In [114…

```
"""
Members :

- EKANE Emile
- LOUOKDOM FOKAM Neal
- SIEWE DAHE William
- NZEKET Aïcha

"""
```

Out[114…

```
'\nMembers : \n\n- Ekane Emile\n- LOUOKDOM FOKAM Neal\n- SIEWE DAHE William\n- NZEKET Aïcha\n\n'
```

In [115…

```
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import word_tokenize, pos_tag
from gensim.models.ldamulticore import LdaMulticore
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
import glob
import gensim.corpora as corpora
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
import random
pyLDAvis.enable_notebook()
%matplotlib inline
```

In [116…

```
df = pd.read_csv('question_responce.csv')
df = df[["responce", "question"]]
```

In [117…

```
def cleaningfunc (column):
    cleaned_data = []
    sentence = []
    data = []

    for line in column:
        cleaned_data.append(re.sub(r'[^\w\s]', ' ',re.sub(r"(?:\@|https?\://)\S+", "", re.sub("#[A-Za-z0-9_]+",

    for line in cleaned_data:
        for word in line.split():
            if word not in stopwords.words("english"):
                sentence.append(word)
        data.append((" ").join(sentence))
        sentence = []
    cleaned_data = data

    return cleaned_data
```

In [118…

```
def tokenizingfunc(data_cleaned):
    data_tokenized = [word_tokenize(line) for line in data_cleaned]
    data_tokenized = [word for word in data_tokenized if word not in stopwords.words("english")]
    return data_tokenized

def tokenizingfuncquestion(text):
    cleaned_data = []
    data = re.sub(r'[^\w\s]', ' ',re.sub(r"(?:\@|https?\://)\S+", "", re.sub("#[A-Za-z0-9_]+","", text)).lower

    for word in data.split():
        if word not in stopwords.words("english"):
                cleaned_data.append(word)

    return cleaned_data
```

In [119…

```
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    return ''

def lemmatizingfunc(data_tokenized):
    lemmatizer = WordNetLemmatizer()

    data_lemmatized = []
    line_lemmatized = []

    for line in data_tokenized:
        for i, word in enumerate(line):
            pos = get_wordnet_pos(pos_tag([word])[0][1])
            if pos != '':
                line_lemmatized.append(lemmatizer.lemmatize(word, pos))
            else:
                line_lemmatized.append(word)
        data_lemmatized.append(line_lemmatized)
        line_lemmatized = []

    return data_lemmatized

def lemmatizingfuncquestion(data_tokenized):
    lemmatizer = WordNetLemmatizer()

    data_lemmatized = []

    for word in data_tokenized:
        pos = get_wordnet_pos(pos_tag([word])[0][1])
        if pos != '':
            data_lemmatized.append(lemmatizer.lemmatize(word, pos))
        else:
            data_lemmatized.append(word)

    return data_lemmatized
```

In [120…

```
def preprocessingquestion(text):
    return lemmatizingfuncquestion(tokenizingfuncquestion(text))
```

In [121…

```
question_cleaned = cleaningfunc(df["question"])
question_tokenized = tokenizingfunc(question_cleaned)
question_lemmatized = lemmatizingfunc(question_tokenized)

response_cleaned = cleaningfunc(df["responce"])
response_tokenized = tokenizingfunc(response_cleaned)
response_lemmatized = lemmatizingfunc(response_tokenized)
```

In [122…

```
def doc2vecfunc():
    documents = glob.glob("corpus/*")
    documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(df['responce'])]
    model = Doc2Vec(vector_size=10, alpha=0.025,min_alpha=0.00025,min_count=10,dm=0, epochs=150)
    model.build_vocab(documents)
    model.train(documents, total_examples=model.corpus_count, epochs=model.epochs)
    return model
```

In [123…

```
def ldafunc(data_lemmatized):
    id2word = corpora.Dictionary(data_lemmatized)

    # Create Corpus
    texts = data_lemmatized

    # Term Document Frequency
    corpus = [id2word.doc2bow(text) for text in texts]

    lda = LdaMulticore(corpus = corpus, id2word=id2word, num_topics=25, passes=100)

    pyLDAvis.enable_notebook()
    vis = gensimvis.prepare(lda, corpus, id2word)

    return lda,corpus,vis
```

In [124…

```
model = doc2vecfunc()
lda,corpus,vis = ldafunc(question_lemmatized)
```

```
/usr/local/lib/python3.9/site-packages/pyLDAvis/_prepare.py:246: FutureWarning: In a future version of pandas a
ll arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  default_term_info = default_term_info.sort_values(
```

In [125…

```
print('Perplexity: ', lda.log_perplexity(corpus))
```

```
Perplexity:  -8.134501835893126
```

In [126…

```
for idx, topic in lda.print_topics(-1):
    print('Topic: {} \nWords: {}'.format(idx, topic))
```

```
Topic: 0
Words: 0.023*"dm" + 0.023*"sent" + 0.014*"get" + 0.014*"back" + 0.014*"flight" + 0.013*"would" + 0.011*"ok" +
0.011*"issue" + 0.009*"late" + 0.008*"like"
Topic: 1
Words: 0.019*"flight" + 0.014*"take" + 0.014*"seat" + 0.012*"1" + 0.009*"time" + 0.009*"people" + 0.008*"one" +
0.007*"make" + 0.007*"plane" + 0.007*"thank"
Topic: 2
Words: 0.035*"flight" + 0.015*"get" + 0.011*"seat" + 0.010*"know" + 0.009*"tomorrow" + 0.009*"amp" + 0.009*"ear
ly" + 0.008*"update" + 0.008*"customer" + 0.008*"delay"
Topic: 3
Words: 0.021*"seat" + 0.016*"flight" + 0.010*"get" + 0.010*"card" + 0.009*"thanks" + 0.009*"exit" + 0.008*"row"
+ 0.008*"gate" + 0.006*"way" + 0.006*"home"
Topic: 4
Words: 0.020*"day" + 0.018*"one" + 0.016*"ticket" + 0.012*"way" + 0.011*"get" + 0.010*"fly" + 0.009*"thank" +
0.009*"try" + 0.009*"airline" + 0.009*"need"
Topic: 5
Words: 0.043*"flight" + 0.036*"hour" + 0.027*"miss" + 0.018*"make" + 0.015*"2" + 0.015*"bad" + 0.014*"delayed"
+ 0.014*"delay" + 0.013*"3" + 0.011*"get"
Topic: 6
Words: 0.022*"flight" + 0.019*"get" + 0.018*"time" + 0.009*"aa" + 0.009*"ticket" + 0.009*"fit" + 0.008*"one" +
0.008*"could" + 0.008*"make" + 0.008*"found"
Topic: 7
Words: 0.030*"help" + 0.028*"thank" + 0.023*"flight" + 0.021*"great" + 0.018*"please" + 0.018*"club" + 0.013*"a
dmiral" + 0.012*"dm" + 0.011*"day" + 0.010*"another"
Topic: 8
Words: 0.031*"today" + 0.022*"bag" + 0.020*"get" + 0.018*"pay" + 0.018*"make" + 0.012*"right" + 0.011*"told" +
0.011*"thanks" + 0.010*"aa" + 0.010*"go"
Topic: 9
Words: 0.036*"flight" + 0.019*"book" + 0.018*"hey" + 0.018*"know" + 0.016*"get" + 0.014*"holiday" + 0.011*"fly"
+ 0.011*"travel" + 0.010*"cancel" + 0.010*"airline"
Topic: 10
Words: 0.037*"flight" + 0.026*"service" + 0.024*"customer" + 0.020*"get" + 0.011*"time" + 0.009*"thanks" + 0.00
8*"still" + 0.008*"really" + 0.008*"wait" + 0.008*"attendant"
Topic: 11
Words: 0.012*"lounge" + 0.011*"flight" + 0.010*"take" + 0.008*"yes" + 0.008*"class" + 0.008*"new" + 0.008*"than
ks" + 0.008*"business" + 0.008*"great" + 0.008*"today"
Topic: 12
Words: 0.037*"bag" + 0.027*"check" + 0.027*"carry" + 0.019*"overhead" + 0.017*"thanks" + 0.016*"charge" + 0.015
*"get" + 0.015*"try" + 0.014*"bin" + 0.014*"seat"
Topic: 13
Words: 0.033*"flight" + 0.015*"even" + 0.014*"fare" + 0.011*"fly" + 0.011*"difference" + 0.011*"book" + 0.010
*"refund" + 0.010*"aa" + 0.008*"economy" + 0.008*"great"
Topic: 14
Words: 0.062*"flight" + 0.031*"gate" + 0.026*"get" + 0.014*"plane" + 0.012*"bag" + 0.010*"us" + 0.010*"say" +
0.009*"agent" + 0.008*"aa" + 0.008*"wait"
Topic: 15
Words: 0.015*"flight" + 0.014*"bag" + 0.011*"check" + 0.010*"time" + 0.009*"day" + 0.009*"us" + 0.008*"agent" +
0.008*"gt" + 0.007*"trip" + 0.007*"yet"
Topic: 16
Words: 0.035*"flight" + 0.012*"seat" + 0.012*"customer" + 0.011*"service" + 0.010*"fly" + 0.010*"go" + 0.010*"h
our" + 0.009*"make" + 0.008*"get" + 0.008*"2"
Topic: 17
Words: 0.076*"flight" + 0.049*"cancel" + 0.013*"need" + 0.013*"refund" + 0.012*"check" + 0.011*"make" + 0.010
*"ticket" + 0.009*"seat" + 0.008*"say" + 0.008*"december"
Topic: 18
Words: 0.022*"fly" + 0.012*"upgrade" + 0.011*"never" + 0.009*"go" + 0.009*"gate" + 0.008*"take" + 0.007*"agent"
+ 0.007*"customer" + 0.007*"flight" + 0.007*"w"
Topic: 19
Words: 0.031*"bag" + 0.029*"checked" + 0.017*"baggage" + 0.016*"thanks" + 0.015*"flight" + 0.013*"pay" + 0.012
*"get" + 0.011*"first" + 0.011*"happy" + 0.011*"fee"
Topic: 20
Words: 0.021*"flight" + 0.013*"airline" + 0.011*"thanks" + 0.011*"make" + 0.010*"response" + 0.009*"wait" + 0.0
08*"much" + 0.008*"fly" + 0.008*"system" + 0.008*"aa"
Topic: 21
Words: 0.036*"flight" + 0.032*"hour" + 0.031*"get" + 0.031*"plane" + 0.023*"delayed" + 0.019*"gate" + 0.017*"ho
me" + 0.016*"sit" + 0.013*"great" + 0.013*"2"
Topic: 22
Words: 0.010*"charge" + 0.010*"make" + 0.008*"much" + 0.008*"travel" + 0.007*"class" + 0.007*"even" + 0.006*"pa
y" + 0.006*"try" + 0.006*"platinum" + 0.006*"ticket"
Topic: 23
Words: 0.019*"get" + 0.019*"flight" + 0.010*"nothing" + 0.010*"time" + 0.010*"service" + 0.009*"call" + 0.008
*"need" + 0.008*"go" + 0.008*"customer" + 0.008*"aa"
Topic: 24
Words: 0.034*"flight" + 0.015*"jfk" + 0.014*"delay" + 0.012*"nice" + 0.010*"seat" + 0.010*"work" + 0.010*"get"
+ 0.010*"really" + 0.009*"thx" + 0.009*"thanks"
```

In [127…

```
ourData = {"intents":[
    {
        "tag" : "Thanks",
        "patterns" : ["thanks", "thanksgiving", "thank", "happy", "great", "beautiful", "nice"],
        "responses" : ["I am happy to hear that", "We will continue to work hard to maintain your satifaction"]
    },
    {
        "tag" : "Delay",
        "patterns" : ["delay", "late", "wait"],
        "responses" : ["We are really sorry to hear that", "We sincerely apologize about this delay"]
    },
    {
        "tag" : "Cancel",
        "patterns" : ["cancel"],
        "responses" : ["We're sorry about that inconvenient situation. Rebook another flight"]
    },
    {
        "tag" : "Guidance",
        "patterns" : ["guide"],
        "responses" : ["We'll see what we could do", "We'll send you someone as soon as possible"]
    },
    {
        "tag" : "Goodbye",
        "patterns" : ["bye", "goodbye"],
        "responses" : ["You're welcome, goodbye and have a nice stay"]
    },
    {
        "tag" : "Please",
        "patterns" : ["please"],
        "responses" : ["I will do the best I can"]
    },
    {
        "tag" : "Disappointment",
        "patterns" : ["disappoint", "upset"],
        "responses" : ["We are really sorry to hear that, we shall do better next time"]
    },
    {
        "tag" : "Refund",
        "patterns" : ["refund"],
        "responses" : ["We are really sorry, our team will study your request and then, give you a feedback"]
    },
    {
        "tag" : "Help",
        "patterns" : ["help"],
        "responses" : ["I'll do my best for that", "We'll send you someone asap"]
    }
]}
```

In [128…

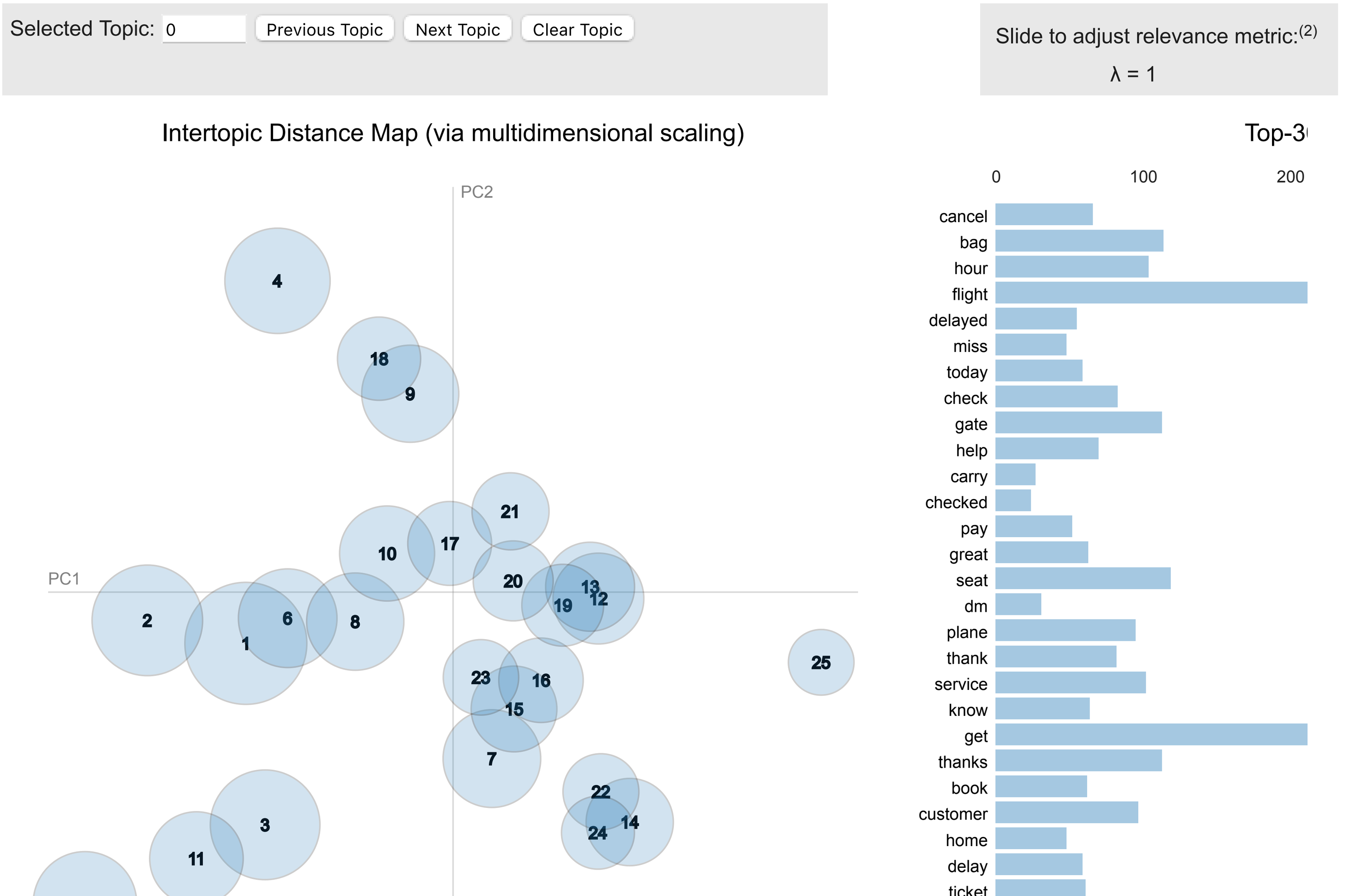
```
vis
```

Out[128…

Selected Topic: 0 | Previous Topic | Next Topic | Clear Topic

Slide to adjust relevance metric:(2)

λ = 1



In [129…

```
for index, score in sorted(lda[corpus[65]], key=lambda tup: -1*tup[1]):
    print("\nScore: {}\t \nTopic: {}".format(score, lda.print_topic(index, 10)))
```

```
Score: 0.9039932489395142
Topic: 0.037*"bag" + 0.027*"check" + 0.027*"carry" + 0.019*"overhead" + 0.017*"thanks" + 0.016*"charge" + 0.015
*"get" + 0.015*"try" + 0.014*"bin" + 0.014*"seat"
```

In [130…

```
topics = []
for i in range(len(df["responce"])):
    topics.append(sorted(lda[corpus[i]], key = lambda x : -1*x[1])[0][0])

df['topics'] = topics
```

In [131…

```
id = 100
question = lemmatizingfuncquestion(tokenizingfuncquestion(df["question"].loc[id]))
```

In [132…

```
id2word = corpora.Dictionary([question])
corpus = [id2word.doc2bow(question)]
print(lda[corpus])
```

```
<gensim.interfaces.TransformedCorpus object at 0x139551a30>
```

In [133…

```
model.dv.most_similar(model.infer_vector(question))
```

Out[133…

```
[(117, 0.893161952495575),
 (975, 0.8799808025360107),
 (1380, 0.8645240664482117),
 (736, 0.8626986742019653),
 (1436, 0.8451420664787292),
 (1313, 0.8388538956642151),
 (1814, 0.828161895275116),
 (1789, 0.8265056610107422),
 (829, 0.8218382000923157),
 (630, 0.8138636946678162)]
```

In [134…

```
print("question : " + df["question"].loc[id] + "\n")
print("response : " + df["responce"].loc[id])
```

```
question : @AmericanAir Great thank you. I had a GREAT flight.

response : @119479 We love hearing this and look forward to welcoming you on board again soon.
```

In [135…

```
df["responce"].loc[model.dv.most_similar(model.infer_vector(question))[0][0]]
```

Out[135…

```
"@120197 Thanks for the info David. We'll pass this on to our team so she gets the recognition she deserves."
```

In [139…

```
def getRes(text, fJson):
  listOfIntents = fJson["intents"]
  for w in text:
    for i in listOfIntents:
      for el in i["patterns"]:
        if el == w:
          ourResult = random.choice(i["responses"])
          return ourResult

getRes(question, ourData)
```

```
We will continue to work hard to maintain your satifaction
```

Out[139…

```
'We will continue to work hard to maintain your satifaction'
```