” is changed to “apart”, “building” is changed to “build”.

```
great location staff location price make older hotel good choice staff friendly looking modern place old studio apartment building
great locat staff locat price make older hotel good choic staff friendli look modern place old studio apart build run hotel great
```

Figure 3. Stemming Example 1

The second example shows that “inconsistent” is changed to “inconsist”, “arrived” is changed to “arriv”, “spending” is changed to “spend”, “conservatory” is changed to “conservatori”, “initially” is changed to “initi”.

```
inconsistent arrived eliot spending wonderful night lenox eliot closer boston conservatory initially wanted stay no room night stayed
inconsist arriv eliot spend wonder night lenox eliot closer boston conservatori initi want stay no room night stay lenox small charm l
```

Figure 4. Stemming Example 2

2.2.6 Lemmatization

Lemmatization is another text normalization process which is similar to stemming, however, it considers the word context and uses vocabulary and morphological analysis of the words. The library we used here is the Natural Language Toolkit (NLTK). And the package is WordNetLemmatizer.
The first example shows that compared to stemming, the change of words seem to make more sense, mostly it just removes the “s” at the end of the word.

```
nice complimentary shuttle copley place helpful generally convenient suites comfortable spacious hotel dining extremely expensive ov
nice complimentary shuttle copley place helpful generally convenient suite comfortable spacious hotel dining extremely expensive ove
```

Figure 5. Lemmatization Example

Comparing the stemming and lemmatization, lemmatization seems more informative, while stemming requires less knowledge. We decided to keep both processes since they both generate the foundation of inflected words.

2.2.7 Remove Most Frequent Words

The next step is to find the most frequent words that are common for all the ratings. The rationale is that if the words appear frequently in all the ratings, then it is meaningless for the classification. The below words are the results, we removed most of these words, we kept the word “good” since it is a key opinion word which determines if the review is positive or negative.

```
{'stayed', 'day', 'time', 'hotel', 'stay', 'room', 'good', 'rooms', 'nt', 'staff', 'night', 'service'}
```

Figure 6. Most Frequent Words

2.2.8 Word Cloud

After all the data cleaning steps performed, we generated word clouds for each rating. Below are the results. The library being used is WordCloud.
For rating 1, we can see there are some more frequent words of “bad”, “old”, “dirty”, “worst” which indicate negative opinion.

Figure 7. Word Cloud of Rating 1

For rating 2, we can see there are more frequent words of “bad”, “problem”, “ok”, although there are also frequent words of “good”.

Figure 8. Word Cloud of Rating 2

For rating 3, we can see there are more frequent words on “great”, “nice”, “good”, “clean”, but there are also some frequent words of “bad”, “small”, which shows a mix of positive and negative opinions.

Figure 9. Word Cloud of Rating 3

For rating 4, we can see the frequent words are “nice”, “good”, “great”, there are not so many words with negative reviews.

Figure 10. Word Cloud of Rating 4

Similar to rating 4, there are a lot more positive words such as “nice”, “great”, “good”, “best”, “fantastic”, “excellent”, “wonderful” for rating 5, while not so many negative words.

Figure 11. Word Cloud of Rating 5

2.3 Polarity analysis (Zhiyan)

Sentiment label or tag can be defined in two ways - polarity analysis that has categories of positive, negative and neutral, and emotion analysis that categorize more specific feelings of the opinion holder. Due to the nature of the dataset, only polarity analysis is performed.
The polarity analysis is implemented using Valence Aware Dictionary and Sentiment Reasoner (VADER). The SentimentIntensityAnalyzer from vaderSentiment library is used to initialize a sentiment analyzer. This analyzer will determine a sentiment score between -1 (most negative) and 1 (most positive), based on the words in a review. The definition of "positive", "negative", and "neutral" tends to be more arbitrary. In this project, reviews with a sentiment score greater than 0.5 is defined as "positive" and those with a sentiment score lower than 0.5 is defined as "negative". All the remaining reviews are categorized as "neutral". The total number of the three sentiments is shown in Figure 11. The number of the three sentiments for reviews with each rating is shown in Figure 12.

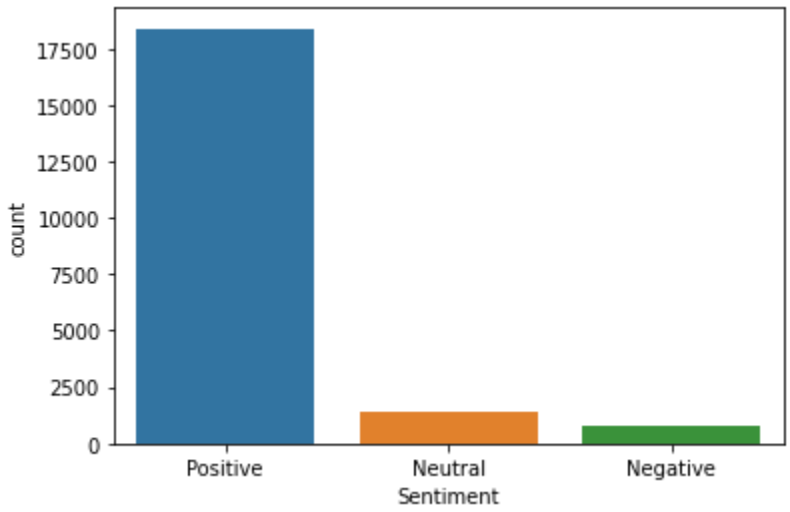


Figure 12. Total number of "positive", "negative", and "neutral" reviews

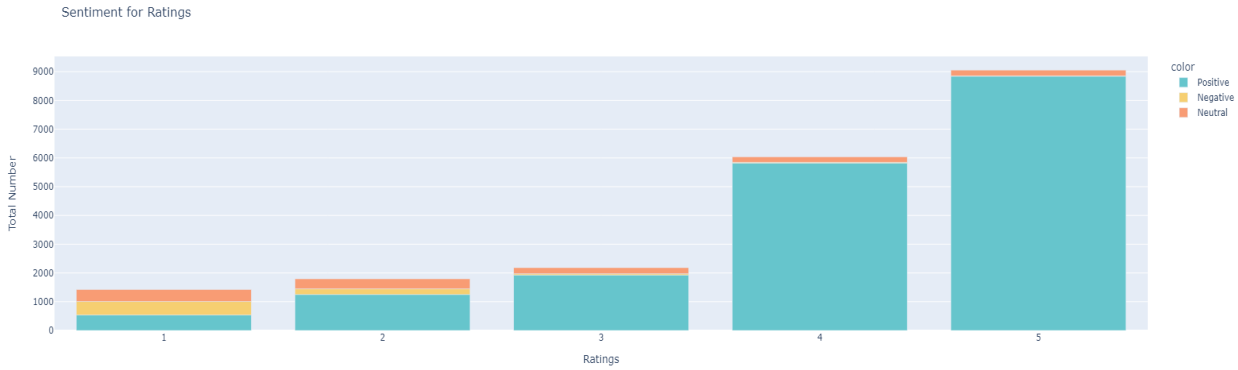


Figure 13. Number of reviews with various polarity for each rating

The original dataset does not include labels for polarity analysis. Thus the performance regarding VADER cannot be evaluated this time. Instead part of the results from the polarity analysis are shown as below:

Review No.	Review content (after data cleaning)	Sentiment score	Polarity
1	**nice** hotel expensive parking got **good deal** stay hotel anniversary arrived late evening took advice previous reviews did valet parking check **quick easy** little **disappointed** non-existent view room room **clean** **nice** size bed **comfortable** woke stiff neck high pillows **not soundproof** like heard music room night morning loud bangs doors opening closing hear people talking hallway maybe just **noisy** neighbors aveda bath products **nice** did not goldfish stay **nice** touch taken advantage staying longer location **great** walking distance shopping overall **nice** experience having pay 40 parking night	0.9919	Positive
101	dump stayed weekend expected charming 1929 property based web-site reviews expedia did n't expect **charming** **mean incompetent** staff showers randomly **scalding cold** medium pressure just trickle **tacky** acoustic ceilings moldy **non-functional** windows shower **no** screens windows **no a/c** having room left unlocked maid service having wait 18 hours iron gave 30 minutes make wedding having car 8am parking thought desk reality n't etc. etc.expedia apparantly wo n't publish **negative** reviews like gather having tried 3 times publish on-line calling customer service exchanging e-mails beware reviews expedia especially beware hotels like	0.2732	Neutral
2004	**terrible** customer service second stay resort realized offered aaa discount inquiring desk said late change rate 1/2 way 4 week stay asked talk managment recieved **bad sad** response **not reccomend** hotel	-0.8658	Negative

Human evaluation was used to mark the positive (in black bold) and negative (in red bold) words. Review No.1 includes more words on the positive side than the negative side. It explains why it has a very high sentiment score and is determined as “positive”. As a comparison, Review No.101 has more words on the negative side, and it is classified as “neutral”. Review No.2004 only has serious and negative words such as terrible and sad. This is why it is classified as “negative”. In general, the polarity analysis performed using VADER appears to be reasonable.

2.4 Overall rating prediction - Logistic Regression (Zhiyan)

The first approach adopted for overall rating prediction is logistic regression. The basic idea behind is that each ngram word within a vocabulary is treated as a numerical feature with values representing word count. The overall rating (1 through 5) is treated as the categorical response variable. At the training phase, each feature is assigned as a numerical coefficient, which represents the weight of this n-gram word. In other words, these words correlate to more positive opinions. At the prediction phase, the test

review will be assigned the corresponding weights. Then the weighted sum of all words in the test review is essentially the predicted overall rating.

2.4.1 Preprocessing

The first step in preprocessing is to establish a document-term matrix. Each row of the matrix represents a document, and each column represents a n-gram word (i.e., feature) within a vocabulary. “N-gram” means the N contiguous words (minimum unit) can be combined as a phrase. Because logistic regression results are a weighted sum, using n-gram words instead of single words as features helps capture semantics more accurately. For example, if a review contains words “not good”, it is better to treat them as a single feature to capture the negative sentiment. If treated separately, “not” must have a lower weight to counteract the positive impact by “good”. However, this is not always the case, as “not” can also be combined with other words to show neutral or positive meaning, such as “not expensive”. In this project, 4-gram words were adopted and can be shown to produce decent performance. Other important arguments in the document-term matrix include “max_features”, “min_df”, and “max_df”. Theoretically more features lead to less training error, but at the risk of overfitting and at the expense of more computation. “Min_df” and “max_df” rule out the most frequently occurring words (stopwords) and very rare words. With all factors considered, the above three arguments were set as 10,000, 0.001, and 0.5, respectively. The resulting document-term matrix has dimensions of 20,491 x 10,000.
The training/test sets were split with a ratio of 9:1. That is, 90% of the entire dataset was used as a training set and the remaining 10% dataset was used for testing. The split was random and can be controlled by state.

2.4.2 Training

For the training phase, function LogisticRegression from the library sklearn.linear_model was used. Major arguments were left as default and are listed as below:
penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None.

2.4.3 Results

Because multiclass prediction is implemented, the predicted ratings are presented in the form of a confusion matrix. A resulting confusion matrix is presented as below. It has dimensions of 5x5. Each row represents the ground truth rating and each column represents the predicted rating. The entry at (i, j) refers to the number of reviews that have a rating of i+1 and is predicted with rating of j+1. In other words, numbers on the diagonal are the correctly predicted reviews and those off the diagonal are incorrectly predicted.

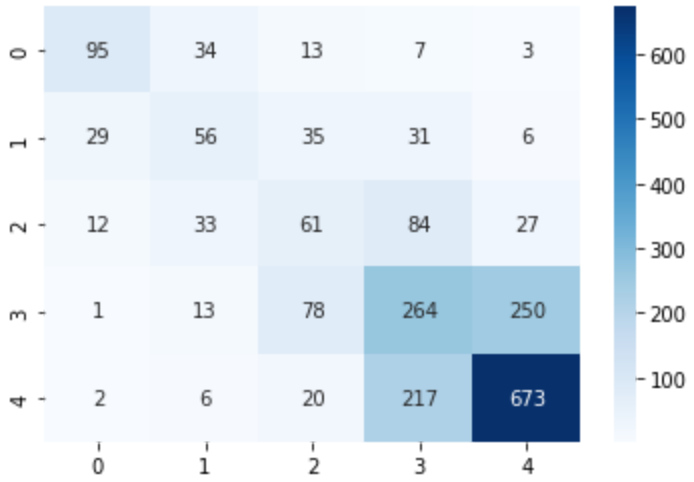


Figure 14. A typical confusion matrix

Figure 13 illustrates different prediction performance for different ratings. With a preliminary estimation, the reviews with labels of 5 stars have the best performance. The reviews with labels of 1 star have the second-to-best performance. Its performance for 3-star reviews is the worst, which is very close to random guess.

Another form to exhibit the performance is to calculate precision and recall:

Precision = True Positives / (True Positives + False Positives)

Recall = True Positives / (True Positives + False Negatives)

Note that the precision and recall are originally defined only for binary classification. To apply them in the multi-classification as in this project, each ground truth rating must be analyzed individually. It can be seen that predictions for Rating 5 and Rating 1 have the highest accuracy, followed by those for Rating 2 and Rating 4. Reviews with Rating 3 are the most difficult ones to predict by logistic regression.

Rating 1:

	Logistic regression (1 star)	Logistic regression (not 1 star)
Ground truth (1 star)	97	55
Ground truth (not 1 star)	45	1854
Precision (P)	0.68	
Recall (R)	0.64	
F-1	0.66	

Rating 2:

	Logistic regression (2 star)	Logistic regression (not 2 star)
Ground truth (2 star)	56	101
Ground truth (not 2 star)	86	1787
Precision (P)	0.39	
Recall (R)	0.36	
F-1	0.37	

Rating 3:

	Logistic regression (3 star)	Logistic regression (not 3 star)
Ground truth (3 star)	61	156
Ground truth (not 3 star)	146	1787
Precision (P)	0.29	
Recall (R)	0.28	
F-1	0.28	

Rating 4:

	Logistic regression (4 star)	Logistic regression (not 4 star)
Ground truth (4 star)	264	342
Ground truth (not 4 star)	339	1105
Precision (P)	0.44	
Recall (R)	0.44	
F-1	0.44	

Rating 5:

	Logistic regression (5 star)	Logistic regression (not 5 star)
Ground truth (5 star)	673	245

Ground truth (not 5 star)	286	846
Precision (P)	0.70	
Recall (R)	0.73	
F-1	0.71	

One possible reason is that the training set for 5-star reviews has the largest size, which helps achieve better training performance. Another reason is that the 3-star reviews are essentially neutral and more susceptible to the errors in the training model. For each rating, it is interesting to see the precision is close to recall. As a summary, the established logistic regression model yields decent results and also has room for improvement.

2.4.4 Interpretation

An advantage of logistic regression is that the results are typically easy to interpret. In this case, the coefficient of each feature corresponds to the positive impact on rating. Larger coefficients denote negative sentiment and less contribution to achieving a higher score. To demonstration this, the features with top 20 largest coefficients and top 20 smallest coefficients for Rating 1 are listed in the table below:

Table 2 N-gram words with top 10 largest and top 10 smallest coefficients for Rating 1

N-gram word	Coefficient	N-gram word	Coefficient
worst	1.99	**reading**	**-0.92**
avoid	1.64	helpful	-0.93
rude	1.50	impressed	-0.94
terrible	1.45	value	-0.94
dump	1.33	liked	-0.95
unhelpful	1.33	outstanding	-0.95
good thing	**1.32**	matter	-0.96
awful	1.31	happy	-0.97
stains	1.31	**strange**	**-0.97**
joke	1.29	**agree**	**-0.97**
moldy	1.23	expectations	-0.99
dirty	1.18	spacious	-1.00

disgusting	1.16	beautiful	-1.02
turned	1.13	clean	-1.03
worststayed	1.06	enjoyed	-1.04
unpleasant	1.03	excellent	-1.08
stuck	1.01	loved	-1.16
horrible	1.01	comfortable	-1.17
stay away	1.00	**bit**	**-1.21**
apartment	**0.99**	great	-1.22

The top 20 N-gram with largest coefficients generally contain negative sentiment. For example, “worst” has the largest coefficient of 1.99. On the other hand, the top 20 N-gram with smallest coefficients generally convey positive feelings. For example, “great” has the smallest coefficient of -1.22. However, it must be pointed out that some words that appear to be neutral are marked as either positive or negative. For example, “apartment” is marked as negative, while “reading”, “strange”, “agree”, and “bit” are labeled as positive. It is astonishing that “good thing” is marked as negative. Possible reason is that this n-gram word is used in conjunction with some other words to express negative sentiment.

2.5 Overall rating prediction - LSTM (Peng)

2.5.1 word embedding

A word embedding is a learned representation for text where words that have the same meaning have a similar representation. It is an approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.
Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.
Key to the approach is the idea of using a dense distributed representation for each word.
Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one-hot encoding.

2.5.2 LSTM layer

LSTM networks were designed specifically to overcome the long-term dependency problem faced by recurrent neural networks RNNs (due to the vanishing gradient problem). LSTMs have feedback connections which make them different to more traditional feedforward neural networks. This property enables LSTMs to process entire sequences of data (e.g. time series) without treating each point in the

sequence independently, but rather, retaining useful information about previous data in the sequence to help with the processing of new data points. As a result, LSTMs are particularly good at processing sequences of data such as text, speech and general time-series.

2.5.3 Dense layer

In any neural network, a dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most commonly used layer in artificial neural network networks.

2.5.4 Model implementation

In this part, we implement three different models. First model is the simplest model that only contains 1 embedding layer, 1 LSTM layer, and 1 dense layer. Second model is built upon first model with a dropout layer added. The last model is the most complex model among these three. It's a bidirectional LSTM with Dropout model.
Here's the code for building model1, model2, and model3:

```
model1 = tf.keras.models.Sequential([tf.keras.layers.Embedding(total_word,
8),
                                     tf.keras.layers.LSTM(16),
                                     tf.keras.layers.Dense(3,
activation='softmax')])

model2 = tf.keras.models.Sequential([tf.keras.layers.Embedding(total_word,
8),
                                     tf.keras.layers.LSTM(16),
                                     tf.keras.layers.Dropout(0.5),
                                     tf.keras.layers.Dense(3,
activation='softmax')])



model3 = tf.keras.models.Sequential([tf.keras.layers.Embedding(total_word,
8),

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
                                     tf.keras.layers.Dropout(0.5),
                                     tf.keras.layers.Dense(8,
kernel_regularizer=l2(0.001),

bias_regularizer=l2(0.001), activation='relu'),
```

```
                                    tf.keras.layers.Dropout(0.5),
                                    tf.keras.layers.Dense(3,
activation='softmax')])
```

2.5.5 results

We evaluate our different models in two aspects: 1. Running time. 2. Accuracy.
Below are the running times for each of the models.
Model1:

```
Epoch 1/3
513/513 [==============================] - 408s
Epoch 2/3
513/513 [==============================] - 407s
Epoch 3/3
513/513 [==============================] - 406s
<keras.callbacks.History at 0x7fe975f8f490>
```

Model2:

```
Epoch 1/3
513/513 [==============================] - 413s
Epoch 2/3
513/513 [==============================] - 403s
Epoch 3/3
513/513 [==============================] - 412s
<keras.callbacks.History at 0x7fe975fb8610>
```

model3:

```
Epoch 1/3
513/513 [==============================] - 722s
Epoch 2/3
513/513 [==============================] - 752s
Epoch 3/3
513/513 [==============================] - 727s
<keras.callbacks.History at 0x7fe979bab4f0>
```

We can see that model 1 and model 2 used much less time than model 3. The differences between model 1 and model 2 are not significant. This is because the added dropout layer won't increase much of the complexity of the model. The dropout layer can help with reducing the overfit of LSTM layer. The model 3 has much longer running time and it is because it has the bidirectional LSTM layer. The model 3 needs to calculate not only the forward, but also backward. This bidirectional LSTM layer can account for the impact from both the word in the front and word in the later portion of the text.

For accuracy, we listed all models precision, recall, f1-scrore and accuracy:
Model1:

	precision	recall	f1-score	support
0	0.63	0.76	0.69	648
1	0.87	0.94	0.91	3040
2	0.26	0.03	0.06	411
accuracy			0.82	4099
macro avg	0.59	0.58	0.55	4099
weighted avg	0.77	0.82	0.79	4099

Model2:

	precision	recall	f1-score	support
0	0.49	0.20	0.28	648
1	0.77	0.97	0.86	3040
2	0.00	0.00	0.00	411
accuracy			0.75	4099
macro avg	0.42	0.39	0.38	4099
weighted avg	0.65	0.75	0.68	4099

Model3:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	648
1	0.74	1.00	0.85	3040
2	0.00	0.00	0.00	411
accuracy			0.74	4099
macro avg	0.25	0.33	0.28	4099
weighted avg	0.55	0.74	0.63	4099

We can see that model 1 has the best performance regarding accuracy. However, once we added the dropout layer to reduce the overfit, we see the accuracy score comes down to a reasonable level. Even though model 3 has the bidirectional LSTM model and it's the most complex model among those three, model 3 does not have a higher accuracy than model 2.

3. Conclusion

We successfully performed sentiment analysis of the TripAdvisor Hotel Review dataset. We first performed data cleaning to obtain the most important and meaningful text data to prepare for the analysis model. Then we did polarity analysis to categorize the reviews to "negative", "neutral" and "positive". Next we performed Logistic Regression which shows decent performance for overall rating prediction. Lastly we implemented a LSTM model with three different configurations with the simple model showing the highest accuracy. For future improvement, we can further tune the parameters to see if the accuracy can be improved.

4. References

1. Course project instructions. (https://www.coursera.org/learn/cs-410/supplement/vtHNZ/course-project-overview), accessed on Dec.4, 2022.

2. TripAdviosr dataset (https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews), accessed on Dec.4, 2022.

3. Wang, H., Lu, Y., and Zhai, C. Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach. KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. PP 783-793. https://doi.org/10.1145/1835804.1835903

4. TripAdvisor project (https://deepnote.com/@abid/Trip-Advisor-Data-AnalysisML-f6060b39-d76c-4579-9648-a54bc8b5ffb5), accessed on Dec.4, 2022.

5. Getting Started with Sentiment analysis using Python (https://cnvrg.io/sentiment-analysis-python/),

accessed on Dec.4,2022.

6. Stemming and Lemmatization
(https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html).
Accessed on Dec.5, 2022.

7. Sentiment Using Sklearn and Tensorflow
(https://www.kaggle.com/code/mfaaris/sentiment-using-sklearn-and-tensorflow).
Accessed on Dec.5, 2022.

8. Hotel Reviews Sentiment Prediction
(https://www.kaggle.com/code/shahraizanwar/hotel-reviews-sentiment-prediction)
Accessed on Dec.5,2022

9.Introduction to Convolutional Neural Network (CNN) using Tensorflow
(https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83)
Accessed on Dec.5,2022

10.A Complete Understanding of Dense Layers in Neural Networks
(https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/#:~:text=Sign%20up-,What%20is%20a%20Dense%20Layer%3F,in%20artificial%20neural%20network%20networks.)
Accessed on Dec.5, 2022

11.What Are Word Embeddings for Text?
(https://machinelearningmastery.com/what-are-word-embeddings/#:~:text=A%20word%20embedding%20is%20a,challenging%20natural%20language%20processing%20problems.)
Accessed on Dec.5, 2022

12.Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
(https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/)
Accessed on Dec.5, 2022