Technology: Semantic searching

	Open source	Accuracy	Speed
Sentence Transformer	√	√	√
Scapy	√	x	x
Openai	x	／	／

Top results for query COVID 19

Sentence Transformer:

We are doing research, but I have not heard anything specific to Covid-19. (Score: 0.7449)
We are still wrestling with what effects COVID-19 has had. (Score: 0.6983)
We are currently living in unprecedented times with COVID-19. (Score: 0.6779)

Scapy:

DR. WOODARD: We added the standard 5 percent administrative cap; the 8 percent are the indirect fees.
Senate Committee on Health and Human Services April 27, 2021 BARRY GOLD (AARP): We support A.B.
ASSEMBLYWOMAN TITUS MADE A MOTION TO AMEND AND DO PASS ASSEMBLY BILL 216.

Technology: nltk, textblob, wordcloud

nltk

1. Word_tokenize: split a given sentence into words

```
from nltk.tokenize import word_tokenize

words = word_tokenize(text)
print(words)
```

```
['A', 'Topic', 'in', 'Kafka', 'is', 'something', 'where', 'a', 'message',
'is', 'sent', '.', 'The', 'consumer', 'applications', 'which', 'are', 'in
terested', 'in', 'that', 'topic', 'pulls', 'the', 'message', 'inside', 't
hat', 'topic', 'and', 'can', 'do', 'anything', 'with', 'that', 'data',
'.', 'Up', 'to', 'a', 'specific', 'time', ',', 'any', 'number', 'of', 'co
nsumer', 'applications', 'can', 'pull', 'this', 'message', 'any', 'numbe
r', 'of', 'times', '.']
```

Technology: nltk, textblob, wordcloud

nltk

2. Pos_tag: assign each word in a text corpus to a grammatical category

```
pos = nltk.pos_tag(tokens)
pos
```

```
[('This', 'DT'),
 ('is', 'VBZ'),
 ('an', 'DT'),
 ('article', 'NN'),
 ('on', 'IN'),
 ('Sentiment', 'NN'),
 ('Analysis', 'NN')]
```

Technology: nltk, textblob, wordcloud

nltk

3. Stopwords: a corpus, a list of words that are very common but don't provide useful information for most text analysis procedures

In [7]:
```
import nltk
STOP_WORDS = nltk.corpus.stopwords.words('english')
STOP_WORDS.append('Test')

print(len(STOP_WORDS))
print(STOP_WORDS)
```

```
180
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", 'Test']
```

Technology: nltk, textblob, wordcloud

nltk

4. Freqdist: a function which gives you the frequency of words within a text

```
1  FreqDist(all_song_tokens)
```

executed in 142ms, finished 16:13:06 2020-12-20

```
FreqDist({'i': 1365, 'the': 1359, 'a': 1020, 'you': 932, 'it': 895, 'my': 8
65, 'im': 667, 'in': 632, 'like': 630, 'to': 624, ...})
```

Technology: nltk, textblob, wordcloud

Analysis in nltk divided into two parts:

1. Cleaning part: use the Word_tokenize, pos_tag technology combined with stopwords corpus, clean stop words, punctuations, lemmatized words, md words in the original json file.
2. Analysis part: use freqdist to do word frequency analysis, implement TF-IDF technology(term frequency-inverse document frequency) for key word extraction

Technology: nltk, textblob, wordcloud

Textblob

- Use the textblob to complete sentiment analysis for the text.We will get sentiment polarity of every sentence, positive or negative, ranging from [-1,1]

```
TextBlob("The movie is good").sentiment
```

```
Sentiment(polarity=0.7, subjectivity=0.6000000000000001)
```

```
TextBlob("This movie is bad").sentiment
```

```
Sentiment(polarity=-0.6999999999999998, subjectivity=0.6666666666666666)
```

Technology: nltk, textblob, wordcloud

wordcloud

- In visualization part, we use wordcloud technology , a data visualization technology to represent data by the frequency..

Technology: Plotly Dash

1. dash_bootstrap_components(Card): a library of Bootstrap components for use with Plotly Dash.
2. Html:html components to help us organize and display the output
3. Dcc(dropdown, input): (ConfirmDialog component)send a dialog to the browser asking the user to confirm or cancel with a custom message
4. dash.dependencies.input,dash.dependencies.output