Introduction to Text Summarization

Text summarization is the process of creating a shorter version of a longer text while retaining its core information. In this project, we explored the use of SpaCy for text summarization.

SpaCy: A Brief Overview

SpaCy is an open-source natural language processing library written in Python. It is designed specifically for production use and provides fast and efficient processing of large volumes of text data.

Some of the key features of SpaCy include:

- Tokenization
- Part-of-speech tagging
- Named entity recognition
- Dependency parsing
- Text classification

Text Summarization Techniques

Extractive Summarization

Extractive summarization involves selecting the most important sentences or phrases from the original text and creating a summary based on them. This technique is widely used as it is simple and easy to implement. It involves techniques such as sentence scoring, graph-based approaches, and clustering algorithms.

Abstractive Summarization

Abstractive summarization involves generating a summary that is not present in the original text. This technique is more complex and involves natural language generation techniques, such as deep learning models, to generate summaries that are more human-like. However, this technique is still in its early stages of development and requires large amounts of data and computing power.

Future Work

Enhancing Model Performance

There is a scope of improving the performance of the text summarization model by experimenting with different neural network architectures and hyperparameters.

Multi-Document Summarization

Future work could focus on developing a multi-document summarization model that could summarize large volumes of text from multiple sources into a concise summary.

Real-Time Summarization

Developing a real-time summarization model that could summarize news articles, social media posts, and other text data as soon as they are published could be a potential area of future work. This could be useful in scenarios where quick decision-making is required based on large volumes of text data.

Connecting Machine Learning Back-end to Flask

Flask

Flask is a lightweight web framework for Python that allows you to easily build web applications. It is particularly well-suited for building APIs and web services, which makes it a great choice for connecting your machine learning back-end to a web server.

Connecting to the Back-end

To connect to your machine learning back-end, you can use Flask's built-in HTTP client to make requests to your API endpoints. You can also use Flask's request object to handle incoming requests and pass data to your back-end for processing.

Calculating Word Frequency

This code calculates the word frequency in a document and stores it in a dictionary, then divides each word frequency by the maximum frequency to normalize the values.

Selecting the Summary

This code snippet uses nlargest and sent_scores to create a summary of a text, with a length of 30% of the original text. It then prints the summary and returns the summary, the original text, and their respective lengths.

Thank You!

Room-Number 419
Project-Name:- Text Summarization with Spacy

Team details

K.Pavan sai kumar
P.vijay
M.Siva Gopi
B.vishnuvardan
E mani kanta
M. Lakshmi Nadh

Git-hub link of project:-https://github.com/KIET-NEST-PROJECTS-KBH/E-4F-419

Linkedin link:-