Run in Google Colab View on Github View raw on Github

# Module 13: Texts

We'll use `spaCy` and `wordcloud` to play with text data. `spaCy` is probably the best python package for analyzing text data. It's capable and super fast. Let's install them.

```
pip install wordcloud spacy
```

To use spaCy, you also need to download models. Run:

```
python -m spacy download en_core_web_sm
```

## SpaCy basics

```
import spacy
import wordcloud

nlp = spacy.load('en_core_web_sm')
```

Usually the first step of text analysis is *tokenization*, which is the process of breaking a document into "tokens". You can roughly think of it as extracting each word.

```
doc = nlp(u'Apple is looking at buying U.K. startup for $1 billion')

for token in doc:
    print(token)
```

```
Apple
is
looking
at
buying
U.K.
startup
for
$
1
billion
```

As you can see, it's not exactly same as `doc.split()`. You'd want to have `$` as a separate token because it has a particular meaning (USD). Actually, as shown in an example (https://spacy.io/usage/spacy-101#annotations-pos-deps), `spaCy` figures out a lot of things about these tokens. For instance,

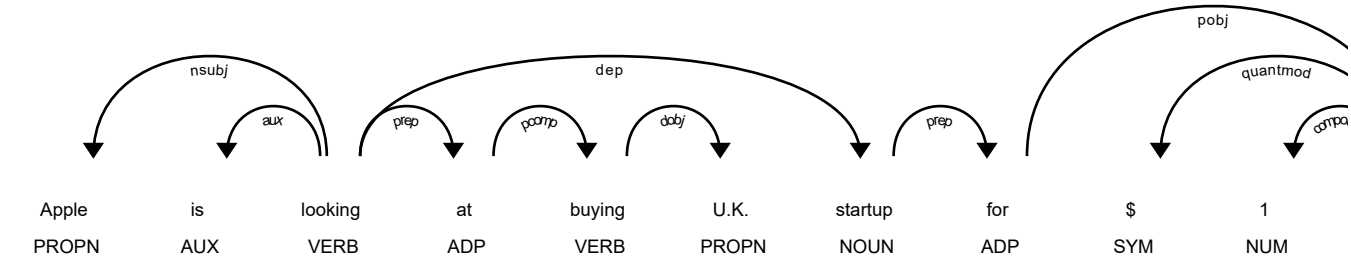
```
for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_)
```

```
Apple Apple PROPN NNP
is be AUX VBZ
looking look VERB VBG
at at ADP IN
buying buy VERB VBG
U.K. U.K. PROPN NNP
startup startup NOUN NN
for for ADP IN
$ $ SYM $
1 1 NUM CD
billion billion NUM CD
```

It figured it out that `Apple` is a proper noun ("PROPN" and "NNP"; see here for the part of speech tags).

`spaCy` has a visualizer too.

```
from spacy import displacy
displacy.render(doc, style='dep', jupyter=True, options={'distance': 100})
```




It even recognizes entities and can visualize them.

```
text = """But Google is starting from behind. The company made a late push
into hardware, and Apple's Siri, available on iPhones, and Amazon's Alexa
software, which runs on its Echo and Dot devices, have clear leads in
consumer adoption."""

doc2 = nlp(text)
displacy.render(doc2, style='ent', jupyter=True)
```

But Google ORG is starting from behind. The company made a late push into hardware, and Apple ORG 's Siri PERSON , available on iPhones ORG , and Amazon ORG 's Alexa ORG software, which runs on its Echo LOC and Dot devices, have clear leads in consumer adoption.

## Let's read a book

Shall we load some serious book? You can use any books that you can find as a text file.

```
import urllib.request

book = urllib.request.urlopen('https://sherlock-holm.es/stories/plain-text/stud.txt').read()
```

```
book[:1000]
```

```
b'\n\n\n\n                              A STUDY IN SCARLET\n\n                                    Arthur Conan Doyle\n\n\n\n\n\n\n\n
Table of contents\n\n         Part I\n        Mr. Sherlock Holmes\n        The Science Of Deduction\n        The Lauriston Garden
Mystery\n        What John Rance Had To Tell\n        Our Advertisement Brings A Visitor\n        Tobias Gregson Shows What He Can
Do\n        Light In The Darkness\n\n         Part II\n        On The Great Alkali Plain\n        The Flower Of Utah\n        John
Ferrier Talks With The Prophet\n        A Flight For Life\n        The Avenging Angels\n        A Continuation Of The Reminiscences
Of John Watson, M.D.\n        The Conclusion\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n                                    PART
I\n\n                  (Being a reprint from the reminiscences of\n                                    John H. Watson, M.D.,\n
late of the Army Medical Department.)\n\n\n\n\n\n          CHAPTER I\n          Mr. Sherlock Holmes\n\n\n     In t'
```

Looks like we have successfully loaded the book. You'd probably want to remove the parts at the beginning and at the end that are not parts of the book if you are doing a serious analysis, but let's ignore them for now. Let's try to feed this directly into `spaCy`.

```
doc = nlp(book)
```

```
---------------------------------------------------------------------------
ExtraData                                 Traceback (most recent call last)
<ipython-input-35-080488d7679a> in <cell line: 1>()
----> 1 doc = nlp(book)

 5 frames
/usr/local/lib/python3.10/dist-packages/srsly/msgpack/_unpacker.pyx in srsly.msgpack._unpacker.unpackb()

ExtraData: unpack(b) received extra data.
```

Next steps: Explain error

## On encodings

What are we getting this error? What does it mean? It says `nlp` function expects `str` type but we passed `bytes`.

```
type(book)
```

```
bytes
```

Indeed, the type of `metamorphosis_book` is `bytes`. But as we have seen above, we can see the book contents right? What's going on?

Well, the problem is that a byte sequence is not yet a proper string until we know how to decode it. A string is an abstract object and we need to specify an encoding to write the string into a file. For instance, if I have a string of Korean characters like "안녕", there are several encodings that I can specify to write that into a file, and depending on the encoding that I choose, the byte sequences can be totally different from each other. This is a really important (and confusing) topic, but because it's beyond the scope of the course, I'll just link a nice post about encoding: http://kunststube.net/encoding/

```
"안녕".encode('utf8')
```

```
b'\xec\x95\x88\xeb\x85\x95'
```

```
# b'\xec\x95\x88\xeb\x85\x95'.decode('euc-kr') <- what happen if you do this?
b'\xec\x95\x88\xeb\x85\x95'.decode('utf8')
```

```
'안녕'
```

```
"안녕".encode('euc-kr')
```

```
b'\xbe\xc8\xb3\xe7'
```

```
b'\xbe\xc8\xb3\xe7'.decode('euc-kr')
```

```
'안녕'
```

You can decode with "wrong" encoding too.

```
b'\xbe\xc8\xb3\xe7'.decode('latin-1')
```

```
'¾È³ç'
```

As you can see the same string can be encoded into different byte sequences depending on the encoding. It's a really ~~annoying~~ fun topic and if you need to deal with text data, you must have a good understanding of it.

There is a lot of complexity in encoding. But for now, just remember that `utf-8` encoding is the most common encoding. It is also compatible with ASCII encoding as well. That means you can *decode* both ASCII and utf-8 documents with utf-8 encoding. So let's decode the byte sequence into a string.

```
# YOUR SOLUTION HERE
book_str = book.decode('utf-8')

print(book_str[:1000])
```

```
                                Table of contents

                 Part I
                Mr. Sherlock Holmes
                The Science Of Deduction
                The Lauriston Garden Mystery
                What John Rance Had To Tell
                Our Advertisement Brings A Visitor
                Tobias Gregson Shows What He Can Do
                Light In The Darkness

                 Part II
                On The Great Alkali Plain
```

```
                                PART I

                    (Being a reprint from the reminiscences of
                              John H. Watson, M.D.,
                         late of the Army Medical Department.)



            CHAPTER I
            Mr. Sherlock Holmes


         In t
```

```
type(book_str)
```

```
str
```

Shall we try again?

```
doc = nlp(book_str)

words = [token.text for token in doc
         if token.is_stop != True and token.is_punct != True]
```

## Let's count!

```
from collections import Counter

Counter(words).most_common(5)
```

```
[('\n     ', 3107),
 ('\n\n     ', 772),
 ('said', 207),
 ('man', 155),
 ('Holmes', 98)]
```

a lot of newline characters and multiple spaces. A quick and dirty way to remove them is split & join. The idea is that you split the document using `split()` and then join with a single space . Can you implement it and print the 10 most common words?

```
# YOUR SOLUTION HERE
cleaned_text = " ".join(book_str.split())

doc = nlp(cleaned_text)

words = [token.text.lower() for token in doc
         if not token.is_stop and not token.is_punct]

word_counts = Counter(words)

most_common_words = word_counts.most_common(10)

print(most_common_words)
```

```
[('said', 207), ('man', 155), ('holmes', 98), ('little', 82), ('time', 77), ('come', 70), ('way', 69), ('came', 67), ('face', 67),
```

Let's keep the object with word count.

```
word_cnt = Counter(words)
```

## Some wordclouds?

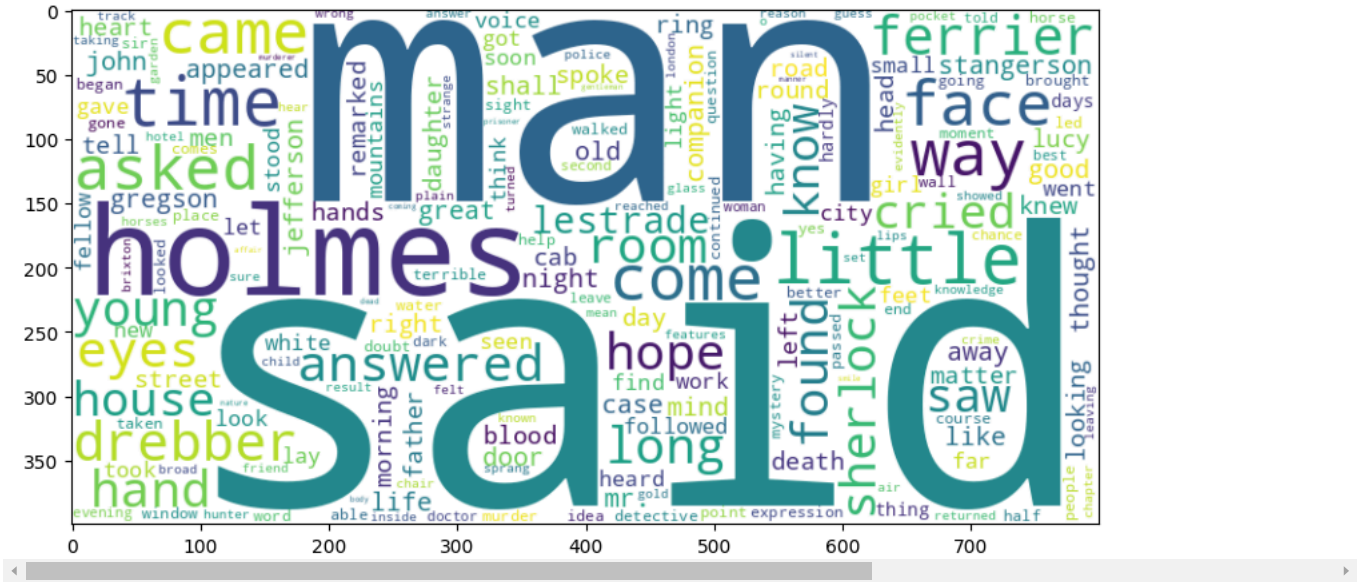
```
import matplotlib.pyplot as plt
```

Can you check out the `wordcloud` package documentation and create a word cloud from the word count object that we created from the book above and plot it?

```
# Implement: create a word cloud object

# YOUR SOLUTION HERE
from wordcloud import WordCloud
import matplotlib.pyplot as plt

wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(word_cnt)
```

```
# Implement: plot the word cloud object

# YOUR SOLUTION HERE
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
```

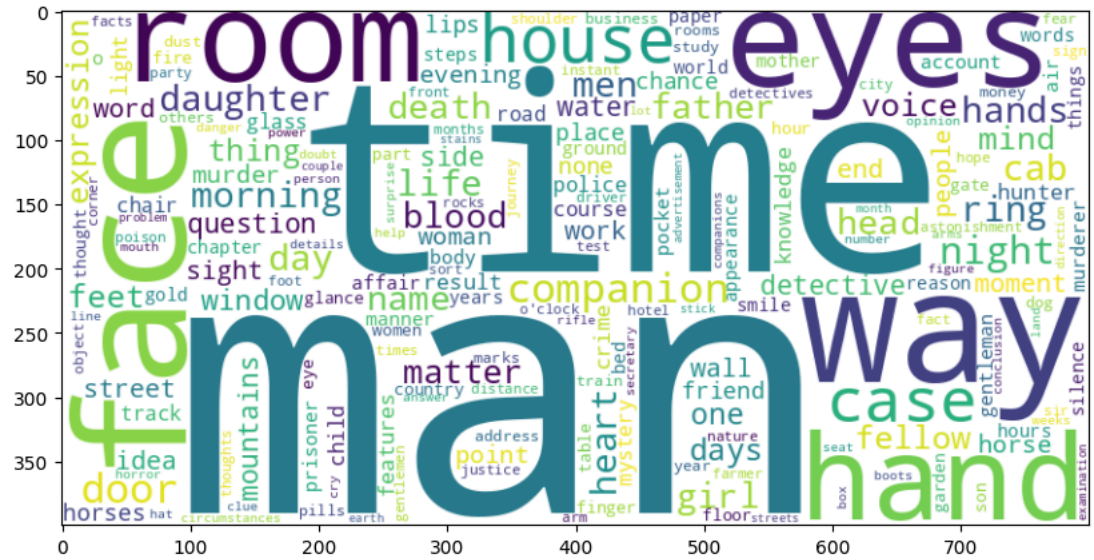
```
<matplotlib.image.AxesImage at 0x7a880c8b1bd0>
```




### Q: Can you create a word cloud for a certain part of speech, such as nouns, verbs, proper nouns, etc. (pick one)?

```
# YOUR SOLUTION HERE
nouns = [token.text.lower() for token in doc if token.pos_ == "NOUN"]

noun_counts = Counter(nouns)

wordcloud_nouns = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(noun_counts)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_nouns, interpolation='bilinear')
```

```
<matplotlib.image.AxesImage at 0x7a8807528490>
```




doc

```
He little knew how far that iron grasp could reach, or how soon it was to close upon them and crush them. About the middle of the second day of their flight their scanty store of provisions began to run out. This gave the hunter little uneasiness, however, for there was game to be had among the mountains, and he had frequently before had to depend upon his rifle for the needs of life. Choosing a sheltered nook, he piled together a few dried branches and made a blazing fire, at which his companions might warm themselves, for they were now nearly five thousand feet above the sea level, and the air was bitter and keen. Having tethered the horses, and bade Lucy adieu, he threw his gun over his shoulder, and set out in search of whatever chance might throw in his way. Looking back he saw the old man and the young girl crouching over the blazing fire, while the three animals stood motionless in the back-ground. Then the intervening rocks hid them from his view. He walked for a couple of miles through one ravine after another without success, though from the marks upon the bark of the trees, and other indications, he judged that there were numerous bears in the vicinity. At last, after two or three hours' fruitless search, he was thinking of turning back in despair, when casting his eyes upwards he saw a sight which sent a thrill of pleasure through his heart. On the edge of a jutting pinnacle, three or four hundred feet above him, there stood a creature somewhat resembling a sheep in appearance, but armed with a pair of gigantic horns. The big-horn--for so it is called--was acting, probably, as a guardian over a flock which were invisible to the hunter; but fortunately it was heading in the opposite direction, and had not perceived him. Lying on his face, he rested his rifle upon a rock, and took a long and steady aim before drawing the trigger. The animal sprang into the air, tottered for a moment upon the edge of the precipice, and then came crashing down into the valley beneath. The creature was too unwieldy to lift, so the hunter contented himself with cutting away one haunch and part of the flank. With this trophy over his shoulder, he hastened to retrace his steps, for the evening was already drawing in. He had hardly started, however, before he realized the difficulty which faced him. In his eagerness he had wandered far past the ravines which were known to him, and it was no easy matter to pick out the path which he had taken. The valley in which he found himself divided and sub-divided into many gorges, which were so like each other that it was impossible to distinguish one from the other. He followed one for a mile or more until he came to a mountain torrent which he was sure that he had never seen before. Convinced that he had taken the wrong turn, he tried another, but with the same result. Night was coming on rapidly, and it was almost dark before he at last found himself in a defile which was familiar to him. Even then it was no easy matter to keep to the right track, for the moon had not yet risen, and the high cliffs on either side made the obscurity more profound. Weighed down with his burden, and weary from his exertions, he stumbled along, keeping up his heart by the reflection that every step brought him nearer to Lucy, and that he carried with him enough to ensure them food for the remainder of their journey. He had now come to the mouth of the very defile in which he had left them. Even in the darkness he could recognize the outline of the cliffs which bounded it. They must, he reflected, be awaiting him anxiously, for he had been absent nearly five hours. In the gladness of his heart he put his hands to his mouth and made the glen re-echo to a loud halloo as a signal that he was coming. He paused and listened for an answer. None came save his own cry, which clattered up the dreary silent ravines, and was borne back to his ears in countless repetitions. Again he shouted, even louder than before, and again no whisper came back from the friends whom he had left such a short time ago. A vague, nameless dread came over him, and he hurried onwards frantically, dropping the precious food in his agitation. When he turned the corner, he came full in sight of the spot where the fire had been lit. There was still a glowing pile of wood ashes there, but it had evidently not been tended since his departure. The same dead silence still reigned all round. With his fears all changed to convictions, he hurried on. There was no living creature near the remains of the fire: animals, man, maiden, all were gone. It was only too clear that some sudden and terrible disaster had occurred during his absence--a disaster which had embraced them all, and yet had left no traces behind it. Bewildered and stunned by this blow, Jefferson Hope felt his head spin round, and had to lean upon his rifle to save himself from falling. He was essentially a man of action, however, and speedily recovered from his temporary impotence. Seizing a half-consumed piece of wood from the smouldering fire, he blew it into a flame, and proceeded with its help to examine the little camp. The ground was all stamped down by the feet of horses, showing that a large party of mounted men had overtaken the fugitives, and the direction of their tracks proved that they had afterwards turned back to Salt Lake City. Had they carried back both of his companions with them? Jefferson Hope had almost persuaded himself that they must have done so, when his eye fell upon an object which made every nerve of his body tingle within him. A little way on one side of the camp was a low-lying heap of reddish soil, which had assuredly not been there before. There was no mistaking it for anything but a newly-dug grave. As the young hunter approached it, he perceived that a stick had been planted on it, with a sheet of paper stuck in the cleft fork of it. The inscription upon the paper was brief, but to the point: JOHN FERRIER, Formerly of Salt Lake City, Died August 4th, 1860. The sturdy old man, whom he had left so short a time before, was gone, then, and this was all his epitaph. Jefferson Hope looked wildly round to see if there was a second grave, but there was no sign of one. Lucy had been carried back by their terrible pursuers to fulfil her original destiny, by becoming one of the harem of the Elder's son. As the young fellow realized the certainty of her fate, and his own powerlessness to prevent it, he wished that he, too, was lying with the old farmer in his last silent resting-place. Again, however, his active spirit shook off the lethargy which springs from despair. If there was nothing else left to him, he could at least devote his life to revenge. With indomitable patience and perseverance, Jefferson Hope possessed also a power of sustained vindictiveness, which he may have learned from the Indians amongst whom he had lived. As he stood by the desolate fire, he felt that the only one thing which could assuage his grief would be thorough and complete retribution, brought by his own hand upon his enemies. His strong will and untiring energy should, he determined, be devoted to that one end. With a grim, white face, he retraced his steps to where he had dropped the food, and having stirred up the smouldering fire, he cooked enough to last him for a few days. This he made up into a bundle, and, tired as he was, he set himself to walk back through the mountains upon the track of the avenging angels. For five days he
```

## Topic modeling

Another basic text analysis is *topic modeling*. Imagine you have a bunch of documents and you want to know what are the main topics that are discussed in these documents. Topic modeling methods aim to extract these "topics" automatically based on the words that appear in the documents. In topic modeling, each topic can be thought of as a distribution over words. For instance, a topic can be "politics" and it can be represented as a probability distribution over words like "election", "vote", "president", etc.

Let's do a simple topic modeling with `bertopic` package. Although there is a long history of topic modeling, with many methods such as LDA, just like many other NLP tasks, topic modeling is also being replaced by LLM-based methods like BERTopic, so we'd like to try that as well. You should already have `scikit-learn` installed. Let's install `bertopic`.

```
pip install bertopic
```

Let's first get the dataset included in scikit-learn.

```
from sklearn.datasets import fetch_20newsgroups

newsgroup_data = fetch_20newsgroups(subset='all')
```

We can inspect what's inside. Feel free to play with it.

```
newsgroup_data.keys()
```

```
dict_keys(['data', 'filenames', 'target_names', 'target', 'DESCR'])
```

```
len(newsgroup_data.filenames)
```

```
18846
```

```
print(newsgroup_data.DESCR)
```

```
    should be taken into consideration when using the dataset, reviewing the
    output, and the bias should be documented.

.. rubric:: Examples

* :ref:`sphx_glr_auto_examples_model_selection_plot_grid_search_text_feature_extraction.py`
* :ref:`sphx_glr_auto_examples_text_plot_document_classification_20newsgroups.py`
* :ref:`sphx_glr_auto_examples_text_plot_hashing_vs_dict_vectorizer.py`
* :ref:`sphx_glr_auto_examples_text_plot_document_clustering.py`
```

You can look into the data as well.

```
print(newsgroup_data.data[0])
```

```
From: Mamatha Devineni Ratnam <mr47+@andrew.cmu.edu>
Subject: Pens fans reactions
Organization: Post Office, Carnegie Mellon, Pittsburgh, PA
Lines: 12
NNTP-Posting-Host: po4.andrew.cmu.edu



I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However, I am going to put an end
to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they
are killing those Devils worse than I thought. Jagr just showed you why
he is much better than his regular season stats. He is also a lot
fo fun to watch in the playoffs. Bowman should let JAgr have a lot of
fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disappointed not to see t
regular season game.          PENS RULE!!!
```

You can load a more "clean" data like following.

```
docs = fetch_20newsgroups(subset='all',  remove=('headers', 'footers', 'quotes'))['data']
```

```
print(docs[0])
```

```
I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However, I am going to put an end
to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they
are killing those Devils worse than I thought. Jagr just showed you why
he is much better than his regular season stats. He is also a lot
fo fun to watch in the playoffs. Bowman should let JAgr have a lot of
fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disappointed not to see t
regular season game.          PENS RULE!!!
```

BERTopic follows the convention of the scikit-learn API. You can fit the model with `fit_transform` method.

```
!pip install bertopic
```

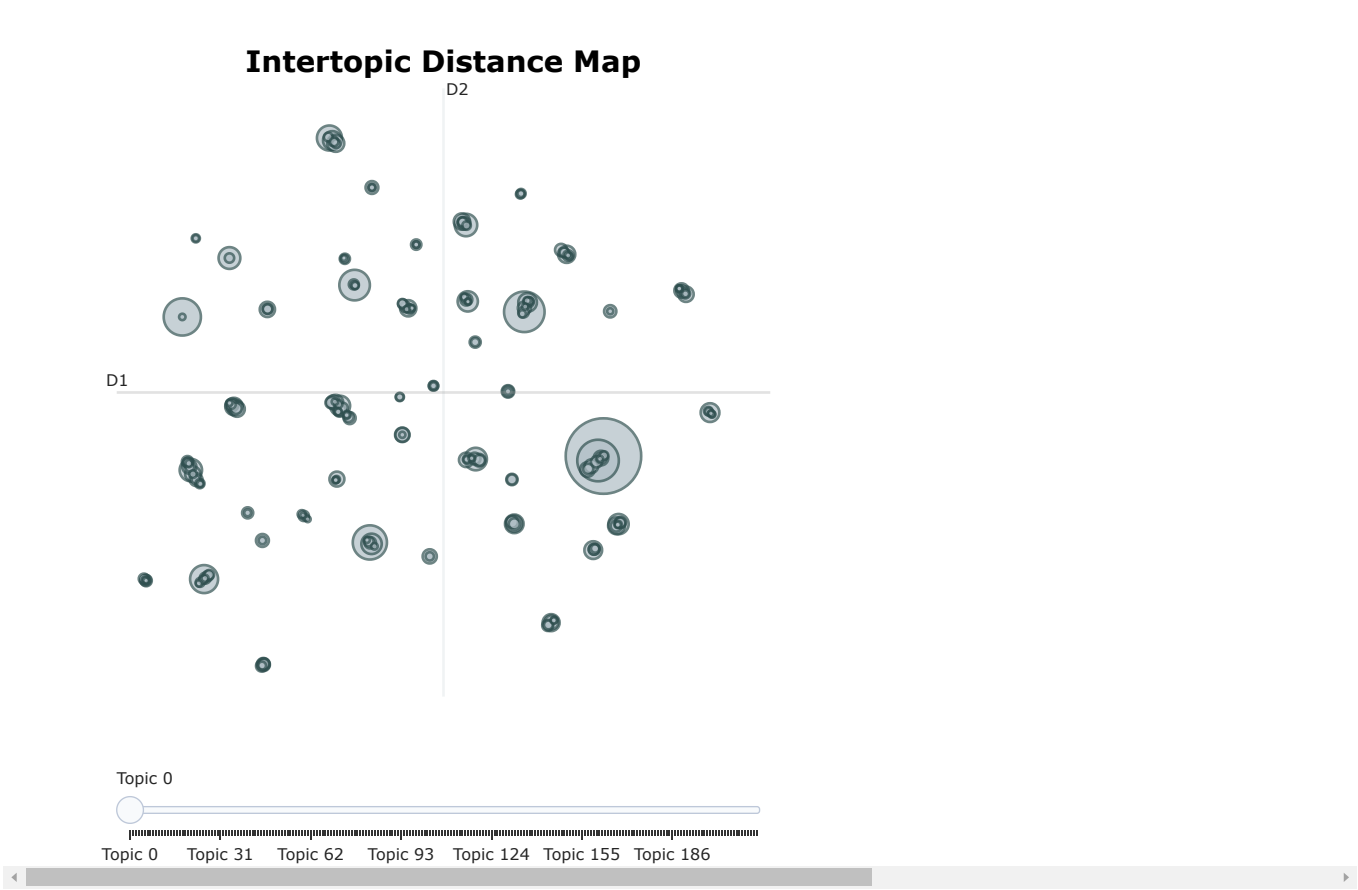
```
Requirement already satisfied: bertopic in /usr/local/lib/python3.10/dist-packages (0.16.4)
Requirement already satisfied: hdbscan>=0.8.29 in /usr/local/lib/python3.10/dist-packages (from bertopic) (0.8.40)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.10/dist-packages (from bertopic) (1.26.4)
Requirement already satisfied: pandas>=1.1.5 in /usr/local/lib/python3.10/dist-packages (from bertopic) (2.2.2)
Requirement already satisfied: plotly>=4.7.0 in /usr/local/lib/python3.10/dist-packages (from bertopic) (5.24.1)
Requirement already satisfied: scikit-learn>=0.22.2.post1 in /usr/local/lib/python3.10/dist-packages (from bertopic) (1.5.2)
Requirement already satisfied: sentence-transformers>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from bertopic) (3.2.1)
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.10/dist-packages (from bertopic) (4.66.6)
Requirement already satisfied: umap-learn>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from bertopic) (0.5.7)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.10/dist-packages (from hdbscan>=0.8.29->bertopic) (1.13.1)
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.10/dist-packages (from hdbscan>=0.8.29->bertopic) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.5->bertopic) (2.8
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.5->bertopic) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.5->bertopic) (2024.2)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly>=4.7.0->bertopic) (9.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly>=4.7.0->bertopic) (24.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22.2.post1->ber
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from sentence-transformers>=0
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from sentence-transformers>=0.4.1->bertopic
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from sentence-transformers>=0.4.1
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from sentence-transformers>=0.4.1->bertopic) (11.0
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-packages (from umap-learn>=0.5.0->bertopic) (0.60.0)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.10/dist-packages (from umap-learn>=0.5.0->bertopic) (0.5.1
```

```
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0->sentence-transform
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0->sentence-t
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0->sentence-transform
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.20.0-
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.2->umap-learn
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=1.1.5->bert
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence-transformers>=0.4.1
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence-transformers>=0.4.1-
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence-transformers>=
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch>=1.11.0->ser
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.41.0->sente
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.41.0->sent
Requirement already satisfied: tokenizers<0.21,>=0.20 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.41.0-
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.11.0->sentence-tran
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.20.0->sent
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.20.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.20.0
```

```
from bertopic import BERTopic

topic_model = BERTopic()
topics, probs = topic_model.fit_transform(docs)
```

The package also has a quick visualizer, where you can see the topics and the words that are associated with the topics.

```
topic_model.visualize_topics()
```

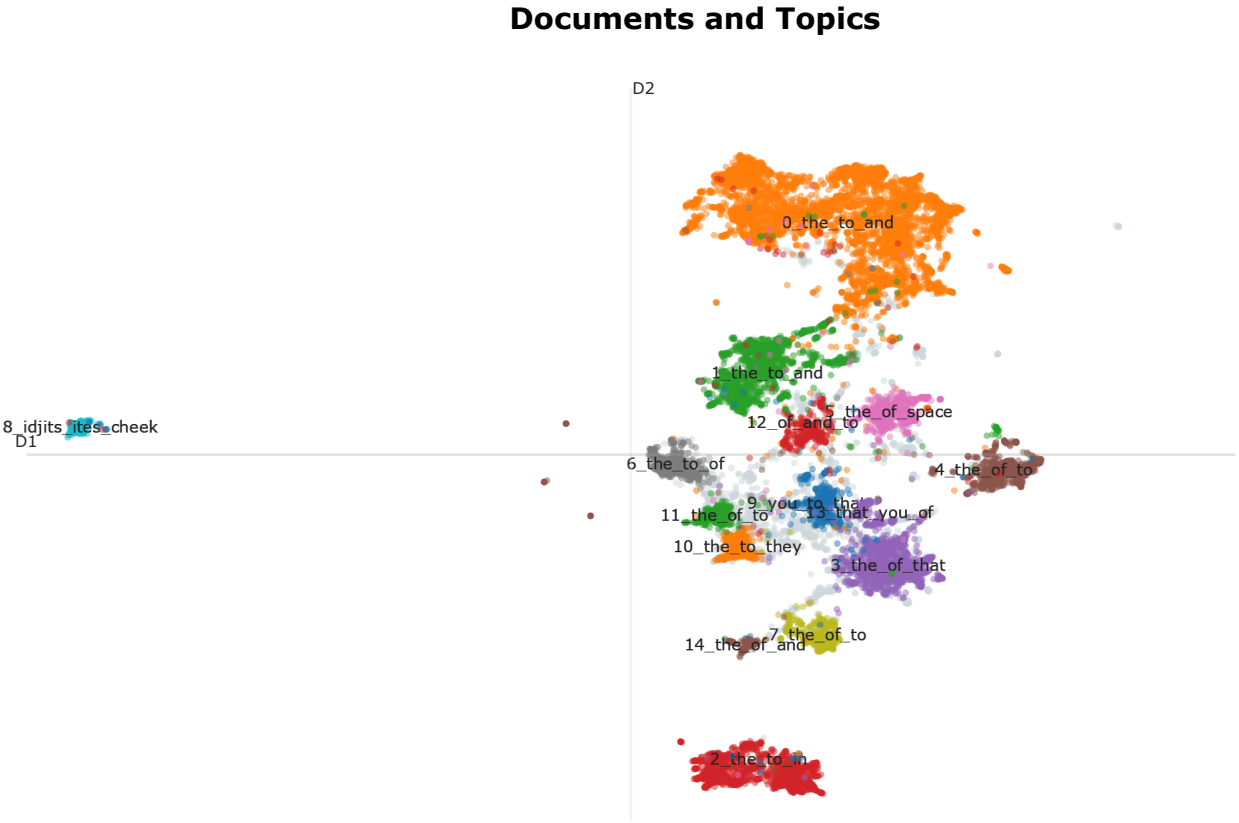



Another visualization we can do (see https://maartengr.github.io/BERTopic/getting_started/visualization/visualization.html#visualize-documents) is to use the sentence embedding model to embed the documents into a vector space and visualize all the documents using the UMAP algorithm, along with the topics.

This time, I'll set the `min_topic_size` to 100 to reduce the number of topics it finds and to make the visualization simpler.

```
from sentence_transformers import SentenceTransformer
from umap import UMAP

# Prepare embeddings
sentence_model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = sentence_model.encode(docs, show_progress_bar=False)

topic_model = BERTopic(min_topic_size=100).fit(docs, embeddings)
```

```
# Run the visualization with the original embeddings
topic_model.visualize_documents(docs, embeddings=embeddings)

# Reduce dimensionality of embeddings, this step is optional but much faster to perform iteratively:
reduced_embeddings = UMAP(n_neighbors=10, n_components=2, min_dist=0.0, metric='cosine').fit_transform(embeddings)
topic_model.visualize_documents(docs, reduced_embeddings=reduced_embeddings)
```

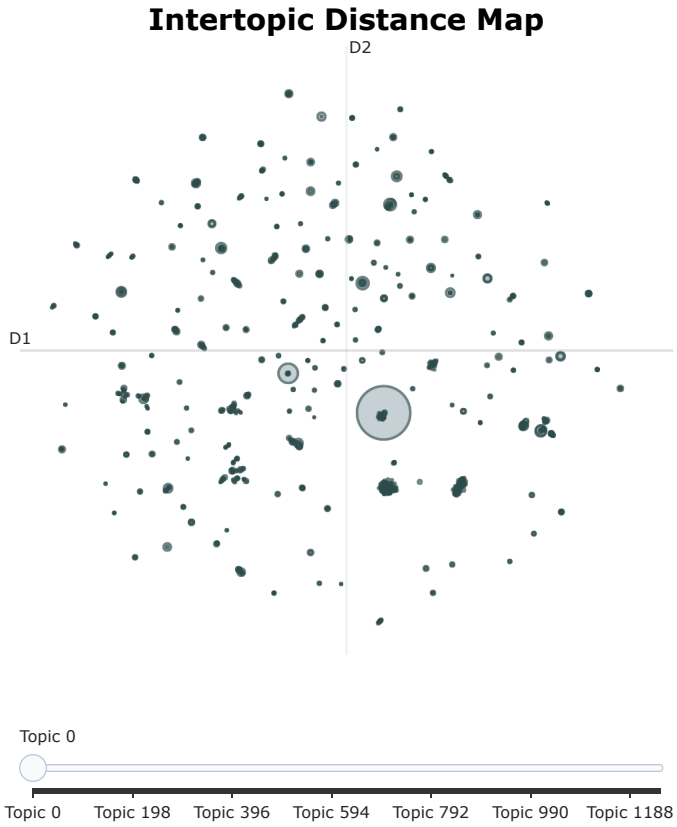



## Q: Can you identify an interesting dataset of documents and apply BERTopic & produce the visualization?

```
from nltk.corpus import shakespeare
import nltk

# Download the Shakespeare dataset
nltk.download("shakespeare")

# Process the Shakespeare dataset
docs = [" ".join(line.split()) for fileid in shakespeare.fileids() for line in shakespeare.raw(fileid).splitlines() if line.strip()]

sentence_model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = sentence_model.encode(docs, show_progress_bar=True)

topic_model = BERTopic(min_topic_size=5)
topics, probs = topic_model.fit_transform(docs, embeddings)
```

```
[nltk_data] Downloading package shakespeare to /root/nltk_data...
[nltk_data]   Package shakespeare is already up-to-date!
Batches: 100%  1475/1475 [00:15<00:00, 108.31it/s]
```

```
topic_model.visualize_topics().show()
```




```
reduced_embeddings = UMAP(n_neighbors=10, n_components=2, min_dist=0.0, metric='cosine').fit_transform(embeddings)
fig = topic_model.visualize_documents(docs, reduced_embeddings=reduced_embeddings)
fig.update_traces(text=None, hovertemplate=None)

fig.update_layout(showlegend=True)

# Show the visualization
fig.show()
```