Member-only story

Hands-On Topic Modeling with Python

A tutorial on topic modeling using Latent Dirichlet Allocation (LDA) and visualization with pyLDAvis

Idil Ismiguzel · Follow

Published in Towards Data Science
11 min read · Dec 14, 2022

Listen Share

Photo by Bradley Singleton on Unsplash

Topic modeling is a popular technique in Natural Language Processing (NLP) and text mining to extract topics of a given text. Utilizing topic modeling we can scan large volumes of unstructured text to detect keywords, topics, and themes.

Topic modeling is an unsupervised machine learning technique and does not need labeled data for model training. It should not be confused with *topic classification* which is a supervised machine learning technique and needs labeled data for training to fit and learn. In some cases, topic modeling can be used together with topic classification, where we perform topic modeling first to detect topics in a given text and label each record with its corresponding topic. Then this labeled data is used for training a classifier and performing topic classification on unseen data.

In this article, we will focus on topic modeling and cover how to prepare data with text preprocessing, assign the best number of topics with coherence score, extract topics using Latent Dirichlet Allocation (LDA), and visualize topics using pyLDAvis.

While following the article, I encourage you to check out the Jupyter Notebook on my GitHub for full analysis and code.

We have lots of things to cover, let's get started! 🤓

1. Data

We will use the Disneyland Reviews data set which can be downloaded from Kaggle. It has 42,000 reviews and ratings for the Disneyland branches in Paris, California, and Hong Kong. The rating column includes rating scores and can be used for topic classification to classify unseen reviews as positive, negative, or neutral. This is out of the scope of this article but, if you are interested in topic classification you can check the article below.

Applying Text Classification using Logistic Regression: A comparison between BoW and Tf-Idf

Creating “language-aware data products” are becoming more and more important for businesses and organizations…

medium.com

Let’s read the data and have a look at the first few rows.

```
# Read the data
reviews = pd.read_csv('/content/DisneylandReviews.csv', encoding='latin-1')
```


	Review_ID	Rating	Year_Month	Reviewer_Location	Review_Text	Branch
0	670772142	4	2019-4	Australia	If you've ever been to Disneyland anywhere you'll find Disneyland Hong Kong ...	Disneyland_HongKong
1	670682799	4	2019-5	Philippines	Its been a while since d last time we visit HK Disneyland .. Yet, this time ...	Disneyland_HongKong
2	670623270	4	2019-4	United Arab Emirates	Thanks God it wasn t too hot or too humid when I was visiting the park o...	Disneyland_HongKong
3	670607911	4	2019-4	Australia	HK Disneyland is a great compact park. Unfortunately there is quite a bit of...	Disneyland_HongKong
4	670607296	4	2019-4	United Kingdom	the location is not in the city, took around 1 hour from Kowlon, my kids lik...	Disneyland_HongKong

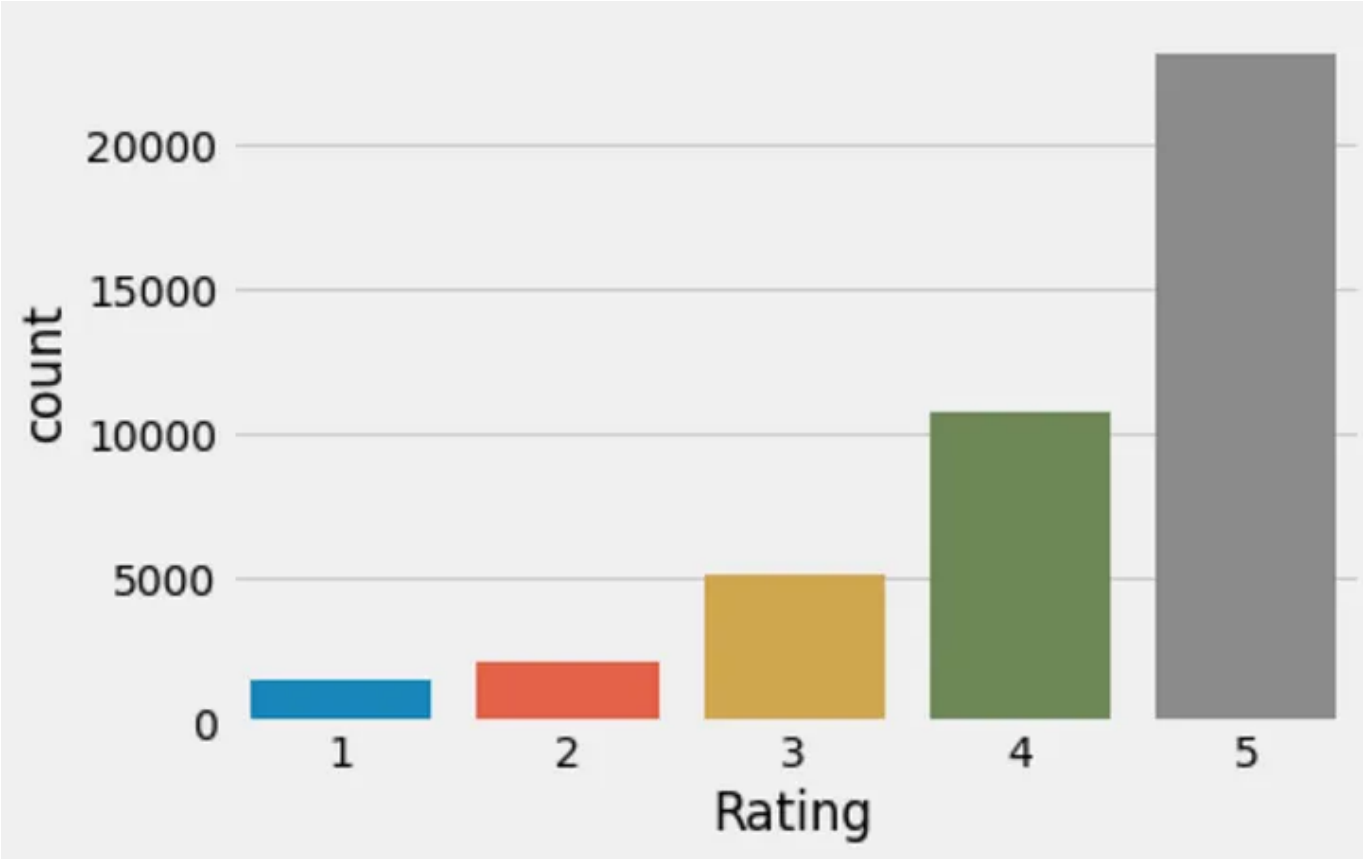
The First 5 rows of data set

Let’s filter only the “Review” and “Rating” columns.

```
# Filter only related columns and drop duplicated reviews
reviews = reviews[["Review_Text", "Rating"]]
reviews = reviews.drop_duplicates(subset='Review_Text')
```

Let’s print a value counts bar plot using `countplot` from seaborn to learn the overall emotion of the reviews.

```
# Create a bar plot with value counts
sns.countplot(x='Rating', data=reviews)
```



The majority are positive but there are some negative ratings

2. Data Cleaning and Preprocessing

Before starting topic modeling, we need to prepare the text and perform cleaning and preprocessing. This is a crucial step in all text mining pipelines and the end model's performance highly depends on it. The steps we will follow for this dataset are:

1. **Lowercase** each word
2. **Replace contractions** with their longer forms
3. **Remove special characters and unwanted words**
4. **Tokenize** each word by using `nltk.WordPunctTokenizer()` we will extract tokens from strings of words or sentences.
5. **Lemmatize** each word by using `nltk.stem.WordNetLemmatizer()` we will restore words to their dictionary forms so all words with similar meanings will be linked to one word.

To apply all the listed steps, I will use the following functions. However, to increase modularity and easy debugging you can define each task in a separate function.

```
def text_preprocessing(text):

    # Convert words to lower case
    text = text.lower()

    # Expand contractions
    if True:
        text = text.split()
        new_text = []
        for word in text:
            if word in contractions:
                new_text.append(contractions[word])
            else:
                new_text.append(word)
        text = " ".join(new_text)

    # Format words and remove unwanted characters
    text = re.sub(r'https?:\/\/.*[\r\n]*', '', text, flags=re.MULTILINE)
    text = re.sub(r'\<a href', ' ', text)
    text = re.sub(r'&amp;', '', text)
    text = re.sub(r'[_"\-;%()|+&=*%.,!?:#$@\[\]/]', ' ', text)
    text = re.sub(r'<br />', ' ', text)
    text = re.sub(r'\'', ' ', text)

    # Tokenize each word
    text = nltk.WordPunctTokenizer().tokenize(text)

    # Lemmatize each word
    text = [nltk.stem.WordNetLemmatizer().lemmatize(token, pos='v') for token in

return text
```

```
def to_string(text):
    # Convert list to string
    text = ' '.join(map(str, text))

    return text

# Create a list of review by applying text_preprocessing function
reviews['Review_Clean_List'] = list(map(text_preprocessing, reviews.Review_Text)
```

```
# Return to string with to_string function
reviews['Review_Clean'] = list(map(to_string, reviews['Review_Clean_List']))
```

Let's have a look at new columns by printing a random row.

	Review_Text	Rating	Review_Clean_List	Review_Clean
9375	Meeting Disney Character is like a dream come true! Me went from 12 pm to the closing, it has so many things to offer, and I won't ever forget the beautiful fireworks.Of course it's not really an adventure for adrenaline, but it sure is fun. Really really fun!I'm looking forward to come back.	5	[meet, character, like, dream, come, true, go, 12, pm, close, many, things, offer, ever, forget, beautiful, fireworks, course, really, adventure, adrenaline, sure, fun, really, really, fun, look, forward, come, back]	meet character like dream come true go 12 pm close many things offer ever forget beautiful fireworks course really adventure adrenaline sure fun really really fun look forward come back

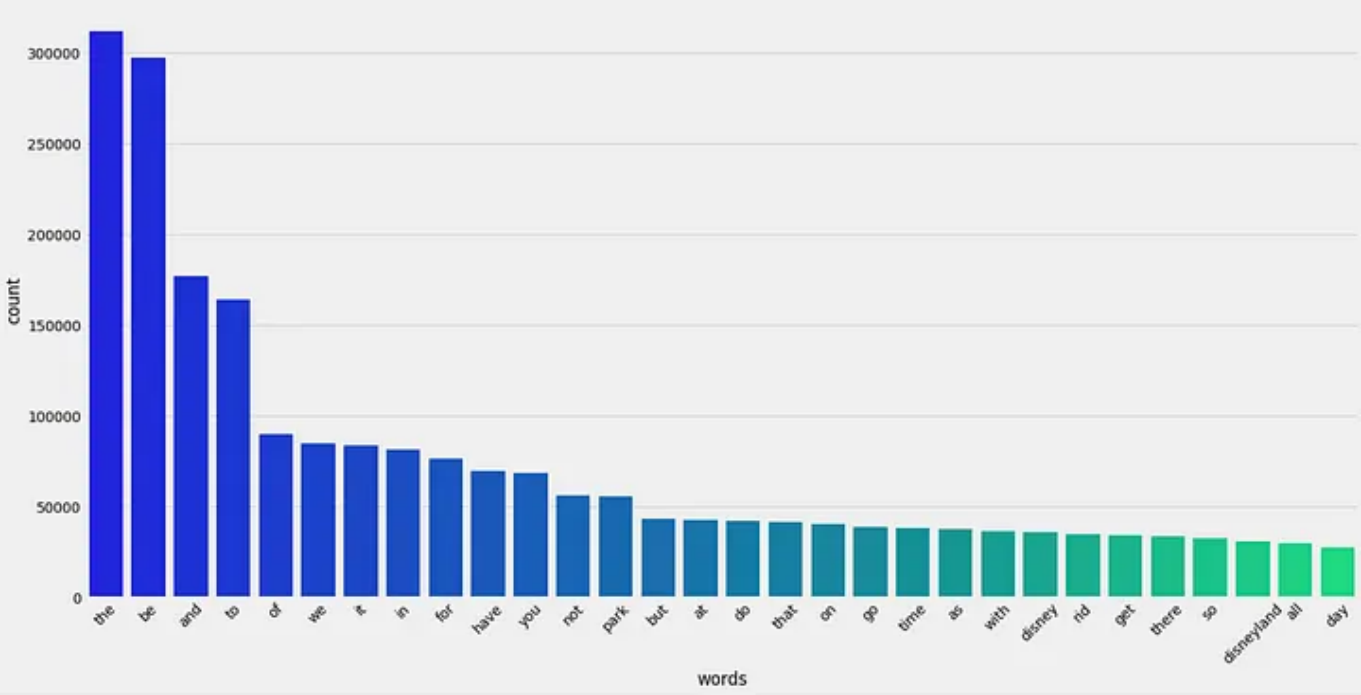
Last but not least, we need to remove stopwords before moving to the next step. Stopwords are language-specific common words (i.e. "the", "a", "and "an" in English) that neither add value nor improve the interpretation of the review and tend to introduce bias in modeling. We will load the English stopwords list from `nltk` library and drop those words from our corpus.

Since we are removing stopwords, we may want to check the most frequent words in our corpus and evaluate if we want to remove some of them too. Some of these words might just repeat very often and do not add any value to the meaning.

We will use `Counter` from collections library to count words.

```
# Import Counter
from collections import Counter

# Join all word corpus
review_words = ','.join(list(reviews['Review_Clean'].values))

# Count and find the 30 most frequent
Counter = Counter(review_words.split())
most_frequent = Counter.most_common(30)

# Bar plot of frequent words
fig = plt.figure(1, figsize = (20,10))
_ = pd.DataFrame(most_frequent, columns=("words","count"))
```

```
    sns.barplot(x = 'words', y = 'count', data = _, palette = 'winter')
    plt.xticks(rotation=45);
```

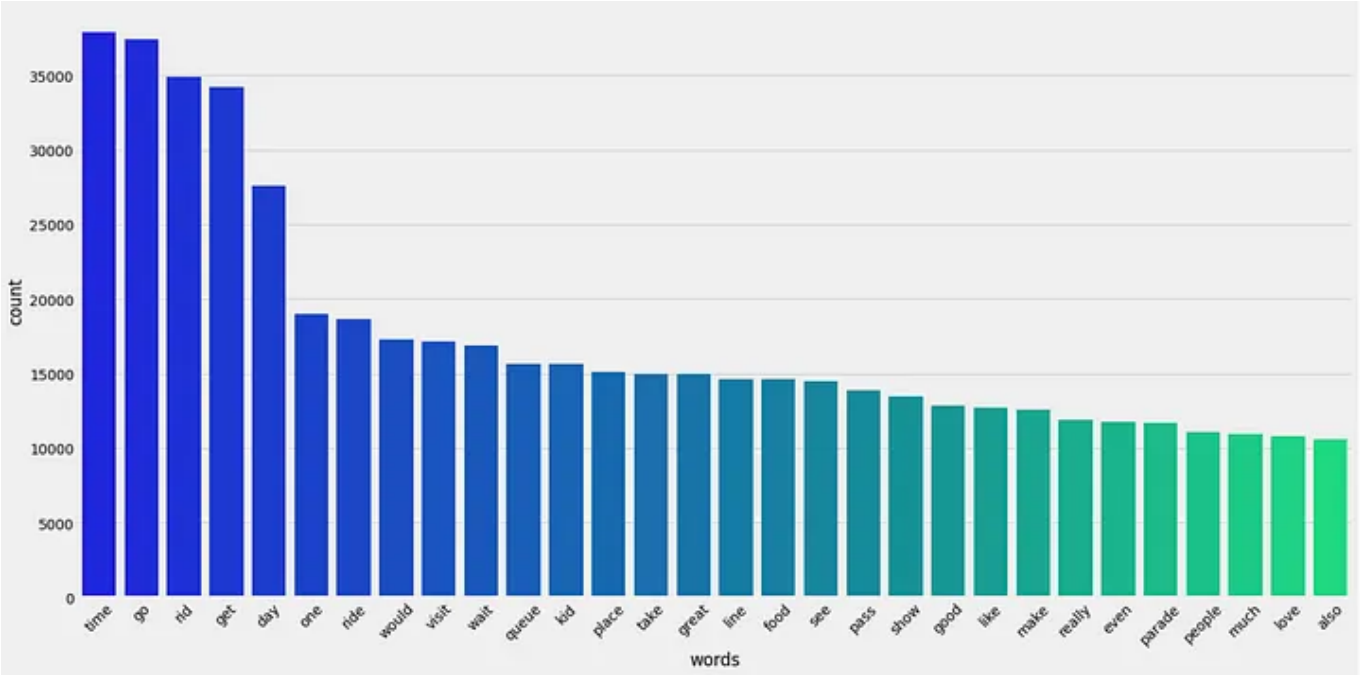


30 most frequent words (before removing stopwords)

As expected there are frequent words in the top 30, related to Disney and park content such as “park”, “disney” and “disneyland”. We will remove these words by adding them to the stopwords list. You can also create a separate list.

```
# Load the list of stopwords
nltk.download('stopwords')

stopwords_list = stopwords.words('english')
stopwords_list.extend(['park', 'disney', 'disneyland'])

reviews['Review_Clean_List'] = [[word for word in line if word not in stopwords_
reviews['Review_Clean'] = list(map(text_as_string, reviews['Review_Clean_List']))

# Join all word corpus
review_words = ','.join(list(reviews['Review_Clean'].values))

# Count and find the 30 most frequent
Counter = Counter(review_words.split())
most_frequent = Counter.most_common(30)
```

```
# Bar plot of frequent words
fig = plt.figure(1, figsize = (20,10))
_ = pd.DataFrame(most_frequent, columns=("words","count"))
sns.barplot(x = 'words', y = 'count', data = _, palette = 'winter')
plt.xticks(rotation=45);
```



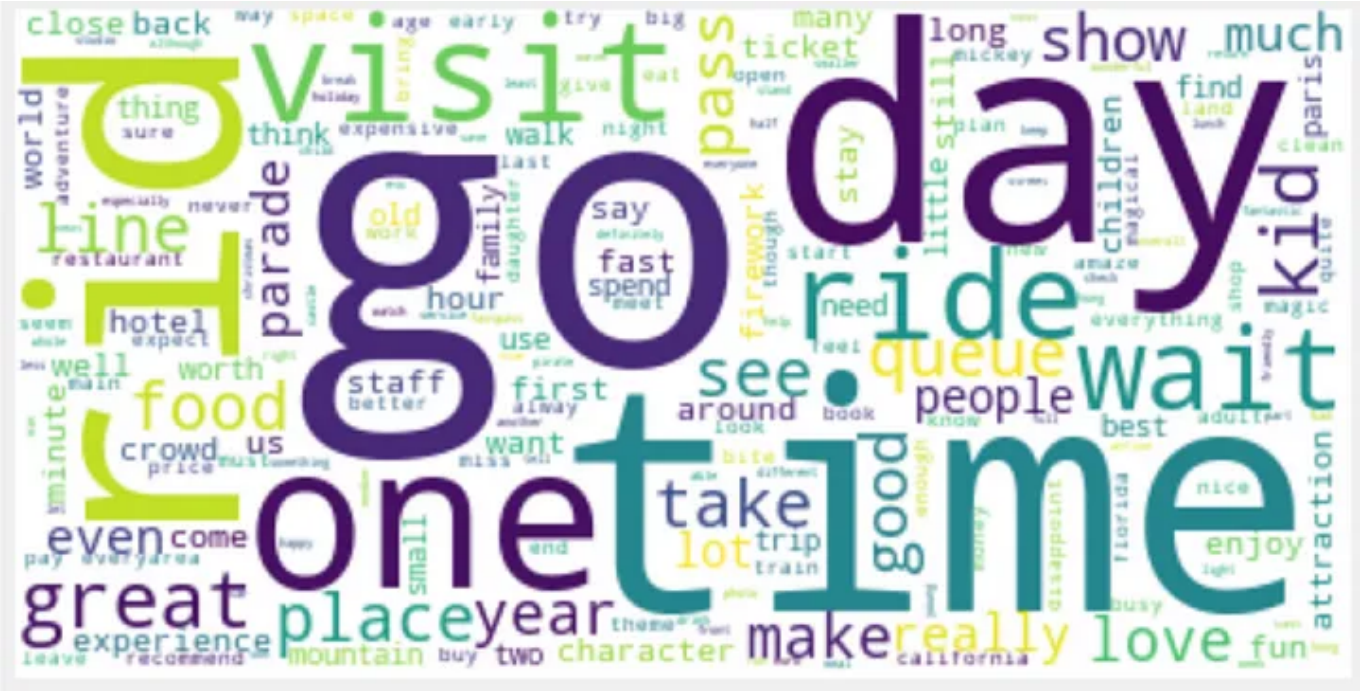
30 most frequent words (after removing stopwords and some frequent words)

Bonus

Let's create a word cloud of the preprocessed text corpus using `review_words` created previously. ☁️ ☁️ ☁️

```
# Generate the word cloud
wordcloud = WordCloud(background_color="white",
                      max_words= 200,
                      contour_width = 8,
                      contour_color = "steelblue",
                      collocations=False).generate(review_words)

# Visualize the word cloud
fig = plt.figure(1, figsize = (10, 10))
plt.axis('off')
```

```
plt.imshow(wordcloud)
plt.show()
```



Wordcloud after text preprocessing

3. Bag-of-Words

In order to use text as an input to machine learning algorithms, we need to present it in a numerical format. Bag-of-words is a vector space model and represents the occurrence of words in the document. In other words, bag-of-words converts each review into a collection of word counts without giving importance to the order or meaning.

We will first create our dictionary using `corpora.Dictionary` of Gensim and then use `dictionary.doc2bow` to create bag-of words.

```
# Create Dictionary
id2word = gensim.corpora.Dictionary(reviews['Review_Clean_List'])

# Create Corpus: Term Document Frequency
corpus = [id2word.doc2bow(text) for text in reviews['Review_Clean_List']]
```

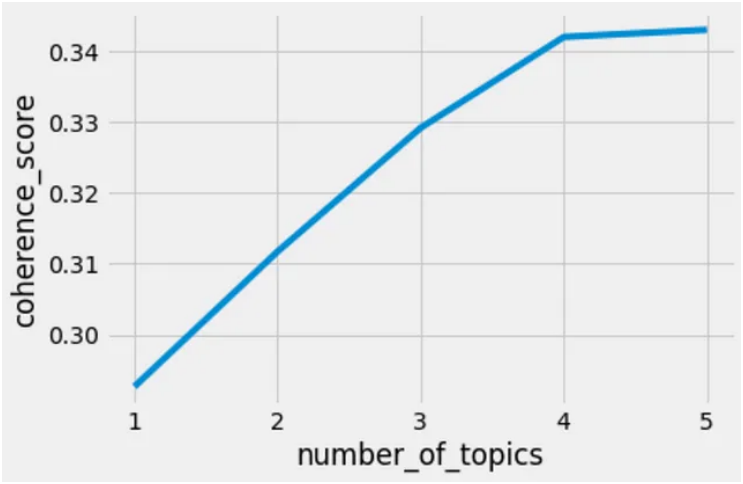
By creating the dictionary we map each word with an integer id (aka id2word) and then we call the doc2bow function on each dictionary to create a list of (id, frequency) tuples.

4. Determining the Number of Topics

Deciding on the number of topics for the topic modeling can be difficult. Since we have initial knowledge of the context, determining the number of topics for modeling wouldn't be too outraging. However, if this number is too much then the model might fail to detect a topic that is actually broader and if this number is too less then topics might have large overlapping words. Because of these reasons, we will use the topic coherence score.

```
from gensim.models import CoherenceModel

# Compute coherence score
number_of_topics = []
coherence_score = []
for i in range(1,10):
  lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                              id2word=id2word,
                                              iterations=50,
                                              num_topics=i)
  coherence_model_lda = CoherenceModel(model=lda_model,
                                       texts=reviews['Review_Clean_List'],
                                       dictionary=id2word,
                                       coherence='c_v')
  coherence_lda = coherence_model_lda.get_coherence()
  number_of_topics.append(i)
  coherence_score.append(coherence_lda)

# Create a dataframe of coherence score by number of topics
topic_coherence = pd.DataFrame({'number_of_topics':number_of_topics,
                                'coherence_score':coherence_score})

# Print a line plot
sns.lineplot(data=topic_coherence, x='number_of_topics', y='coherence_score')
```

number_of_topics	coherence_score
1	0.292660
2	0.311713
3	0.329169
4	0.341985
5	0.342998



Coherence score by the number of topics

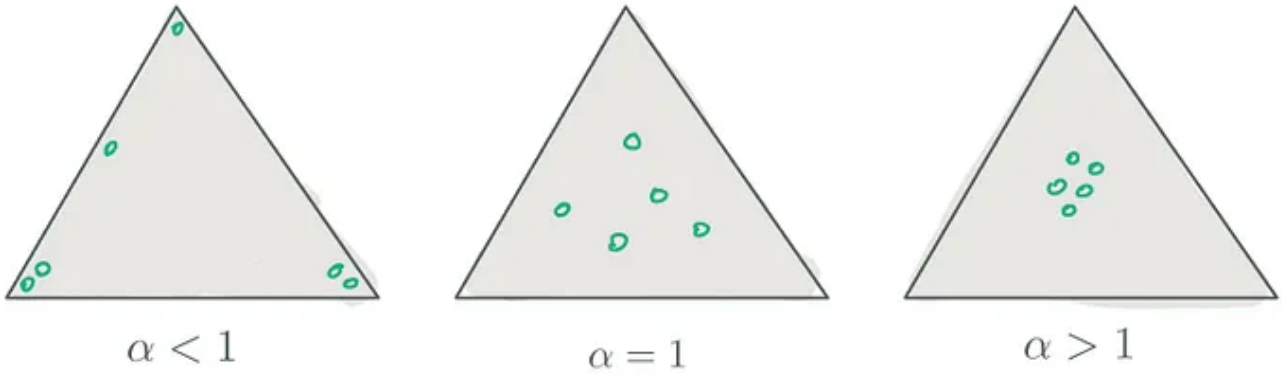
Since a very high coherence score (0.3429) is achieved with four topics, and there is no big jump from four to five topics, we will construct our LDA model with four topics. However, it is important to note that we defined the coherence hyperparameter as `coherence='c_v'` but there are other options as well such as *'u_mass', 'c_uci', 'c_npmi'*, and it would be the best practice to validate them. (Check Gensim's doc for detailed information.)

5. Topic Modeling with LDA

Latent Dirichlet Allocation is a popular statistical unsupervised machine learning model for topic modeling. It assumes each topic is made up of words and each document (in our case each review) consists of a collection of these words. Therefore, LDA tries to find words that best describe each topic and matches reviews that are represented by these words.

LDA uses Dirichlet distribution, a generalization of Beta distribution that models probability distribution for two or more outcomes (K). For example, K = 2 is a special case of Dirichlet distribution for beta distribution.

Dirichlet distribution denoted with Dir(α) where $\alpha < 1$ (symmetric) indicates sparsity, and it is exactly how we want to present topics and words for topic modeling. As you can see below, with $\alpha < 1$ we have circles on sides/corners separated from each other (in other words sparse), and with $\alpha > 1$ we have circles in the center very close to each other and difficult to distinguish. You can imagine these circles as topics.



LDA uses two Dirichlet distributions where

- K is the number of topics
- *M* denotes the number of documents
- *N* denotes the number of words in a given document
- Dir(alpha) is the Dirichlet distribution per-document topic distribution
- Dir(beta) is the Dirichlet distribution per-topic word distribution

It then uses multinomial distributions for each word position

- to choose a topic for the *j*-th word in document i; z_{i,j}
- to choose a word for the specific word; w_{i,j}

$$z_{i,j} \sim Multinomial\left(\theta_i\right)$$

$$\omega_{i,j} \sim Multinomial\left(\varphi_{z_{i,j}}\right)$$

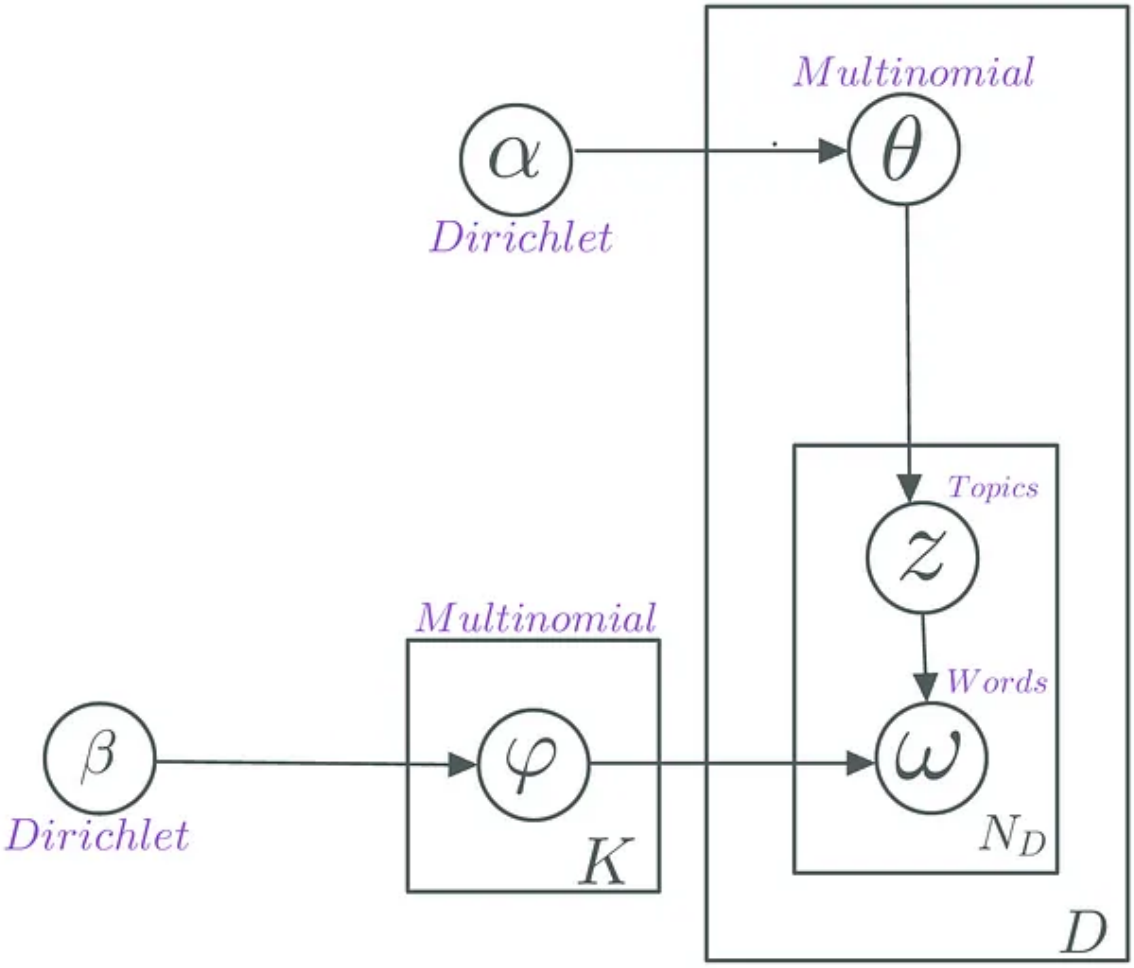


Plate notation of LDA from literature

If we bring all the pieces together, we get the formula below, which describes the probability of a document with two Dirichlet distributions followed by multinomial distributions.

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$

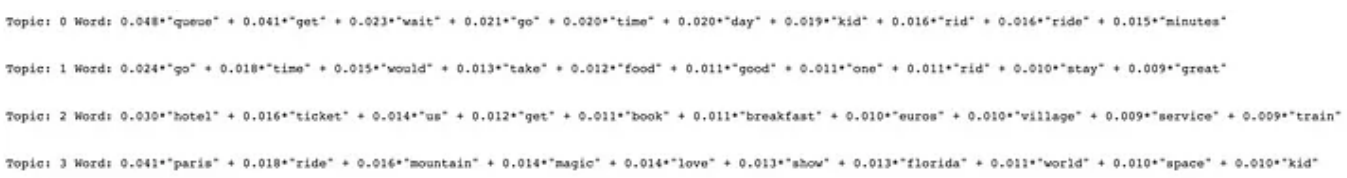
Probability of a document

Enough theory! 🤓 Let's see how to perform the LDA model in Python using `ldaModel` from Gensim.

```
# Define the number of topics
n_topics = 4

# Run the LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=n_topics,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=10,
                                            passes=10,
                                            alpha='symmetric',
                                            iterations=100,
                                            per_word_topics=True)
```

Let's explore words occurring in each topic with their relative weight.

```
for idx, topic in lda_model.print_topics(-1):
    print("Topic: {} Word: {}".format(idx, topic))
```



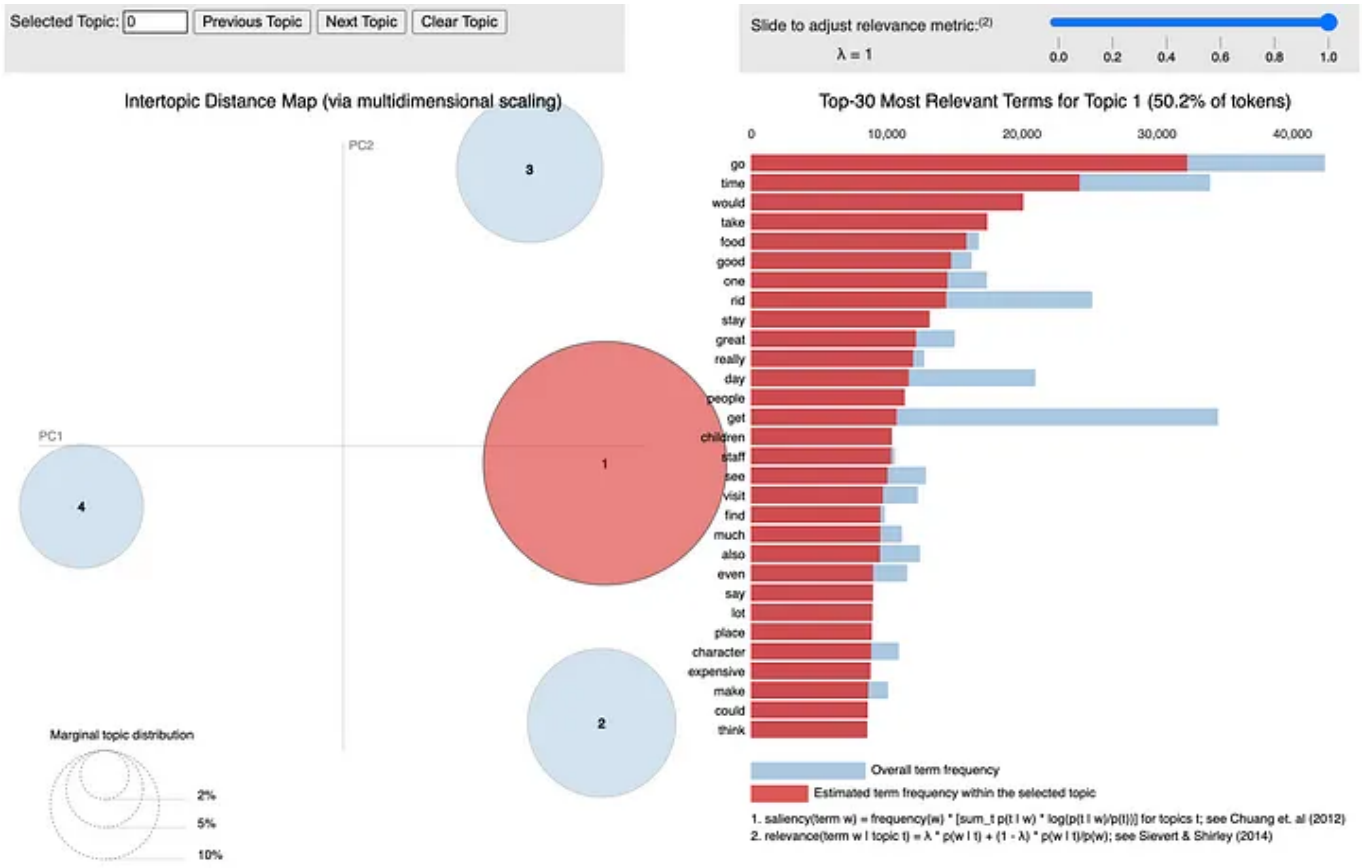
Words occurrence and their relative weight in each topic

We can see that one topic is related to queuing and waiting; the next one is related to visiting, staying, and food; another one is related to hotels, tickets, and villages; and the last one is related to magic, love, and shows highlighting Paris and Florida.

6. Visualising with pyLDAvis

pyLDAvis is an interactive web-based visualization tool that is used to visualize topic models. You can easily install in Python using `pip install pyldavis` and enable running the visualization on python notebook with `enable_notebook()`.

```
# Import and enable notebook to run visualization
import pyLDAvis.gensim_models
pyLDAvis.enable_notebook()

vis = pyLDAvis.gensim_models.prepare(lda_model,
                                     corpus,
                                     dictionary=lda_model.id2word)
vis
```



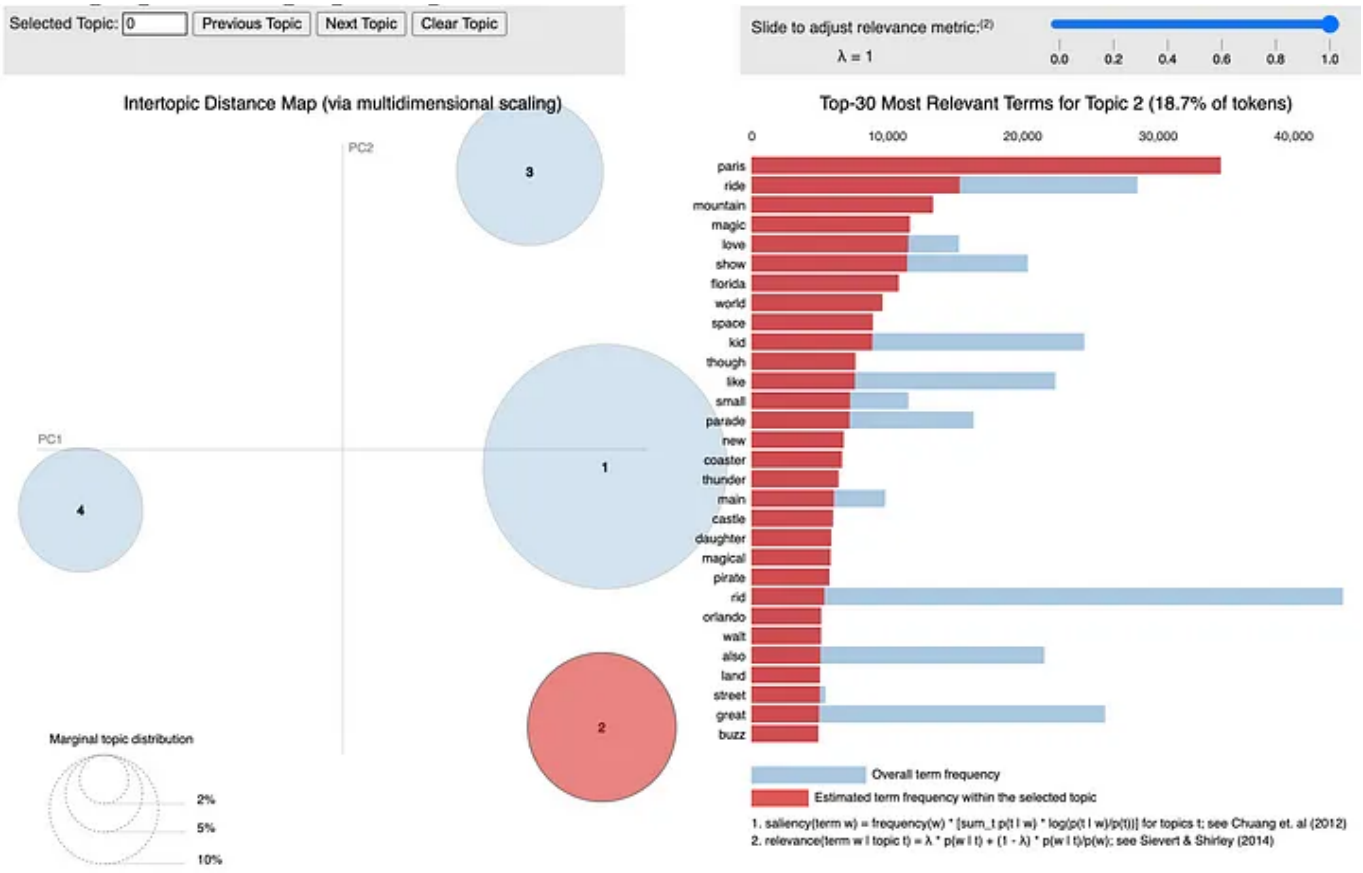
pyLDAvis representation of Topic 1 (with λ = 1)

On the left, we can see each topic represented as a bubble on the intertopic distance map (multidimensional scaling onto the x and y-axis) and if we click on a topic, visualization automatically adjusts to that specific topic. The distance between bubbles represents the semantic distance between topics, and in case bubbles are overlapping that means there are a lot of common words. In our case topics are well separated and do not overlap. In addition, the area of the topic bubbles represents

coverage of each topic, and topic 1 covers around 50% of reviews while the rest of the topics share nearly equal amounts.

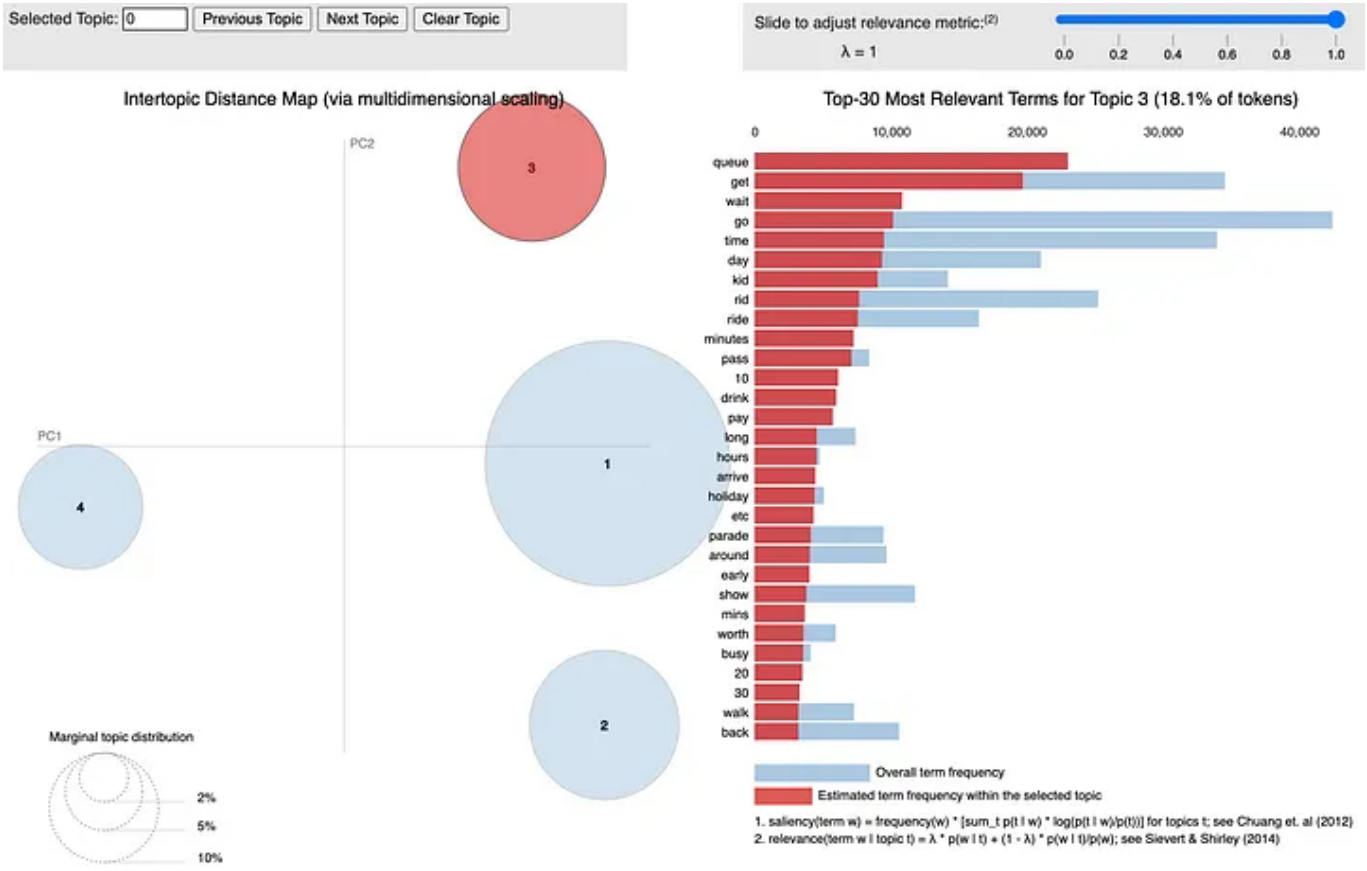
The visualization on the right side shows the top 30 most relevant words per topic the blue shaded bar represents the occurrence of the word in all reviews and the red bar represents the occurrence of the word within the selected topic. On top of it, you can see a slide to adjust the relevance metric λ (where $0 \leq \lambda \leq 1$) and $\lambda = 1$ tunes the visualization for the words most likely to occur in each topic, and $\lambda = 0$ tunes for the words only specific for the selected topic.

Let's check Topic 2 👀

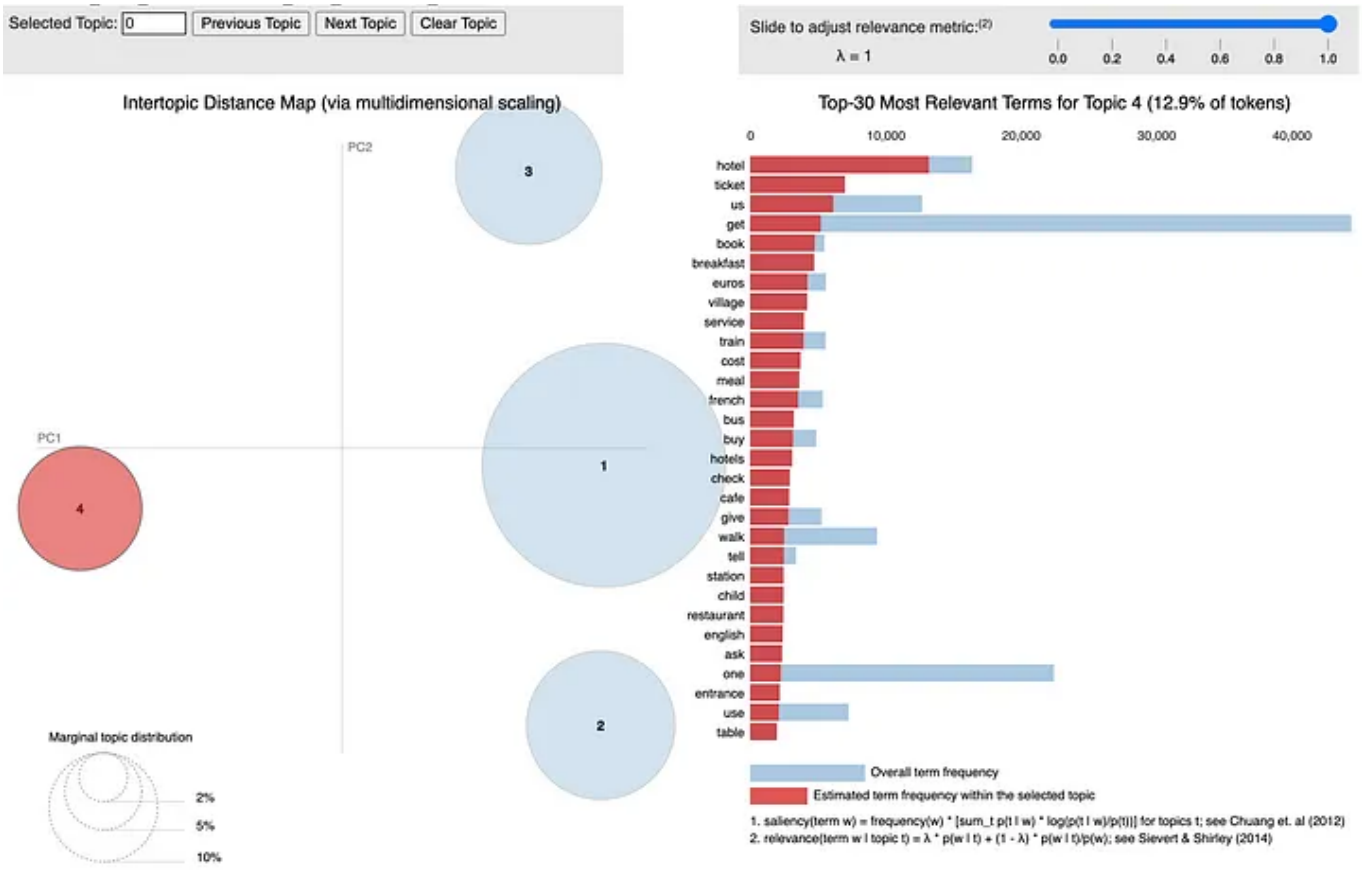


pyLDAvis representation of Topic 2 (with λ = 1)

Topic 3 👀



and lastly Topic 4 👀



Conclusion

In this article, we explored how to detect the themes and keywords from text data in order to understand the content without the need of scanning the entire text. We covered how to apply preprocessing including cleaning the text, lemmatization, and removing stopwords & most common words to prepare the data for machine learning. We also created a word cloud, which helped us to visualize the overall content. To find topics of the Disneyland Reviews data set we used Latent Dirichlet Allocation (LDA), a probabilistic method for topic modeling assuming *topics* can be represented as distributions over words in the text corpus. And each document (in our case review) can exhibit more than one topic with a difference in proportions. The topic with the highest proportion is selected as the *topic* of that document. We defined the number of topics by using the coherence score and finally visualized our topics and keywords using pyLDAvis.

LDA is a relatively simple technique for topic modeling and thanks to pyLDAvis, you can show the results to others who are not familiar with the technical scope. The visualization also helps to describe the functioning principle and makes topic models more interpretable and explainable.

While we only covered the LDA technique, there are many other techniques available for topic modeling. To name a few, Latent Semantic Analysis (LSA), Non-Negative Matrix Factorization, Word2vec. If you are interested in the topic, I strongly recommend exploring these methods too, they all have different strengths & weaknesses depending on the use case.

I hope you enjoyed reading and learning about topic modeling and find the article useful! ✨

Enjoy this article? Become a member for more!

*You can **read my other articles here** and **follow me on Medium.*** Let me know if you have any questions or suggestions.✨

References

1. Disneyland Reviews data set from Kaggle. License: CC0: Public Domain
2. Header Photo by Bradley Singleton on Unsplash
3. All other images are by the author

NLP Data Science Machine Learning Data Visualization Hands On Tutorials

Follow

Written by Idil Ismiguzel

902 Followers · Writer for Towards Data Science

Data Scientist @ Kolibri Games | Writing on Data Science & Machine Learning | Top 2000 Writer on Medium | MSc, MBA | https://de.linkedin.com/in/idilismiguzel

More from Idil Ismiguzel and Towards Data Science

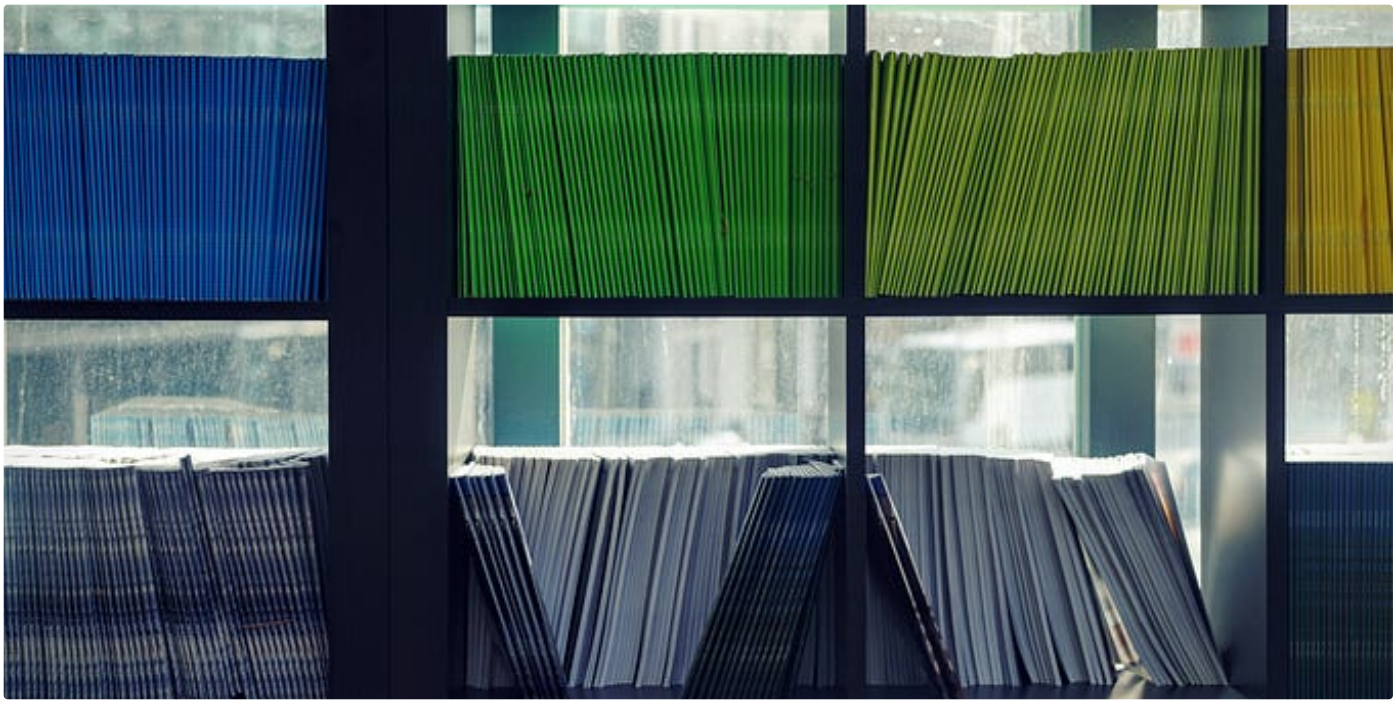
Idil Ismiguzel in Towards Data Science

A Guide to Association Rule Mining

Create insights from frequent patterns using market basket analysis with Python

· 10 min read · Apr 5

62 2

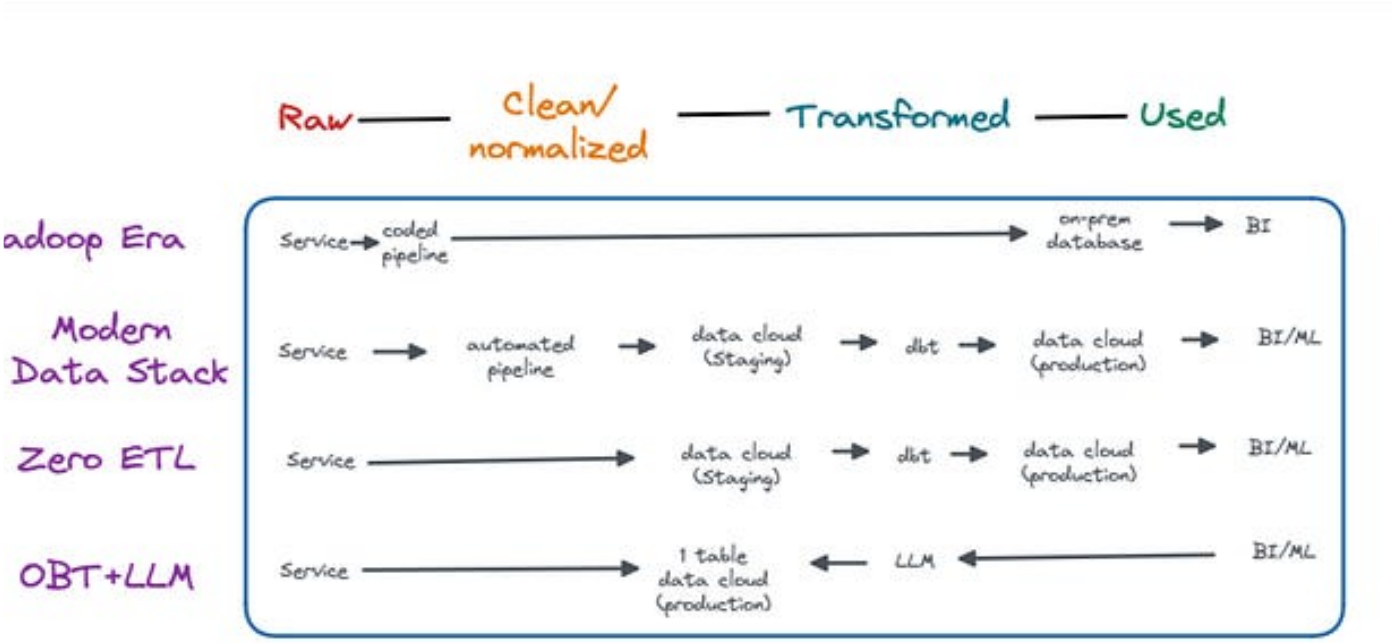


Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25



Barr Moses in Towards Data Science

Zero-ETL, ChatGPT, And The Future of Data Engineering

The post-modern data stack is coming. Are we ready?

9 min read · Apr 3

1.1K 22

Idil Ismiguzel in Towards Data Science

Outlier Detection with Simple and Advanced Techniques

A tutorial on how to detect outliers using standard deviation, interquartile range, isolation forest, DBSCAN, and local outlier factor

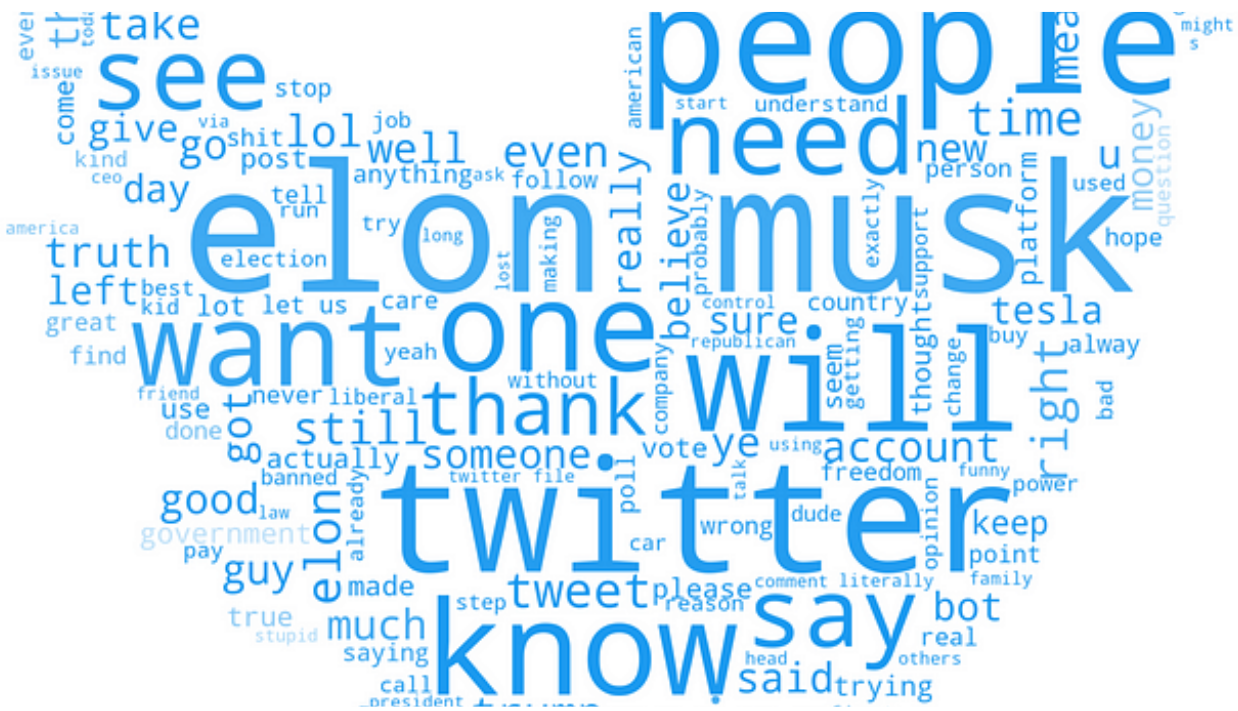
· 10 min read · Nov 17, 2022

129 1

See all from Idil Ismiguzel

See all from Towards Data Science

Recommended from Medium



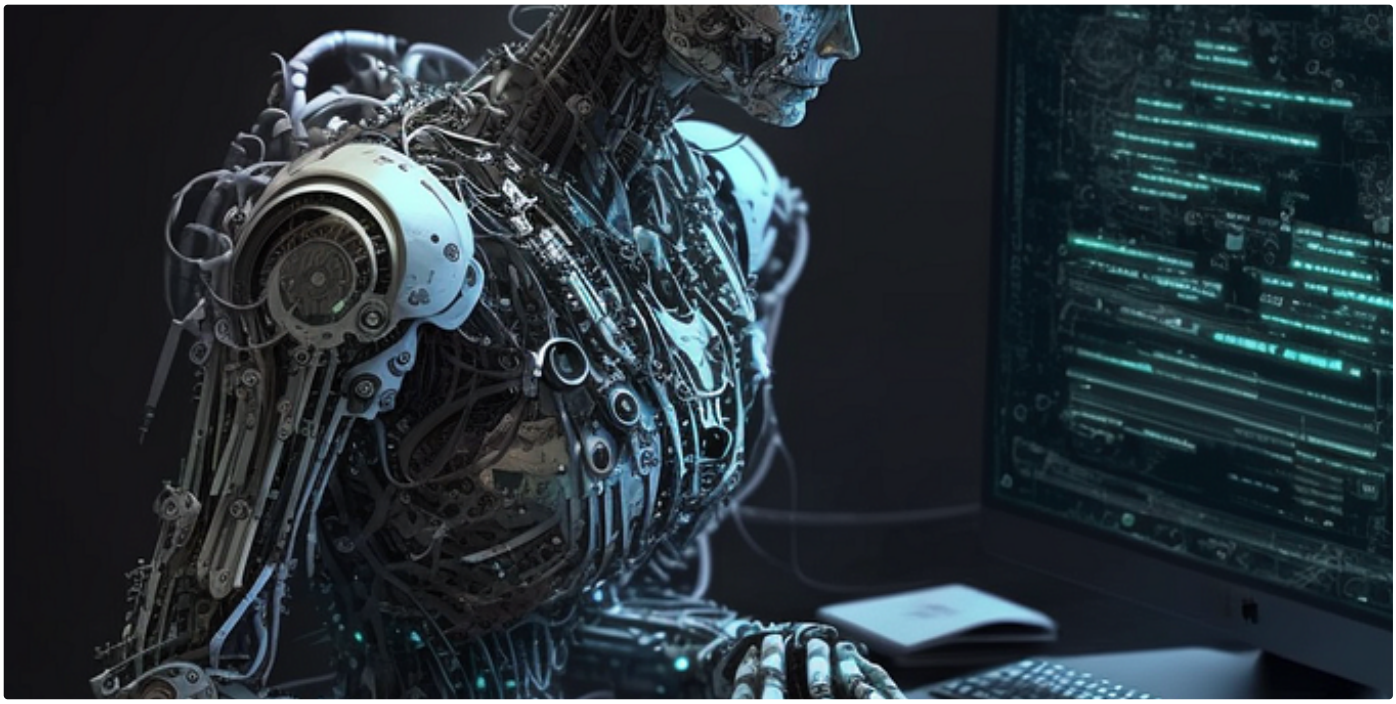
Clément Delteil in Towards AI

Unsupervised Sentiment Analysis With Real-World Data: 500,000 Tweets on Elon Musk

Guided walkthrough in a real-world Natural Language Processing project.

✨ · 15 min read · Feb 11

261 3



Eric Kleppen in Python in Plain English

Topic Modeling For Beginners Using BERTopic and Python

How to make sense of your text data by reducing it to topics

· 11 min read · Feb 12

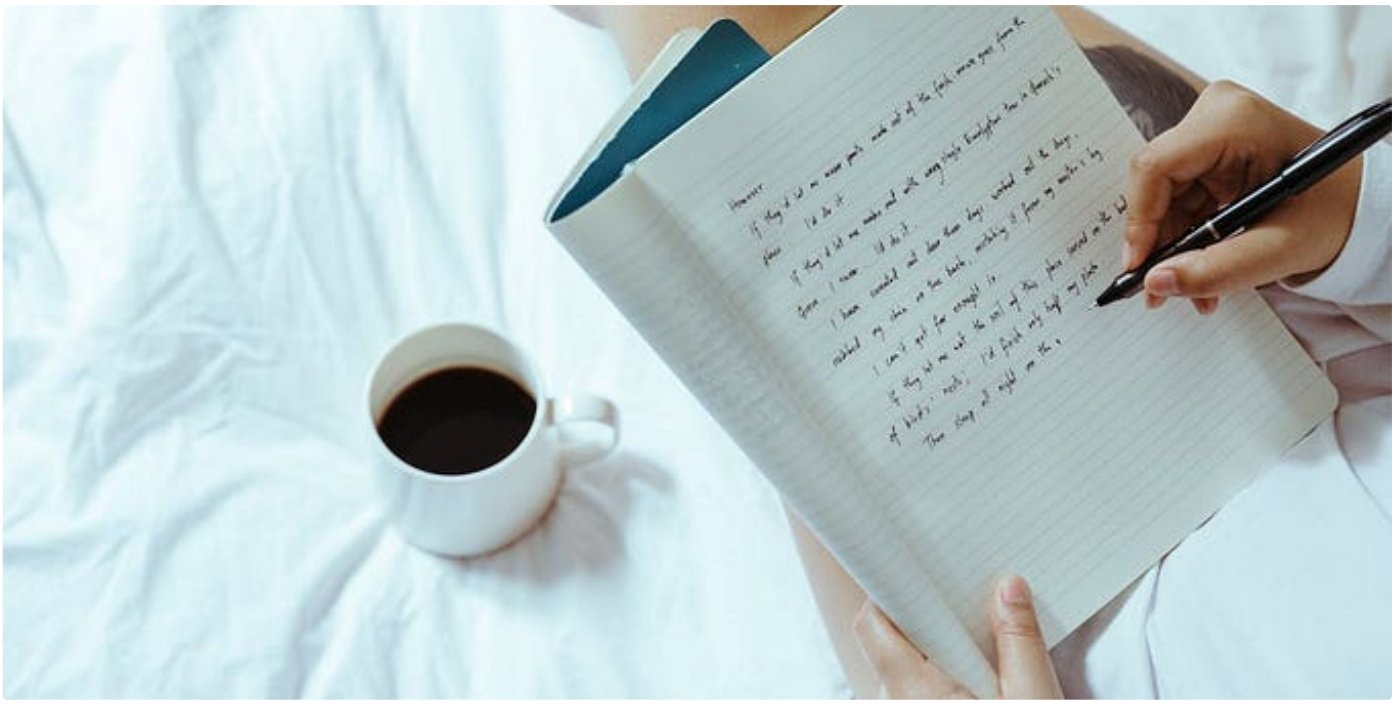
Seungjun (Josh) Kim in Geek Culture

Let us Extract some Topics from Text Data — Part II: Gibbs Sampling Dirichlet Multinomial Mixture...

Learn how to use GSDMM for Topic Modeling and how it compares to LDA

· 9 min read · Nov 7, 2022

56



Seungjun (Josh) Kim in Towards Data Science

Let us Extract some Topics from Text Data — Part IV: BERTopic

Learn more about the family member of BERT for topic modelling

· 10 min read · Dec 19, 2022

64 1

Vatsal in Towards Data Science

Supervised & Unsupervised Approach to Topic Modelling in Python

Build a Topic Modelling Pipeline from Scratch in Python

· 11 min read · Jan 31

Eugenia Anello in Towards AI

Topic Modeling for E-commerce Reviews using BERTopic

A step-by-step guide for training your unsupervised model using a new and improved way to deal with the dataset

✨ · 8 min read · Jan 6

118 1