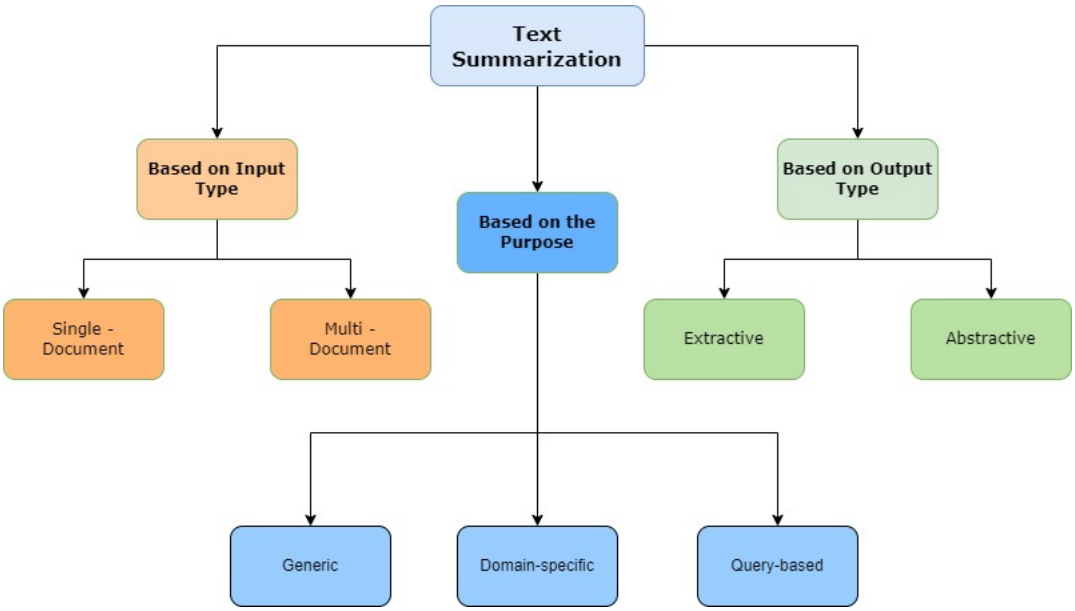
What is text summarization?

Text summarization is the process of creating a short, accurate, and fluent summary of a longer text document. It is the process of distilling the most important information from a source text. Automatic text summarization is a common problem in machine learning and natural language processing (NLP). Automatic text summarization methods are greatly needed to address the ever-growing amount of text data available online to both better help discover relevant information and to consume relevant information faster.

Why automatic text summarization?

- Summaries reduce reading time.
- While researching using various documents, summaries make the selection process easier.
- Automatic summarization improves the effectiveness of indexing.
- Automatic summarization algorithms are less biased than human summarizers.
- Personalized summaries are useful in question-answering systems as they provide personalized information.
- Using automatic or semi-automatic summarization systems enables commercial abstract services to - increase the number of text documents they are able to process.

Type of summarization



- An Extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form.
- An Abstractive summarization is an understanding of the main concepts in a document and then express those concepts in clear natural language.
- The Domain-specific summarization techniques utilize the available knowledge specific to the domain of text. For example, automatic summarization research on medical text generally attempts to utilize the various sources of codified medical knowledge and ontologies.
- The Generic summarization focuses on obtaining a generic summary or abstract of the collection of documents, or sets of images, or videos, news stories etc.
- The Query-based summarization, sometimes called query-relevant summarization, summarizes objects specific to a query.
- The Multi-document summarization is an automatic procedure aimed at extraction of information from multiple texts written about the same topic. Resulting summary report allows individual users, such as professional information consumers, to quickly familiarize themselves with information contained in a large cluster of documents.
- The Single-document summarization generates a summary from a single source document.

How to do text summarization

- Text cleaning
- Sentence Tokenization

- Word tokenization
- Word-frequency table
- Summarization

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

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

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

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

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

```
tokens = [token.text for token in doc]
print(tokens)
```

```
['\n', 'There', 'are', 'broadly', 'two', 'types', 'of', 'extractive', 'summarization', 'tasks', 'depending', 'on', 'what', 'the', '
```

```
punctuation = punctuation + '\n'
punctuation
```

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

```
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
```

```
print(word_frequencies)
```

```
{'broadly': 1, 'types': 1, 'extractive': 1, 'summarization': 11, 'tasks': 1, 'depending': 2, 'program': 1, 'focuses': 2, 'generic':
```

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

```
11
```

```
for word in word_frequencies.keys():
  word_frequencies[word] = word_frequencies[word]/max_frequency
```

```
print(word_frequencies)
```

```
{'broadly': 0.09090909090909091, 'types': 0.09090909090909091, 'extractive': 0.09090909090909091, 'summarization': 1.0, 'tasks': 0.
```

```
sentence_tokens = [sent for sent in doc.sents]
print(sentence_tokens)
```

```
[
There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on., The first is
, An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given
, Image collection summarization is another application example of automatic summarization., It consists in selecting a representat
]
```

```
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
```

```
{
 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.).: 3.9999999999999987,
 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.: 3.09090909090909,
 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.: 2.727272727272727,
 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.[3] A summary in this context is useful to show the most representative images of results in an image collection exploration system.: 2.999999999999999,
 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}
```

```
from heapq import nlargest
```

```
select_length = int(len(sentence_tokens)*0.33)
select_length
```

```
4
```

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

```
[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 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.]
```

```
final_summary = [word.text for word in summary]
summary = ' '.join(final_summary)
```

```
print(text)
```

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

```
print(summary)
```

```
An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given d
```

```
len(text)
```

```
1869
```

```
len(summary)
```

```
605
```