**DOCUMENTATION FOR FEW LINES OF CODE**

Climate change refers to long-term shifts in temperatures and weather patterns. Such shifts can be natural, due to changes in the sun’s activity or large volcanic eruptions. But since the 1800s, human activities have been the main driver of climate change, primarily due to the burning of fossil fuels like coal, oil and gas. Burning fossil fuels generates greenhouse gas emissions that act like a blanket wrapped around the Earth, trapping the sun’s heat and raising temperatures. The main greenhouse gases that are causing climate change include carbon dioxide and methane. These come from using gasoline for driving a car or coal for heating a building, for example. Clearing land and cutting down forests can also release carbon dioxide. Agriculture, oil and gas operations are major sources of methane emissions. Energy, industry, transport, buildings, agriculture and land use are among the main sectors causing greenhouse gases.

Climate change research involves analyzing vast amounts of scientific literature, reports, and news articles. Keyword extraction helps researchers quickly identify and extract key terms and phrases related to climate change. This enables them to understand the most relevant topics and concepts within the field.

At its core, this code leverages the prowess of cutting-edge technology and the vast repository of human language. By utilizing the Hugging Face platform and their keyphrase extraction model, it taps into a sophisticated algorithm trained to identify significant words and phrases within a given text.

Let's delve into the intricacies of this code. It begins by defining a function named "keyword\_extraction" that takes a text input as its parameter. Within this function, a list called "keywords" is initialized, acting as a repository for the extracted keyphrases. We also set the API URL, providing the pathway to access the keyphrase extraction model.

To authenticate our access to the model, we define the necessary headers, including an authorization token. This ensures that we have the necessary permissions to interact with the model and retrieve the desired information.

The magic happens with the "query" function, which facilitates communication with the API. This function constructs a payload consisting of the text input and additional options, such as waiting for the model to be ready before generating the output. The payload is then sent via a POST request to the API URL, accompanied by the headers. The response we receive is in the form of a JSON object, containing the extracted keyphrases.

The code then proceeds to iterate through the output, extracting the desired information. It identifies the value associated with the key "word" in each iteration and appends it to the "keywords" list. In case an exception occurs, such as the absence of the "word" key, the code gracefully continues to the next iteration.

To eliminate duplicate entries and ensure a concise representation, the code converts the "keywords" list into a set and then back into a list, storing the resulting unique keyphrases in a variable called "keyphrases."

Finally, a message is printed, indicating the successful execution of the code.

Ladies and gentlemen, this code exemplifies the power of automated text analysis and information retrieval. It has diverse applications, ranging from data integration and knowledge organization to sentiment analysis and policy-making. By harnessing the capabilities of natural language processing, we can unlock the hidden gems within vast amounts of textual data and gain insights that would have otherwise remained concealed.

In conclusion, the "keyword\_extraction" code represents a gateway to a wealth of knowledge and understanding. Its ability to extract keyphrases from text opens doors to enhanced decision-making, improved information retrieval, and deeper comprehension of complex topics. Let us embrace this code and explore the endless possibilities it offers in the realm of natural language processing.

!git clone https://github.com/petermr/semanticClimate.git

!pip install pandas

!pip install numpy

%pip install transformers

import torch

import os

import pandas as pd

import json

import requests

import numpy as np

print('\033[1;32m We have successfully finished running this cell.')

text = “”

def keyword\_extraction(text):

keywords = []

API\_URL = "https://api-inference.huggingface.co/models/ml6team/keyphrase-extraction-kbir-inspec"

headers = {"Authorization": "Bearer hf\_IRdcHKWETBdPHwNGBUKWxjcEzUSQFpYamD"}

def query(payload):

response = requests.post(API\_URL, headers=headers, json=payload)

return response.json()

output = query({

"inputs": text,

"options": "wait\_for\_model=true"

})

print(output)

for f in output:

try:

word = (f['word'])

keywords.append(word)

except:

pass

keyphrases = [\*set(keywords)]

return keyphrases

print('\033[1;32m We have successfully finished running this cell.')

key=keyword\_extraction(text)

print(key)

df = pd.read\_csv('/content/semanticClimate/paragraphLinking/total\_pages\_groups\_table.csv')

df["anchor\_keywords"] = df["anchor\_text"].apply(lambda x: keyword\_extraction(x))

df["target\_keywords"] = df["target\_text"].apply(lambda x: keyword\_extraction(x))

df.to\_csv('/content/total\_pages\_groups\_table.csv', index=None)

print('\033[1;32m We have successfully finished running this cell.')

Rake keywords

TF- IDF keywords

Token classifiers

N- grams

Ml algorithm in NLP

Token classification model