Week 3

Text Analytics II

Dr Anagi Gamachchi

Discipline of Information Systems and Business Analytics,
Deakin Business School

DEAKIN BUSINESS SCHOOL

AACSB ACCREDITED

EFMD EQUIS ACCREDITED

Making Sense of Textual Data

As an Airport Manager

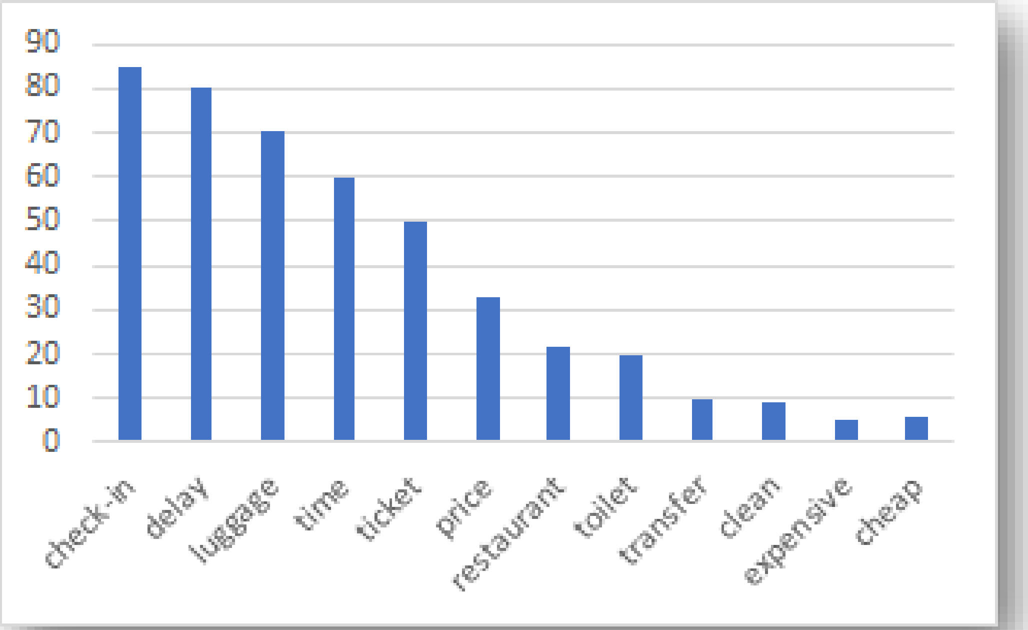
what you would be interested in knowing from customer's comments/discussions?

Airport Reviews (source: Kaggle.com)

airport_name	author	author_country	**content**
aalborg-airport	Klaus Malling	Denmark	A small very effective airport with few flights. Check-in is notorious quick and staff friendly arrival very quick and busses to Aalborg frequent. Usually no problems getting taxis as well. There used to be a cafeteria but nowadays just a kiosk - but good cafeteria with reasonable prizes inside terminal. Security check quick and friendly as well. There is a nice viewing pavilion at one end of the airport. Outside note the famous "kiss and goodbye signs". Restrooms outside terminal however few.
aalborg-airport	S Kroes	Netherlands	This is a nice and modern airport at the moment they are expanding the airport so there is a lot of building going on but in the departure area you will not notice this very much. The Airport has got free Wifi and a small restaurant with shop on the land side. Airside you will find a small shop with pre-packed sandwiches and hot dogs and other small stuff a small duty free shop is also around but not very cheap. There is no Lounge to be found at the moment but after the expansion is completed there will be one available (around May 2013). Check-in procedures are fast and the waiting area after check-in is fine with a view on the tarmac. All in all a nice modern but small airport with expensive restaurants and shop.
aalborg-airport	M Andersen	Denmark	A very nice airy terminal - that seems modern enough. Free WIFI and free parking. Everything within walking distance. Most people travel domestic to Copenhagen but a rising number of international routes e.g. AAL-AMS makes for a lot of possibilities. Check-in is very quick and so is Security. All in all a nice experience.

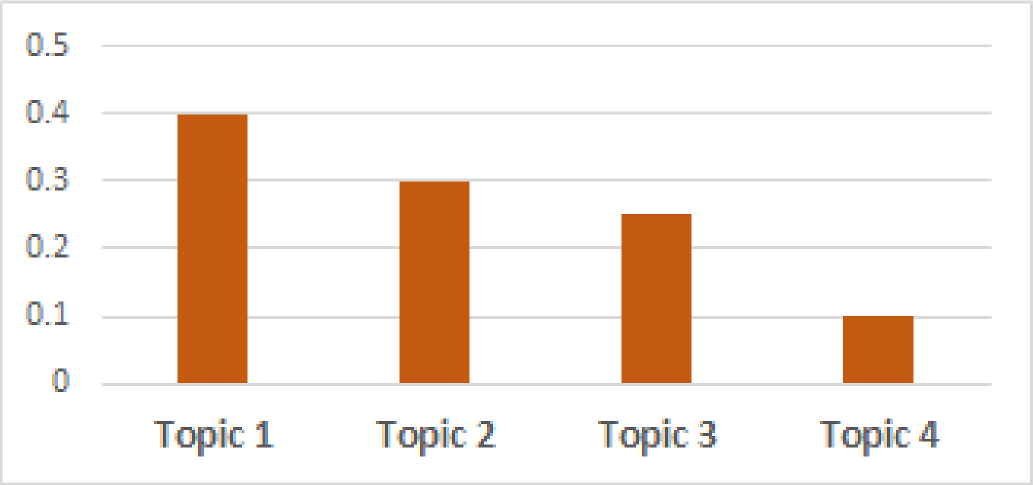
Making Sense of Textual Data (cont.)

Can we just rely on word count for insights?



Topics

Group 1	**Group 2**	**Group 3**	**Group 4**
ticket	restaurant	transfer	toilet
price	cheap	time	clean
expensive	clean	delay	



A topic is a **group of words** that are likely to appear in the same **context**

Topic Modelling

- **Topic modelling (TM):** a method for finding a group of words (topic) from a collection of documents.
- The concept of topic modeling was first introduced under the name "**latent semantic indexing**"[Papadimitriou et al., 2000].
- **Topic modeling** techniques can be grouped into two categories depending their mathematical foundation:
  - Linear Algebra:

    Singular Value Decomposition (SVD) [Dumais, 2005]

    Non-negative Matrix Factorization [Arora, Ge, & Moitra, 2012]
  - Probability :

    Probabilisticlatent semantic analysis (PLSA) [Hofmann, 1999]

    ***Latent Dirichlet Allocation* (LDA)** [Blei, Ng, & Jordan, 2003]

https://www.kdnuggets.com/2016/07/text-mining-101-topic-modeling.html

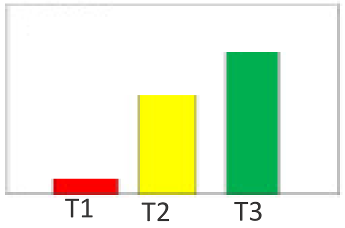
Latent Dirichlet Allocation

LDA model topics based on **Probability Distribution**:

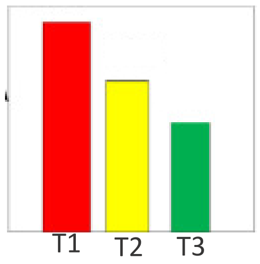
- A text data set is assumed to have number of **topics** with various proportions (probabilities)
- Each **topic** is a group of words frequently appear together.
- Each **document** may contain a mixture of multiple topics
- LDA takes one parameter (***number_of_topics***) and estimate the probability values of
  - Topic Distribution of data set
  - Word Distributions of Topics
  - Topic Distributions of a document

Document corpus

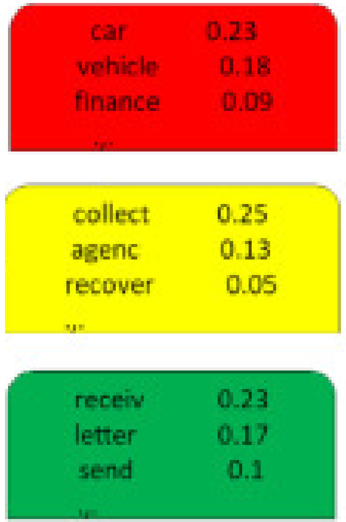
Topic Distributions of a document



Topic Distributions of data set



Word Distributions of Topics



T1

T2

T3

A toy example

Example Texts: **Titles from 9 technical documents (2 categories)**

c1: *Human* machine *interface* for ABC *computer* applications

c2: A *survey* of *user* opinion of *computer system response time*

c3: The *EPS user interface* management *system*

c4: *System* and *human system* engineering testing of *EPS*

c5: Relation of *user* perceived *response time* to error measurement

Computer user interface

m1: The generation of random, binary, ordered *trees*

m2: The intersection *graph* of paths in *trees*

m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering

m4: *Graph minors*: A *survey*

graph theory

(Key words are highlighted in red)

Topic Modelling With LDA (1 of 3)

Construct **Document x Term** Matrix (Bag-of-Word representation):

c1: *Human* machine *interface* for ABC *computer* applications

12 columns (words)

9 rows (documents)

	human	interface	computer	user	system	response	time	EPS	survey	trees	graph	minors
c1	1	1	1	0	0	0	0	0	0	0	0	0
c2	0	0	1	1	1	1	1	0	1	0	0	0
c3	0	1	0	1	1	0	0	1	0	0	0	0
c4	1	0	0	0	2	0	0	1	0	0	0	0
c5	0	0	0	1	0	1	1	0	0	0	0	0
m1	0	0	0	0	0	0	0	0	0	1	0	0
m2	0	0	0	0	0	0	0	0	0	1	1	0
m3	0	0	0	0	0	0	0	0	0	1	1	1
m4	0	0	0	0	0	0	0	0	1	0	1	1

Notice terms frequently appear together in each document category

Topic Modeling With LDA (2 of 3)

Input into LDA is **Document x Term** matrix

	human	interface	computer	user	system	response	time	EPS	survey	trees	graph	minors
c1	1	1	1	0	0	0	0	0	0	0	0	0
c2	0	0	1	1	1	1	1	0	1	0	0	0
c3	0	1	0	1	1	0	0	1	0	0	0	0
c4	1	0	0	0	2	0	0	1	0	0	0	0
c5	0	0	0	1	0	1	1	0	0	0	0	0
m1	0	0	0	0	0	0	0	0	0	1	0	0
m2	0	0	0	0	0	0	0	0	0	1	1	0
m3	0	0	0	0	0	0	0	0	0	1	1	1
m4	0	0	0	0	0	0	0	0	1	0	1	1

Train LDA

Assume we trained an LDA model with 2 Topics

LDA captured well 2 dominant topics

Topic Distribution of data set

T1	T2
0.5558	0.4442

Sum to 1

Word Distributions of Topic

	T1	T2
human	0.0992	0.0083
interface	0.0992	0.0083
computer	0.0992	0.0083
user	0.1468	0.0083
system	0.1944	0.0083
response	0.0992	0.0083
time	0.0992	0.0083
EPS	0.0992	0.0083
survey	0.0516	0.1083
trees	0.0040	0.3083
graph	0.0040	0.3083
minors	0.0040	0.2083

Sum to 1

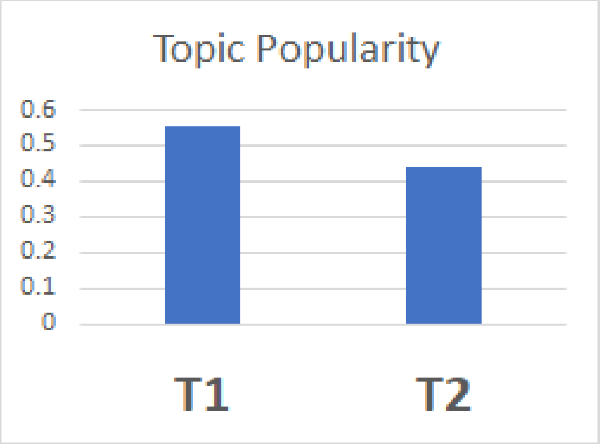
Topic Distributions of document

	T1	T2
c1	0.9998	0.0002
c2	0.9999	0.0001
c3	0.9998	0.0002
c4	0.9998	0.0002
c5	0.9998	0.0002
m1	0.0008	0.9992
m2	0.0004	0.9996
m3	0.0003	0.9997
m4	0.0003	0.9997

Sum to 1

Result Report

T1	T2
0.5558	0.4442



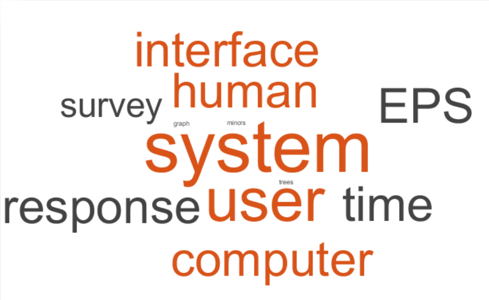
Topic Distribution of data set

	T1	T2
human	0.0992	0.0083
interface	0.0992	0.0083
computer	0.0992	0.0083
user	0.1468	0.0083
system	0.1944	0.0083
response	0.0992	0.0083
time	0.0992	0.0083
EPS	0.0992	0.0083
survey	0.0516	0.1083
trees	0.0040	0.3083
graph	0.0040	0.3083
minors	0.0040	0.2083

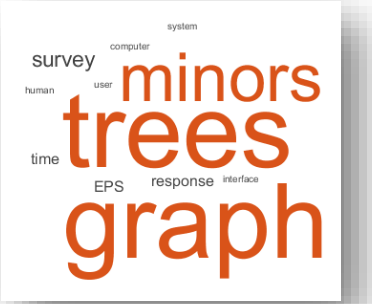
	T1	T2
c1	0.9998	0.0002
c2	0.9999	0.0001
c3	0.9998	0.0002
c4	0.9998	0.0002
c5	0.9998	0.0002
m1	0.0008	0.9992
m2	0.0004	0.9996
m3	0.0003	0.9997
m4	0.0003	0.9997

Word Distributions of Topic

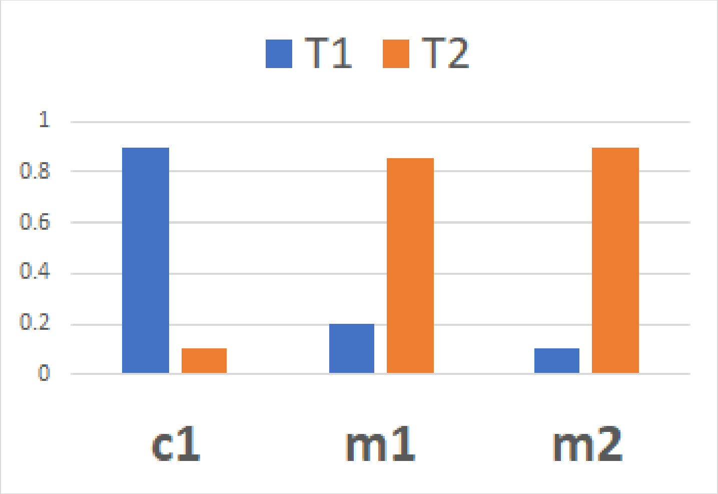
Topic 1 User interface of computer system



Topic 2 Graph theory



Topic Distributions of document



Topic Modeling With LDA (3 of 3)

What if we set the Topic Number = 4?

3 dominant topics

Topic Distribution of data set

T1	T2	T3	T4
0.444	0.333	0.222	0.000

Word Distributions of Topic

Dominant words in T2

Dominant words in T1

Dominant words in T3

	T1	T2	T3	T4
human	0.008	0.174	0.008	0.000
interface	0.008	0.174	0.008	0.000
computer	0.008	0.090	0.108	0.000
user	0.008	0.090	0.208	0.000
system	0.008	0.257	0.108	0.000
response	0.008	0.007	0.208	0.000
time	0.008	0.007	0.208	0.000
EPS	0.008	0.174	0.008	0.000
survey	0.108	0.007	0.108	0.000
trees	0.308	0.007	0.008	0.000
graph	0.308	0.007	0.008	0.000
minors	0.208	0.007	0.008	0.000

Topic Distributions of document

Dominant topic in c1

Dominant topic in c5

Dominant topic in m4

	T1	T2	T3	T4
c1	0.0002	0.9998	0.0001	0.0000
c2	0.0001	0.0001	0.9999	0.0000
c3	0.0001	0.9998	0.0001	0.0000
c4	0.0001	0.9998	0.0001	0.0000
c5	0.0002	0.0001	0.9997	0.0000
m1	0.9994	0.0003	0.0002	0.0000
m2	0.9997	0.0002	0.0001	0.0000
m3	0.9998	0.0001	0.0001	0.0000
m4	0.9998	0.0001	0.0001	0.0000

Topic Modelling in Python

Supervised or Unsupervised?

We use the "**Airport**" review dataset, available from Kaggle.

```
import pandas as pd

df = pd.read_csv('AirportReview.csv')
df.head()
```

Textual Review Comments

	airport_name	author	author_country	content	overall_rating	recommended
0	aalborg-airport	Klaus Malling	Denmark	A small very effective airport with few flight...	9.0	1
1	aalborg-airport	S Kroes	Netherlands	This is a nice and modern airport at the momen...	9.0	1
2	aalborg-airport	M Andersen	Denmark	A very nice airy terminal - that seems modern ...	9.0	1
3	aalborg-airport	Paul Van Alsten	France	AMS-AAL and quite satisfied with this regional...	5.0	0
4	aalborg-airport	K Fischer	NaN	Very quick check-inn and security screening. N...	4.0	0

Business Problem:

What problems/issues are concerning travellers most when transiting through an airport?

➔ Improve visitor experience and promote revisit

Review Preprocessing (1 of 2)

Clean text data by removing the *punctuations*, *numbers*, *special characters*, and *short words*

```
from nltk.stem import PorterStemmer #Stemming Package
import re  #Regular expression operation package

porter = PorterStemmer()

documents = df['content']
Cleaned_doc = []
for r in range(len(documents)):
    review = documents[r]
    try:
        # removing everything except alphabets
        review = re.sub('[^A-Za-z]', ' ', review)
        # make all text lowercase
        review = review.lower()
        # apply tokenization
        Tokens = review.split()
        # apply stemming operation (Optional)
        #for t in range(len(Tokens)):
        #    Tokens[t] = porter.stem(Tokens[t])
        # removing short words
        Filtered_token = [w for w in Tokens if len(w)>3]
        review = ' '.join(Filtered_token)
    except:
        continue
    #Save cleaned text
    Cleaned_doc.append(review)
    print('-[Review Text]: ', review)
```

remove *stop-word*

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

# Remove Stop Words
for r in range(len(Cleaned_doc)):
    each_item = []
    for t in Cleaned_doc[r].split():
        if t not in stop_words:
            each_item.append(t)
    Cleaned_doc[r] = ' '.join(each_item)
    print('-[Cleaned Text]: ', Cleaned_doc[r])
```

```
-[Cleaned Text]:  small effective airport flights check notorious quick staff friendly a
-[Cleaned Text]:  nice modern airport moment expanding airport building going departure
-[Cleaned Text]:  nice airy terminal seems modern enough free wifi free parking everythi
-[Cleaned Text]:  quite satisfied regional airport flights baggage reclaim understandabl
-[Cleaned Text]:  quick check security screening nice airy free parking need show airpor
-[Cleaned Text]:  aalborg lufthavn smallish airport near city aalborg usually people lea
-[Cleaned Text]:  nice cafe first floor great view overall bright free access computers
-[Cleaned Text]:  depressing airport depressing town maersk operators using besides priv
-[Cleaned Text]:  amazed find little place gets reviews staged april enroute bright clea
-[Cleaned Text]:  travelling airport every week years easy access downtown aalborg right
-[Cleaned Text]:  nice small friendly airport good transport links city meets flights pl
-[Cleaned Text]:  airport gets worse monthly basis spent million pounds front door needs
```

Review Preprocessing (2 of 2)

create a **document-term** matrix

```
from sklearn.feature_extraction.text import CountVectorizer

count_vectorizer = CountVectorizer()# Fit and transform the processed titles

count_data = count_vectorizer.fit_transform(Cleaned_doc)
count_data
```

```
<5000x12809 sparse matrix of type '<class 'numpy.int64'>'
	with 227688 stored elements in Compressed Sparse Row format>
```

Full Matrix Format
(5x6 = 30 values)

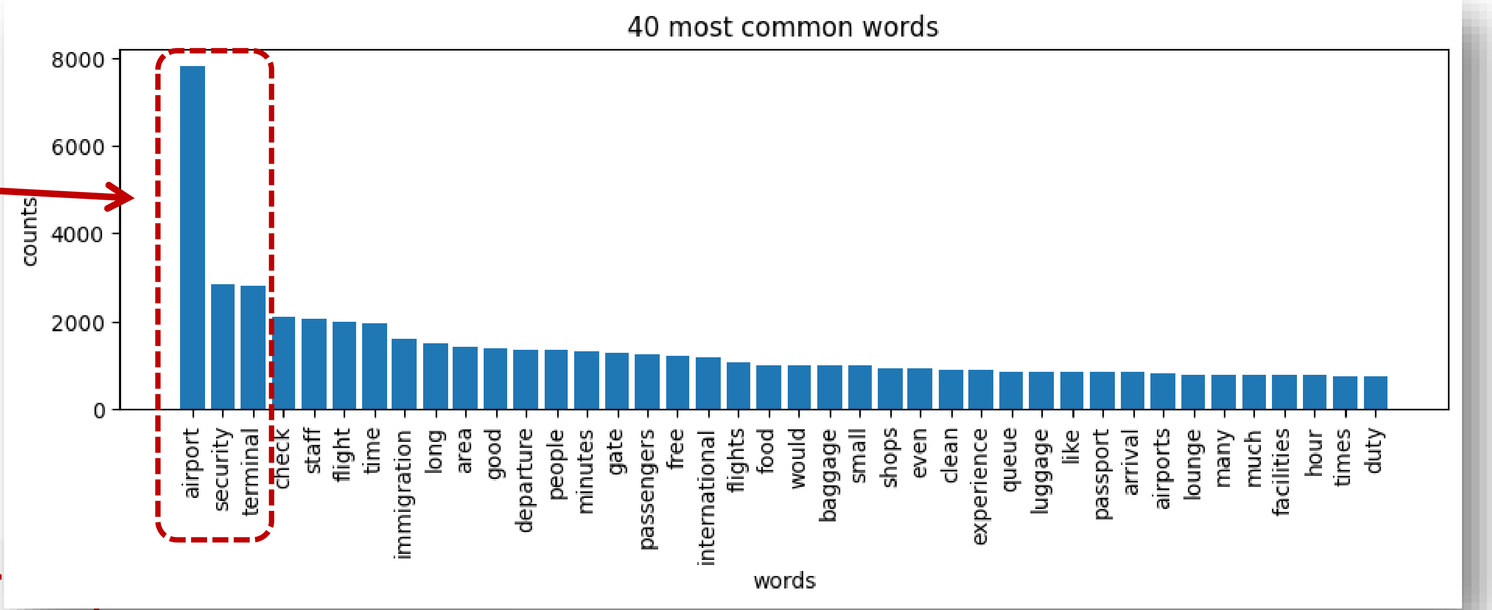
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

Rows	Columns	Values
5	6	6
0	4	9
1	1	8
2	0	4
2	3	2
3	5	5
4	2	2

Sparse Matrix Format
(7x3 = 21 values)

Removing insignificant words

Highly frequent (> 50%) or infrequent words (<0.01%) do not carry much value and thus should be discarded.




```
keepIndex = [];
for t in range(len(total_counts)):
    if total_counts[t] < 1000 and total_counts[t] > 50:
        keepIndex.append(t)

print('Number of Terms Remained: ', len(keepIndex))

#Save the remain ing term and frequency data
ReducedTerm = [terms[t] for t in keepIndex]
ReducedCount = count_data[:,keepIndex]
ReducedCount
```

```
Number of Terms Remained:  906

<5000x906 sparse matrix of type '<class 'numpy.int64'>'
	with 139281 stored elements in Compressed Sparse Row format>
```

Training LDA Model

Train an LDA model with 10 topics, based on "**sklearn**" library in python

```
from sklearn.decomposition import LatentDirichletAllocation as LDA

# Tweak the two parameters below
number_topics = 10

lda = LDA(n_components=number_topics, n_jobs=-1, random_state=2023)
lda.fit(ReducedCount)
```

Bag of Word Features
(Document x Term Matrix)

Specify the random seed to ensure the same result at every run

Topic Interpretation

View Popular Terms in each topic

```
for topic_idx, topic in enumerate(Word_Topics_Pro):
    print("\nTopic #%d:" % topic_idx)
    count_dict = (zip(ReducedTerm, topic))
    count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:5]
    for w in count_dict:
        print(w[0], ': {0:.3f}'.format(w[1]))
```

View top 5 words

```
Topic #0:
clean : 0.030
shops : 0.020
easy : 0.017
nice : 0.017
friendly : 0.017
```

```
Topic #1:
customs : 0.018
domestic : 0.014
times : 0.011
wait : 0.011
many : 0.010
```

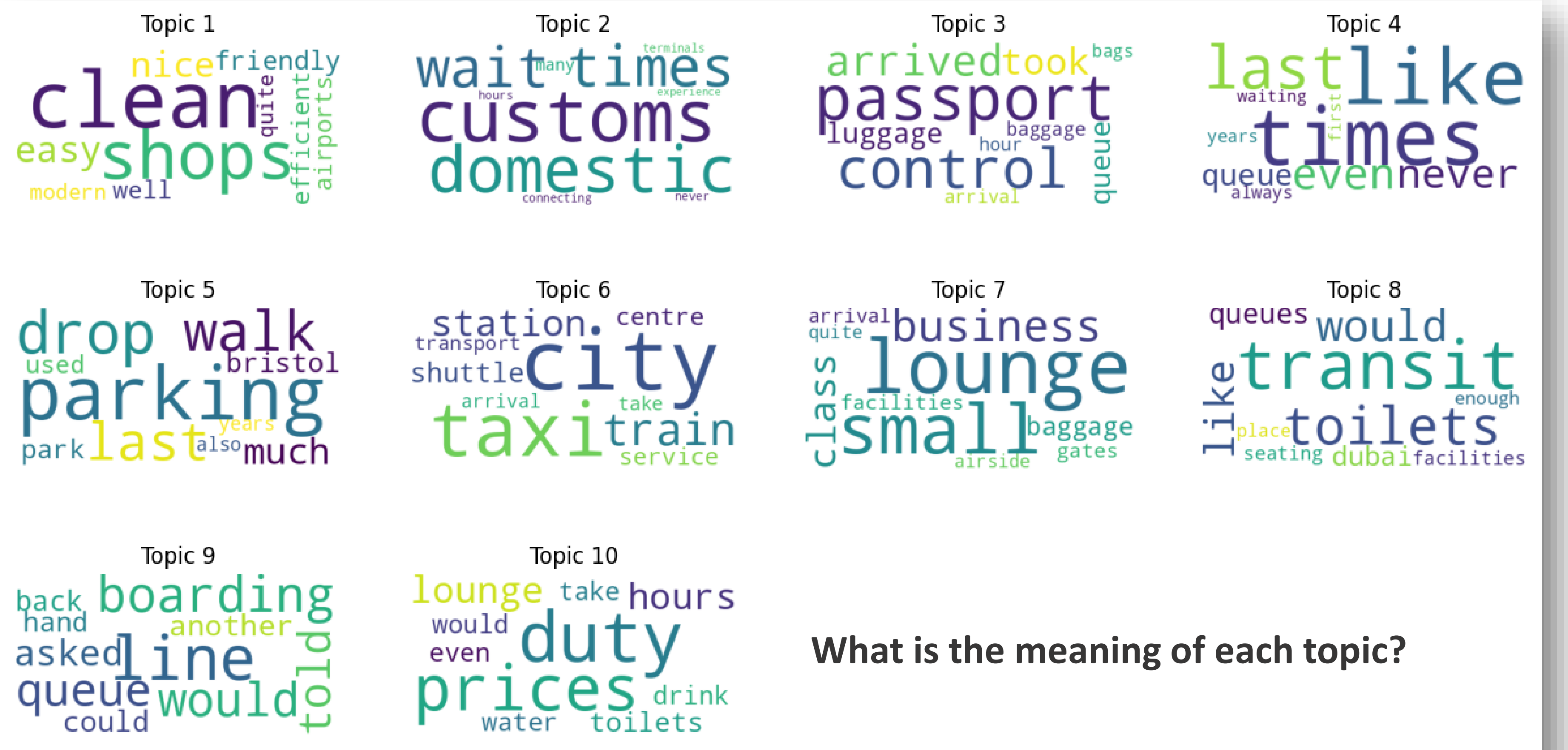
```
Topic #2:
passport : 0.036
control : 0.028
arrived : 0.021
took : 0.020
luggage : 0.018
```

```
Topic #3:
times : 0.016
like : 0.015
last : 0.013
even : 0.012
never : 0.011
```

Making sense of topic meaning (1 of 2)

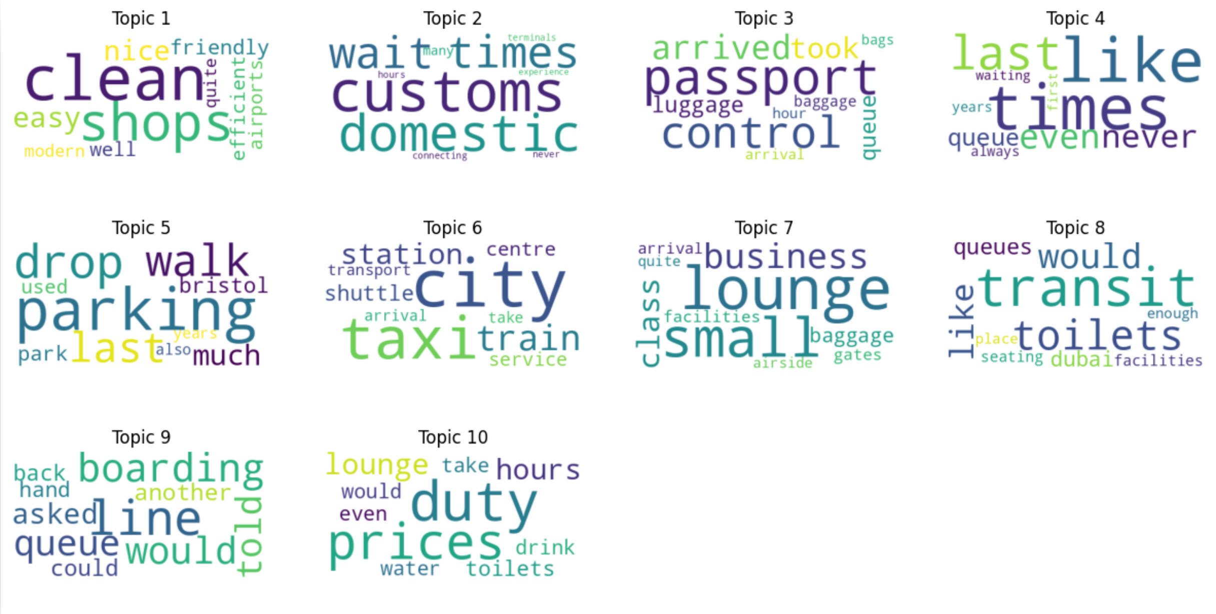
```
#install package wordcloud
!pip install wordcloud
```

Visualize the top (most popular) words in each topic



What is the meaning of each topic?

Making sense of topic meaning (2 of 2)



Topic Meaning:

T1- Shop cleanliness and friendliness

T2- Waiting time at customs

T3- Passport control

T4- Discussion about last time of visit.

T5- Parking and drop-off

T6- Transportation to city

T7- Business lounge

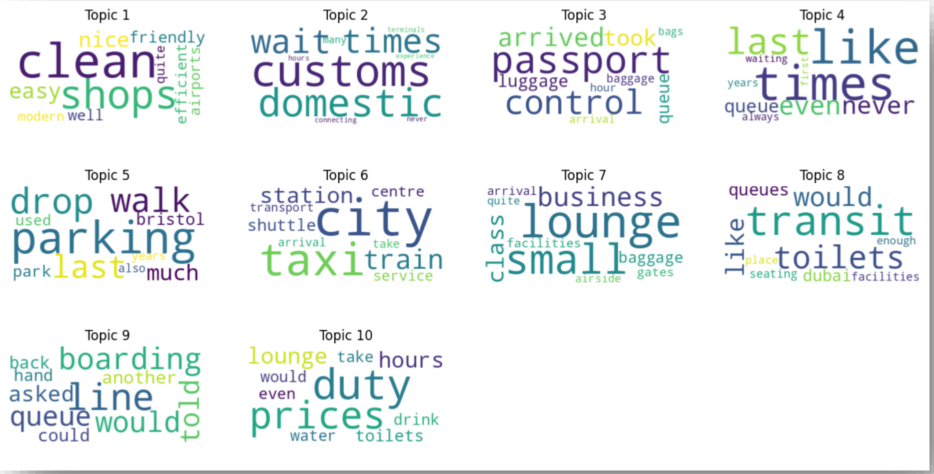
T8- Transiting experiences (queues, toilets, facilities)

T9- Queuing for boarding.

T10 – Duty free

How would these findings benefit airport managers?

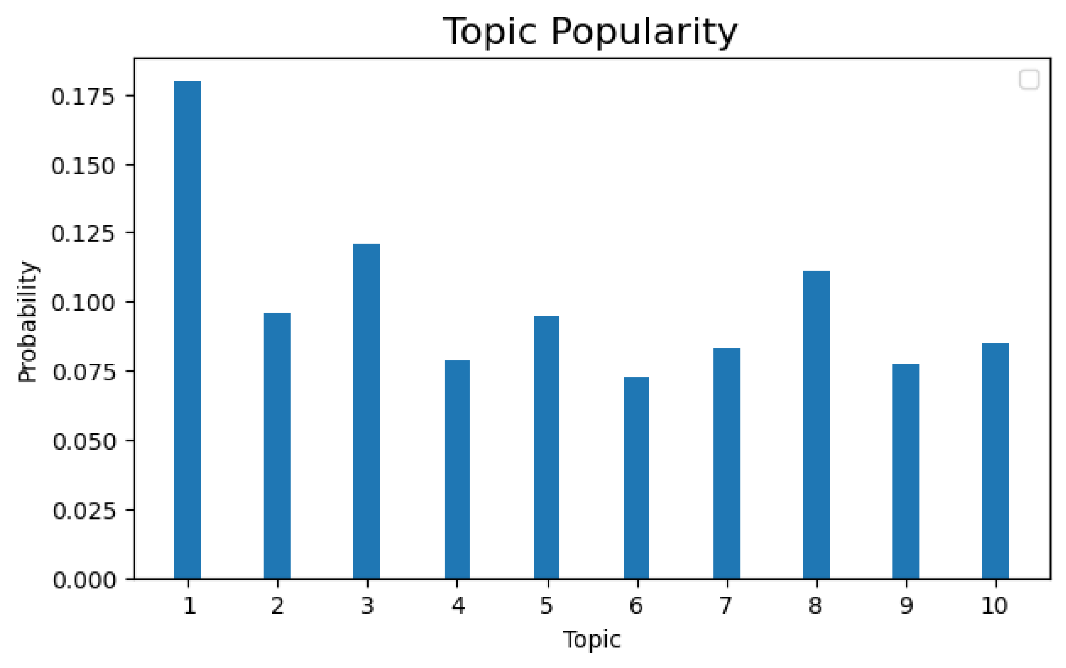
Topic Popularity




```
#Compute topic distribution for each document
TopicDis_Doc = lda.transform(ReducedCount)

#Compute overall topic distribution for all each documents
Overall_Topic_Dis = sum(TopicDis_Doc)/sum(sum(TopicDis_Doc))
Overall_Topic_Dis
```

```
array([0.17943903, 0.09602315, 0.12120395, 0.07895196, 0.09464621,
       0.0728363 , 0.08321962, 0.11145589, 0.07731108, 0.08491281])
```



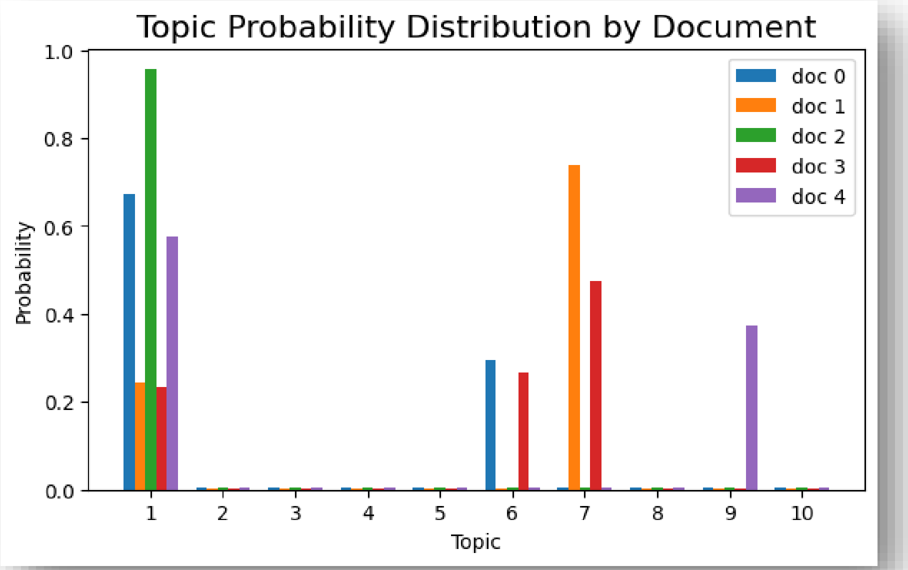
Which topics are most popular in the entire data set?

Topic distribution in document

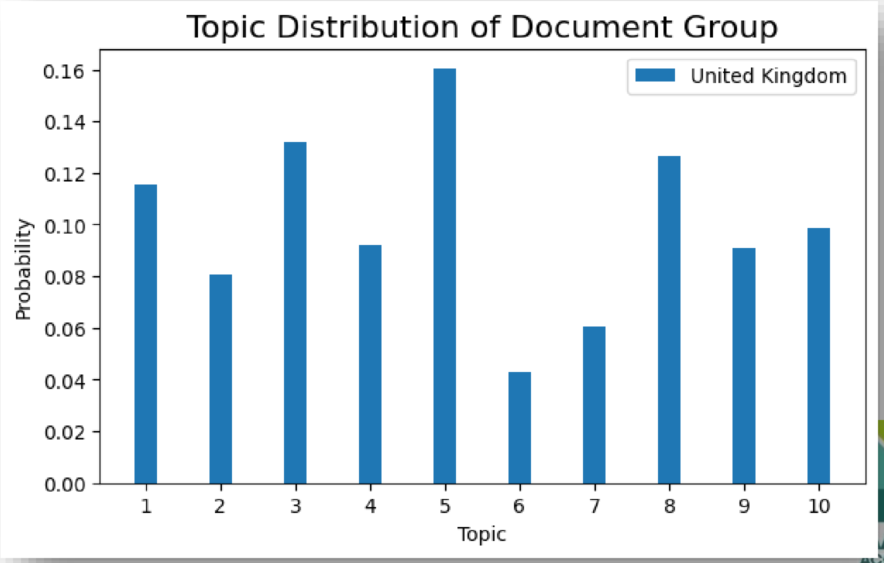
```
# View full Topic Probabilities by Document Matrix
TopicDis_Doc = lda.transform(ReducedCount)
df_document_topics = pd.DataFrame(TopicDis_Doc)
df_document_topics
```

	0	1	2	3	4	5	6	7	8	9
0	0.673977	0.004001	0.004001	0.004001	0.004001	0.294017	0.004001	0.004000	0.004000	0.004001
1	0.242688	0.002326	0.002326	0.002326	0.002326	0.002326	0.738705	0.002326	0.002326	0.002326
2	0.954991	0.005001	0.005000	0.005002	0.005002	0.005001	0.005001	0.005001	0.005000	0.005001
3	0.234488	0.003334	0.003334	0.003334	0.003334	0.267217	0.474957	0.003334	0.003334	0.003334
4	0.574567	0.006668	0.006669	0.006668	0.006671	0.006668	0.006669	0.006668	0.372084	0.006668
...	...	...	...	...	...	...	...	...	...	...
4995	0.004348	0.004349	0.251519	0.004349	0.004349	0.004349	0.004349	0.713691	0.004348	0.004349
4996	0.175661	0.005000	0.005001	0.005001	0.005001	0.005001	0.378757	0.332429	0.083147	0.005002
4997	0.066682	0.002565	0.002565	0.002565	0.002565	0.002565	0.404641	0.510724	0.002565	0.002565
4998	0.287207	0.691164	0.002704	0.002704	0.002703	0.002704	0.002704	0.002704	0.002703	0.002703
4999	0.003449	0.003450	0.320907	0.003450	0.130240	0.172356	0.003449	0.355802	0.003449	0.003449

5000 rows × 10 columns



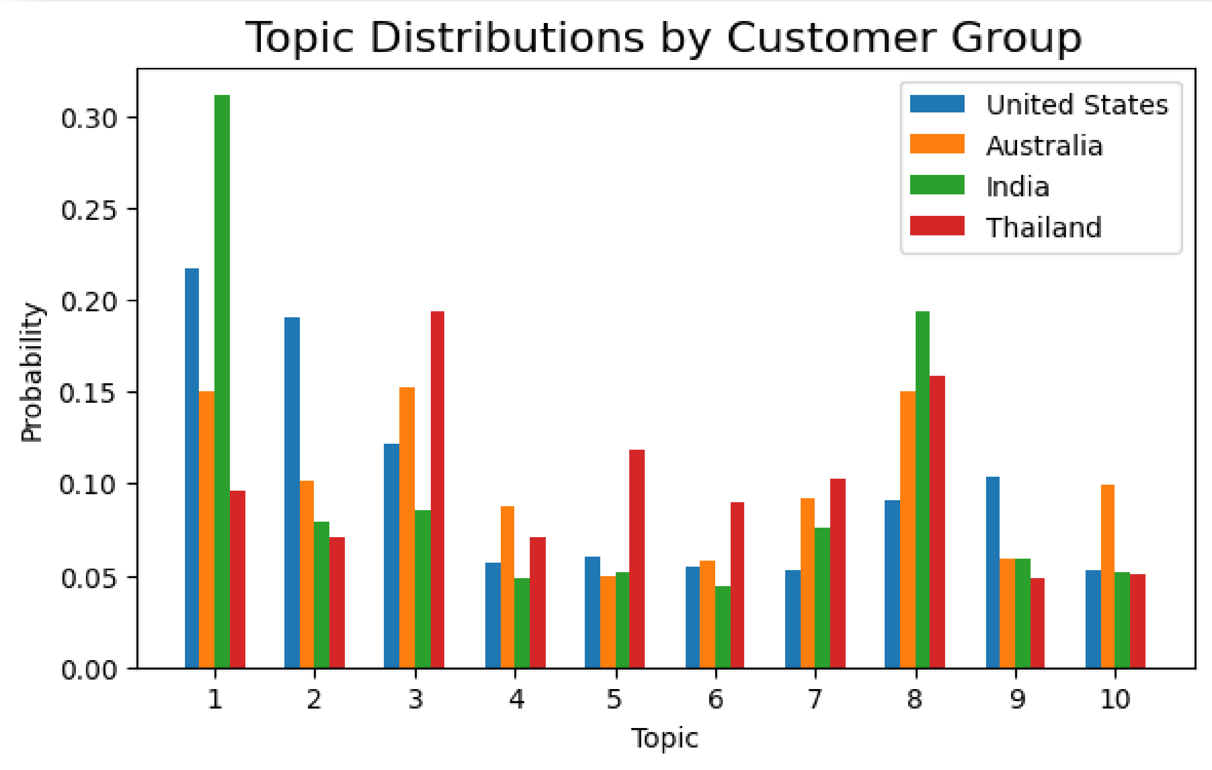
Visualize topic distribution for the first 5 documents

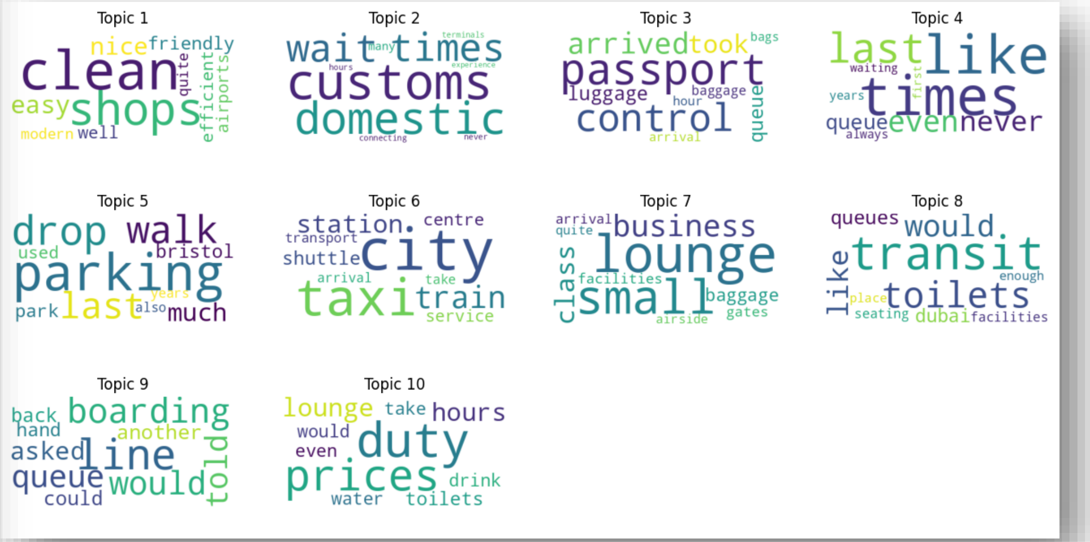


Topic distributions in reviews of travellers from **United Kingdom**

What are most discussed concerns/issues/interests of travellers from United Kingdom?

Topic distribution in document (cont.)





Which group of travellers are most interested in shopping experiences at airports?

Which group of travellers are most interested in duty free?

Comparing to other groups of travellers, what are the issues most concerned by travellers from Thailand?

Choosing Topic Number

- An important issue in topic modeling is to chose the best number of topics **k:**
  - We can use **topic coherence score** to evaluate LDA model
  - Measuring the degree of semantic similarity between high scoring words in the topic
  - A higher score indicates a better topic model
- **“sklearn”** library does not provide function to compute topic coherence. We can use an alternative LDA library, named **gensim**, to construct and evaluate topic models with different topic numbers.

Install Gensim Library

```
#This only needs to run once to install Gensim package
#Make sure that your computer is connected to the Internet
!pip install gensim
```

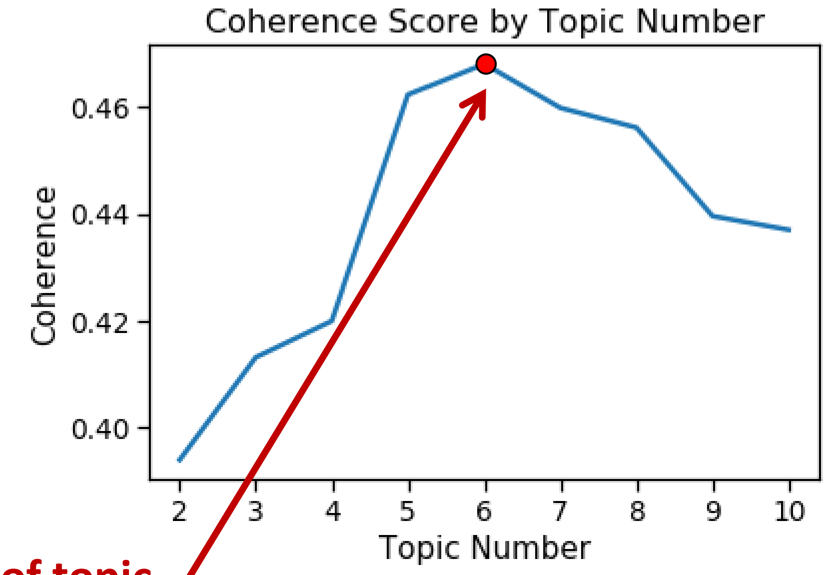
```
Requirement already satisfied: gensim in c:\programdata\anaconda3\lib\site-packages (3.8.
Requirement already satisfied: six>=1.5.0 in c:\programdata\anaconda3\lib\site-packages (
im) (1.12.0)
```


https://www.kdnuggets.com/2018/03/5-things-sentiment-analysis-classification.html

Topic Coherence Score

```
#Evaluation models with topics numbers from 2 to 10
Topics = list(range(2,11,1))
coherence_scores = []
Trained_Models = []
for top in Topics:
    lda_model = gensim.models.ldamodel.LdaModel(corpus=Corpus,
                                                id2word=id2word,
                                                num_topics=top,
                                                random_state=100)
    #Keep the trained models
    Trained_Models.append(lda_model)
    #Compute coherence score for each model
    coherence_model_lda = CoherenceModel(model=lda_model,
                                         texts=Cleaned_doc_new,
                                         dictionary=id2word,
                                         coherence='c_v')
    coherence = coherence_model_lda.get_coherence()
    #Save and print the coherence scores
    coherence_scores.append(coherence)
    print('Topic Number: {0} -- Coherence: {1}'.format(top, coherence))
```

```
Topic Number: 2 -- Coherence: 0.39382943595121905
Topic Number: 3 -- Coherence: 0.41309049439957124
Topic Number: 4 -- Coherence: 0.4199217216431408
Topic Number: 5 -- Coherence: 0.46235027940669704
Topic Number: 6 -- Coherence: 0.4681074959035363
Topic Number: 7 -- Coherence: 0.45985879666094753
Topic Number: 8 -- Coherence: 0.456154378472523
Topic Number: 9 -- Coherence: 0.43952356328025866
Topic Number: 10 -- Coherence: 0.4369607018839415
```



The best number of topic based on the current data set is 6

How about setting Topic Number = 8 ?

In this lecture, we have covered:

- Introduction to the concepts of topic modeling
- Topic modeling and analysis with LDA techniques
- Optimization procedure to chose a suitable topic number.

Summary