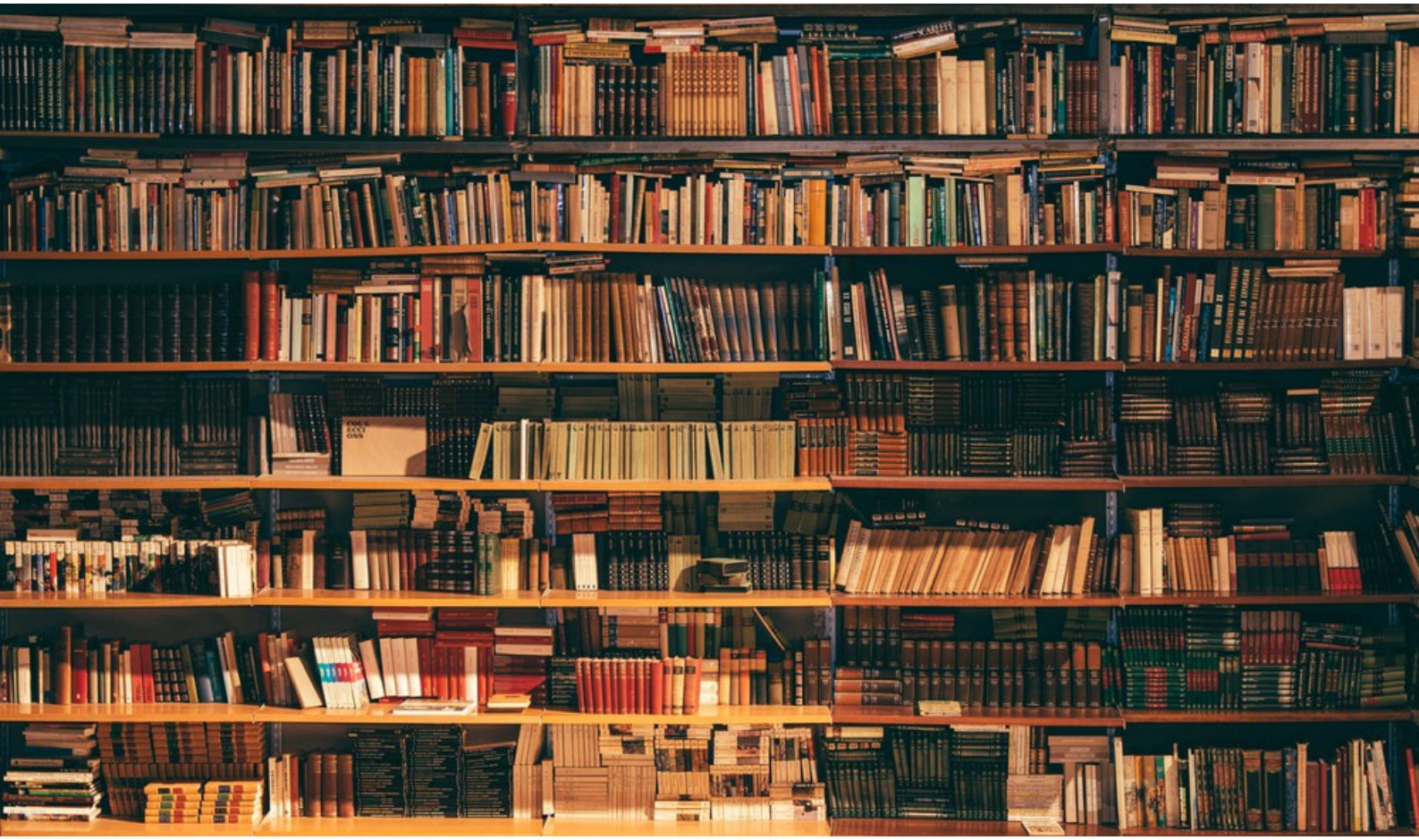
Understand Text Summarization and create your own summarizer in python

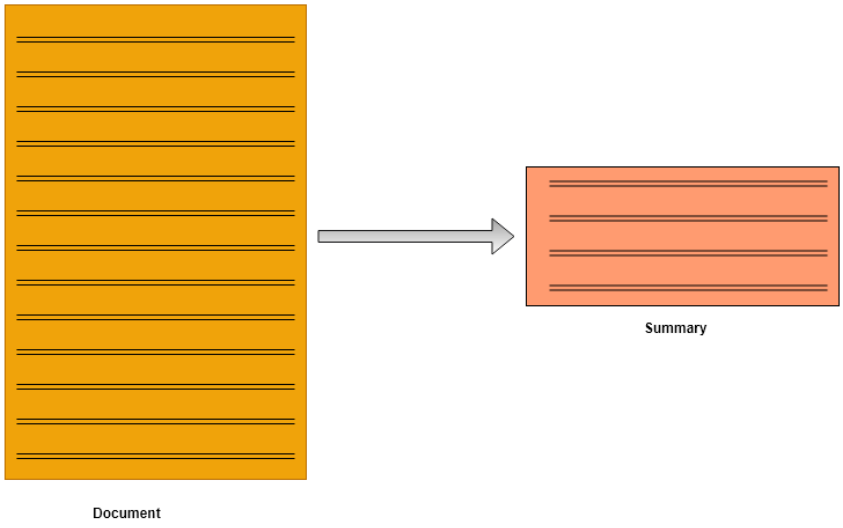
An Introduction to Text Summarization

Praveen Dubey Follow

Dec 23 · 7 min read



We all interact with applications which uses text summarization. Many of those applications are for the platform which publishes articles on daily news, entertainment, sports. With our busy schedule, we prefer to read the summary of those article before we decide to jump in for reading entire article. Reading a summary help us to identify the interest area, gives a brief context of the story.



Summarization can be defined as a task of producing a concise and fluent summary while preserving key information and overall meaning.

Impact

Summarization systems often have additional evidence they can utilize in order to specify the most important topics of document(s). For example, when summarizing blogs, there are discussions or comments coming after the blog post that are good sources of information to determine which parts of the blog are critical and interesting.

In scientific paper summarization, there is a considerable amount of information such as cited papers and conference information which can be leveraged to identify important sentences in the original paper.

. . .

How text summarization works

In general there are two types of summarization, **abstractive** and **extractive** summarization.

1. **Abstractive Summarization:** Abstractive methods select words based on semantic understanding, even those words did not appear in the source documents. It aims at producing important material in a new way. They interpret and examine the text using advanced natural language techniques in order to generate a new shorter text that conveys the most critical information from the original text.

It can be correlated to the way human reads a text article or blog post and then summarizes in their own word.

Input document → understand context → semantics → create own summary.

2. Extractive Summarization: Extractive methods attempt to summarize articles by selecting a subset of words that retain the most important points.

This approach weights the important part of sentences and uses the same to form the summary. Different algorithm and techniques are used to define weights for the sentences and further rank them based on importance and similarity among each other.

Input document → sentences similarity → weight sentences → select sentences with higher rank.

The limited study is available for abstractive summarization as it requires a deeper understanding of the text as compared to the extractive approach.

Purely extractive summaries often times give better results compared to automatic abstractive summaries. This is because of the fact that abstractive summarization methods cope with problems such as semantic representation,
inference and natural language generation which is relatively harder than data-driven approaches such as sentence extraction.

There are many techniques available to generate extractive summarization. To keep it simple, I will be using an **unsupervised learning** approach to find the sentences similarity and rank them. One benefit of this will be, you don't need to train and build a model prior start using it for your project.

It's good to understand **Cosine similarity** to make the best use of code you are going to see. **Cosine similarity** is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. Since we will be representing our sentences as the bunch of vectors, we can use it to find the similarity among sentences. Its measures cosine of the angle between vectors. Angle will be **0** if sentences are similar.

All good till now..? Hope so :)

Next, Below is our code flow to generate summarize text:-

Input article → split into sentences → remove stop words → build a similarity matrix → generate rank based on matrix → pick top N sentences for summary.

Let's create these methods.

1. Import all necessary libraries

```
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx
```

2. Generate clean sentences

```
def read_article(file_name):
    file = open(file_name, "r")
    filedata = file.readlines()
    article = filedata[0].split(". ")
    sentences = []

   for sentence in article:
     print(sentence)
     sentences.append(sentence.replace("[^a-zA-Z]", "
").split(" "))
     sentences.pop()

    return sentences
```

3. Similarity matrix

This is where we will be using cosine similarity to find similarity between sentences.

```
def build_similarity_matrix(sentences, stop_words):
    # Create an empty similarity matrix
    similarity_matrix = np.zeros((len(sentences),
len(sentences)))

    for idx1 in range(len(sentences)):
        for idx2 in range(len(sentences)):
            if idx1 == idx2: #ignore if both are same
sentences
                continue
            similarity_matrix[idx1][idx2] =
sentence_similarity(sentences[idx1], sentences[idx2],
stop_words)

return similarity_matrix
```

4. Generate Summary Method

Method will keep calling all other helper function to keep our summarization pipeline going. Make sure to take a look at all `# Steps` in below code.

```
def generate_summary(file_name, top_n=5):
    stop_words = stopwords.words('english')
    summarize_text = []
```

```
    # Step 1 - Read text and tokenize
    sentences =  read_article(file_name)

    # Step 2 - Generate Similary Martix across sentences
    sentence_similarity_martix = build_similarity_matrix(sentences, stop_words)

    # Step 3 - Rank sentences in similarity martix
    sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_martix)
    scores = nx.pagerank(sentence_similarity_graph)

    # Step 4 - Sort the rank and pick top sentences
    ranked_sentence = sorted(((scores[i],s) for i,s in enumerate(sentences)), reverse=True)
    print("Indexes of top ranked_sentence order are ", ranked_sentence)

for i in range(top_n):
      summarize_text.append(" ".join(ranked_sentence[i][1]))

    # Step 5 - Offcourse, output the summarize texr
    print("Summarize Text: \n", ". ".join(summarize_text))
```

All put together, here is the complete code.

```python
#!/usr/bin/env python
# coding: utf-8

from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx

def read_article(file_name):
    file = open(file_name, "r")
    filedata = file.readlines()
    article = filedata[0].split(". ")
    sentences = []

    for sentence in article:
        print(sentence)
        sentences.append(sentence.replace("[^a-zA-Z]", " "))
    sentences.pop()

    return sentences

def sentence_similarity(sent1, sent2, stopwords=None):
    if stopwords is None:
        stopwords = []

    sent1 = [w.lower() for w in sent1]
    sent2 = [w.lower() for w in sent2]

    all_words = list(set(sent1 + sent2))

    vector1 = [0] * len(all_words)
    vector2 = [0] * len(all_words)

    # build the vector for the first sentence
    for w in sent1:
        if w in stopwords:
            continue
        vector1[all_words.index(w)] += 1

    # build the vector for the second sentence
    for w in sent2:
        if w in stopwords:
            continue
        vector2[all_words.index(w)] += 1

    return 1 - cosine_distance(vector1, vector2)

def build_similarity_matrix(sentences, stop_words):
    # Create an empty similarity matrix
    similarity_matrix = np.zeros((len(sentences), len(sente

```

```
    for idx1 in range(len(sentences)):
        for idx2 in range(len(sentences)):
            if idx1 == idx2: #ignore if both are same sente
```

. . .

Let's look at it in action.

The complete text from an article titled ***Microsoft Launches Intelligent Cloud Hub To Upskill Students In AI & Cloud Technologies***

```
In an attempt to build an AI-ready workforce,
Microsoft announced Intelligent Cloud Hub which has
been launched to empower the next generation of
students with AI-ready skills. Envisioned as a three-
year collaborative program, Intelligent Cloud Hub will
support around 100 institutions with AI
infrastructure, course content and curriculum,
developer support, development tools and give students
access to cloud and AI services. As part of the
program, the Redmond giant which wants to expand its
reach and is planning to build a strong developer
ecosystem in India with the program will set up the
core AI infrastructure and IoT Hub for the selected
campuses. The company will provide AI development
tools and Azure AI services such as Microsoft
Cognitive Services, Bot Services and Azure Machine
Learning.According to Manish Prakash, Country General
Manager-PS, Health and Education, Microsoft India,
said, "With AI being the defining technology of our
time, it is transforming lives and industry and the
jobs of tomorrow will require a different skillset.
This will require more collaborations and training and
working with AI. That's why it has become more
critical than ever for educational institutions to
integrate new cloud and AI technologies. The program
is an attempt to ramp up the institutional set-up and
build capabilities among the educators to educate the
workforce of tomorrow." The program aims to build up
the cognitive skills and in-depth understanding of
developing intelligent cloud connected solutions for
applications across industry. Earlier in April this
year, the company announced Microsoft Professional
Program In AI as a learning track open to the public.
The program was developed to provide job ready skills
to programmers who wanted to hone their skills in AI
and data science with a series of online courses which
featured hands-on labs and expert instructors as well.
This program also included developer-focused AI school
that provided a bunch of assets to help build AI
skills.
```

(source: analyticsindiamag.com)

and the summarized text with *2 lines* as an input is

```
Envisioned as a three-year collaborative program,
Intelligent Cloud Hub will support around 100
institutions with AI infrastructure, course content
and curriculum, developer support, development tools
and give students access to cloud and AI services. The
company will provide AI development tools and Azure AI
services such as Microsoft Cognitive Services, Bot
Services and Azure Machine Learning. According to
Manish Prakash, Country General Manager-PS, Health and
Education, Microsoft India, said, "With AI being the
defining technology of our time, it is transforming
lives and industry and the jobs of tomorrow will
require a different skillset.
```

As you can see, it does a pretty good job. You can further customized it to reduce to number to character instead of lines.

It is important to understand that we have used **textrank** as an approach to rank the sentences. TextRank *does not rely on any previous training data* and can work with any arbitrary piece of text. TextRank is a general purpose *graph-based ranking* algorithm for NLP.

There are much-advanced techniques available for text summarization. If you are new to it, you can start with an interesting research paper named Text Summarization Techniques: A Brief Survey

Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation is a much more detailed research paper which you can go through for better understanding.

Hope this would have given you a brief overview of text summarization and sample demonstration of code to summarize the text. You can start with the above research papers for advance knowledge and approaches to solve this problem.

The code shown here is available on my GitHub. You can download and play around with it.

. . .

You can follow me on Medium, Twitter, and LinkedIn, For any question, reach out to me on email (praveend806 [at] gmail [dot] com).