A revised version of tutorial material (Sept. 21, 2022) made by Prof. Junegak Joung (Hanyang Univ.)

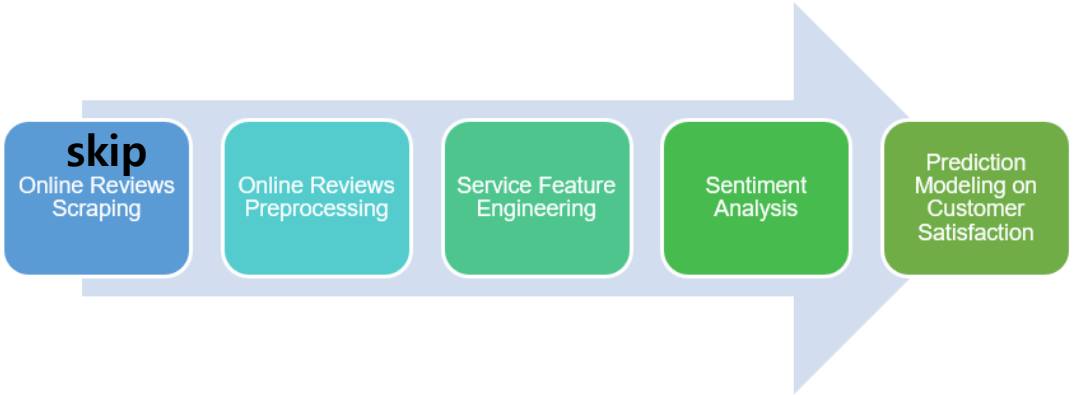
Online Review Mining Tutorial

Jongkyung Shin

shinjk1156@unist.ac.kr

Learning Objectives

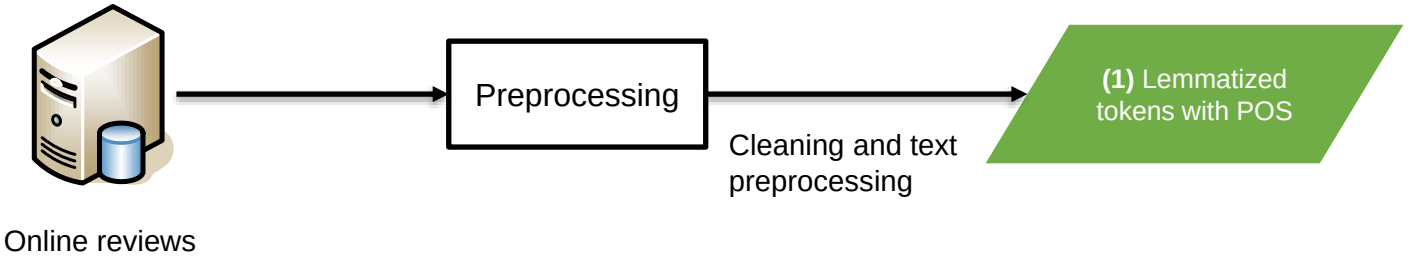
- Get to know:
  - How to preprocess online reviews
  - How to execute LDA topic modeling as feature engineering
  - How to execute VADER sentiment analysis
  - How to build the logit model as prediction modeling on customer satisfaction



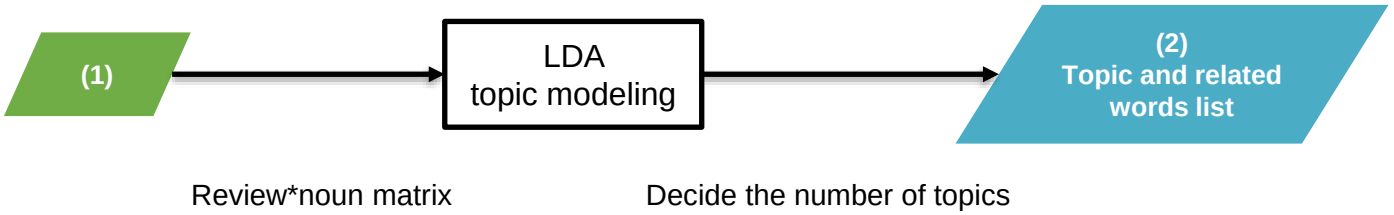
Online Review Mining Framework for Service Quality Improvement

Tutorial of Online Review Mining for Service Quality Improvement

(1) Online reviews preprocessing

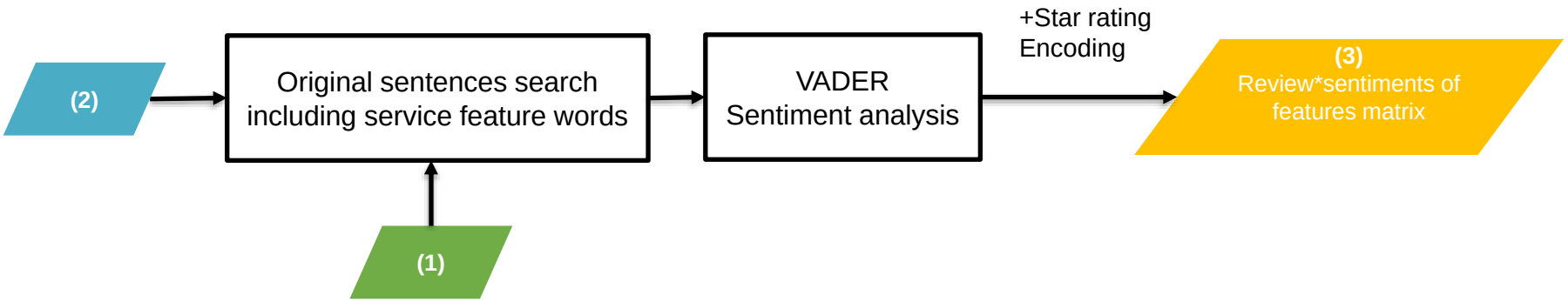


(2) Service Feature Engineering



Tutorial of Online Review Mining for Service Quality Improvement

(3) Sentiment Analysis



(4) Prediction modeling on customer satisfaction

Dataset

- Singapore hotels under four ratings
- Time: 2010.01 – 2019.12
- Number of reviews: about 30,000

Dataset

- HotelRev_less4STAR.xlsx

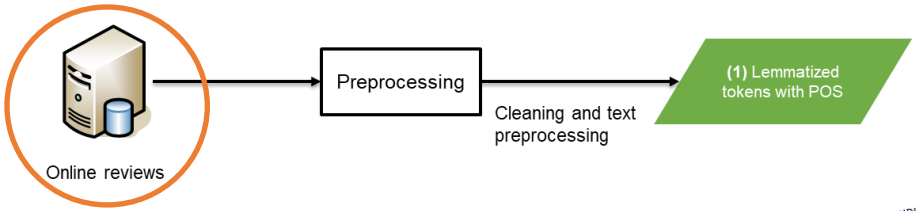
Class	Name	id	date	date_of_s	title	body	rating	trip_type	aspect	aspect_ra
2	Arianna H	Zee7	10월-11	10월-11	Expect Pu	Arrived at	10	Travelled	['Value', 'I	['10', '10',
2	Arianna H	triippyy	10월-11	10월-11	Not bad, i	The hotel	50	Travelled	['Value', 'I	['40', '40',
2	Arianna H	Minibreal	10월-11	10월-11	Better the	Expected	30	Travelled	['Value', 'I	['40', '30',
2	Arianna H	Masrura F	12월-11	10월-11	Really in t	My reasoı	30		['Value', 'I	['30', '20',
2	Arianna H	Genevie A	1월-12	10월-11	Average	Don't exp	30	Travelled	['Value', 'I	['40', '30',
2	Arianna H	Sapphire4	11월-11	11월-11	Overprice	The hotel	20	Travelled	['Value', 'I	['30', '20',
2	Arianna H	PCUY	2월-12	12월-11	Comforta	Stayed 5 r	40	Travelled	['Value', 'I	['50', '40',
2	Arianna H	keenaz	12월-11	12월-11	ok la	although	30	Travelled	['Value', 'I	['30', '30',
2	Arianna H	cinta_alar	2월-12	12월-11	No frills b	my family	30	Travelled	['Value', 'I	['30', '30',
2	Arianna H	ariellek_b	1월-12	1월-12	Budget tr	Singapore	40	Travelled	['Value', 'I	['40', '30',
2	Arianna H	OnaBeech	2월-12	2월-12	A lot of nc	You might	20	Travelled	['Value', 'I	['30', '20',
2	Arianna H	RonnySigu	4월-12	3월-12	The only c	Singapore	30	Travelled	['Value', 'I	['30', '20',
2	Arianna H	Luah G	3월-12	3월-12	Bad decis	Was our f	30	Travelled	['Value', 'I	['20', '20',
2	Arianna H	Danno_9(	4월-12	3월-12	Small, sm	Arianna is	20	Travelled	['Value', 'I	['30', '20',
2	Arianna H	cj_shiju	3월-12	3월-12	Great Loc	After reac	30	Travelled	['Value', 'I	['20', '30',
2	Arianna H	brendon l	2월-13	3월-12	Not good	I stayed h	10	Travelled	['Value', 'I	['10', '10',

2class 3class 4class

Loading Dataset

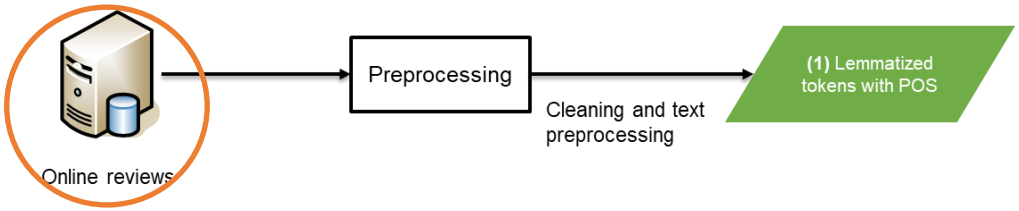
- Loading hotel reviews from the excel file
- Data=[review 1, review 2, … review n],
- Review n=[class, hotel name, id, date1, date2, title, body, rating, trip_type, aspect, aspect_rating]

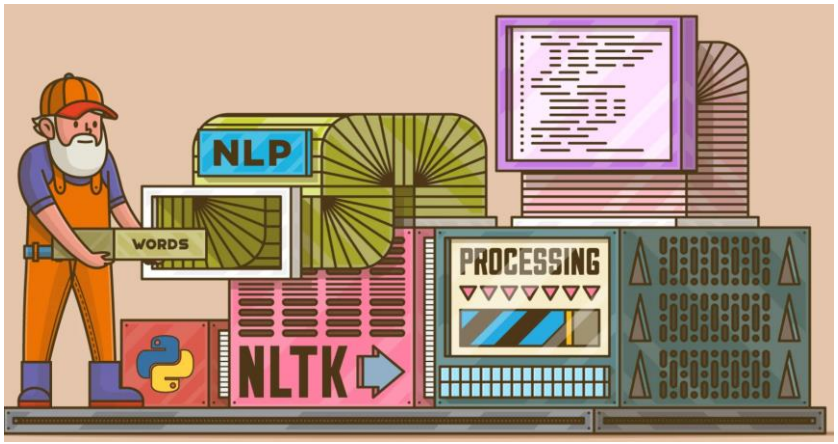
	Class	Name	id	date	date_of_stay	title	body	rating	trip_type	aspect	aspect_rating
0	2	Arianna Hotel	Zee7	2011-10-01	2011-10-01	Expect Pure Misery	Arrived at the hotel at 2PM for my stay on Thu...	10	Travelled solo	['Value', 'Rooms', 'Location', 'Cleanliness', ...	['10', '10', '10', '10', '10', '10']
1	2	Arianna Hotel	triippyy	2011-10-01	2011-10-01	Not bad, not bad at all	The hotel was cosy and it was perfect for a so...	50	Travelled solo	['Value', 'Rooms', 'Location', 'Cleanliness', ...	['40', '40', '40', '40', '40', '40']
2	2	Arianna Hotel	Minibreak_11	2011-10-01	2011-10-01	Better then 1*	Expected the worst after booking a 1* rated ho...	30	Travelled solo	['Value', 'Rooms', 'Location', 'Cleanliness', ...	['40', '30', '40', '30', '40', '30']
3	2	Arianna Hotel	Masrura Ramidjal	2011-12-01	2011-10-01	Really in the Centre of Shopping Centre	My reason to stay in this hotel was the Locati...	30	NaN	['Value', 'Rooms', 'Location', 'Cleanliness', ...	['30', '20', '50', '30', '20']
4	2	Arianna Hotel	Genevie A	2012-01-01	2011-10-01	Average	Don't expect the luxuries of a hotel if you ar...	30	Travelled as a couple	['Value', 'Rooms', 'Location', 'Cleanliness', ...	['40', '30', '30', '20', '30', '20']
...	...	...	...	...	...	...	...	...	...	...	...



Online Reviews Preprocessing

- Remove review data that is not in the analysis period
- Remove unnecessary attributes
- Splitting sentences and clearing
- Text preprocessing
  - Lowercasing
  - Tokenization and Lemmatization
  - Removal of stop words
  - POS (Part-Of-Speech) tagging
  - Noun extraction





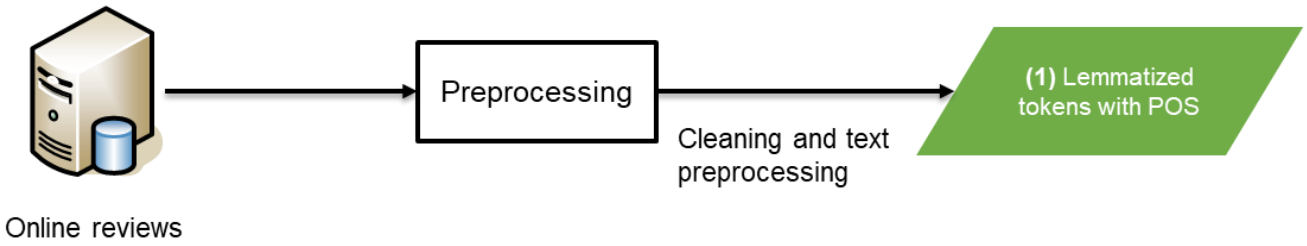
Online Reviews Preprocessing

- Remove review data that is not in the analysis period (Depending on your analysis purpose)
  - Make sure “date of stay” falls within the analysis period
- Remove unnecessary attributes
- Splitting sentences and clearing

```
def makeClearSent(sent):
    sent = str(sent)
    sent = sent.replace("\n", "")
    sent = sent.replace("\r", "")
    sent = sent.replace("  ", " ")
    return sent


def makeReviewList(df):
    review_list = []
    for i in range(len(df)):
        review_list.append(makeClearSent(df.loc[i, "title"]) + ". " + makeClearSent(df.loc[i, "body"]))

    '''
    --- same as ----
    review_list = [makeClearSent(df.loc[i, "title"]) + ". " + makeClearSent(df.loc[i, "body"]) for i in range(len(df))]
    '''
    return review_list
```

Online Reviews Preprocessing

- Merging the data




```
star2_review_list = makeReviewList(df_star2)
star3_review_list = makeReviewList(df_star3)
star4_review_list = makeReviewList(df_star4)

star2_rating_list = (df_star2.loc[:,"rating"]//10).to_list()
star3_rating_list = (df_star3.loc[:,"rating"]//10).to_list()
star4_rating_list = (df_star4.loc[:,"rating"]//10).to_list()
```

Text Preprocessing: Lowercasing, Tokenization, and Lemmatization

- Lowercasing is required to handle words mixed with uppercase letters as semantically equal
  - ex) Room <-> room, Staff <-> staff

ex="Expected the worst after booking a 1* rated hotel. Surprised with good location, easy access to shopping and MRT. Staff very friendly, rooms basic, yet clean with all the basic amenities. ."

```
#Lowercasing
ex=ex.lower()
print(ex)
```

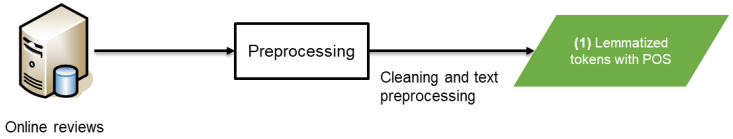
```
expected the worst after booking a 1* rated hotel.  surprised with good location, easy access to shopping and mrt.  staff very f
riendly, rooms basic, yet clean with all the basic amenities.
```

```
print("Original sentence : ",ExampleSentence + "\n")

LowerCasedSentence = ExampleSentence.lower() # for string
print("Lowercased Sentence : ", LowerCasedSentence)

print("\nOriginal word token list : ", ExampleWordTokenList)

LowerCasedTokenList = []
for token in ExampleWordTokenList:
    LowerCapToken = token.lower() # for string
    LowerCasedTokenList.append(LowerCapToken)

print("\nLowercased word token list : ", LowerCasedTokenList)

ExampleWordTokenList = LowerCasedTokenList
```

```
Original sentence :  Again, he laughed and proceeded to recommend other hotels in the area.

Lowercased Sentence :  again, he laughed and proceeded to recommend other hotels in the area.

Original word token list :  ['Again', ',', 'he', 'laughed', 'and', 'proceeded', 'to', 'recommend', 'other', 'hotels', 'in', 'the', 'area', '.']

Lowercased word token list :  ['again', ',', 'he', 'laughed', 'and', 'proceeded', 'to', 'recommend', 'other', 'hotels', 'in', 'the', 'area', '.']
```

Text Preprocessing: Lowercasing, Tokenization, and Lemmatization

- Tokenization is required to handle words as tokens
  - ex) What a nice hotel! <-> [‘What’, ‘a’, ‘nice’, ‘hotel’, ‘!’]




```
ExampleSentence = SentenceTokenList[10]

ExampleWordTokenList = word_tokenize(ExampleSentence)

print("The number of words in a sentence : {}\n".format(len(ExampleWordTokenList)))
print("Original sentence : ",ExampleSentence + "\n")
print("Tokenized sentence : ", ExampleWordTokenList)
```

```
The number of words in a sentence : 14

Original sentence :  Again, he laughed and proceeded to recommend other hotels in the area.

Tokenized sentence :  ['Again', ',', 'he', 'laughed', 'and', 'proceeded', 'to', 'recommend', 'other', 'hotels', 'in', 'the', 'area', '.']
```

- Lemmatization is required to handle inflectional forms as semantically equal
  - ex) rooms <-> room, staffs <-> staff

```
lemma = WordNetLemmatizer()

LemmatizationTokenList = []
for token in ExampleWordTokenList:
    LemmatizedToken = lemma.lemmatize(token)
    LemmatizationTokenList.append(LemmatizedToken)

print("\nOriginal word token list : ", ExampleWordTokenList)

print("\nLemmatized word token list : ", LemmatizationTokenList)

ExampleWordTokenList = LemmatizationTokenList
```

```
Original word token list :  ['again', ',', 'he', 'laughed', 'and', 'proceeded', 'to', 'recommend', 'other', 'hotels', 'in', 'the', 'area', '.']

Lemmatized word token list :  ['again', ',', 'he', 'laughed', 'and', 'proceeded', 'to', 'recommend', 'other', 'hotel', 'in', 'the', 'area', '.']
```

Text Preprocessing: Removal of Stop Words

- Stop words are a set of commonly used words in any language
  - ex) the, is, and, she, he, they, that, etc.

```
import nltk
from nltk.corpus import stopwords
print(stopwords.words('english'))
```

{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}

```
StopWordRemovalTokenList = []
for token in ExampleWordTokenList:
    if token not in StopWordList:
        StopWordRemovalTokenList.append(token)

print("\nOriginal word token list : ", ExampleWordTokenList)

print("\nStopword Removed token list : ", StopWordRemovalTokenList)


ExampleWordTokenList = StopWordRemovalTokenList
```

```
Original word token list :  ['again', ',', 'he', 'laughed', 'and', 'proceeded', 'to', 'recommend', 'other', 'hotel', 'in', 'the', 'area', '.']

Stopword Removed token list :  [',', 'laughed', 'proceeded', 'recommend', 'hotel', 'area', '.']
```

Text Preprocessing: POS (Part-Of-Speech) Tagging

- POS tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech

CC	Coordinating conjunction	NNS	Noun, plural	UH	Interjection
CD	Cardinal number	NNP	Proper noun, singular	VB	Verb, base form
DT	Determiner	NNPS	Proper noun, plural	VBD	Verb, past tense
EX	Existential there	PDT	Predeterminer	VBG	Verb, gerund or present participle
FW	Foreign word	POS	Possessive ending	VBN	Verb, past participle
IN	Preposition or subordinating conjunction	PRP	Personal pronoun	VBP	Verb, non-3rd person singular present
JJ	Adjective	PRP$	Possessive pronoun	VBZ	Verb, 3rd person singular present
JJR	Adjective, comparative	RB	Adverb	WDT	Wh-determiner
JJS	Adjective, superlative	RBR	Adverb, comparative	WP	Wh-pronoun
LS	List item marker	RBS	Adverb, superlative	WP$	Possessive wh-pronoun
MD	Modal	RP	Particle	WRB	Wh-adverb
NN	Noun, singular or mass	SYM	Symbol		
		TO	to		

```
PosTaggedTokenList = pos_tag(ExampleWordTokenList)

print("\nOriginal word token list : ", ExampleWordTokenList)

print("\nPos tagged token list : ", PosTaggedTokenList)
```

```
Original word token list :  [',', 'laughed', 'proceeded', 'recommend', 'hotel', 'area', '.']

Pos tagged token list :  [(',', ','), ('laughed', 'VBD'), ('proceeded', 'JJ'), ('recommend', 'JJ'), ('hotel', 'NN'), ('area', 'NN'), ('.', '.')]
```

Online Reviews Preprocessing

- POS tagged tokens

Review example =“Expected the worst after booking a 1* rated hotel. Surprised with good location, easy access to shopping and MRT. Staff very friendly, rooms basic, yet clean with all the basic amenities.”

POS tagged tokens for one review=
[[('expected', 'VBN'), ('the', 'DT'), ('worst', 'JJS'), ('after', 'IN'), ('booking', 'VBG'), ('a', 'DT'), ('1', 'CD'), ('*', 'NN'), ('rated', 'VBN'), ('hotel', 'NN'), ('.', '.’)],
[('surprised', 'VBN'), ('with', 'IN'), ('good', 'JJ'), ('location', 'NN'), (',', ','), ('easy', 'JJ'), ('access', 'NN'), ('to', 'TO'), ('shopping', 'NN'), ('and', 'CC'), ('mrt', 'NN'), ('.', '.’)],
[('staff', 'NN'), ('very', 'RB'), ('friendly', 'RB'), (',', ','), ('rooms', 'NNS'), ('basic', 'VBP'), (',', ','), ('yet', 'CC'), ('clean', 'JJ'), ('with', 'IN'), ('all', 'PDT'), ('the', 'DT'), ('basic', 'JJ'), ('amenities', 'NNS'), ('.', '.’)]]

Text Preprocessing: Noun Extraction

- Noun Extraction is the process to extract service feature candidates in review data

Pos tag

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

```
NounTokenList = []
for token, pos in PosTaggedTokenList:
    if pos[0] == "N":
        NounTokenList.append(token)

print("\nOriginal word token list : ", ExampleWordTokenList)

print("\nNoun token list : ", NounTokenList)
```

```
Original word token list :  [',', 'laughed', 'proceeded', 'recommend', 'hotel', 'area', '.']

Noun token list :  ['hotel', 'area']
```

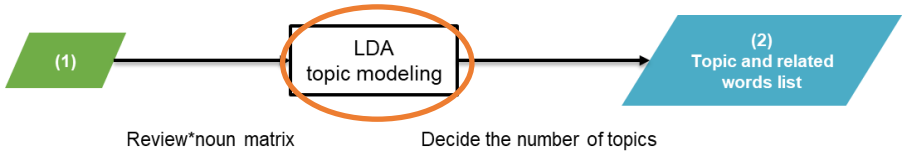
Service Feature Engineering

- LDA topic modeling
  - LDA_input=[[noun1, noun2, …noun n], [noun1, noun2, …noun n], …[noun1, noun2, …noun n]]

Review 1 Review 2 Review M

Review*keyword matrix (input)

	$Keyword_1$	…	$Keyword_m$
$Review_1$			
…			
$Review_n$			



Review*topic matrix (output)

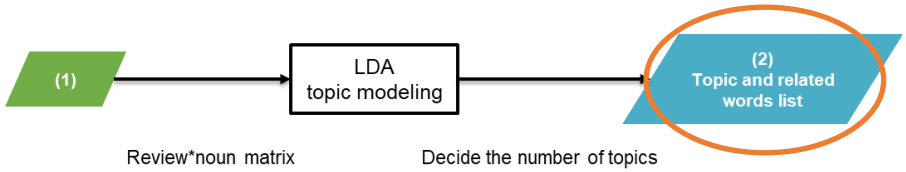
	$Topic_1$	…	$Topic_m$
$Review_1$			
…			
$Review_n$			

Topic*keyword matrix (output)

	$Keyword_1$	…	$Keyword_m$
$Topic_1$		…	
…			
$Topic_n$			

Service Feature Engineering

- Execute LDA using genism library

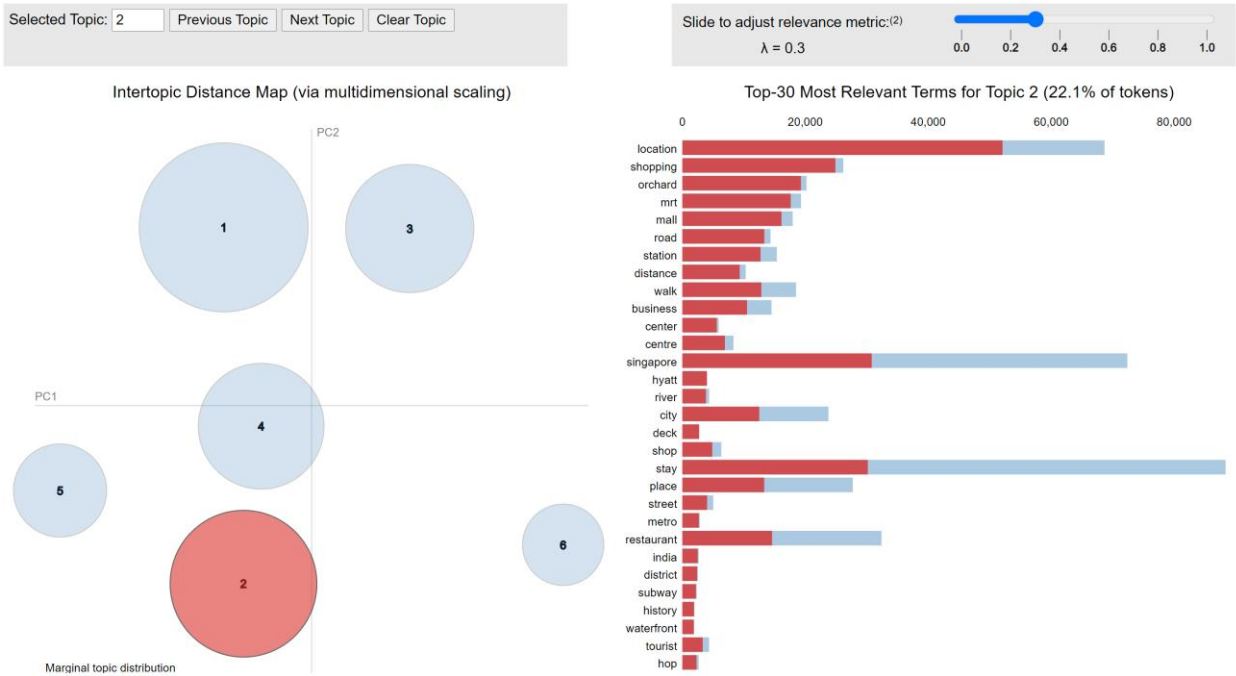



```
import gensim
import gensim.corpora as corpora

'''Create dictionary'''
id2word= corpora.Dictionary(LDA_input)
corpus = [id2word.doc2bow(rev) for rev in LDA_input]
```

```
#Peform LDA model
model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                        id2word=id2word,
                                        num_topics=9,
                                        iterations=3000,
                                        alpha=0.1,
                                        eta=0.01)

model.print_topics()
```

Service Feature Engineering

```
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
#Visualize LDA model

pyLDAvis.enable_notebook()

vis= pyLDAvis.gensim.prepare(model, corpus, id2word)
vis
# pyLDAvis.save_html(vis, 'LDA visualization.html')
```



Service Feature Engineering

- LDA output (example)

Word list of Topic 1 (name : 'location')

```
['location', 'accessibility', 'station', 'mrt', 'distance', 'proximity', 'chinatown', 'mtr', 'underground', 'mrts', 'downtown',
'transportation', 'cbd', 'metro', 'subway', 'transport', 'heart', 'place', 'train', 'shopping', 'shoping', 'funan', 'hawker',
'paragon', 'tekka', 'newton', 'shop', 'mustafa', 'mustapha', 'mustaffa', 'closeby', 'zhongshan', 'katong', 'riverfront', 'conve
ntion', 'eatery', 'supermarket', 'shopper', 'suntec', 'cinema', 'area', 'surroundings', 'mall', 'complex', 'centre', 'arcade',
'center', 'plaza', 'paradise', 'hub', 'landmark', 'grocery', 'mcdonald', 'starbucks', 'mcdonalds', 'starbuck', 'smrt', 'railwa
y', 'bus', 'tube', 'stn', 'stop', 'interchange']
```

Hotel service features in Singapore (example)

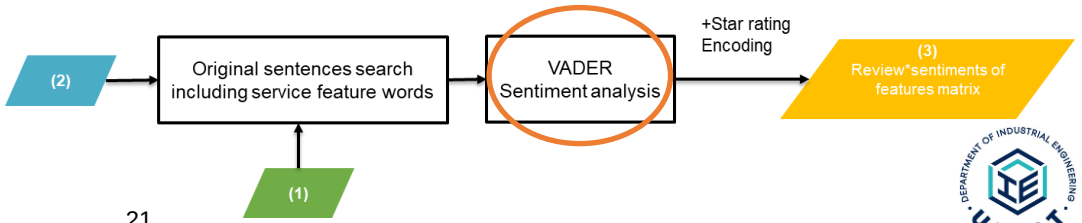
	Feature	Frequent word	# of words	# of reviews
f_1	Location	location, $\cdots$	63	26,700
f_2	View	view, outlook,$\cdots$	15	6,527
f_3	Breakfast	breakfast, buffet,$\cdots$	24	13,484
f_4	Sleep quality	bed, mattress,$\cdots$	20	10,707
f_5	Bathroom	bathroom, toilet,$\cdots$	24	11,466
f_6	Service	service, staff,$\cdots$	32	20,864
f_7	Check	check, checkin,$\cdots$	19	12,651
f_8	Value	value, price,$\cdots$	6	11,477
f_9	Internet	internet, wifi,$\cdots$	32	6,137

Sentiment Analysis

- VADER
  - A full list of emoticons (e.g., ";-)" and ":-(”), acronyms and initialisms (e.g., “LOL” and “BRB”), and frequently used slang (e.g., “sux” and “meh”) on social media was also incorporated
  - −1 (extremely negative) and 1 (extremely positive)

```
# Example
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()

example1 = "Location was great, nearby landmark"
example2 = "Awesome view, Wow"
example3 = "City veiw was not good"
example4 = "Terrible room condition"
example5 = "great quality internet"

print("Sentence  : [Sentiment score] ")
print("{0} : {1}".format(example1, analyser.polarity_scores(example1)['compound']))
print("{0} : {1}".format(example2, analyser.polarity_scores(example2)['compound']))
print("{0} : {1}".format(example3, analyser.polarity_scores(example3)['compound']))
print("{0} : {1}".format(example4, analyser.polarity_scores(example4)['compound']))
print("{0} : {1}".format(example5, analyser.polarity_scores(example5)['compound']))
```

```
Sentence  : [Sentiment score]
Location was great, nearby landmark : 0.6597
Awesome view, Wow : 0.836
City veiw was not good : -0.3412
Terrible room condition : -0.4767
great quality internet : 0.6249
```



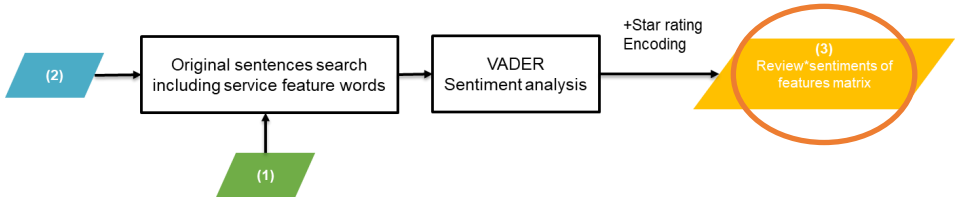
Sentiment Analysis

```python
#Sentiment analysis
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()

def show_sentence_sentiment (TopicWordList_topic, TokenDataset, ReviewList):
    for i,review in enumerate(TokenDataset):
        SentenceTokenizedList = nltk.sent_tokenize(ReviewList[i])
        for j,sent in enumerate(review):
            for word in sent:
                if word in TopicWordList_topic:
                    print("word: ",word)
                    print("tokenlist: ",sent)
                    print("Sent: ",SentenceTokenizedList[j])
                    print("Sentiment score: {}\n".format(analyser.polarity_scores(SentenceTokenizedList[j])['compound']))

# Only for topic 1
show_sentence_sentiment(TopicWordList[0], TokenDataset,  ReviewList)
```

```
word:  stay
tokenlist:  ['hotel', 'cosy', 'perfect', 'solo', 'stay', '.']
Sent:  The hotel was cosy and it was perfect for a solo stay.
Sentiment score: 0.5719

word:  shopping
tokenlist:  ['24', 'hour', 'shopping', 'centre', 'stone', 'throw', 'away', 'add', 'value', 'location', '.']
Sent:  There is a 24 hour shopping centre just a stones throw away which adds more value to its location.
Sentiment score: 0.4005
```

Sentiment Analysis

- Dataset preparation for prediction model



Review 1

S1: Surprised with good **location**, easy access to shopping and MRT. → Location (f_1): 0.772
S2: **Room** was clean. → Room (f_2): (0.402)
S3: **Staffs** were friendly and efficient. → Employee (f_i): (0.719)

$$S_{im} = \begin{cases} 4, & if\ 0.525 \leq \text{Sentiment intensity} \leq 1 \\ 3, & if\ 0.05 \leq \text{Sentiment intensity} < 0.525 \\ 0, & if\ -0.05 < \text{Sentiment intensity} < 0.05 \\ 2, & if\ -0.525 < \text{Sentiment intensity} \leq -0.05 \\ 1, & if\ -1 \leq \text{Sentiment intensity} \leq -0.525 \end{cases}$$

	f_1	f_2	…	f_i	Star ratings
1	4	3		4	1
2	0	2	…	0	0
3	0	0	…	0	1
…	…	…	…	…	…
M	3	4	…	2	1

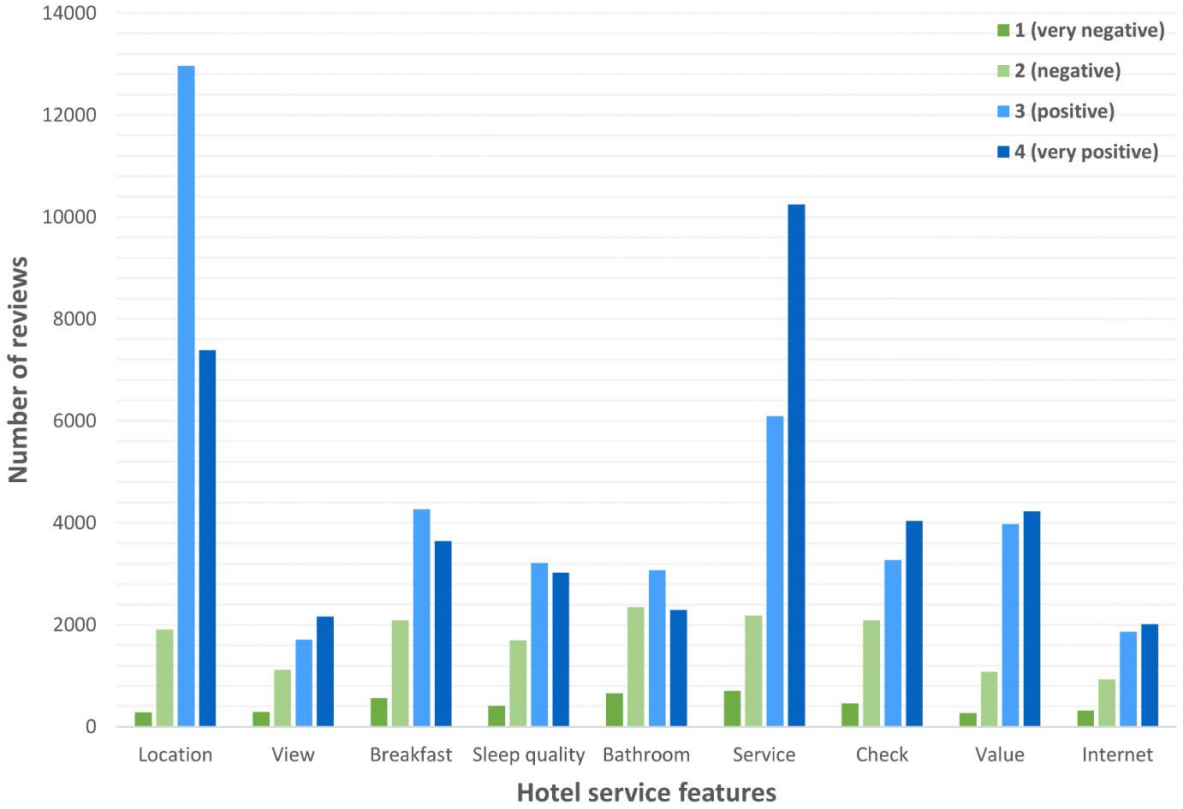
→ Negative label (1, 2, 3 ratings)

→ Positive label (4, 5 ratings)

Input variables

Output variables

Sentiment Analysis



Statistical results of sentiment scores of each feature in reviews of all hotels

Prediction Modeling on Customer Satisfaction

- Execute linear logit model using scikit-learn library

	f_1	f_2	…	f_i	Star ratings
1	4	3		4	0
2	0	2	…	0	0
3	0	0	…	0	1
…	…	…	…	…	…
M	3	4	…	2	1

→ Negative label (1, 2, 3 ratings)

→ Positive label (4, 5 ratings)

Input variables

Output variables

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

logit=LogisticRegression()
logit=logit.fit(All_input,All_y_2la)

print(logit.score(All_input,All_y_2la))
print(logit.coef_)
```

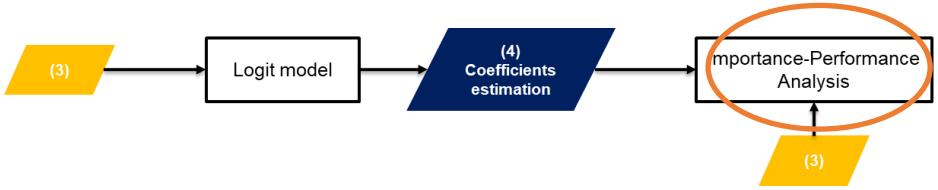
```
0.6298839096242667
[[ 0.18386161  0.06931302  0.0955905  -0.00675778 -0.0964391   0.26514475
  -0.00630123  0.0700245   0.03206919]]
```

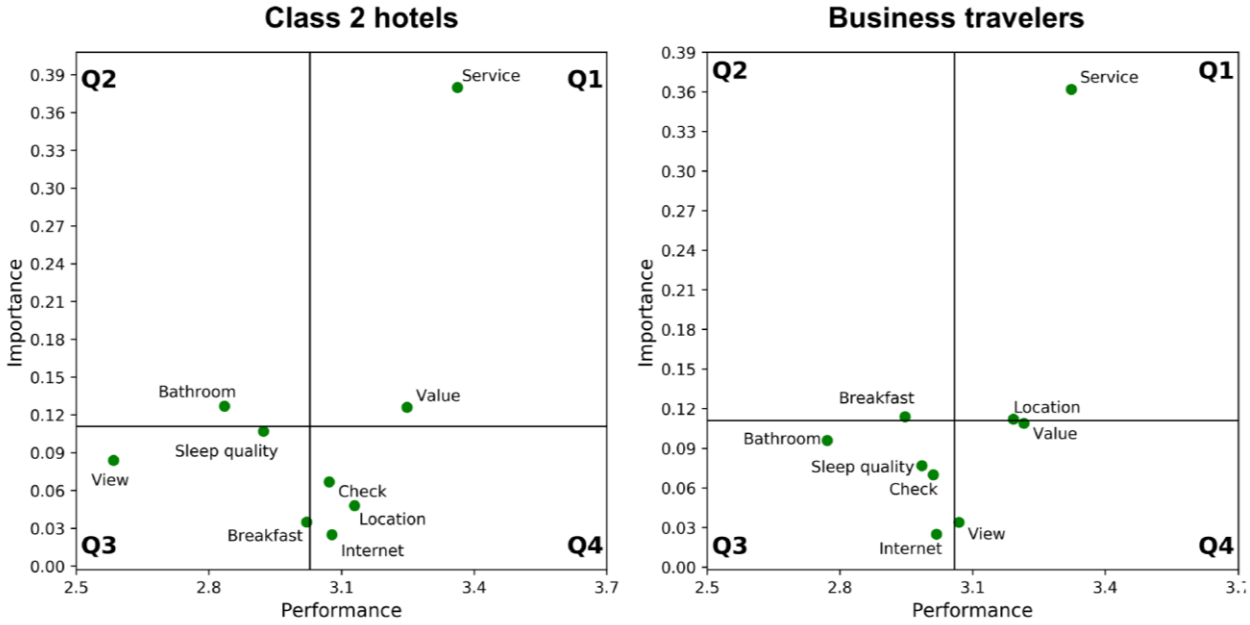
$$Y_i = \beta_0 + \beta_1 A_1 + \beta_2 A_2 + \cdots \beta_i A_i,$$

Y_i: $customer\ satsfaction$
A_1: $performance\ of\ quality\ attributes$

Service Improvement Implications from Online Review Mining

- Importance-Performance Analysis (IPA) for service improvement
  - Q1: “Keep up the good work”
  - Q2: “Concentrate here”
  - Q3: “Low priority”
  - Q4: Possible overkill”





Check ReviewMining_Practice.ipynb file in BlackBoard

Service Intelligence Practice

Loading dataset

1) Online Review Preprocessing

1-1) Tokenization

1-2) Lowercasing

1-3) Lemmatization

1-4) Stopwords removal

1-5) Part-Of-Speech Tagging (Pos tagging)

1-6) Noun Extraction

Your work (1)

Output datasets shape checker

2) LDA topic modeling (service feature engineering)

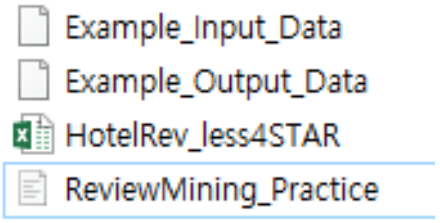
Your work (2)

3) Sentiment Analysis

Your Work (3)

4) Prediction Modeling on Customer Satisfaction

Your work (4)



Following the description,
Enter your code in this box

```
################ Write Your code ####################


#################################################################
```

Thank you