Text Summarization App with Flask and Spacy

Last Updated : 03 Jul, 2024

SpaCy is an open-source library for advanced natural language processing in Python. It is perfect for both industrial and scholarly applications because it is made to process vast amounts of text efficiently. Pre-trained models for multiple languages are provided by SpaCy, making tasks like dependency parsing, named entity identification, and part-of-speech tagging possible. Its modular design makes it an adaptable option for developers, enabling smooth integration with other libraries and tools in the NLP ecosystem.

Flask App For Summarization using Advance NLP

The motive behind this project is to create and develop an application or model that can efficiently summarize a large textual article or text document. This, in

text. For all this, we require a basic knowledge of Flask, HTML, and NLP.

Steps for Creating a Text Summarizer App

Step 1: Create a virtual environment

Open Anaconda Navigator and Launch vs-code or open any other IDE like Pycharm. To create a virtual Environment write the following code in the terminal.

- *python -m venv <enviroment name>*
- *<enviroment name>\Scripts\activate*

```
> python -m venv myenv
> myenv\Scripts\activate
```

Write this line of codes on the terminal

Step2: Developing NLP/ML model for text summarization

app.py: The app.py begins by importing necessary libraries for web handling, form creation, and text processing, and initializes a Flask instance with a secret key for session management while loading the SpaCy English model for NLP tasks. It defines a Form class using Flask-WTF, featuring a text input field and a submit button with validation to ensure the field isn't empty.

The application also downloads essential NLTK resources (stopwords and punkt) for tokenization and stopword removal. The root route (/) of the web application creates an instance of the Form, checks if it has been submitted and validated, processes the input text using the prediction function to generate a summary if valid, and renders the home.html template, passing the form and summary for display.

remove_punc(text): This function starts by tokenizing the input text into individual sentences and then further breaks down each sentence into words. It filters out any punctuation marks from these words. After filtering, it reconstructs the sentences from the remaining words and finally returns the text devoid of punctuation.

remove_tags(text): This function defines a list of HTML tags to be removed. It then tokenizes the input text into sentences and further into words within each sentence. It filters out the specified HTML tags from these words. After filtering, the function reconstructs the sentences from the remaining words and returns the cleaned text.

remove_stpwrds(text): This function begins by loading a set of English stopwords. It then tokenizes the text into sentences and further into words within each sentence. The function filters out any stopwords from these words. After filtering, it reconstructs the sentences from the remaining words and returns the text without stopwords.

extract_keywords(text): This function processes the input text using SpaCy to obtain part-of-speech tags for each token. It then filters tokens based on specified tags (PROPN, ADJ, NOUN, VERB). Finally, it collects and returns the filtered keywords that meet the criteria.

summarize_text(text): This function preprocesses the input text by removing punctuation, HTML tags, and stopwords. It then extracts keywords from the cleaned text and calculates their frequency. The function normalizes the keyword frequencies and assigns a strength score to each sentence based on these frequencies. Finally, it selects and returns the top sentences with the highest scores as the summary.

Python

```python
from flask import Flask, render_template, request
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired
import spacy
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.tokenize import sent_tokenize
from heapq import nlargest
import string
from collections import Counter

app = Flask(__name__)
app.secret_key = 'b83a1e0ea4e74d22c5d6a3a0ff5e6e66'
nlp = spacy.load("en_core_web_sm")

class Form(FlaskForm):
     text = StringField('Enter the text', validators=[DataRequired()])
     submit = SubmitField('Submit')

nltk.download('stopwords')
nltk.download('punkt')

@app.route('/', methods=['GET', 'POST'])
def home():
    form=Form()
    pred= None

    if form.validate_on_submit():
        text=form.text.data
```

```
        pred=prediction(text)
    return render_template('home.html',form=form,pred=pred)

def prediction(text):
    # Function to remove punctuation from the text
    def remove_punc(text):
        new_sent = []
        for sent in sent_tokenize(text):
            words = word_tokenize(sent)
            new_word=[]
            for i in words:
                if i not in string.punctuation:
                    new_word.append(i)
            new_sent.append(' '.join(new_word))
        return ' '.join(new_sent)

    # Function to remove specific HTML tags from the text
    def remove_tags(text):
        br_tags=['<br>','']
        new_sent = []
        for sent in sent_tokenize(text):
            words = word_tokenize(sent)
            new_word=[]
            for i in words:
                if i not in br_tags:
                    new_word.append(i)
            new_sent.append(' '.join(new_word))
        return ' '.join(new_sent)

    # Function to remove stopwords from the text
    def remove_stpwrds(text):
        stop_words = set(stopwords.words('english'))
        new_sent = []
        for sent in sent_tokenize(text):
            words = word_tokenize(sent)
            new_word=[]
            for i in words:
                if i.lower() not in stop_words:
                    new_word.append(i)
            new_sent.append(' '.join(new_word))
        return ' '.join(new_sent)
```

```
    # Function to extract keywords from the text
    def extract_keywords(text):
        doc = nlp(text)
        keywords = []
        tags = ['PROPN', 'ADJ', 'NOUN', 'VERB']
        for token in doc:
            if token.pos_ in tags:
                keywords.append(token.text)
        return keywords

    # Function to summarize the text based on keyword frequency
    def summarize_text(text):
        doc = nlp(text)
        text = remove_punc(text)
        text = remove_tags(text)
        text = remove_stpwrds(text)
        keywords = extract_keywords(text)
        freq = Counter(keywords)
        max_freq = freq.most_common(1)[0][1]
        for i in freq.keys():
            freq[i] = freq[i] / max_freq

        sent_strength = {}

        for sent in doc.sents:
            for word in sent:
                if word.text in freq.keys():
                    if sent in sent_strength.keys():
                        sent_strength[sent] += freq[word.text]
                    else:
                        sent_strength[sent] = freq[word.text]

        summarized_sentences = nlargest(4, sent_strength, key=sent_strength.get)
        return summarized_sentences

    # Call the summarization function and return the result
    summary = summarize_text(text)
    return summary
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Step 3: Setting up GUI

home.html: The provided code sets up a Flask web application that allows users to input text and receive a summarized version in three key points, which are then displayed on a webpage. The form uses the POST method to submit data to the root URL (/). It includes a text area for input and a submit button.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Text Summarizer</title>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

    <style>
```

```
        .container {
            background-color: rgb(235, 235, 235);
            padding: 20px;
            border-radius: 10px;
            margin-top: 20px;
            color: rgb(11, 1, 10);
        }

        body {
            background-color: rgb(12, 12, 228);
            background-image: linear-gradient(to bottom right, rgb(66, 114, 186),rgb(73, 154, 198), rgb(92, 144, 149));
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column; /* Added */
        }

        .header {
            color: white;
            font-size: 60px;
            margin-bottom: 20px;
        }

        .line {
            width: 50%;
            height: 2px;
            background-color: white;
            margin-bottom: 20px;
        }
    </style>

</head>
<body>
    <div class="header">Text Summarizer</div>
    <div class="line"></div>


    <div class="container">


        <form method="POST" action="/">
```

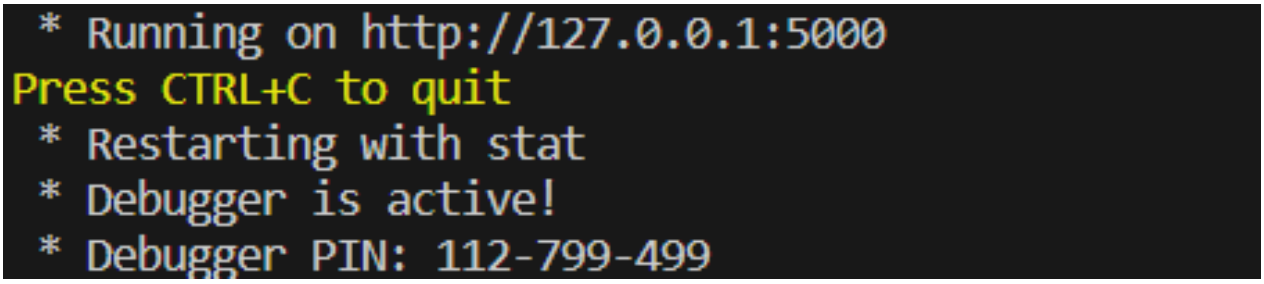
```
            <div class="form-group">
                <label for="text">Enter the text</label>
                <textarea class="form-control" id="text" name="text" rows="5" placeholder="Enter your text here" required>{{ text }}</textarea>
            </div>


            <button type="submit" class="btn btn-primary">Submit</button>

            {{ form.hidden_tag() }}
        </form>


        <!-- Content here -->
        {% if pred %}
        <h2>Summary:</h2>
        <ul>
            {% for sentence in pred %}
                <li>{{ sentence }}</li>
            {% endfor %}
        </ul>
        {% endif %}
      </div>


</body>
</html>
```

Output:

Step 4: Running the app on local host.

Line of code to run the flask app

Just write "python app.py" on the terminal and this would be generated.



Code running on local host

After that just click on the "http://127.0.0.1:5000" and you would be redirected to a webpage, which would the homepage of the application.

Output:

Looking to dive into the world of programming or sharpen your Python skills? Our **Master Python: Complete Beginner to Advanced Course** is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!

Previous Article

Scraping Reddit with Python and BeautifulSoup

Next Article

Similar Reads

Python | Extractive Text Summarization using Gensim

Summarization is a useful tool for varied textual applications that aims to highlight important information within a large corpus. With the outburst of...

3 min read

Python | Word Similarity using spaCy

Word similarity is a number between 0 to 1 which tells us how close two words are, semantically. This is done by finding similarity between word vectors in the...

2 min read

Python | Perform Sentence Segmentation Using Spacy

The process of deciding from where the sentences actually start or end in NLP or we can simply say that here we are dividing a paragraph based on sentences....

2 min read

How to Integrate Flask-Admin and Flask-Login

In order to merge the admin and login pages, we can utilize a short form or any other login method that only requires the username and password. This is know...

8 min read

Build a Text Translator Web App using Flask and Azure Cognitive Services

We're going to create a website that will translate text into multiple languages using Artificial Intelligence(AI). We'll use Flask framework for the Front end and...

7 min read

Documenting Flask Endpoint using Flask-Autodoc

Documentation of endpoints is an essential task in web development and being able to apply it in different frameworks is always a utility. This article discusses...

4 min read

How to use Flask-Session in Python Flask ?

Flask Session - Flask-Session is an extension for Flask that supports Server-side Session to your application.The Session is the time between the client logs in to...

4 min read

Minify HTML in Flask using Flask-Minify

Flask offers HTML rendering as output, it is usually desired that the output HTML should be concise and it serves the purpose as well. In this article, we would...

12 min read

Flask URL Helper Function - Flask url_for()

In this article, we are going to learn about the flask url_for() function of the flask URL helper in Python. Flask is a straightforward, speedy, scalable library, used f...

11 min read

How to Build a Web App using Flask and SQLite in Python

Python-based Flask is a microweb framework. Typically, a micro-framework has little to no dependencies on outside frameworks. Despite being a micro...

4 min read

Article Tags : Python Python Flask Flask Projects

Practice Tags : python

GeeksforGeeks Community
Android Tutorial
Tutorials Archive

DSA
Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML
Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial
Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps
Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design
High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation
Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects
Mathematics
Physics
Chemistry
Biology

GeeksforGeeks Videos
DSA
Python
Java
C++