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.

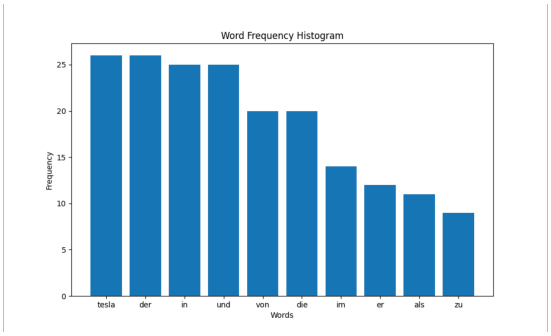
5. Sentiment analysis - Функцията `sentiment_analysis()` използва TextBlob, за да анализира настроенията на предоставения текст. Тя връща полярността и субективността на настроенията.

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 откритията 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.



Пример за хистограмата.