Summarize

of articles in

Python

using NLTK

**Summarize of Articles in**

**Python using NLTK**

**A PROJECT REPORT SUBMITTED TO,**

**GITAM UNIVERSITY**

**Visakhapatnam**

**Submitted in partial fulfilment of the requirements**

**For the award of the degree of**

**“MASTER OF BUSINESS ADMISTRATION”**

**Submitted by:**

**AMMAN NAIDU**

**Regd No. :121923901004**

**Under the esteemed guidance of**

**Sri. Divakar Allavarapu**

**(Assistant Professor)**

****

**GITAM INSTITUTE OF MANAGEMENT**

**GITAM UNIVERSITY**

**(U/s 3 of UGC Act )**

**VISAKHAPATANM**

**(2019-2020)**

**DECLARATION**

I, the undersigned, hereby declare that, the project titled **“Summarize of Articles in Python using NLTK”** submitted to GITAM Institute of Management, GITAM University for the award of the Masters of Business Administration (FinTech) is the original work done by us, under the guidance of Mr. Divakar Allavarapu, Assistant Professor, GITAM Institute of Management The empirical findings in this report are based on the data collected by us. It was not copied from any other project. The project has not been submitted to any Substitute/University for the award of any Diploma/Degree.

Date: AMMAN NAIDU

Place: - Visakhapatnam (121923901004)

CERTIFICATE

This is to certify that the project entitled, **“Summarize of Articles in Python using NLTK”,** is a Bonafide work done by **AMMAN NAIDU (121923901004**) and is submitted in partial fulfilment for the **Masters of Business Administration (Fintech**) of GITAM University. It has not beensubmitted for the award of anydiploma/degree in any other Institution/University.

Date: Mr. Divakar Allavarapu

Place: Visakhapatnam Assistant Professor

ACKNOWLEDGEMENT

I am thankful to Prof. Y. Gowtham Rao, Director, GITAM Institute of Management, GITAM University, Vishakhapatnam, for giving me an opportunity to learn and have a study on very important topic of management. I am thankful to Mr. Divakar Allavarapu, Program Coordinator for his guidance.

K. AMMAN NAIDU (121923901004)

Contents

[Introduction 7](#_Toc37941881)

[Types of text summarization 7](#_Toc37941882)

[1.Extraction-based summarization 7](#_Toc37941883)

[2. Abstraction-based summarization 8](#_Toc37941884)

[Executive Summary 9](#_Toc37941885)

[Execution procedure 10](#_Toc37941886)

[Source Code 11](#_Toc37941887)

[Conclusion 14](#_Toc37941888)

# Introduction

Text summarization is the technique for generating a concise and precise summary of voluminous texts while focusing on the sections that convey useful information, and without losing the overall meaning. Automatic text summarization aims to transform lengthy documents into shortened versions, something which could be difficult and costly to undertake if done manually.   
Machine learning algorithms can be trained to comprehend documents and identify the sections that convey important facts and information before producing the required summarized texts. For example, the image below is of [this news article](https://www.autosport.com/motogp/news/142779/marquez-calls-austin-crash-hard-to-understand) that has been fed into a machine learning algorithm to generate a summary.

# Types of text summarization

Text summarization is mainly classified into 2 types.

1. Extraction-based summarization
2. Abstraction-based summarization

## 1.Extraction-based summarization

In extraction-based summarization, a subset of words that represent the most important points is pulled from a piece of text and combined to make a summary.

Highlighter = Extractive-based summarization

In machine learning, extractive summarization usually involves weighing the essential sections of sentences and using the results to generate summaries. Different types of algorithms and methods can be used to find the weights of the sentences and then rank them according to their relevance and similarity with one another.

**Example**

**TEXT: Amman and Balaji** took a taxi to **attend** the night **party** in the city. While in the party, **Balaji** collapsed and was **rushed** to the **hospital**.

**SUMMARY: Amman and Balaji attend party. Balaji rushed hospital**

Extracted summary is composed of the words highlighted in bold, the results may not be grammatically accurate.

## 2. Abstraction-based summarization

In abstraction-based summarization, advanced deep learning techniques are applied to paraphrase and shorten the original document, just like humans do.

Pen = Abstraction-based summarization

Since abstractive machine learning algorithms can generate new phrases and sentences that represent the most important information from the source text, they can assist in overcoming the grammatical inaccuracies of the extraction techniques.

**Example**

**TEXT:** Amman and Balaji took a taxi to attend the night party in the city. While in the party, Balaji collapsed and was rushed to the hospital.

**SUMMARY:** Balaji was hospitalized after attending a party with Amman.

Abstraction performs better at text summarization, developing its algorithms requires complicated deep learning techniques and sophisticated language modelling. To generate plausible outputs, abstraction-based summarization approaches must address a wide variety of NLP problems, such as natural language generation, semantic representation, and inference permutation.

# Executive Summary

Project was objectified to creating a code to Summarize of Articles in Python using NLTK certain library are imported for this project like Heapq, BS4, URLLIB Request, Regular Expressions. Using BS4 and URLLIB Request we can get data and html code from website like Investopedia, Encyclopaedia etc after getting data using regular expressions, we must clean the data once the data is cleaned using Heapq the summary will be executed.

Project is creating an app that can reads an article of text and creating a summary for that article using NLTK via Natural processing Language(NLP). Identifying the problem that humans take more time to make smart decisions. My idea is identifying the gap where humans take more time to take decision So innovatively creating an app which make smart decisions easily instead of humans in less amount of time. Consequently, developing the app increases the scale of the app and grabbing the opportunity in future so the by-product for the app is revenue.

App can be used in finance in different sectors taking the stock previous data and can know the trend, pattern and moment of the stock (investment Decisions). As an Investor while investing in a company he has to know the analysis of the company here this app will suggest weather to invest or not based on the previous data. This app can be useful to the high Networth portfolio investors like Equity Traded Funds(ETF), Mutual Funds to make easy decision in short span of time to achieve profits. So here upgrading the stock market with AI/ML technology can sustain in market for the longer period and also scalability of customers increases for the app and finally the by-product revenue is generated.

# Execution procedure

Steps involved inSummarize of Articles in Python using NLTK

**Step1:** Collection of data from different websites through URL

**Step2:** Getting the HTML Parser to extract the paragraph from URL

**Step3:** Grouping all paragraph into one paragraph

**Step4:** Data cleaning using regular expression

**Step5:** tokenization of the entire paragraph using NLTK

**Step6:** Text Processing by removing Stop Words

**Step7:** Giving frequency to the words

**Step8:** Getting the summary

# Source Code

**from urllib import request**

**from bs4 import BeautifulSoup as bs**

The above URLLIB is a python library packages are for getting the data using URL. [Beautiful Soup](http://www.crummy.com/software/BeautifulSoup/) is a Python library for getting data out of HTML and XML files.

**url="https://www.investopedia.com/terms/d/deep-learning.asp"**

**htmldoc=request.urlopen(url)**

**object=bs(htmldoc, 'html.parser')**

**print(object)**

Based on url request html code will execute.

**paragraph=object.find\_all('p')**

**print(paragraph)**

from the above code all paragraph from obtained using find\_all(‘p’)

**allparagraph=""**

**for paragraphs in paragraph:**

**allparagraph += paragraphs.text**

**print(allparagraph)**

Here all paragraph is combined and formed into one single paragraph.

**import re**

**allparagraph\_cleandata=re.sub(r'\[[0-9]\*\]',' ', allparagraph)**

**allparagraph\_cleandata=re.sub(r'\s+',' ', allparagraph\_cleandata)**

**allparagraph\_cleandata=re.sub(r'[^a-zA-Z]', ' ', allparagraph\_cleandata)**

**allparagraph\_cleandata=re.sub(r'\s+', ' ', allparagraph\_cleandata)**

By importing re(regular Expressions) we are removing the quotations comma punctuations as data cleaning process.

**import nltk**

**token=nltk.sent\_tokenize(allparagraph\_cleandata)**

**print(allparagraph\_cleandata)**

here paragraph will break into sentence and sentence into words simple Tokenize means breaking

**nltk.download('stopwords')**

**word\_token=nltk.word\_tokenize(allparagraph\_cleandata)**

**print(word\_token)**

**stopwords=nltk.corpus.stopwords.words('english')**

**print(stopwords)**

obtaining the stopwords

**word\_frequencies = {}**

**for word in word\_token:**

**if word not in stopwords:**

**if word not in word\_frequencies.keys():**

**word\_frequencies[word] = 1**

**else:**

**word\_frequencies[word] += 1**

**print(word\_frequencies )**

**maximum\_frequency\_word =max(word\_frequencies.values())**

**for word in word\_frequencies.keys():**

**word\_frequencies[word]=(word\_frequencies[word]/maximum\_frequency\_word)**

**print(word\_frequencies)**

getting the weighted frequencies to the words

**sentence\_scores={}**

**for sentence in token:**

**for word in nltk.word\_tokenize(sentence.lower()):**

**if word in word\_frequencies.keys():**

**if (len(sentence.split(' '))) < 30:**

**if sentence not in sentence\_scores.keys():**

**sentence\_scores[sentence] = word\_frequencies[word]**

**else:**

**sentence\_scores[sentence] += word\_frequencies[word]**

**print(sentence\_scores)**

calculating the score for the sentence here we are taking sentence which are leaser than 30

**import heapq**

**summary=heapq.nlargest(10,sentence\_scores,key=sentence\_scores.get)**

**print(summary)**

**import heapq**

**summary=heapq.nlargest(40,sentence\_scores,key=sentence\_scores.get)**

**print(summary)**

This heapq is an implementation of the heap queue algorithm, also known as the priority queue algorithm. I will arrange the all the sentence into a meaningful order

# 

# Conclusion

A basic Code is created for Summarize of Articles in Python using this code in Education field while writing a thesis of a project has to add the references at the end so for that article has to completely read so if we use this app it will tell whether the article is related to thesis or not by taking key words. Also we can use in finance field , weather forecasting, market sentiment analysis.