Text Summarization for Wikipedia Articles

1. Описание на проекта:

Разработката представлява AI модел, който като входни данни ще получава URL на статия в Wikipedia, а като изходни ще изкарва сравнително къс текстов низ под формата на обобщение на текста (в рамките на няколко изречения).

2. Описание на използваните технологии (модели, frameworks и т.н.):

Този проект е разработен на Python и е базиран на техники за обработка на естествени езици (NLP или Natural Language Processing). За имплементацията ще бъде използван Python пакет NLTK (Natural Language Toolkit), който се използва за обработка на естествени езици.)

Проекта използава още следните технологии:

- **bs4 (BeautifulSoup):** За парсване и обработка на HTML/XML документи.
- **urllib:** За отваряне и четене на URL адреси.
- **re (Regular expressions):** За обработка на текст, премахване на числови референции и други.
- **nltk (Natural Language Toolkit):** За токенизация, работа със стоп думи и анализ на текста.
- **heapq:** За избиране на най-важните изречения според техните оценки.

Ще бъде използван и DataSet съдържащ StopWords(думи, които обикновено се игнорира в текста):
https://www.kaggle.com/datasets/heeraldedhia/stop-words-in-28-languages

3. Архитектура и/или описание на техническата разработка:

Процесът на работа включва следните стъпки:

1. Потребителят въвежда URL адрес на статия от Wikipedia.
2. Програмата използва urllib.request за отваряне и четене на съдържанието на статията.
3. Съдържанието се обработва с помощта на BeautifulSoup за извличане на текст от HTML документа.
4. Текстът се форматира и подлага на различни обработки (например, премахване на числови референции и други символи).
5. Използват се nltk.sent_tokenize и nltk.word_tokenize за токенизация на изречения и думи.
6. Стоп думите за български и английски се зареждат от външни файлове и комбинират в общ списък от стоп думи.
7. Изчислява се честотата на всяка дума и нормализира спрямо най-честата дума.
8. За всяко изречение се изчислява оценка за важността му базирана на честотата на думите в него.
9. Използва се модулът heapq за избиране на най-важните изречения.
10. Генерира се текстово резюме от избраните изречения.

4. Изходен код на разработката:

```
# За локален старт трябва да се инсталират:
# $ pip install beautifulsoup4
# $ pip install lxml
# $ pip install nltk
# Можете също да се използва "pip3", ако "pip" не работи
```

4.1 Извличане на статията от Wikipedia чрез подаден URL

```
import bs4 as bs          # библиотека за обработка на HTML/XML
import urllib.request     # библиотека за отваряне и четене на URL адреси
import re                 # Regular expressions библиотека

# Въведеният URL адреса на Wikipedia трябва да бъде във формата 'https://'
userLink = input("Коя статия в Wikipedia бихте искали да резюмирам? Въведете URL: ")
```

```
# Отваря връзката към URL адрес и изтегля съдържанието му
raw_data = urllib.request.urlopen(userLink)
# Чете съдържанието на статията след като е било изтеглено
document = raw_data.read()

# Използва се BeautifulSoup, за да се анализира HTML документаи
#създаде обект, с който лесно може да се манипулира HTML съдържанието
parsed_document = bs.BeautifulSoup(document,'lxml')

# Използва BeautifulSoup, за да открие всички параграфи (<p>) в HTML документа
article_paras = parsed_document.find_all('p')
# Инициализиране на променлива, която ще съдържа текста на статията
article_text = ""

# Итерира през всички параграфи и добавя текста им към променливата article_text
for para in article_paras:
    article_text += para.text

print(article_text[:1000])
```

4.2 Почистване на текста

```
# Форматиране на текста,така че да бъдат премахнати числовите референции
# в квадратни скоби и да бъдат замени всички последователности от празни
# пространства (включително нови редове) с едно единствено интервал
article_text = re.sub(r'\[[0-9]*\]', ' ',  article_text)
article_text = re.sub(r'\s+', ' ',  article_text)

# Форматиране на текста,така че да съдържа само буквени символи и
# интервали, а всички други символи да са заменени с интервали
formatted_text = re.sub('[^a-яA-Яа-zA-Z]', ' ', article_text)
formatted_text = re.sub(r'\s+', ' ', formatted_text)

print(formatted_text[:1000])
```

4.3 Откриване на повтарящи се думи

```
import nltk

def get_stopwords_list(stop_file_path):
    """Load stop words"""
    with open(stop_file_path, 'r', encoding="utf-8") as f:
        stopwords = f.readlines()
        stop_set = set(m.strip() for m in stopwords)
    return stop_set
```

```
# Зареждане на стоп думите
stopwords_bg =
get_stopwords_list("/kaggle/input/stop-words-in-28-languages/bulgarian.txt")
stopwords_en =
get_stopwords_list("/kaggle/input/stop-words-in-28-languages/english.txt")

# Обединяване на стоп думите за двата езика
stopwords_combined = stopwords_bg.union(stopwords_en)
stop_words = nltk.corpus.stopwords.words('english')

# Punkt tokenizer е модел за токенизация, който е разработен от
# NLTK и се използва за разделяне на текст на отделни токени
# (думи или интерпункционни знаци)

# Токенизация на текста - разделя текста на изречения, използвайки
# модула за токенизация на изречения от библиотеката NLTK. Този метод
# използва предварително обучен модел, който разпознава пунктуационни знаци
# и ги използва за разделяне на текста на отделни изречения.
all_sentences = nltk.sent_tokenize(formatted_text)
for sentence in all_sentences:
    words = nltk.word_tokenize(sentence)

# Създава празен речник, в който ще се съхранява броят на срещанията на всяка дума
word_frequency = {}

for word in words:
    # За всяка дума (след като е конвертирана в малки букви) проверява дали
    # не е "stopword" (дума, която обикновено се игнорира в текста);
    if word.lower() not in stop_words:
        # Проверява се дали думата се съдържа в речника
        # Ако думата не  съществува в речника, я добавя със стойност 1
        if word not in word_frequency.keys():
            word_frequency[word] = 1
        # Ако думата съществува в речника, увеличава броя в речника с 1
        else:
            word_frequency[word] += 1

# Намира максималната честота на дума в документа
max_freq = max(word_frequency.values())

# За всяка дума преизчислява нормализираната честота, т.е. в диапазон от 0 до 1,
# където 1 представлява най-честата дума
for word in word_frequency.keys():
    word_frequency[word] = (word_frequency[word]/max_freq)
```

4.4 Обработване на резултатите от изреченията

```
# Създава празен речник, в който ще се съхраняват оценките за всяко изречение
sentence_scores = {}

# За всяко изречение кодът токенизира текста на думи, преобразува ги в малки
# букви и след това итерира през всеки токен. Ако токенът присъства в речника
# word_frequency (който съдържа честотата на всяка дума), кодът проверява
# дължината на изречението. Ако дължината на изречението е по-малка от 25
#думи, той обновява оценката за това изречение в речника sentence_scores.

# Този процес създава оценки за всяко изречение въз основа на честотата на
# съдържащите се в него думи и условие за дължина на изречението.
all_sentences = nltk.sent_tokenize(article_text)
for sentence in all_sentences:
    for token in nltk.word_tokenize(sentence.lower()):
        if token in word_frequency.keys():
            if len(sentence.split(' ')) < 25:
                if sentence not in sentence_scores.keys():
                    sentence_scores[sentence] = word_frequency[token]
                else:
                    sentence_scores[sentence] += word_frequency[token]
```

4.5 Извеждане на резултата

```
import heapq

# Избират се 5-те най-високо оценени изречения от речника sentence_scores
selected_sentences= heapq.nlargest(5, sentence_scores, key=sentence_scores.get)

# Създаване на текстово резюме
text_summary = ' '.join(selected_sentences)
print(text_summary)
```

5. Пример за начин на работа:

Вход:

Коя статия в Wikipedia бихте искали да резюмирам? Въведете URL:

```
https://bg.wikipedia.org/wiki/%D0%9A%D1%83%D1%87%D0%B5
```

Изход:

При използване на куче за домашен любимец, децата се учат на отговорност и внимание, на което кучетата отговарят с любов и преданост. Използването на кучета като впрегатни и товарни животни в тези култури често продължава да съществува дори и след въвеждането на коня в Северна Америка. По време на фазата на почивка далакът събира червени кръвни клетки и животното може да диша интензивно, за да се охлади. Не е рядкост кучетата да нападат хора или други животни; това обаче в повечето случаи се дължи на неправилно отглеждане от страна на стопанина. Така например се смята, че заселването на Америка от хората би било невъзможно без използването на впрегатни кучета за пресичането на Беринговия проток.