1. Формулировка на задачата

Задачата, която ни е възложена, е да реализираме програма, която при получен текст ни връща по-съкратена версия на него, съдържащата само най-важните неща или накратно, резюме на текст.

2. Използвани алгоритми

• *Предварителна обработка на текста*: -

- *Tokenization*: Използва функциите nltk.tokenize.word_tokenize и nltk.tokenize.sent_tokenize съответно за токенизиране на думи и изречения.

- *Lemmatization*: Прилага функцията nltk.stem.WordNetLemmatizer, за да сведе думите до тяхната основна или коренна форма, като подпомага групирането на различни форми на една и съща дума.

• *Анализ на настроенията*:

- *Анализът на настроенията* се извършва с помощта на библиотеката `TextBlob`. TextBlob предоставя прост API за общи задачи за обработка на естествен език (NLP), включително анализ на настроенията. За обобщението се създава обект `TextBlob`, а неговият атрибут `sentiment` се използва за получаване на полярността (позитивност/негативност) и оценката за субективност.

• *Обобщаване на текст*: - Скриптът използва механизъм за оценяване на изреченията въз основа на наличието на важни думи:

- *TF-IDF Score*: Функцията `score_sentence()` изчислява оценка за всяко изречение, като отчита броя на важните думи в изречението и го нормализира по броя на думите.

- *Multiprocessing*: За да се ускори процесът, кодът използва библиотеката `multiprocessing` за паралелизиране на точкуването на изреченията, което го прави по-ефективен.

• *Моделиране на теми*:

- *Latent Dirichlet Allocation (LDA)*: Библиотеката `gensim` се използва за прилагане на LDA за моделиране на теми. LDA е вероятностен модел, който приписва теми на документи и думи на теми. В този случай той се използва за откриване на теми в токенизирания и изчистен текст.

• *Анализ на честотата на думите*:

- *Counter*: Класът `Counter` от модула `collections` се използва за преброяване на срещите на всяка дума в текста. След това тази информация се използва за генериране на хистограма на честотата на думите.

• *Записване на обобщение във файл*:

- *File I/O*: Кодът предлага на потребителя да запише обобщението във файл. Ако потребителят избере да запише резюмето, се записва в текстов файл, като се използват операциите за въвеждане/извеждане на файлове на Python.

• *Разпознаване на езика*:

- *langdetect*: Библиотеката `langdetect` се използва за автоматично откриване на езика на входния текст. Това е полезно за персонализиране на списъка със стоп думи въз основа на открития език.

• *Построяване на хистограма на честотата на думите*:

- *Matplotlib*: Модулът `matplotlib.pyplot` се използва за създаване на стълбовидна диаграма, представяща хистограмата на честотата на думите. Тази визуализация помага да се разбере разпределението на думите в текста.

• *Форматиране на дата и час*:

- *Модулът `datetime`* се използва за генериране на времева марка за името на файла по подразбиране, при запазване на обобщението. Това гарантира, че всяко запазено обобщение има уникално име на файла въз основа на текущата дата и час.

3. Описание на програмната реализация

1. Импротиране на библиотеките и изтегляне на ресурсите.

2. Предварителна обработка на текст. Приема се входният текст и езикът като параметри, от функцията ‘preprocess_text()’. Тази функция токенизира текста на изречения и лематизира всяка дума, като използва WordNet.

3. Оценяване на изреченията: функцията ‘score_sentence()’, която приема изречение и набор от важни думи като параметри. Тя изчислява броя на думите и оценява изречението въз основа на наличието на важни думи, като взема предвид честотата на думите.

4. Обобщаване на текст: Функцията ‘summarize()’ приема входния текст, опцията за обобщаване(short,medium,long) и езика като параметри. Тази функция извършва обработката на текста, оценява важността на всяко изречение и генерира резюмето с избраната от нас дължина, използвайки 'multiprocessing' за ефективност.

5. Sentiment analysis - Функцията `sentiment_analysis()` използва TextBlob, за да анализира настроенията на предоставения текст. Тя връща полярността и субективността на настроенията.

6. Моделиране на теми: Функцията ‘topic_modeling()’ токенизира и почиства текста, след като прилага Latent Dirichlet Allocation (LDA) с помощта на Genism. Тази функция връща идентифицираните теми от модела на LDA.

7. Анализ на честотата на думите: Чрез функцията ‘plot_word_frequency()’ се генерира хистограма на честотата на думите въз основа на входния текст като използва Matplotlib, за да визуализира най-важните думи и тяхната честота.

8. Записване на обобщението във файл: Функцията ‘save_summary_to_file()’ предлага на потребителя избора да запази резюмето си във файл, като при съгласие файлът се записва с име по подразбиране или с такова, което избере потребителят и се запазва в същата директория. Името по подразбиране сме го направили да бъде датата и часът, в момента на запазване до ***секундата***, за да може всяко едно запазване на програмата да бъде само по себе си с уникално име и да няма дубликати.

9. Main block: Кодът първоначално чете входния текст от файла, който сме му подали (в нашия случай той е с име ‘*input.txt*’). Той открива езика на този текст, използвайки langdetect след което идва ред на избраната от потребителя големина на резюмето (short, medium, long). След това функцията ‘summarize()’ генерира резюмето благодарение на отркития език и зададената от потребителя големина. ‘sentiment_analysis()’ прави анализ на настроението върху направеното резюме, изчислява се и се отпечатват статистически данни за оригиналния текст като например дължина, дължина на резюмето и процент на редуциране. Най-накрая генерираното съобщение се записва във файл с избрано от потрбителя име или default името.

4. Примери, илюстриращи работата на програмната система

```
import nltk
import os
import multiprocessing
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk import ne_chunk, pos_tag
from langdetect import detect
from functools import partial
from gensim import corpora, models
from textblob import TextBlob
from langdetect import detect
import string
import matplotlib.pyplot as plt
from collections import Counter
from datetime import datetime

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('wordnet')
```

```python
def preprocess_text(text, language):
    sentences = sent_tokenize(text)
    lemmatizer = WordNetLemmatizer()

    # Extend stopwords list with common but less informative words
    extended_stopwords = set(stopwords.words(language)) if language in stopwords.fileids() else set()
    extended_stopwords.update(["the", "to", "a", "of", "in", "and", "is", "for", "on", "with", "that", "this", "it"])

    lemmatized_sentences = []
    for sentence in sentences:
        words = word_tokenize(sentence)
        lemmatized = [lemmatizer.lemmatize(word) for word in words if word.isalpha()]
        filtered = [word for word in lemmatized if word.lower() not in extended_stopwords]
        lemmatized_sentences.append(' '.join(filtered))

    return sentences, lemmatized_sentences
```

```python
def score_sentence(sentence, important_words):
    word_count = len([word for word in word_tokenize(sentence.lower()) if word.isalpha()])
    word_score = sum(1 for word in word_tokenize(sentence.lower()) if word in important_words)
    return sentence, word_score / max(word_count, 1)

def summarize(text, mode='medium', language='english'):
    sentences, important_words = preprocess_text(text, language)
    pool = multiprocessing.Pool()
    score_partial = partial(score_sentence, important_words=important_words)
    sentence_scores = dict(pool.map(score_partial, sentences))
    pool.close()
    pool.join()

    summary_length = {'short': 0.25, 'medium': 0.5, 'long': 0.75}
    num_sentences = int(len(sentences) * summary_length.get(mode, 0.5))

    ranked_sentences = sorted(sentence_scores, key=sentence_scores.get, reverse=True)
    summary = " ".join(ranked_sentences[:num_sentences])
    return summary
```

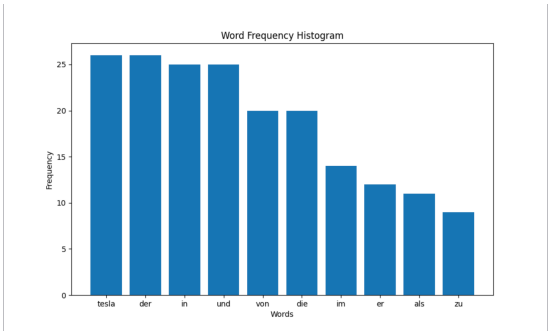
```python


def sentiment_analysis(text):
    analysis = TextBlob(text)
    return analysis.sentiment

def topic_modeling(text, num_topics=5, num_words=3):
    # Tokenize and clean text
    tokenized_text = [word_tokenize(doc.lower()) for doc in sent_tokenize(text)]
    tokenized_text = [[word for word in doc if word.isalpha() and word not in string.punctuation] for doc in tokenized_te

    dictionary = corpora.Dictionary(tokenized_text)
    corpus = [dictionary.doc2bow(doc) for doc in tokenized_text]

    # Apply LDA model
    lda = models.ldamodel.LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=10)
    topics = lda.print_topics(num_words=num_words)
    return topics
```

```python
def plot_word_frequency(text, num_words=10, file_name='word_frequency_histogram.png'):
    words = word_tokenize(text)
    words = [word.lower() for word in words if word.isalpha()]
    frequency = Counter(words)
    most_common = frequency.most_common(num_words)

    words, counts = zip(*most_common)
    plt.figure(figsize=(10, 6))
    plt.bar(words, counts)
    plt.title("Word Frequency Histogram")
    plt.xlabel("Words")
    plt.ylabel("Frequency")

    # Save plot to a file instead of showing it
    plt.savefig(file_name)
    print(f"Histogram saved as {file_name}")
```

```python
def save_summary_to_file(summary, default_filename):
    while True:
        user_input = input("Do you want to save the summary to a file? (y/n): ").strip().lower()
        if user_input in ['y', 'yes']:
            filename = input(f"Enter filename (default: {default_filename}): ").strip()
            filename = filename if filename else default_filename

            with open(filename, 'w', encoding='utf-8') as file:
                file.write(summary)
            print(f"Summary saved to {filename}")
            break
        elif user_input in ['n', 'no', '']:
            print("Summary not saved.")
            break
```

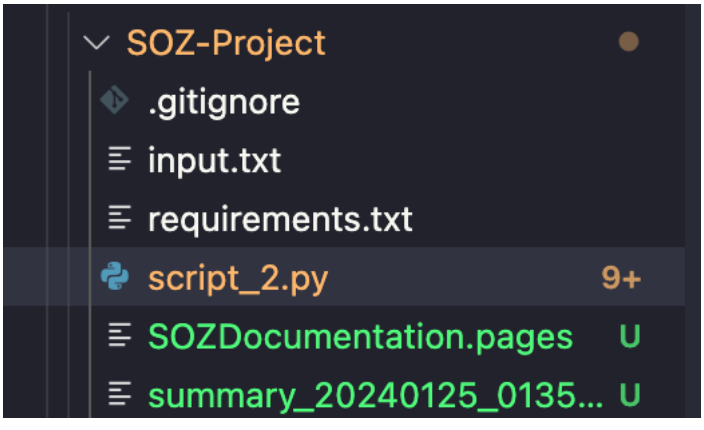
```python
if __name__ == "__main__":
    with open('input.txt', 'r', encoding='utf-8') as file:
        text = file.read()

    detected_language = detect(text)
    print(f"Detected Language: {detected_language}")

    valid_modes = ['short', 'medium', 'long']
    mode = input("Enter the summary mode (short, medium, long): ").lower()

    if mode not in valid_modes:
        print(f"Invalid mode selected. Defaulting to 'medium'.")
        mode = 'medium'

    summary = summarize(text, mode=mode, language=detected_language)
    print("\nSummary:\n", summary)

    # Sentiment Analysis
    sentiment = sentiment_analysis(summary)
    print("\nSentiment Analysis:")
    print("Polarity: ", sentiment.polarity)
    print("Subjectivity: ", sentiment.subjectivity)

    # Print summary and statistics
    original_len = len(text)
    summary_len = len(summary)
    reduction_percentage = ((original_len - summary_len) / original_len) * 100

    print(f"\nOriginal Text Length: {original_len} characters")
    print(f"Summary Length: {summary_len} characters")
    print(f"Reduction: {reduction_percentage:.2f}%")

    # Save summary to a file
    current_time = datetime.now().strftime("%Y%m%d_%H%M%S")
    default_filename = f"summary_{current_time}.txt"
    save_summary_to_file(summary, default_filename)
```

```
martinmenchev@Martins-MacBook-Air SOZdom % source venv/bin/activate
(venv) martinmenchev@Martins-MacBook-Air SOZdom % cd SOZ-Project
(venv) martinmenchev@Martins-MacBook-Air SOZ-Project % python3 script_2.py
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/martinmenchev/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/martinmenchev/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/martinmenchev/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /Users/martinmenchev/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     /Users/martinmenchev/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/martinmenchev/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Detected Language: de
Enter the summary mode (short, medium, long): medium
```




```
Summary:
 Tesla wurde als viertes von fünf Kindern serbischstämmiger Eltern in dem Dorf Smiljan in der Lika unweit von Gospić im heutigen Kroatien g
eboren. Seine Eltern waren der serbisch-orthodoxe Priester Milutin Tesla (1819–1879) und dessen Frau Georgina (Rufname Đuka, geborene Mandi
ć, 1822–1892). Das Geburtshaus war das Pfarrhaus der Sankt-Peter-und-Paul-Kirche von Smiljan. [2] Im Taufregister der orthodoxen Kirche wur
de das Geburtsdatum noch nach dem julianischen Kalender eingetragen. [3] Er hatte drei Schwestern und einen älteren Bruder, der aber schon
 im Alter von zwölf Jahren bei einem Reitunfall starb, als Nikola fünf Jahre alt war. Tesla besuchte die Grundschule und die Mittelschule in
 Gospić und ab 1870 das Gymnasium in Karlovac. Während seiner Gymnasialzeit lebte er bei seiner Tante Stanka und deren Mann Dane Branković,
 einem pensionierten Oberst. 1875 nahm er ein Studium Generale an der Kaiserlich-Königlichen Technischen Hochschule in Graz auf und belegte
 im ersten Jahr überdurchschnittlich viele Vorlesungen. Am 21. Juli 1876 erhielt er die Zugangsberechtigung zum Hauptstudium in Maschinenba
u. Beim Physikprofessor Jakob Pöschl lernte er die Gramme-Maschine kennen, einen damals neuartigen Gleichstromgenerator von Zénobe Gramme.
Im zweiten Studienjahr nahm seine Studienaktivität deutlich ab. Im dritten Studienjahr schloss er keine Prüfung ab und wurde schließlich vo
n der Hochschule 1877/78 exmatrikuliert, nachdem er das Unterrichtsgeld nicht bezahlt hatte. [4]

Erste berufliche Anstellungen

Reisepass von Nikola Tesla (1883)
Tesla zog nach Marburg an der Drau, wo er eine Anstellung als Maschinenbauer fand. Seine Freizeit verbrachte er als Karten- und Billardspie
ler in einschlägigen Lokalen. Am 24. März 1879 wurde er per polizeilicher Anordnung aus Marburg verwiesen und in seine Heimatgemeinde Gospi
ć zurückgeschickt. Einen Monat später, im April 1879, starb sein Vater. Nach dessen Tod blieb Tesla zunächst in Gospić und nahm eine Anstel
lung als Aushilfslehrer an. 1880 ging Tesla mit finanzieller Unterstützung durch seinen Onkel Dane Branković nach Prag, um an der dortigen,
 damals deutschsprachigen Karls-Universität sein Studium abzuschließen. Allerdings ist weder der Besuch noch der Abschluss der von ihm besu
chten Vorlesungen belegt; auch die notwendigen Studiengebühren wurden nie bezahlt. Von 1881 bis 1882 lebte Tesla in Budapest, wo er 1882 ei
ne Anstellung als Telegrafenamtstechniker bei Tivadar Puskás fand, der zu jener Zeit Repräsentant der Firmen von Thomas Alva Edison in Euro
pa war. Mit einer Empfehlung von Puskás zog Tesla Ende 1882 nach Paris zu Charles Batchelor, der eine der führenden Edison-Firmen in Frankr
eich betrieb. Neben weiteren Tätigkeiten betreute Tesla von November 1883 bis Februar 1884 die neu installierte elektrische Beleuchtung am
Gare de l'Est in Paris. [5]


Nikola Tesla um ca. 1885
Umzug nach New York
Am 6. Juni 1884 zog Tesla praktisch ohne Finanzmittel nach New York.

Sentiment Analysis:
Polarity:  0.13636363636363635
Subjectivity:  0.45454545454545453

Original Text Length: 6744 characters
Summary Length: 2946 characters
Reduction: 56.32%
Do you want to save the summary to a file? (y/n): y
Enter filename (default: summary_20240125_013553.txt):
Summary saved to summary_20240125_013553.txt
```

Тъй като е добра практика на програмния език Python, желателно е проектът да се пусне от virtual environment, заради факта, че има доста външни библиотеки за използване. Точно заради тази цел сме направили файла 'requirements.txt' , в който сме написали всички dependencies на всяка една от използваните от нас библиотеки. Ако нямате някоя от библиотеките, няма да тръгне самата задача. Точно заради това с virtual environment всичко става много лесно.



То трябва да се пусне и като се нагласи да бъде в същата директория се пише в него pip install -r requirments.txt и то тегли всичко необходимо, за да тръгне нашето приложение. Ако нямате virtual environment готов за ползване, той се тегли изключително лесно и бързо. В терминала

на VSC(Visual Studio Code), след като се намирате в директорията, която е и проекта, се пише python -m venv venv (ако това не стане пробрайте python3 -m venv venv). След като се инсталира, за да се пусне пишем пак в терминала в същата директория “ .\venv\Scripts\activate ”. След като тръгне се пише, както е споменато по-горе - “pip install -r requirements.txt “ и след това се пуска програмата от терминала: python script_2.py или python3 script_2.py.

5. Литература

https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f

https://machinelearningmastery.com/gentle-introduction-text-summarization/