```
# !pip install spacy
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages (3.8.7)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.13)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.10)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (8.3.6)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.16.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.11.5
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.5.0)
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2.0->spacy)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->sp
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (8.2.1)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (1.5
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (13.9.4)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spa
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.2->langcodes<4.0
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0,>=0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0,>
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typ
```

```
#!python  -m spacy downmload en_core_web_sm
```

```
import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp("data science and a great carrer ahead")
```

```
# import PyQt5
```

```
doc
```

```
data science and a great carrer ahead
```

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

```
data
science
and
a
great
carrer
ahead
```

```
import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp("data science and a great carrer ahead")
for token in doc:
  print(token.text,token.lemma_,token.pos_,token.pos_,token.tag_,token.dep_,token.shape_,token.is_alpha,token.is_stop)
```

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

```
NOUN
NOUN
CCONJ
```

```
DET
ADJ
NOUN
ADV
```

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

```
data NOUN data
science NOUN science
and CCONJ and
a DET a
great ADJ great
carrer NOUN carrer
ahead ADV ahead
```

```
text = """here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first
An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given docume
Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set
```

```
text
```

```
'here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is
generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of im
ages, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which
summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic mac
hine-generated summaries depending on what the user needs.\nAn example of a summarization problem is document summarization, which
attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from
```

```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
```

```
stopwords = list(STOP_WORDS)
stopwords
```

```
"'s"
```

```
  's',
 'less',
 'though',
 'although',
 'but',
 'latterly',
 'more',
 'sometime',
 'using',
 'since']
```

```
len(stopwords)
```

```
326
```

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

```
text
```

```
'here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.\nAn example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). Th
```

```
doc = nlp(text)
doc
```

```
here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.
An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). This problem is called multi-document summarization. A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.
Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set of images from a larger set of images.[4] A summary in this context is useful to show the most representative images of results in an image collection exploration system. Video summarization is a related domain, where the system automatically creates a trailer of a long video. This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions. Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured
```

```
# lets gets the token  from text
tokens =[token.text for token in doc]
print(tokens)
```

```
['here', 'are', 'broadly', 'two', 'types', 'of', 'extractive', 'summarization', 'tasks', 'depending', 'on', 'what', 'the', 'summariz
```

```
tokens
```

```
Unsplash
```

```
 'boring',
 'or',
 'repetitive',
 'actions',
 '.',
 'Similarly',
 ',',
 'in',
 'surveillance',
 'videos',
 ',',
 'one',
 'would',
 'want',
 'to',
 'extract',
 'important',
 'and',
 'suspicious',
 'activity',
 ',',
 'while',
 'ignoring',
 'all',
 'the',
 'boring',
 'and',
 'redundant',
 'frames',
 'captured']
```

```
len(tokens)
```

```
322
```

```
punctuation # also called noisy characters
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
doc
```

```
here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.
An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). This problem is called multi-document summarization. A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.
Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set of images from a larger set of images.[4] A summary in this context is useful to show the most representative images of results in an image collection exploration system. Video summarization is a related domain, where the system automatically creates a trailer of a long video. This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions. Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured
```

```
word_frequencies = {}
for word in doc:
  if word.text.lower() not in stopwords:
    if word.text.lower() not in punctuation:
      if word.text not in word_frequencies.keys():
       word_frequencies[word.text] = 1
      else:
        word_frequencies[word.text] += 1
```

```
word_frequencies
```

```

```

```
 'web': 1,
 'concisely': 1,
 'represents': 1,
 'latest': 1,
 'Image': 1,
 'automatic': 1,
 'consists': 1,
 'selecting': 1,
 'representative': 2,
 'set': 2,
 'larger': 1,
 'images.[4': 1,
 'context': 1,
 'useful': 1,
 'results': 1,
 'image': 1,
 'exploration': 1,
 'Video': 1,
 'domain': 1,
 'creates': 1,
 'trailer': 1,
 'long': 1,
 'video': 1,
 'applications': 1,
 'consumer': 1,
 'personal': 1,
 'want': 2,
 'skip': 1,
 'boring': 2,
 'repetitive': 1,
 'actions': 1,
 'Similarly': 1,
 'surveillance': 1,
 'extract': 1,
 'important': 1,
 'suspicious': 1,
 'activity': 1,
 'ignoring': 1,
 'redundant': 1,
 'frames': 1,
 'captured': 1}
```

```
len(word_frequencies)
```

```
103
```

```
word_frequencies
```

```
 'some words': 1,
```

```
 'consumer': 1,
'personal': 1,
'want': 2,
'skip': 1,
'boring': 2,
'repetitive': 1,
'actions': 1,
'Similarly': 1,
'surveillance': 1,
'extract': 1,
'important': 1,
'suspicious': 1,
'activity': 1,
'ignoring': 1,
'redundant': 1,
'frames': 1,
'captured': 1}
```

```
max_frequency = max(word_frequencies.values())
max_frequency
```

```
11
```

```
# to get normalised/weighted frequencies you should divivde all frequency with all
 for word in word_frequencies.keys():
  word_frequencies[word] = word_frequencies[word]/max_frequency
```

```
# print(word_frequencies)
word_frequencies
```

```
'given': 2,
'interested': 1,
'generating': 1,
'single': 1,
'source': 2,
'use': 1,
'multiple': 1,
'cluster': 1,
'articles': 3,
'topic': 2,
'multi': 1,
'related': 2,
```

```
captured : 1}
```

```
sentence_tokens = [sent for sent in doc.sents]
sentence_tokens
```

```
[here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on.,
 The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.).,
 The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.,
 Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.,
 An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.,
 Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic).,
 This problem is called multi-document summarization.,
 A related application is summarizing news articles.,
 Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.,
 Image collection summarization is another application example of automatic summarization.,
 It consists in selecting a representative set of images from a larger set of images.[4],
 A summary in this context is useful to show the most representative images of results in an image collection exploration system.,
 Video summarization is a related domain, where the system automatically creates a trailer of a long video.,
 This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions.,
 Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured]
```

```
len(sentence_tokens)
```

```
15
```

```
sentence_tokens
```

```
[here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on.,
 The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.).,
 The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.,
 Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.,
 An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.,
 Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic).,
 This problem is called multi-document summarization.,
 A related application is summarizing news articles.,
 Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.,
 Image collection summarization is another application example of automatic summarization.,
 It consists in selecting a representative set of images from a larger set of images.[4],
 A summary in this context is useful to show the most representative images of results in an image collection exploration system.,
 Video summarization is a related domain, where the system automatically creates a trailer of a long video.,
 This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions.,
 Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured]
```

```
# we are going to calculate the sentence score , to calculate the sentence score
sentence_scores = {}

for sent in sentence_tokens:
    for word in sent:
        if word.text.lower() in word_frequencies.keys():
            if sent not in sentence_scores.keys():
                sentence_scores[sent] = word_frequencies[word.text.lower()]
            else:
                sentence_scores[sent] += word_frequencies[word.text.lower()]
```

```
sentence_scores
```

```
{here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on.: 31,
 The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.).: 44,
 The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.: 43,
 Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.: 36,
 An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.: 44,
 Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic).: 28,
 This problem is called multi-document summarization.: 20,
 A related application is summarizing news articles.: 12,
 Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the
```

```
latest news as a summary.: 32,
 Image collection summarization is another application example of automatic summarization.: 32,
 It consists in selecting a representative set of images from a larger set of images.[4]: 13,
 A summary in this context is useful to show the most representative images of results in an image collection exploration system.: 20,
 Video summarization is a related domain, where the system automatically creates a trailer of a long video.: 25,
 This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions.: 13,
 Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured: 16}


# lets say our case study was 30% sentence with maximum scores
from heapq import nlargest


select_length = int(len(sentence_tokens)*0.4)
select_length

6


summary = nlargest(select_length,sentence_scores,key = sentence_scores.get)



summary

[The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.).,
 An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.,
 The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.,
 Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.,
 Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.,
 Image collection summarization is another application example of automatic summarization.]


sentence_scores

{here are broadly two types of extractive summarization tasks depending on what the summarization program focuses on.: 31,
 The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.).: 44,
 The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.: 43,
 Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.: 36,
 An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.: 44,
 Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic).: 28,
 This problem is called multi-document summarization.: 20,
 A related application is summarizing news articles.: 12,
 Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.: 32,
 Image collection summarization is another application example of automatic summarization.: 32,
 It consists in selecting a representative set of images from a larger set of images.[4]: 13,
 A summary in this context is useful to show the most representative images of results in an image collection exploration system.: 20,
 Video summarization is a related domain, where the system automatically creates a trailer of a long video.: 25,
 This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions.: 13,
 Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured: 16}


# if i need to combine these three top 3 sentences then :
final_summary = [word.text for word in summary]


final_summary

['The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.).',
 'An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.',
 'The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.',
 'Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.\n',
 'Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.\n',
 'Image collection summarization is another application example of automatic summarization.']


print(summary)

[The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents,
, Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the ]
```

, Image collection summarization is another application example of automatic summarization.]

Start coding or generate with AI.