Bi-weekly Report 2

-Introduction To Brain and Machine Learning-

Group 2

Members and their Role

Shareen Rai - Making a report

김동환 - Data analysis, preprocessing and model constructing

서범석 - Reference paper survey and model learning

송윤아 - Define the question and select a topic and summarize the result

Problem definition

Due to the pandemic, the vacation in the form of "Staycation" has increased worldwide [1], hotels are now not only just accommodations but also being served as a travel destination. As such, we expect consumers to consider a variety of factors in the process of deciding on a hotel. We thought that hotels needed the means to examine what factors satisfy consumers, and also consumers needed the means to help select the rational hotel.

In general, when people choose a hotel, they refer to the hotel review. However, as the number of review-set is way too large, and every reviewer is interested in each different factor. So the evaluation of those reviews is mandatory for every individual who wants to get information from them. To solve this problem, we use machine learning to make this easier for those who have to select the hotel to spend memorable days.

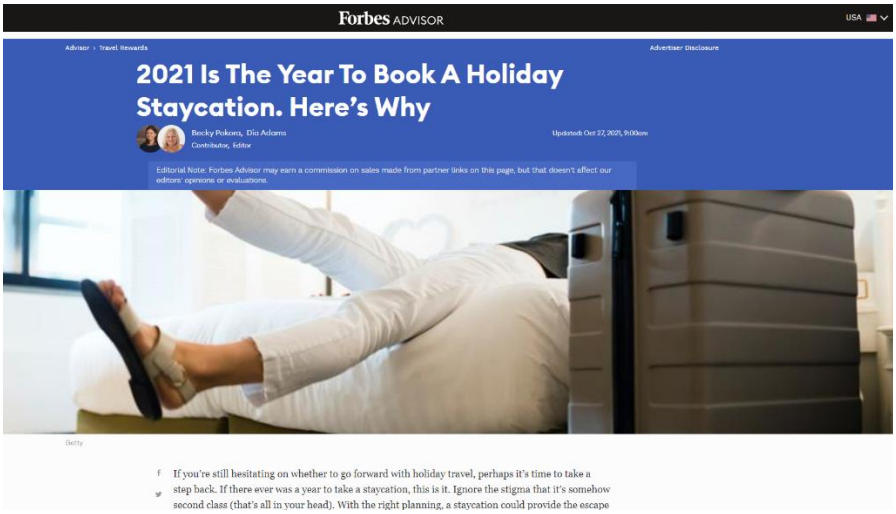

Fig 1 : Article about "Staycation" [1]

Topic

Using machine learning, will examine in detail the factors that hotel consumers think positively and negatively.

Model

To solve this problem, we use the LDA (Latent Dirichlet Allocation) model for the topic. Because LDA technique does not have to know in advance what the topics will look like. And we can explore topic formation and resulting document clusters by tuning the LDA parameters to fit different dataset shapes [2].

LDA is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA assumes the following generative process for each document w in a corpus D [3]:

1. Choose N ~ Poisson(ξ).

2. Choose θ ~ Dir(α).

3. For each of the N words w_n:

   (a) Choose a topic $z_n \sim Multinomial(\theta)$.

   (b) Choose a word w_n from $p(z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

'sklearn.decomposition.LatentDirichletAllocation' module of Scikit-learn is a very good tool for implementing the LDA model as Python [4]. This module receives the number of topics and Term Frequency-Inverse Document Frequency (TF-IDF) of a document and finds the topic distribution of a given document and the frequency value of words used in the topic.

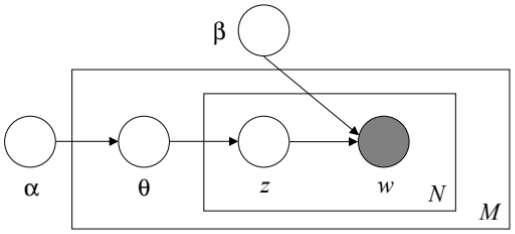

Fig 2 : Graphical model representation of LDA [3]

Setting parameter values

Several search methods such as random search and grid search or genetic algorithm can be used to find the optimal parameter value during model learning. Among them, grid search was used to find the most optimal result value in this project.

Grid search is a traditional method of hyperparameters optimization, which simply makes a complete search over a given subset of the hyperparameters space of the training algorithm [5]. This process takes a long time, but it's the surest way to find the optimal parameter values.

'sklearn.model_selection.GridSearchCV' module of Scikit-learn is a very good tool for implementing search methods as Python [6]. This module receives various parameters as param_grid and compares the result values of substituting them with stratified k-fold cross-validation to create optimal parameters and models made of those parameters.

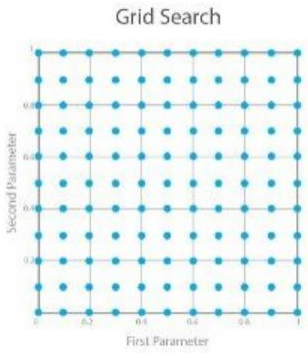

Fig 3 : An illustration of a grid search space [5]

Data set

We use data named ‘Tripadvisor Hotel Revisions’ in Kaggle. This data is from a site called ‘TripAdvisor’ that crawled over 20,000 reviews [7]. And we classified them as positive/negative reactions using the number of stars and reviews in the data set.

To see the use of specific words for each data, we created a word cloud and checked. At this time, ‘hotel’, ’everything’, ’anything’, ‘nothing’, ’thing’, ‘need’, ‘stay’, ‘say’, ’go’, ‘day’, ‘night’, ‘time’ were additionally excluded, in terms of selecting main factor that affects the evaluation.

Data Exploration

- **Followed Pictures are the output of our preprocessing units**

	Review	Rating
0	nice hotel expensive parking got good deal sta...	4
1	ok nothing special charge diamond member hilto...	2
2	nice rooms not 4* experience hotel monaco seat...	3
3	unique, great stay, wonderful time hotel monac...	5
4	great stay great stay, went seahawk game aweso...	5
5	love monaco staff husband stayed hotel crazy w...	5
6	cozy stay rainy city, husband spent 7 nights m...	5
7	excellent staff, housekeeping quality hotel ch...	4
8	hotel stayed hotel monaco cruise, rooms genero...	5
9	excellent stayed hotel monaco past w/e delight...	5

Fig 4 : Format of Data

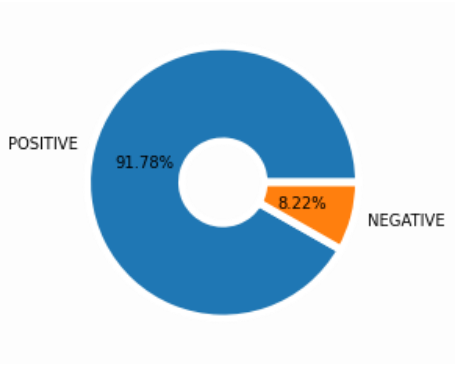

Fig 5 : The ratio of positive and negative

Fig 6 : Word cloud of positive review

Fig 7 : Word cloud of negative review

Training Model

Step1. Import the module that we'll use.

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import LatentDirichletAllocation
```

Step2. Make the 'LDA_model' function.

```
def LDA_model(lemmatized_words):
  """
  CountVectorizer : This give us another standard of selecting words among list
  of word bunches. We could fit with this but we need more than that
  strip_accents : Encoded type of words
  lowercase : Since we made lemmatized above, our word is with all lowercase
  max_df : In our assumption, like as we remove the word 'hotel' for gaining true
  factor of evaluation, the word which appears more than 75% of chance might not
  good for factorize.
  min_df : If the word that doesn't appear more than this value among all other
  groups, it might be that important factor.
  max_df and min_df is a value for efficient and accurate topic selection.
  """
  text2vec = CountVectorizer(strip_accents = 'unicode',
                             lowercase = True,
                             max_df = 0.75,
                             min_df = 100)

  # Apply TfidVectorizer with the parameter of CountVectorizer.
  # It will select proper words.
  text_vectorizer = TfidfVectorizer(**text2vec.get_params())
  text_in_matrix = text_vectorizer.fit_transform(lemmatized_words)

  LDA = LatentDirichletAllocation()

  """
  n_components : the number of topic assumptions dragged out the word from all
  the list of sentences.
  learning_deacy : the rate of learning_deacy that will be applied every iterations
  among learning. { for each word weights, by this we select words that follows the topic }
  learning_method : choosing the learning method, the original paper tried batch
  learning with about 600 reviews, and since we have 15,000 so we choose going with online.
  """
  param_grid = {'n_components' : [1, 2, 3],
                'learning_decay' : [0.5, 0.7, 0.75, 0.8, 0.9],
                'learning_method' : ['online']}

  model = GridSearchCV(LDA, param_grid)
  model.fit(text_in_matrix)

  best_model = model.best_estimator_

  print("Best Parameter : ", model.best_params_)
  print("Model Log Likelihood Score: ", model.best_score_)
  print("Model Perplexity: ", best_model.perplexity(text_in_matrix))

  return best_model, text_vectorizer
```

Step3. Find the best parameter in pos_lem(data) and ned_lem(data).

```
# Since we have more data on POSITIVEs, make our model be on that
LDA_best_pos, LDA_vectorizer_pos = LDA_model(pos_lem)
LDA_best_neg, LDA_vectorizer_neg = LDA_model(neg_lem)
```

```
Best Parameter :  {'learning_decay': 0.75, 'learning_method': 'online', 'n_components': 1}
Model Log Likelihood Score:  -528795.2470952942
Model Perplexity:  339.74406139959166
/usr/local/lib/python3.7/dist-packages/sklearn/feature_extraction/text.py:2032: UserWarning:
  UserWarning,
Best Parameter :  {'learning_decay': 0.7, 'learning_method': 'online', 'n_components': 1}
Model Log Likelihood Score:  -15930.84623987137
Model Perplexity:  51.73220520103631
```

Step4. Make 'display_topics' function.

```
def display_topics(LDA, feature_names, n_top_words):

  topic_dict = {}

  for topic_idx, topic in enumerate(LDA.components_):
    topic_dict["Topic words"]= ['{}'.format(feature_names[i]) for i in topic.argsort()[:-n_top_words - 1:-1]]
    topic_dict["Topic weights"]= ['{:.2f}'.format(topic[i]) for i in topic.argsort()[:-n_top_words - 1:-1]]

  return pd.DataFrame(topic_dict)
```

Step5. Print the result for each positive and negative.

```
pos_result = display_topics(LDA_best_pos, LDA_vectorizer_pos.get_feature_names(), n_top_words = 20)
neg_result = display_topics(LDA_best_neg, LDA_vectorizer_neg.get_feature_names(), n_top_words = 20)
```

```
pos_result
```

```
neg_result
```

```
final_result = pd.concat([pos_result, neg_result], axis = 1)
final_result
```

```
final_result.columns = ['POS Word','POS Weight','NEG Word', 'NEG Weight']
final_result
```

Result

We printed out the words that had the most influence from 1st to 20th. It was confirmed that both positive and negative ranked first and second. The biggest factors that people feel positive and negative are room and staff. It was found that the result was similar to the word cloud result output from data expansion. For the hotel to receive a good review from the users, it will have to be equipped with rooms and staff. To get a positive response, it is necessary to pay attention to location and service. As services and food are at the top of the factors that affect negative reactions, we think it is necessary to consider those factors to reduce negative reactions.

	Topic words	Topic weights
0	room	29098.84
1	staff	12029.83
2	location	8240.86
3	service	6759.41
4	restaurant	6467.17
5	breakfast	6318.61
6	food	5972.94
7	place	5869.68
8	area	4970.54
9	beach	4944.32
10	pool	4740.81
11	resort	4689.28
12	bar	4574.60
13	view	4176.90
14	people	4091.25
15	trip	3770.85
16	bathroom	3645.62
17	minute	3447.22
18	floor	3446.65
19	city	3378.20

	Topic words	Topic weights
0	room	3411.27
1	staff	763.10
2	service	723.79
3	food	516.75
4	place	475.92
5	people	416.54
6	resort	377.95
7	problem	361.72
8	desk	343.29
9	bathroom	330.01
10	water	320.70
11	breakfast	310.93
12	floor	309.05
13	hour	291.91
14	experience	290.75
15	bed	287.54
16	restaurant	271.34
17	pool	268.44
18	beach	267.04
19	day	266.47

Result1. Positive Result	Result2. Negative Result

	POS Word	POS Weight	NEG Word	NEG Weight
0	room	29098.84	room	3411.27
1	staff	12029.83	staff	763.10
2	location	8240.86	service	723.79
3	service	6759.41	food	516.75
4	restaurant	6467.17	place	475.92
5	breakfast	6318.61	people	416.54
6	food	5972.94	resort	377.95
7	place	5869.68	problem	361.72
8	area	4970.54	desk	343.29
9	beach	4944.32	bathroom	330.01
10	pool	4740.81	water	320.70
11	resort	4689.28	breakfast	310.93
12	bar	4574.60	floor	309.05
13	view	4176.90	hour	291.91
14	people	4091.25	experience	290.75
15	trip	3770.85	bed	287.54
16	bathroom	3645.62	restaurant	271.34
17	minute	3447.22	pool	268.44
18	floor	3446.65	beach	267.04
19	city	3378.20	day	266.47

Fig 8. Result

Reference

[1] https://www.forbes.com/advisor/travel-rewards/2021-is-the-year-to-book-a-holiday-staycation-heres-why/

[2] Nguyen, Eric. "Text Mining and Network Analysis of Digital Libraries in R." (2014).

[3] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *the Journal of machine Learning research* 3 (2003): 993-1022.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

[5] Liashchynskyi, Petro, and Pavlo Liashchynskyi. "Grid search, random search, genetic algorithm: a big comparison for NAS." *arXiv preprint arXiv:1912.06059* (2019).

[6] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[7] https://www.kaggle.com/andrewmvd/trip-advisor-hotel-reviews