

# Spacy \_Text Summaerization

```
In [114... !pip install spacy  
!python -m spacy download en_core_web_sm
```

Requirement already satisfied: spacy in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (3.8.2)

Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (3.0.12)

Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (1.0.5)

Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (1.0.10)

Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (2.0.8)

Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (3.0.9)

Requirement already satisfied: thinc<8.4.0,>=8.3.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (8.3.2)

Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (1.1.3)

Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (2.4.8)

Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (2.0.10)

Requirement already satisfied: weasel<0.5.0,>=0.1.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (0.4.1)

Requirement already satisfied: typer<1.0.0,>=0.3.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (0.13.1)

Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (4.67.0)

Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (2.32.3)

Requirement already satisfied: pydantic!=1.8,!1.8.1,<3.0.0,>=1.7.4 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (2.9.2)

Requirement already satisfied: jinja2 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (3.1.4)

Requirement already satisfied: setuptools in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (75.1.0)

Requirement already satisfied: packaging>=20.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (24.1)

Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (3.4.1)

Requirement already satisfied: numpy>=1.19.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from spacy) (2.0.2)

Requirement already satisfied: language-data>=1.2 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from langcodes<4.0.0,>=3.2.0->spacy) (1.2.0)

Requirement already satisfied: annotated-types>=0.6.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from pydantic!=1.8,!1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)

Requirement already satisfied: pydantic-core==2.23.4 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from pydantic!=1.8,!1.8.1,<3.0.0,>=1.7.4->spacy) (2.23.4)

Requirement already satisfied: typing-extensions>=4.6.1 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from pydantic!=1.8,!1.8.1,<3.0.0,>=1.7.4->spacy) (4.11.0)

Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (3.3.2)

Requirement already satisfied: idna<4,>=2.5 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (3.7)

Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.2.3)

Requirement already satisfied: certifi>=2017.4.17 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2024.8.

30)

Requirement already satisfied: blis<1.1.0,>=1.0.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from thinc<8.4.0,>=8.3.0->spacy) (1.0.1)

Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from thinc<8.4.0,>=8.3.0->spacy) (0.1.5)

Requirement already satisfied: colorama in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from tqdm<5.0.0,>=4.38.0->spacy) (0.4.6)

Requirement already satisfied: click>=8.0.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from typer<1.0.0,>=0.3.0->spacy) (8.1.7)

Requirement already satisfied: shellingham>=1.3.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from typer<1.0.0,>=0.3.0->spacy) (1.5.4)

Requirement already satisfied: rich>=10.11.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from typer<1.0.0,>=0.3.0->spacy) (13.7.1)

Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from weasel<0.5.0,>=0.1.0->spacy) (0.20.0)

Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from weasel<0.5.0,>=0.1.0->spacy) (7.0.5)

Requirement already satisfied: MarkupSafe>=2.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from jinja2->spacy) (2.1.3)

Requirement already satisfied: marisa-trie>=0.7.7 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from language-data>=1.2->langcodes<4.0.0,>=3.2.0->spacy) (1.2.1)

Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy) (2.2.0)

Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy) (2.15.1)

Requirement already satisfied: wrapt in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0->spacy) (1.14.1)

Requirement already satisfied: mdurl~=0.1 in c:\users\roy62\anaconda3\envs\tensorflow\_env\lib\site-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy) (0.1.0)

Collecting en-core-web-sm==3.8.0

Using cached [https://github.com/explosion/spacy-models/releases/download/en\\_core\\_web\\_sm-3.8.0/en\\_core\\_web\\_sm-3.8.0-py3-none-any.whl](https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any.whl) (12.8 MB)

[+] Download and installation successful

You can now load the package via spacy.load('en\_core\_web\_sm')

In [115... `import spacy`

In [116... `nlp = spacy.load("en_core_web_sm")`

In [117... `doc = nlp("data science and ai has greate career ahead")`

In [118... `import PyQt5`

In [119... `doc`

Out[119... data science and ai has greate career ahead

In [120... `for token in doc:  
 print(token.text)`

data  
science  
and  
ai  
has  
create  
career  
ahead

In [121...

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
          token.shape_, token.is_alpha, token.is_stop)
```

Apple Apple PROPN NNP nsubj Xxxxx True False  
is be AUX VBZ aux xx True True  
looking look VERB VBG ROOT xxxx True False  
at at ADP IN prep xx True True  
buying buy VERB VBG pcomp xxxx True False  
U.K. U.K. PROPN NNP nsubj X.X. False False  
startup startup VERB VBD ccomp xxxx True False  
for for ADP IN prep xxx True True  
\$ \$ SYM \$ quantmod \$ False False  
1 1 NUM CD compound d False False  
billion billion NUM CD pobj xxxx True False

In [122...

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

PROPN  
AUX  
VERB  
ADP  
VERB  
PROPN  
VERB  
ADP  
SYM  
NUM  
NUM

In [123...

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

Apple PROPN  
is AUX  
looking VERB  
at ADP  
buying VERB  
U.K. PROPN  
startup VERB  
for ADP  
\$ SYM  
1 NUM  
billion NUM

```
In [124... for token in doc:
            print(token.text, token.pos_, token.lemma_)
```

```
Apple PROPN Apple
is AUX be
looking VERB look
at ADP at
buying VERB buy
U.K. PROPN U.K.
startup VERB startup
for ADP for
$ SYM $
1 NUM 1
billion NUM billion
```

```
In [126... text = """There are broadly two types of extractive summarization tasks dependin
An example of a summarization problem is document summarization, which attempts
Image collection summarization is another application example of automatic summa
```

```
In [127... text
```

```
Out[127... 'There are broadly two types of extractive summarization tasks depending on wha
t the summarization program focuses on. The first is generic summarization, whi
ch focuses on obtaining a generic summary or abstract of the collection (whethe
r documents, or sets of images, or videos, news stories etc.). The second is qu
ery relevant summarization, sometimes called query-based summarization, which s
ummarizes objects specific to a query. Summarization systems are able to create
both query relevant text summaries and generic machine-generated summaries depe
nding on what the user needs.\nAn example of a summarization problem is documen
t summarization, which attempts to automatically produce an abstract from a giv
en document. Sometimes one might be interested in generating a summary from a s
ingle source document, while others can use multiple source documents (for exam
ple, a cluster of articles on the same topic). This problem is called multi-doc
ument summarization. A related application is summarizing news articles. Imagin
e a system, which automatically pulls together news articles on a given topic
(from the web), and concisely represents the latest news as a summary.\nImage c
ollection summarization is another application example of automatic summarizati
on. It consists in selecting a representative set of images from a larger set o
f images.[4] A summary in this context is useful to show the most representativ
e images of results in an image collection exploration system. Video summarizat
ion 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 on
e might want to skip the boring or repetitive actions. Similarly, in surveillan
ce videos, one would want to extract important and suspicious activity, while i
gnoring all the boring and redundant frames captured '
```

```
In [128... import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
```

```
In [129... stopwords = list(STOP_WORDS)
stopwords
```

```
Out[129... ['ca',  
            'ten',  
            'though',  
            'about',  
            'using',  
            'anyway',  
            'himself',  
            'since',  
            'here',  
            'out',  
            'only',  
            'up',  
            'nowhere',  
            'n't',  
            'had',  
            'd',  
            'why',  
            'hereby',  
            'namely',  
            'some',  
            'together',  
            'every',  
            'ours',  
            'whose',  
            'n't',  
            'than',  
            'across',  
            'move',  
            'formerly',  
            'anyhow',  
            'thereupon',  
            'few',  
            'amongst',  
            'others',  
            'nevertheless',  
            'empty',  
            'will',  
            'off',  
            'therein',  
            'should',  
            'thru',  
            'what',  
            'your',  
            'may',  
            'is',  
            'ever',  
            'hereafter',  
            'latterly',  
            'could',  
            'seems',  
            'last',  
            'does',  
            'take',  
            'becomes',  
            'yet',  
            'thence',  
            'therefore',  
            'moreover',  
            'whenever',  
            'whom',
```

'enough',  
'at',  
'within',  
'whither',  
'for',  
'amount',  
'm',  
's',  
'such',  
'put',  
'two',  
'm',  
'nor',  
'itself',  
'afterwards',  
'many',  
'us',  
'which',  
'third',  
'him',  
'behind',  
'perhaps',  
'hereupon',  
'through',  
'six',  
'keep',  
'wherever',  
'onto',  
're',  
'next',  
'anywhere',  
'how',  
'nothing',  
'of',  
'everything',  
'seem',  
'become',  
'ourselves',  
'sometimes',  
'seeming',  
'no',  
'almost',  
'whereby',  
'sometime',  
've',  
'serious',  
'really',  
'forty',  
'one',  
'various',  
'by',  
'meanwhile',  
'between',  
'nine',  
's',  
'she',  
'alone',  
'with',  
'still',  
'so',

'might',  
'several',  
'used',  
'top',  
'per',  
'something',  
'do',  
'most',  
'the',  
'see',  
'their',  
'when',  
'that',  
'latter',  
'go',  
'otherwise',  
'herein',  
'least',  
'if',  
'former',  
'beside',  
'mostly',  
'must',  
'too',  
'indeed',  
'another',  
'due',  
'already',  
'this',  
'myself',  
'yourselves',  
'once',  
'noone',  
'during',  
'back',  
'even',  
'd',  
'however',  
'nobody',  
'both',  
'none',  
'll',  
'whatever',  
'from',  
've',  
'never',  
'made',  
'hundred',  
'doing',  
'beyond',  
'everywhere',  
'elsewhere',  
'among',  
'less',  
'our',  
'say',  
'fifteen',  
'please',  
'over',  
'its',



"'ll",  
'everyone',  
'give',  
'seemed',  
'much',  
'i',  
'them',  
'done',  
'twelve',  
'then',  
'each',  
'not',  
'all',  
'via',  
'were',  
'get',  
'front',  
'hers',  
'around',  
'somewhere',  
're',  
'make',  
'bottom',  
'beforehand',  
'are',  
'towards',  
'while',  
'hence',  
've',  
'upon',  
'd",  
'it',  
'again',  
'sixty',  
'or',  
're",  
'his',  
's",  
'and',  
'unless',  
'whereafter',  
'any',  
'those',  
'have',  
'neither',  
'five',  
'against',  
'was',  
'thereby',  
'fifty',  
'also',  
'four',  
'first',  
'side',  
'can',  
'call',  
'thus',  
'anyone',  
'her',  
'yours',

'who',  
'always',  
'thereafter',  
'you',  
'name',  
'besides',  
'has',  
'a',  
'eight',  
'throughout',  
'be',  
'whereas',  
'full',  
'after',  
'yourself',  
'as',  
'under',  
'more',  
'where',  
'becoming',  
'very',  
'an',  
'we',  
'someone',  
'he',  
'been',  
'part',  
'quite',  
'just',  
'well',  
'would',  
'down',  
'own',  
'wherein',  
'me',  
'until',  
'above',  
'eleven',  
'being',  
'now',  
'into',  
'rather',  
're',  
'n't',  
'although',  
'they',  
'whence',  
'there',  
'else',  
'show',  
'anything',  
'regarding',  
'm',  
'whether',  
'without',  
'cannot',  
'whole',  
'further',  
'somehow',  
'either',

```
'to',
'am',
'three',
'did',
'mine',
'along',
'herself',
'because',
'often',
'became',
'twenty',
'same',
'below',
'themselves',
'other',
'in',
'toward',
'whoever',
'but',
'these',
'before',
'11',
'on',
'my',
'except',
'whereupon']
```

```
In [130... len(stopwords)
```

```
Out[130... 326
```

```
In [131... nlp = spacy.load('en_core_web_sm')
```

```
In [132... text
```

```
Out[132... 'There 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). 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.\nImage 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 '
```

```
In [133... doc = nlp(text)
doc
```

```
Out[133... There 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 d
ocuments, or sets of images, or videos, news stories etc.). The second is query
relevant summarization, sometimes called query-based summarization, which summa
rizes objects specific to a query. Summarization systems are able to create bot
h query relevant text summaries and generic machine-generated summaries dependi
ng 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 othe
rs can use multiple source documents (for example, a cluster of articles on the
same topic). This problem is called multi-document summarization. A related app
lication is summarizing news articles. Imagine a system, which automatically pu
lls together news articles on a given topic (from the web), and concisely repre
sents the latest news as a summary.
Image collection summarization is another application example of automatic summ
arization. It consists in selecting a representative set of images from a large
r set of images.[4] A summary in this context is useful to show the most repres
entative images of results in an image collection exploration system. Video sum
marization is a related domain, where the system automatically creates a traile
r of a long video. This also has applications in consumer or personal videos, w
here one might want to skip the boring or repetitive actions. Similarly, in sur
veillance videos, one would want to extract important and suspicious activity,
while ignoring all the boring and redundant frames captured
```

```
In [134... # Lets get the tokens from text
tokens = [token.text for token in doc]
print(tokens)
#when we execute everything we created tokens from the text & not removed any of
```

[ 'There', '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', '.', '\n', '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', '.', '\n', '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']

In [135... tokens

```
Out[135... ['There',
            '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',  
'.',  
'\n',  
'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',  
'.',  
'\n',  
'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']
```

In [136... `len(tokens)`

Out[136... 322

In [137... `punctuation # also called as noisy characters`

Out[137... `'!\"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'`

In [138... `#we have to calculate the frequency of each and every word, how many time word`

```
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
```

In [139... `#print(word_frequencies)`

```
word_frequencies
```

```
Out[139... {'broadly': 1,
            'types': 1,
            'extractive': 1,
            'summarization': 11,
            'tasks': 1,
            'depending': 2,
            'program': 1,
            'focuses': 2,
            'generic': 3,
            'obtaining': 1,
            'summary': 4,
            'abstract': 2,
            'collection': 3,
            'documents': 2,
            'sets': 1,
            'images': 3,
            'videos': 3,
            'news': 4,
            'stories': 1,
            'etc': 1,
            'second': 1,
            'query': 4,
            'relevant': 2,
            'called': 2,
            'based': 1,
            'summarizes': 1,
            'objects': 1,
            'specific': 1,
            'Summarization': 1,
            'systems': 1,
            'able': 1,
            'create': 1,
            'text': 1,
            'summaries': 2,
            'machine': 1,
            'generated': 1,
            'user': 1,
            'needs': 1,
            '\n': 2,
            'example': 3,
            'problem': 2,
            'document': 4,
            'attempts': 1,
            'automatically': 3,
            'produce': 1,
            'given': 2,
            'interested': 1,
            'generating': 1,
            'single': 1,
            'source': 2,
            'use': 1,
            'multiple': 1,
            'cluster': 1,
            'articles': 3,
            'topic': 2,
            'multi': 1,
            'related': 2,
            'application': 2,
            'summarizing': 1,
            'Imagine': 1,
```

```
'system': 3,  
'pulls': 1,  
'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}
```

```
In [140... len(word_frequencies)
```

```
Out[140... 103
```

```
In [141... word_frequencies
```

```
Out[141... {'broadly': 1,
            'types': 1,
            'extractive': 1,
            'summarization': 11,
            'tasks': 1,
            'depending': 2,
            'program': 1,
            'focuses': 2,
            'generic': 3,
            'obtaining': 1,
            'summary': 4,
            'abstract': 2,
            'collection': 3,
            'documents': 2,
            'sets': 1,
            'images': 3,
            'videos': 3,
            'news': 4,
            'stories': 1,
            'etc': 1,
            'second': 1,
            'query': 4,
            'relevant': 2,
            'called': 2,
            'based': 1,
            'summarizes': 1,
            'objects': 1,
            'specific': 1,
            'Summarization': 1,
            'systems': 1,
            'able': 1,
            'create': 1,
            'text': 1,
            'summaries': 2,
            'machine': 1,
            'generated': 1,
            'user': 1,
            'needs': 1,
            '\n': 2,
            'example': 3,
            'problem': 2,
            'document': 4,
            'attempts': 1,
            'automatically': 3,
            'produce': 1,
            'given': 2,
            'interested': 1,
            'generating': 1,
            'single': 1,
            'source': 2,
            'use': 1,
            'multiple': 1,
            'cluster': 1,
            'articles': 3,
            'topic': 2,
            'multi': 1,
            'related': 2,
            'application': 2,
            'summarizing': 1,
            'Imagine': 1,
```

```
'system': 3,
'pulls': 1,
'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}
```

```
In [142... max_frequency = max(word_frequencies.values())
max_frequency
```

```
Out[142... 11
```

```
In [143... #to get normalized/weighted frequencies you should divide all frequencies with 1
for word in word_frequencies.keys():
    word_frequencies[word] = word_frequencies[word]/max_frequency
```

```
In [144... #print(word_frequencies)
word_frequencies
#this is the normalized frequencies of each word
```

```
Out[144...] {'broadly': 0.09090909090909091,
'types': 0.09090909090909091,
'extractive': 0.09090909090909091,
'summarization': 1.0,
'tasks': 0.09090909090909091,
'depending': 0.18181818181818182,
'program': 0.09090909090909091,
'focuses': 0.18181818181818182,
'generic': 0.2727272727272727,
'obtaining': 0.09090909090909091,
'summary': 0.36363636363636365,
'abstract': 0.18181818181818182,
'collection': 0.2727272727272727,
'documents': 0.18181818181818182,
'sets': 0.09090909090909091,
'images': 0.2727272727272727,
'videos': 0.2727272727272727,
'news': 0.36363636363636365,
'stories': 0.09090909090909091,
'etc': 0.09090909090909091,
'second': 0.09090909090909091,
'query': 0.36363636363636365,
'relevant': 0.18181818181818182,
'called': 0.18181818181818182,
'based': 0.09090909090909091,
'summarizes': 0.09090909090909091,
'objects': 0.09090909090909091,
'specific': 0.09090909090909091,
'Summarization': 0.09090909090909091,
'systems': 0.09090909090909091,
'able': 0.09090909090909091,
'create': 0.09090909090909091,
'text': 0.09090909090909091,
'summaries': 0.18181818181818182,
'machine': 0.09090909090909091,
'generated': 0.09090909090909091,
'user': 0.09090909090909091,
'needs': 0.09090909090909091,
'\n': 0.18181818181818182,
'example': 0.2727272727272727,
'problem': 0.18181818181818182,
'document': 0.36363636363636365,
'attempts': 0.09090909090909091,
'automatically': 0.2727272727272727,
'produce': 0.09090909090909091,
'given': 0.18181818181818182,
'interested': 0.09090909090909091,
'generating': 0.09090909090909091,
'single': 0.09090909090909091,
'source': 0.18181818181818182,
'use': 0.09090909090909091,
'multiple': 0.09090909090909091,
'cluster': 0.09090909090909091,
'articles': 0.2727272727272727,
'topic': 0.18181818181818182,
'multi': 0.09090909090909091,
'related': 0.18181818181818182,
'application': 0.18181818181818182,
'summarizing': 0.09090909090909091,
'Imagine': 0.09090909090909091,
```



```
'system': 0.2727272727272727,  
'pulls': 0.09090909090909091,  
'web': 0.09090909090909091,  
'concisely': 0.09090909090909091,  
'represents': 0.09090909090909091,  
'latest': 0.09090909090909091,  
'Image': 0.09090909090909091,  
'automatic': 0.09090909090909091,  
'consists': 0.09090909090909091,  
'selecting': 0.09090909090909091,  
'representative': 0.18181818181818182,  
'set': 0.18181818181818182,  
'larger': 0.09090909090909091,  
'images.[4': 0.09090909090909091,  
'context': 0.09090909090909091,  
'useful': 0.09090909090909091,  
'results': 0.09090909090909091,  
'image': 0.09090909090909091,  
'exploration': 0.09090909090909091,  
'Video': 0.09090909090909091,  
'domain': 0.09090909090909091,  
'creates': 0.09090909090909091,  
'trailer': 0.09090909090909091,  
'long': 0.09090909090909091,  
'video': 0.09090909090909091,  
'applications': 0.09090909090909091,  
'consumer': 0.09090909090909091,  
'personal': 0.09090909090909091,  
'want': 0.18181818181818182,  
'skip': 0.09090909090909091,  
'boring': 0.18181818181818182,  
'repetitive': 0.09090909090909091,  
'actions': 0.09090909090909091,  
'Similarly': 0.09090909090909091,  
'surveillance': 0.09090909090909091,  
'extract': 0.09090909090909091,  
'important': 0.09090909090909091,  
'suspicious': 0.09090909090909091,  
'activity': 0.09090909090909091,  
'ignoring': 0.09090909090909091,  
'redundant': 0.09090909090909091,  
'frames': 0.09090909090909091,  
'captured': 0.09090909090909091}
```

In [145...

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

Out[145... [There 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]

In [146... `len(sentence_tokens)`

Out[146... 15

In [147... *# 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()]`

In [148... `sentence_scores`

Out[148... {There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on.: 2.818181818181818,  
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.): 3.99999999999999987,  
The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.: 3.909090909090909,  
Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.: 3.27272727272716,  
An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document.: 3.9999999999999996,  
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): 2.545454545454545,  
This problem is called multi-document summarization.: 1.8181818181818183,  
A related application is summarizing news articles.: 1.0909090909090908,  
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.: 2.9090909090909087,  
Image collection summarization is another application example of automatic summarization.: 2.909090909090909,  
It consists in selecting a representative set of images from a larger set of images.[4]: 1.1818181818181817,  
A summary in this context is useful to show the most representative images of results in an image collection exploration system.: 1.818181818181818,  
Video summarization is a related domain, where the system automatically creates a trailer of a long video.: 2.2727272727272725,  
This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions.: 1.1818181818181817,  
Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured: 1.4545454545454544}

```
In [149... #lets say our case study was 30% sentence with maximum scores  
from heapq import nlargest
```

```
In [150... select_length = int(len(sentence_tokens)*0.4)  
select_length
```

Out[150... 6

```
In [151... #we have to select maximum 4 sentences out of all sentences  
summary = nlargest(select_length,sentence_scores, key = sentence_scores.get)
```

```
In [152... summary
```

```
Out[152... [An example of a summarization problem is document summarization, which attempt
s to automatically produce an abstract from a given document.,
The first is generic summarization, which focuses on obtaining a generic summa
ry or abstract of the collection (whether documents, or sets of images, or vide
os, news stories etc.).,
The second is query relevant summarization, sometimes called query-based summa
rization, which summarizes objects specific to a query.,
Summarization systems are able to create both query relevant text summaries an
d generic machine-generated summaries depending on what the user needs.,
Image collection summarization is another application example of automatic sum
marization.,
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.]
```

```
In [153... # if i need to combine these top 3 sentences then

final_summary = [word.text for word in summary]
```

```
In [154... final_summary
```

```
Out[154... ['An example of a summarization problem is document summarization, which attempt
ts to automatically produce an abstract from a given document.',
'The first is generic summarization, which focuses on obtaining a generic summ
ary or abstract of the collection (whether documents, or sets of images, or vid
eos, news stories etc.).',
'The second is query relevant summarization, sometimes called query-based summ
arization, which summarizes objects specific to a query.',
'Summarization systems are able to create both query relevant text summaries a
nd generic machine-generated summaries depending on what the user needs.\n',
'Image collection summarization is another application example of automatic su
mmarization.',
'Imagine a system, which automatically pulls together news articles on a given
topic (from the web), and concisely represents the latest news as a summar
y.\n']
```

```
In [155... print(summary) # we get the final summary by our model
```

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

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```