# Hotel Review Key Phrases Extraction

Data 245 Machine Learning

Team 8

Asuna Ding, Huang Peng, Meng Yuan, Rui Xu

## Introduction

The internet is the main source of choosing travel services in the world and has made a remarkable difference in the process of spreading information about tourism. The majority of people are likely to conduct online research before they make any purchase, such as hotel booking in our case. Guest reviews are becoming a prominent factor affecting people's decision making on purchases; people tend to take other consumers' perspectives into consideration for their purchases, which makes Natural Language Processing very important. As the largest travel website in the world, TripAdvisor.com consists of user-generated data that provides a considerable amount of useful information that can help businesses to keep track and to manage customer opinions.

In this project, we will take hotel text reviews from TripAdvisor.com and propose unsupervised machine learning approaches to extract key phrases in order to solve hotel review analysis problems.

## Data Exploration

### Data Distribution

The dataset we are using for this project is from Kaggle, which is extracted from TripAdvisor.com; it contains 20,491 hotel reviews with corresponding ratings, which range from 1 to 5 stars. This dataset has neither missing values nor duplicated values. We are only working with two features: REVIEW is a categorical feature that indicates the text review from users with a minimum of 9 words, maximum of 1933 words, and means of 104 words; RATING is a numerical feature that indicates the 5 distinct levels

of ratings of the hotel from users with 1st mode of 5-star rating (44.2%) and 2nd mode of 4-star rating (29.5%).

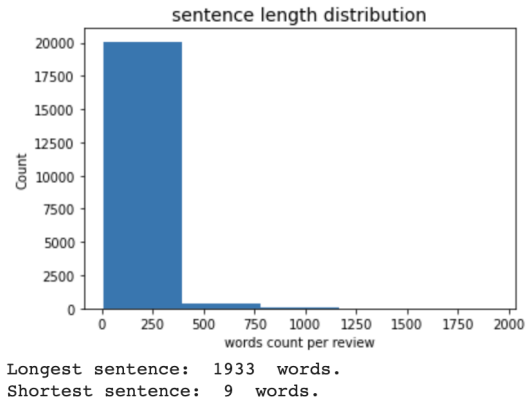



Figure 1. Sentence Length Distribution

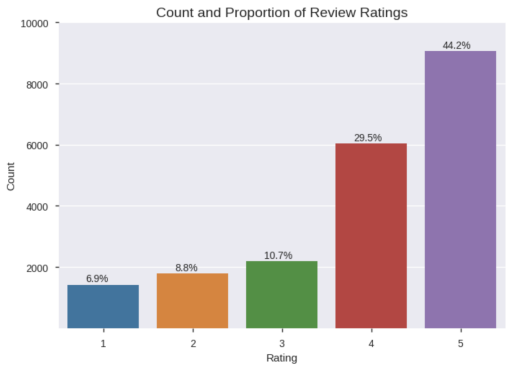



Figure 2. Rating Distribution

**Data Quality Issues**

Detecting and eliminating errors, inaccurate data, missing values, and outliers is called data cleaning. In this project, data cleaning starts with some of the text words that provide little to no useful information.

Figure 2 exhibits the exponential distribution of rates for each review; however, such distribution indicates very unbalanced data. In the following section, we will try different methods to cope with this issue.

Figure 3 illustrates the most frequent words in reviews: Hotel, Room, Resort, Night, Time, etc. It is clear that those common words provide no useful information, which is a relatively common issue of Natural Language Processing. Along with the unnecessary words, numbers, punctuations, stop words and irregular words also need to be removed in order to optimize our study. In the next section, we will be explaining how we apply different methods to cleaning our text data.

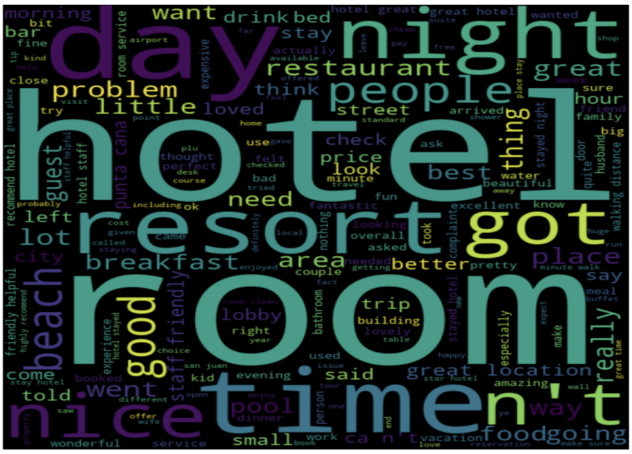



Figure 3. WordCould of most frequent words

## Approaches

### Natural Language Toolkit (NLTK)

NLTK is one of the most comprehensive Python NLP libraries with a lot of corpora, models, and algorithms, which helps majorly in the learning purpose. With the help of the nktl.tokenize.word_tokenize() method, we are able to extract the token from the string of characters from reviews. Moreover, we use the NLTK library for stop-words cleaning and lemmatization. During this process, we build a function to create and clean a collection of words that don't provide much information.

| Pros | Cons |
|---|---|
| • Full NLP library with many 3rd extinctions<br>• Supports the largest number of languages compared to other libraries | • Relatively slow<br>• Only splits test by sentences, without analyzing the semantic structure<br>• No Neural Network models |

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | (hotel, 3588) | (room, 4250) | (hotel, 5058) | (hotel, 14125) | (hotel, 21822) |
| 1 | (room, 3267) | (hotel, 4166) | (room, 4424) | (room, 10172) | (room, 12126) |
| 2 | (stay, 1207) | (good, 1484) | (good, 2453) | (great, 7414) | (great, 10665) |
| 3 | (staff, 958) | (stay, 1367) | (great, 1751) | (good, 6697) | (staff, 7808) |
| 4 | (rooms, 893) | (rooms, 1354) | (nice, 1739) | (nice, 4826) | (stay, 6823) |
| 5 | (night, 874) | (staff, 1283) | (rooms, 1480) | (staff, 4665) | (good, 5705) |
| 6 | (service, 810) | (service, 1217) | (staff, 1468) | (stay, 4331) | (stayed, 4817) |
| 7 | (day, 807) | (night, 1141) | (stay, 1409) | (location, 3865) | (location, 4807) |
| 8 | (like, 713) | (food, 1075) | (location, 1402) | (rooms, 3615) | (rooms, 4652) |
| 9 | (told, 707) | (day, 1059) | (beach, 1260) | (clean, 3452) | (nice, 4407) |
| 10 | (time, 686) | (nice, 1049) | (night, 1219) | (beach, 3438) | (breakfast, 4142) |
| 11 | (desk, 637) | (time, 1044) | (food, 1159) | (breakfast, 3064) | (service, 4138) |
| 12 | (place, 634) | (beach, 1034) | (clean, 1151) | (stayed, 3049) | (time, 3958) |
| 13 | (got, 629) | (like, 1001) | (like, 1121) | (time, 2999) | (clean, 3840) |
| 14 | (resort, 617) | (resort, 980) | (time, 1113) | (food, 2886) | (excellent, 3670) |
| 15 | (stayed, 611) | (great, 908) | (service, 1101) | (night, 2829) | (day, 3586) |
| 16 | (food, 608) | (stayed, 889) | (stayed, 1099) | (day, 2789) | (beach, 3380) |
| 17 | (good, 604) | (got, 843) | (day, 1080) | (service, 2661) | (friendly, 3378) |
| 18 | (people, 534) | (people, 839) | (resort, 1041) | (resort, 2523) | (night, 3342) |
| 19 | (went, 499) | (really, 769) | (breakfast, 1001) | (really, 2517) | (place, 3302) |

Figure 4. Common words by rating using basic nltk tokenization

The NLTK can remove stop words easily. With the help of the word_tonkenize() method, all the punctuations in the sentences are automatically removed. Figure 4 presents the most common word for each rating level after using the NLTK tokenization method. However, the result did not meet our expectations. There are two reasons:

There are too many useless common words left, such as ‘hotel’, ‘room’, ‘time’; their weight seems too high.

Single-word can’t convey many details. With the single word ‘service’, for example, we are unable to come to the conclusion that the service is inadequate.

**Improved NLTK**

After the first approach, we learned that it is not enough to only extract the most common words. Given the structure of the English language, useful information in a hotel review is more likely to hide in the noun and the adjective that modifies it. Therefore, getting the most common

adjective, nouns, and bigrams is a better approach in this case. Figure 5, Figure 6, and Figure 7 illustrate the outcome of the most common words for adjective, noun, and bigram by different rating levels.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | (good, 602) | (good, 1473) | (good, 2436) | (great, 7414) | (great, 10665) |
| 1 | (bad, 435) | (nice, 954) | (great, 1751) | (good, 6670) | (good, 5670) |
| 2 | (small, 363) | (great, 908) | (nice, 1601) | (nice, 4479) | (nice, 4085) |
| 3 | (nice, 329) | (small, 628) | (clean, 957) | (clean, 2870) | (excellent, 3209) |
| 4 | (great, 319) | (clean, 519) | (small, 944) | (small, 2349) | (clean, 3195) |
| 5 | (old, 303) | (bad, 502) | (little, 741) | (little, 2074) | (best, 2586) |
| 6 | (worst, 284) | (little, 472) | (bad, 523) | (comfortable, 1611) | (wonderful, 2583) |
| 7 | (terrible, 256) | (old, 359) | (free, 421) | (helpful, 1591) | (helpful, 2478) |
| 8 | (clean, 236) | (beautiful, 351) | (best, 394) | (excellent, 1472) | (little, 2261) |
| 9 | (new, 235) | (best, 335) | (old, 387) | (free, 1328) | (small, 2233) |
| 10 | (horrible, 220) | (new, 328) | (comfortable, 368) | (large, 1280) | (comfortable, 2166) |
| 11 | (stay, 204) | (big, 281) | (overall, 368) | (friendly, 1248) | (beautiful, 2029) |
| 12 | (little, 204) | (hot, 281) | (ok, 365) | (beautiful, 1199) | (fantastic, 2020) |
| 13 | (available, 185) | (stay, 277) | (friendly, 356) | (best, 1172) | (friendly, 1982) |
| 14 | (open, 181) | (free, 274) | (beautiful, 355) | (big, 1068) | (free, 1825) |
| 15 | (best, 174) | (sure, 269) | (large, 354) | (quiet, 1000) | (large, 1766) |
| 16 | (hot, 167) | (poor, 267) | (helpful, 354) | (new, 994) | (new, 1605) |
| 17 | (poor, 164) | (open, 262) | (big, 352) | (wonderful, 910) | (stay, 1529) |
| 18 | (free, 150) | (ok, 242) | (new, 344) | (overall, 856) | (quiet, 1383) |
| 19 | (beautiful, 148) | (available, 226) | (main, 336) | (main, 844) | (perfect, 1370) |

Figure 5. Common Adjective by Rating

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | (hotel, 3436) | (room, 4250) | (hotel, 4845) | (hotel, 13542) | (hotel, 20949) |
| 1 | (room, 3267) | (hotel, 3979) | (room, 4424) | (room, 10172) | (room, 12126) |
| 2 | (staff, 958) | (rooms, 1354) | (rooms, 1480) | (staff, 4665) | (staff, 7808) |
| 3 | (rooms, 893) | (staff, 1283) | (staff, 1468) | (location, 3751) | (location, 4662) |
| 4 | (night, 874) | (service, 1206) | (location, 1344) | (rooms, 3615) | (rooms, 4652) |
| 5 | (day, 807) | (night, 1141) | (night, 1219) | (time, 2999) | (service, 4081) |
| 6 | (service, 801) | (food, 1075) | (food, 1159) | (food, 2886) | (time, 3958) |
| 7 | (time, 686) | (day, 1059) | (time, 1113) | (night, 2829) | (day, 3586) |
| 8 | (stay, 614) | (time, 1044) | (service, 1081) | (day, 2789) | (breakfast, 3551) |
| 9 | (food, 608) | (resort, 880) | (day, 1080) | (beach, 2632) | (stay, 3446) |
| 10 | (place, 582) | (people, 839) | (beach, 971) | (service, 2624) | (night, 3342) |
| 11 | (resort, 553) | (beach, 794) | (resort, 909) | (breakfast, 2602) | (food, 3232) |
| 12 | (people, 534) | (stay, 661) | (breakfast, 868) | (stay, 2353) | (place, 3118) |
| 13 | (desk, 484) | (location, 654) | (people, 855) | (resort, 2257) | (beach, 2568) |
| 14 | (water, 425) | (water, 632) | (area, 797) | (place, 2082) | (people, 2464) |
| 15 | (days, 365) | (place, 616) | (pool, 770) | (people, 2058) | (resort, 2444) |
| 16 | (experience, 351) | (breakfast, 577) | (place, 767) | (area, 2015) | (pool, 2370) |
| 17 | (bathroom, 346) | (desk, 564) | (stay, 765) | (pool, 1891) | (area, 2324) |
| 18 | (manager, 335) | (pool, 550) | (water, 633) | (restaurants, 1722) | (trip, 2313) |
| 19 | (beach, 335) | (floor, 507) | (bathroom, 609) | (bar, 1631) | (restaurants, 2064) |

Figure 6. Common Nouns by Rating

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | ((punta, cana), 150) | ((punta, cana), 198) | ((great, location), 248) | ((great, location), 808) | ((staff, friendly), 968) |
| 1 | ((credit, card), 110) | ((room, service), 140) | ((staff, friendly), 235) | ((staff, friendly), 726) | ((great, location), 957) |
| 2 | ((customer, service), 103) | ((staff, friendly), 124) | ((punta, cana), 201) | ((hotel, great), 557) | ((great, hotel), 934) |
| 3 | ((stay, hotel), 101) | ((star, hotel), 121) | ((good, location), 166) | ((punta, cana), 548) | ((hotel, great), 866) |
| 4 | ((star, hotel), 95) | ((air, conditioning), 120) | ((room, clean), 136) | ((walking, distance), 512) | ((recommend, hotel), 818) |
| 5 | ((5, star), 88) | ((5, star), 116) | ((room, service), 132) | ((friendly, helpful), 464) | ((friendly, helpful), 735) |
| 6 | ((hotel, staff), 85) | ((stay, hotel), 102) | ((walking, distance), 131) | ((great, hotel), 450) | ((highly, recommend), 727) |
| 7 | ((room, service), 83) | ((san, juan), 97) | ((hotel, great), 130) | ((good, value), 405) | ((walking, distance), 704) |
| 8 | ((worst, hotel), 82) | ((great, location), 96) | ((san, juan), 125) | ((nice, hotel), 400) | ((hotel, staff), 618) |
| 9 | ((stay, away), 75) | ((stayed, hotel), 90) | ((stayed, hotel), 124) | ((room, clean), 396) | ((punta, cana), 582) |
| 10 | ((hotel, room), 75) | ((make, sure), 87) | ((good, value), 120) | ((great, time), 387) | ((stayed, hotel), 550) |
| 11 | ((stayed, hotel), 71) | ((hotel, room), 85) | ((hotel, good), 114) | ((staff, helpful), 382) | ((place, stay), 532) |
| 12 | ((air, conditioning), 70) | ((hotel, staff), 79) | ((make, sure), 112) | ((minute, walk), 373) | ((stay, hotel), 506) |
| 13 | ((got, room), 69) | ((room, small), 79) | ((location, hotel), 110) | ((recommend, hotel), 361) | ((staff, helpful), 504) |
| 14 | ((travel, agent), 66) | ((good, location), 77) | ((star, hotel), 109) | ((room, service), 360) | ((hotel, stayed), 495) |
| 15 | ((san, juan), 61) | ((room, clean), 76) | ((5, star), 104) | ((hotel, good), 358) | ((definitely, stay), 494) |
| 16 | ((hotel, stayed), 60) | ((hotel, good), 74) | ((location, great), 103) | ((stayed, hotel), 358) | ((new, york), 477) |
| 17 | ((booked, hotel), 59) | ((customer, service), 72) | ((room, small), 103) | ((hotel, staff), 348) | ((minute, walk), 471) |
| 18 | ((hot, water), 58) | ((room, ready), 68) | ((minute, walk), 99) | ((good, location), 340) | ((room, service), 469) |
| 19 | ((desk, staff), 56) | ((4, star), 66) | ((nice, hotel), 98) | ((place, stay), 322) | ((great, place), 454) |

Figure 7. Common Bigrams by Rating

When the task becomes more complex, such as adding tags to words, the NLTK performs relatively slower. The results are not optimal but do provide more information compared with our first approach. There are the reasons:

There're still some useless common words like 'punta cana', which is a place name, that needs to be dealt with.

A single adjective like 'horrible' and a single noun like 'service' don't give meaningful information separately; however, if we are able to combine them, we may get some useful phrases.

This approach can't combine n-grams together, for example, 'nice location' and 'beautiful window view' can't appear at the same time.

### Gensim Keywords

Gensim ("Generate Similar") is a popular NLP library used for unsupervised topic modeling. It uses top academic models and modern statistical machine learning like Word2vec and Fast Text to perform various complex tasks, such as building document or word vectors, performing topic identification, performing document comparison (retrieving semantically similar documents), and analyzing plain-text documents for semantic structure, and so on.

**Pros**

The facilities provided by Gensim for building topic models and word embedding is unparalleled.

Allow us to handle large text files without loading the whole file in memory.

Doesn’t require costly annotations or hand tagging of documents because it uses unsupervised models.

The process of **keyword extraction** is listed below:

The text is tokenized first and annotated with part of speech tags — a preprocessing step required to enable the application of syntactic filters.

All lexical units that pass the syntactic filter are added to the graph, and an edge is added between those lexical units that co-occur within a window of words.

After the graph is constructed (undirected unweighted graph), the score associated with each vertex is set to an initial value of 1, and the ranking algorithm described in section 2 is run on the graph for several iterations until it converges — usually for 20–30 iterations, at a threshold of 0.0001.

The score is calculated by the following formula:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Once a final score is obtained for each vertex in the graph, vertices are sorted in reversed order of their score, and the top vertices in the ranking are retained for post-processing.

During post-processing, all lexical units selected as potential keywords by the TextRank algorithm are marked in the text, and sequences of adjacent keywords are collapsed into a multi-word keyword.

In Gensim, the function "keywords" does the above process for us.

The result is still not meeting our expectations and the reasons are:

We tried to combine the reviews together and then extract keywords from them, but the Colab crashed. So, we extract keywords from each review and then combine them together.

A phrase we need maybe "terrible service", but it may not be in high rank in this keyword's extraction algorithm. We need to define a new rule to filter these phrases and may also need to remove some words like "hotel".

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | (hotel, 567) | (room, 680) | (hotel, 878) | (hotel, 2552) | (hotel, 4142) |
| 1 | (room, 507) | (hotel, 662) | (room, 729) | (room, 1866) | (room, 2332) |
| 2 | (rooms, 219) | (rooms, 302) | (rooms, 373) | (great, 1039) | (great, 1734) |
| 3 | (hotels, 174) | (hotels, 201) | (good, 296) | (good, 924) | (stay, 1427) |
| 4 | (stay, 161) | (stay, 198) | (stay, 228) | (stay, 826) | (hotels, 997) |
| 5 | (staff, 107) | (night, 148) | (hotels, 219) | (rooms, 724) | (rooms, 912) |
| 6 | (night, 100) | (good, 128) | (nice, 195) | (nice, 617) | (staff, 837) |
| 7 | (service, 86) | (service, 121) | (great, 185) | (hotels, 606) | (good, 684) |
| 8 | (place, 84) | (resort, 110) | (location, 172) | (location, 482) | (stayed, 573) |
| 9 | (day, 71) | (staff, 103) | (beach, 140) | (staff, 451) | (location, 519) |
| 10 | (resort, 69) | (beach, 98) | (night, 137) | (beach, 380) | (nice, 494) |
| 11 | (food, 61) | (day, 93) | (staff, 114) | (night, 334) | (service, 437) |
| 12 | (like, 60) | (food, 85) | (resort, 104) | (stayed, 280) | (excellent, 433) |
| 13 | (staying, 60) | (time, 77) | (service, 96) | (breakfast, 269) | (night, 403) |
| 14 | (stayed, 59) | (stayed, 73) | (clean, 88) | (resort, 268) | (beach, 383) |
| 15 | (time, 57) | (nice, 73) | (breakfast, 87) | (clean, 255) | (breakfast, 368) |
| 16 | (bed, 54) | (like, 72) | (small, 85) | (service, 249) | (day, 350) |
| 17 | (desk, 50) | (location, 68) | (stayed, 84) | (little, 209) | (place, 347) |
| 18 | (days, 47) | (staying, 57) | (pool, 84) | (walk, 202) | (staying, 329) |
| 19 | (good, 46) | (bed, 57) | (time, 81) | (time, 201) | (resort, 325) |

Figure 8. Keywords by Rating using Gensim

### spaCy

As an advanced NLP library, spaCy is geared toward performance and operating together with deep learning frameworks such as TensorFlow or PyTorch. It comes with pre-trained

statistical models and word vectors. It features tokenization for 50+ languages, convolutional neural network models for tagging, parsing, and named entity recognition. It has some functionalities, such as tokenization, POS, NER, classification, sentiment analysis, dependency parsing, word vectors, etc.

The process of this approach is: *review corpus -> syntactic dependencies analysis -> filter by pattern -> drop meaningless words -> gathering -> sentiment analysis -> ranking*

After syntactic dependencies analysis, one important thing is to filter the target phrases. There're several ways to do this in spaCy, after comparing their functions, we choose the token-based matching way because we can make use of the dependencies in this way. We also try different patterns for the token-based matching, then finally find the one we need and add it to the pipeline. Then we drop some more manually added stop words to optimize the result.

We classify 1, 2 and 3-star reviews as negative reviews and the rest as positive reviews. Figure 9 shows the top 20 most common word combination for each category.

```
collections.Counter(extract_bad_comments).most_common(20)
```

```
[('hot water', 169),
 ('good value', 151),
 ('standard room', 88),
 ('big deal', 79),
 ('good service', 78),
 ('clean room', 75),
 ('free internet', 75),
 ('mini bar', 75),
 ('good food', 75),
 ('good deal', 74),
 ('friendly staff', 72),
 ('double beds', 71),
 ('half hour', 71),
 ('new room', 70),
 ('poor service', 68),
 ('french quarter', 66),
 ('small room', 65),
 ('double room', 64),
 ('great place', 63),
 ('nice room', 59)]
```

```
collections.Counter(extract_good_comments).most_common(20)
```

```
[('great place', 701),
 ('good value', 579),
 ('friendly staff', 467),
 ('great value', 463),
 ('free internet', 397),
 ('great service', 303),
 ('short walk', 300),
 ('flat screen', 285),
 ('nice touch', 283),
 ('helpful staff', 283),
 ('good size', 276),
 ('fresh fruit', 237),
 ('comfortable bed', 223),
 ('little bit', 219),
 ('hot water', 211),
 ('french quarter', 211),
 ('good food', 206),
 ('mini bar', 205),
 ('bottled water', 205),
 ('great food', 203)]
```

Figure 9. Collection of  Bad/Good Comments

**Conclusions**

spaCy is a useful tool for keyphrases extraction.

For short paragraphs like reviews, compared with traditional word vectors, adding syntactic dependencies analysis and sentiment analysis do give us better results.

Need further works on similar words, for example: ‘good deal’ and ‘big deal’. We tried simhash but does not work well because it works better for documents; we also tried TF-idf and Word2vec, but for n-grams it’s hard to compare their similarities.

Similar to point 3, we want to develop a better ranking method, which can take into account similar phrases.

**Yake+BrownClustering**

Yake is a keyword extractor that does not rely on dictionaries nor thesauri, neither is trained against any corpora. Instead, it follows an unsupervised approach that builds upon features extracted from the text, making it thus applicable to documents written in different languages without the need for further knowledge. Thus Yake is a good tool that could help to reduce the dimension. The objective is that we want to find what are the frequent complaints in low rating reviews, and the things that people are most satisfied with when giving high ratings. So we basically processed as following steps:

Take the review corpus by rating

Using YAKE extract keywords from each review. Get the output of the keyword and its weight.

Using a list of tokenized keywords obtained from YAKE as input of BrownClustering, find the most related words of a specific term.

E.g. If we want to know in 1-star reviews, what do people say about “service”. We need to input the term "service", and the model will return the most words which are related to "services".

```
similar_rating1 = get_similar(clustering1,'service',20)
similar_rating1
```

```
t[28 [('told', 384),
 ('called', 383),
 ('desk', 383),
 ('rude', 383),
 ('staff', 383),
 ('gave', 383),
 ('bar', 383),
 ('make', 382),
 ('reservations', 381),
 ('customer', 380),
 ('extremely', 379),
 ('feel', 378),
 ('problem', 377),
 ('speak', 376),
 ('making', 375),
 ('person', 374),
 ('non', 373),
 ('lost', 372),
 ('spoke', 371),
 ('overpriced', 370)]
```

Figure 10. Related words of “service” in 1-star reviews

We input the keywords ‘service’, and from the output we got, some negative adjectives, such as rude, and overpricing, may suggest that the service quality in 1-star hotels is not that satisfying.

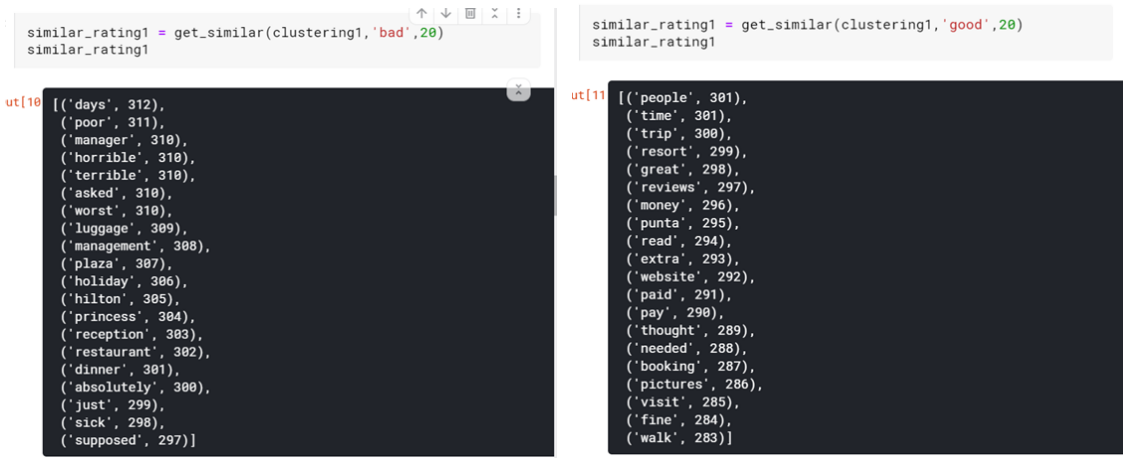



Figure 11. Related words of "good" and "bad" in 1-star reviews

Then we tried to find what people think is good or bad about the hotel, when they are giving 1 star to them. As we can see, service quality, and dining place could be the most dissatisfied factors. But the good thing is that usually, the hotels offer a good price.

**Conclusions**

Yake + BrownClustering is fast.

Yake is a good keyword extracting tool for processing short content, and robust with stop words and punctuation.

Combine the data cleaning and feature engineering, and save a lot of time for data processing.

However, the clusters are only made up of some single words, instead of word pairs. So we only have a comparatively vague understanding of our corpus.

### 3.6 Rating prediction

It is a classification problem. The target feature is Rating. We can use the words TF-IDF and other features extracted from reviews, like sentiment, to predict the rating. In this case, the models we used are Decision Tree, Naive Bayes, and Support Vector Classification (SVC).

We do a data preprocessing to remove the special characters, turn the words into lowercase, lemmatize the words, remove the stopwords, and get the tf-idf of the top 200 words. For the feature extracting, we get the word count, character count, average word length, stopword count, and sentiment (pos, neg, and neu) from each review. The dataset was divided into 2 partitions: the training dataset (70%) and the testing dataset (30%).

From the relationship between features, we can see sentiment_pos is positively correlated with rating (0.58), and sentiment_neg is negatively correlated with rating (-0.60). However, word count, char count, word length, and stopword count are not significantly correlated with rating. So we choose sentiment data as features combined with tf-idf to train and build the models.

| | Rating | words_count | chars_count | words_length | stops_count | pos | neg | neu |
|---|---|---|---|---|---|---|---|---|
| **Rating** | 1.000000 | -0.089711 | -0.079844 | 0.236676 | -0.085622 | 0.575785 | -0.601517 | -0.313964 |
| **words_count** | -0.089711 | 1.000000 | 0.997598 | -0.177120 | 0.995596 | -0.299153 | 0.093955 | 0.307509 |
| **chars_count** | -0.079844 | 0.997598 | 1.000000 | -0.129102 | 0.998537 | -0.294731 | 0.085574 | 0.307583 |
| **words_length** | 0.236676 | -0.177120 | -0.129102 | 1.000000 | -0.126800 | 0.260838 | -0.212779 | -0.181672 |
| **stops_count** | -0.085622 | 0.995596 | 0.998537 | -0.126800 | 1.000000 | -0.301802 | 0.089342 | 0.313829 |
| **pos** | 0.575785 | -0.299153 | -0.294731 | 0.260838 | -0.301802 | 1.000000 | -0.589667 | -0.845826 |
| **neg** | -0.601517 | 0.093955 | 0.085574 | -0.212779 | 0.089342 | -0.589667 | 1.000000 | 0.067924 |
| **neu** | -0.313964 | 0.307509 | 0.307583 | -0.181672 | 0.313829 | -0.845826 | 0.067924 | 1.000000 |

Figure 12. Correlationships between features

We build 3 Decision Tree models using tf-idf data and set the max depth as None, 6, and 10. The accuracy of each DT is 0.4320, 0.4700, and 0.4760. We also build a DT using tf-idf data and sentiment data and set the max depth as 10. The fourth DT results in slightly lower classification accuracy, 0.4770. From Figure 13, when we set the max_depth = 6, or 10, we have a better tree.

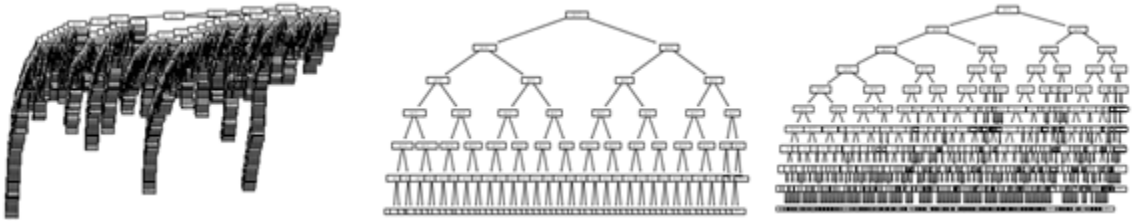


Figure 13. Decision Tree

(Left: set max_depth=None, Middle: set max_depth=6, Right: set max_depth=10)

Then we build 2 Naïve Bayes models. The first NB model using tf-idf data has a slightly higher accuracy (0.5223) than the second NB model using tf-idf data and sentiment data (0.5058).

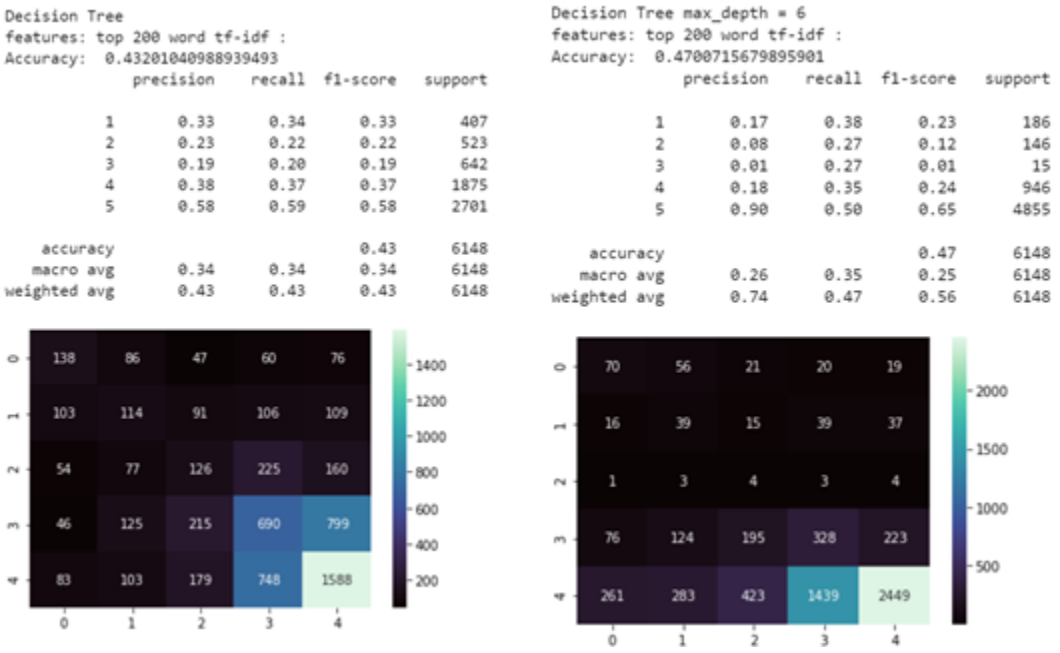
The SVC models have better performance than DT and NB. The accuracy of SVC using tf-idf data is 0.5801, and the accuracy of SVC using tf-idf data and sentiment data is 0.5816.

Overall, the SVC model has the best performance.

## Evaluation

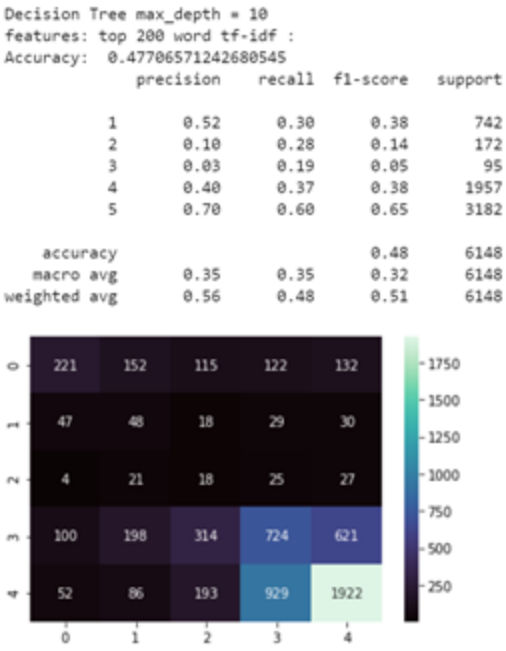
We use unsupervised methods to explore key aspects that make hotels good or bad. So, in this part, we will only talk about rating prediction evaluation.

For DT, the max_depth is an important parameter that can influence the model performance. Figure 14 shows the performance of 3 DT. When the depth of the tree is limited to 6 or 10, the tree has better performance, especially for 5-star reviews. For DT, the words *told, wonderful, excellent, great, perfect, and good* are the key features in predicting rating.




(a) max_depth = None

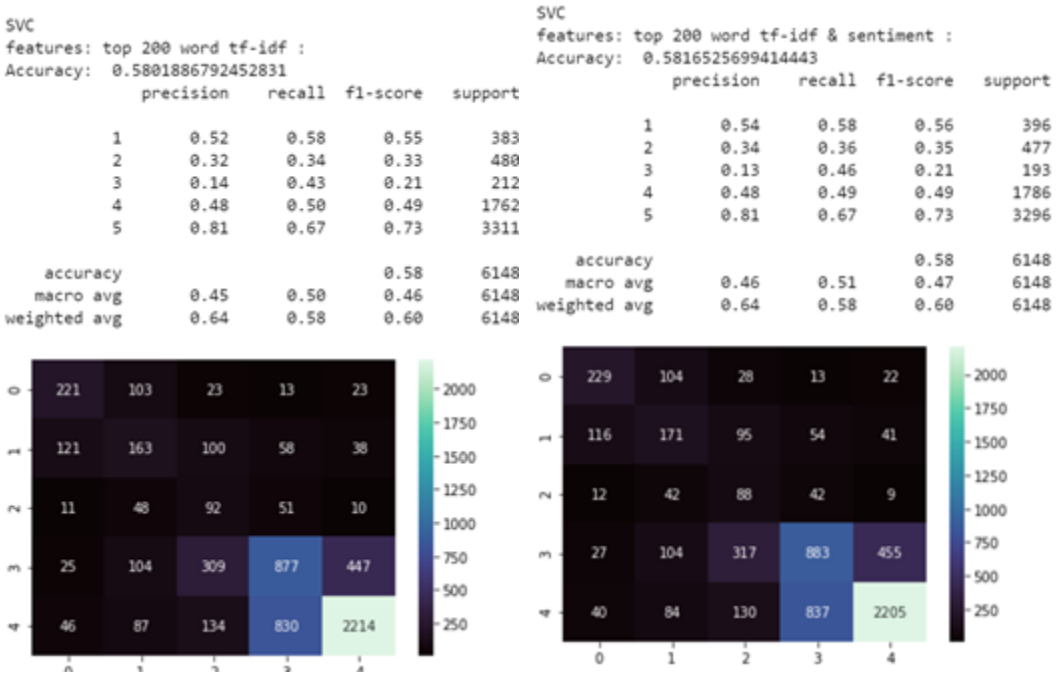
(b) max_depth = 6




(c) max_depth = 10

Figure 14. Confusion Matrix and accuracy score of DT models

Except for the SCV model, DT and NB have a better performance when we only use tf-idf data to build the model. For SCV, when we use tf-idf data and sentiment data to build the model, the accuracy is 0.5816, which is slightly higher than the model using tf-idf (0.5801). So, the sentiment data correlated with rating but have little impact on the rating prediction. We also try to use more features, like word count, character count, and stopword count, to build the prediction model. This results in a worse performance. So the length of the reviews is not a key feature for rating prediction.




(a) SVC using tf-idf (b) SVC using tf-idf and sentiment

Figure 15. Confusion Matrix and accuracy score of SVC