МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА БІБЛІОТЕКИ УЗАГАЛЬНЕННЯ ТЕКСТУ ЗАСОБАМИ PYTHON»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1219

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

А.Є. Тимофєєва
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри загальної математики, доцент, к.ф.-м.н. Стєганцев Є. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент
Лісняк А.О.
(підпис)

« » 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ (СТУДЕНТЦІ)

Тимофєєвій Анастасії Євгенівні
(прізвище, ім'я та по батькові)

1. Тема роботи Розробка бібліотеки узагальнення тексту засобами Python

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н.
(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » 05 2020 року № 576-с

2. Строк подання студентом роботи 30.11.2020

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація програмного забезпечення
4. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	04.06.2020	Виконано
2.	Збір вихідних даних.	15.06.2020	Виконано
3.	Обробка методичних та теоретичних	20.07.2020	Виконано
	джерел.		
4.	Розробка першого та другого розділу.	31.08.2020	Виконано
5.	Розробка третього розділу.	15.10.2020	Виконано
6.	Оформлення та нормоконтроль	30.11.2020	Виконано
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	16.12.2020	Виконано

Студент _____ А.Є. Тимофєєва
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О. В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка бібліотеки узагальнення тексту засобами Python»: 55 с., 28 рис., 10 табл., 38 джерел.

ОБРОБКА ПРИРОДНОЇ МОВИ, АБСТРАКТНІ МЕТОДИ, ЕКСТРАТИВНІ МЕТОДИ, ЧАСТОТНИЙ СЛОВНИК, МІШОК СЛІВ.

Об’єкт дослідження: процес автоматичного узагальнення тексту.

Мета роботи: розробка бібліотеки узагальнення текстів засобами екстракційних алгоритмів.

Методи дослідження – об'єктно орієнтований аналіз, методи програмної інженерії, методи аналізу даних.

У кваліфікаційній роботі розглядаються поширені методи для розв’язання задачі автоматичного узагальнення тексту, наведено їх переваги та недоліки. Розглянуто основні поняття «стемінг», «лематизація», «стоп-слова». На основі цього матеріалу розроблена бібліотека з можливістю створення резюме тексту для двох мов. Результати можуть бути використані для створення анотацій та як модулі в системах обробки природньої мови.

SUMMARY

Master`s Qualifying Thesis «Development of the Python Library for Automatic Text Summarization»: 55 pages, 28 figures, 10 tables, 38 references.

NATURAL LANGUAGE PROCESSING, ABSTRCTIVE METHODS, EXTRACTIVE METHODS, WORD LIST BY FREQUENCY, BAG OF WORDS.

The object of the study is automatic text summarization process.

The aim of the study is to develop the text summarization library, by using extractive algorithms.

The methods of research are object oriented methods, software engineering methods, data analysis methods.

In a masters' qualifying paper common approaches for solving the text summarization task and their advantages and disadvantages are given. Also reviewed such concepts as «stemming», «lemmatization» and «stop words». Based on this theoretical material the text summarization library with two language supporting was developed. The results can be used for annotation creation and as subsystems for natural language processing systems.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

NLP	загальний напрям інформатики, штучного інтелекту та математичної лінгвістики. Вивчає проблеми комп'ютерного аналізу та синтезу природної мови (англ., Natural-language processing, NLP).
Частотний словник	словник, у якому кожне слово характеризується певним числом, що вказує на кількість вживань цього слова в обстеженому масиві текстів, тобто на його абсолютну частоту в цих текстах

ВСТУП

Автоматичне узагальнення тексту – це процес стислого опису вихідного тексту на природній мові.

Узагальнення текстів є досить актуальною областю досліджень, оскільки подібні системи можна використовувати для розширеного пошуку у великих колекціях текстових документів, наприклад, у базах наукових публікацій.

Автоматичне узагальнення тексту є однією з задач машинного навчання та обробки природної мови. Зазвичай відповідні програмні реалізації використовуються при розробці вебсайтів та у мобільних застосунках для створення стрічок новин та коротких резюме статей.

Метою даної роботи є розробка системи автоматичного узагальнення текстів на основі глибинних нейронних мереж та проведення обчислювальних експериментів з метою оптимізації гіперпараметрів при задовільній якості узагальнення.

Для досягнення мети визначено такі завдання:

а) провести огляд існуючих методів моделювання текстів;

б) провести аналіз підходів до узагальнення текстів;

в) спроектувати та розробити програмну бібліотеку узагальнення текстів;

г) провести обчислювальні експерименти з метою оптимізації гіперпараметрів системи.

Об’єктом дослідження є процес автоматичного узагальнення тексту.

Предметом дослідження є методи автоматичного узагальнення тексту.

У першому розділі проведено аналіз технологій для узагальнення тексту. У другому розділі наведено архітектуру та взаємодію компонентів системи. У третьому розділі представлено етапи реалізації та тестування моделі.

1 ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ ОБРОБКИ ТЕКСТОВИХ ДАНИХ

Сучасний етап розвитку інформаційних технологій характеризується досить значним поширенням програмного забезпечення автоматизованої обробки текстів на різних природних мовах. Сюди можна віднести застосунки для машинного перекладу, автоматичної генерації відповідей на запитання користувачів, інформаційного пошуку, розпізнавання мовлення у текст та зворотної генерації мови з тексту тощо. Теоретичною основою наведених практичних застосувань є дослідження на перетині таких областей як штучний інтелект, комп'ютерна лінгвістика, теорія графів, математична логіка та математична статистика. Методи обробки природної мови (англ. Natural-language processing, NLP) поєднують як глибинні нейронні мережі застосування, яких призвело до значного поліпшення якості машинного перекладу, так і ймовірнісні математичні моделі, наприклад, латентне розміщення Діріхле (англ. Latent Dirichlet allocation), яке використовується при тематичному моделюванні.

Незважаючи на різноманітність задач NLP та методів, можна виділити деякі загальні етапи аналізу текстів. Підрозділ 1.1 присвячено більш детальному огляду проблем та підходів до розробки застосунків обробки природньої мови. Огляд алгоритмів узагальнення текстів розглянуто у 1.2, а програмні бібліотеки які застосовуються при розробці систем аналізу текстових даних наведено у 1.3.

1.1 Загальні поняття обробки текстів

Найбільш загальним завданням напряму NLP є проектування та розробка систем голосової взаємодії між машиною та користувачем. Ця проблема включає наступні задачі:

а) машинний переклад. Забезпечення синтаксично, граматично та семантично коректного перекладу між двома мовами. Прикладом є відомий сервіс Google Translate, який дозволяє автоматично перекладати слова, фрази та web-сторінки з однієї мови на іншу;

б) розпізнавання мовлення. Трансляція мовлення у текст та генерація голосового повідомлення з тексту . Таку методику використано у застосунку Google Text-to-Speech. Цей застосунок використовує технологію «синтезу мовлення» (також відому як текст-у-мовлення) аби читати вголос текст з екрану;

в) довідкова система питання-відповідь (англ. Question-answering system). До цього класу можна віднести різноманітні чат-бот системи, які використовують такі фреймворки, як Rasa NLU (англ. Natural Language Understanding). Його головна мета – це перетворення введення користувача на природній мові в об'єкти, з якими може працювати програма. Також популярним для таких цілей є сервіс Dialogflow, що дозволяє створювати чат-ботів для різних платформ і мов на різних пристроях;

г) системи визначення емоційного забарвлення тексту/мовлення (англ. Sentiment analysis);

д) тематичне моделювання. Визначення текстів, які об’єднанні однією темою, контекстом;

е) аналіз схожості. Пошук однакових за семантикою текстів. Наприклад, пошук відповідей на схожі питання користувачів;

ж) класифікація та кластеризація текстів. Прикладом може бути задача визначення або підтвердження авторства;

з) узагальнення тексту. Підготовка короткого резюме, яке дає стислий опис статті або повідомлення.

Загальну схему аналізу тексту засобами NLP зображено на рисунку 1.1.

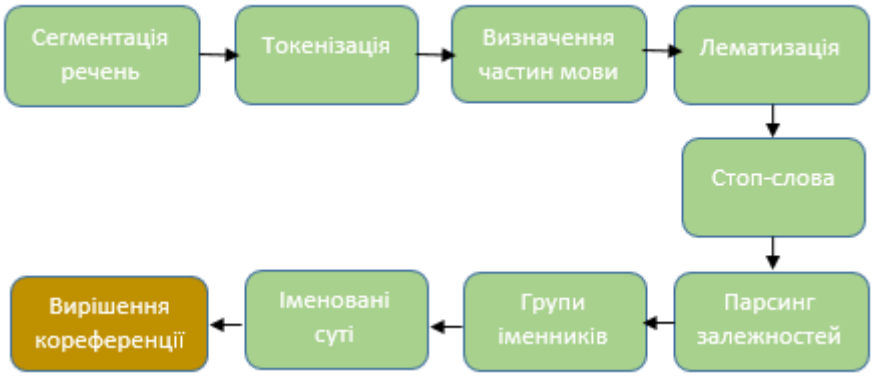

Рисунок 1.1 – Загальна схема аналізу тексту засобами NLP

Як видно з діаграми, основними етапами аналізу тексту є:

а) токенізація – це найперший крок при обробці тексту. Полягає у розбитті (поділу) довгих рядків тексту в більш дрібні: абзаци ділимо на речення, пропозиції на слова. Лематизація і стемінг мають на меті привести всі словоформи, що зустрічаються, до нормальної словникової форми;

б) стемінг – це грубий евристичний процес, який відрізає «зайве» від кореня слів, часто це призводить до втрати словотворчих суфіксів. Це свого роду нормалізація слів. Нормалізація – це метод, при якому набір слів у реченні перетворюється в послідовність, щоб скоротити час пошуку. Слова, які мають те саме значення, але деякі відмінності в залежності від контексту або речення, нормалізуються;

в) лематизація – це більш тонкий процес, який використовує словник і морфологічний аналіз, щоб в результаті привести слово до його канонічної форми – леми. Лематизація зазвичай відноситься до морфологічного аналізу слів, метою якого є видалення флективних закінчень. Це допомагає в поверненні базової або словникової форми слова.

г) стоп-слова – це слова, які викидаються з тексту до / після обробки тексту. Коли ми застосовуємо машинне навчання до текстів, такі слова можуть додати багато «шуму», тому необхідно позбавлятися від нерелевантних слів. Під стоп-словами зазвичай розуміють артиклі, вигуки, сполучники і т.д., які не

несуть смислового навантаження. При цьому треба розуміти, що не існує універсального списку стоп-слів, все залежить від конкретного випадку;

д) під моделюванням тексту розуміється метод переходу від символьного представлення до векторного або матричного числового представлення окремих слів, або тексту в цілому. Одними з найпростіших методів є представлення “мішок слів” та TF-IDF.

Алгоритми машинного навчання не можуть безпосередньо працювати з сирим текстом, тому необхідно конвертувати текст в набори цифр (вектори). Це називається витяганням ознак.

Мішок слів – це популярна і проста техніка вилучення ознак, яка використовується при роботі з текстом. Вона описує входження кожного слова в текст.

Щоб використовувати модель, нам потрібно:

- визначити словник відомих слів (токенов);
- вибрати ступінь присутності відомих слів.

Будь-яка інформація про порядок або структуру слів ігнорується. Ось чому це називається мішки слів. Ця модель намагається зрозуміти, чи зустрічається знайоме слово в документі, але не знає, де саме воно зустрічається. Складність цієї моделі в тому, як визначити словник і як підрахувати входження слів.

У частотного скорингу є проблема: слова з найбільшою частотністю мають, відповідно, найбільшу оцінку. У цих словах може бути не так багато інформаційного виграшу для моделі, як в менш частих словах. Один із способів виправити ситуацію – знижувати оцінку слова, яке часто зустрічається у всіх подібних документах. Це називається TF-IDF.

TF-IDF (скорочення від англ. term frequency – inverse document frequency) - це статистична міра для оцінки важливості слова в документі, який є частиною колекції або корпусу.

Скоринг по TF-IDF зростає пропорційно частоті появи слова в документі, але це компенсується кількістю документів, що містять це слово.

Низку методик мовного моделювання та навчання ознак в обробці природної мови (ОПМ), в яких слова або фрази зі словника відображують у вектори дійсних чисел називають вкладанням слів (англ. word embeeding). Існує три «класичні» варіанти вбудовування слів: Word2Vec, GloVe та FastText.

Дослідники з Google у своїй роботі [1] запропонували метод Word2Vec. У ньому використано невеликі нейронні мережі для обчислення вбудовування слів на основі контексту слів. Існує два підходи до реалізації цього методу - неперервна торба слів (англ. continuous bag-of-words, CBOW) та неперервний пропуск-грам. У першому випадку мережа намагається передбачити, яке слово найімовірніше з огляду на його контекст. Слова, які з однаковою ймовірністю з'являться, можна інтерпретувати як спільні. Якщо ми можемо замінити слово «кішка» на «собака» у реченні, цей підхід передбачає подібну ймовірність для обох. Тому ми робимо висновок, що значення цих слів є подібним принаймні на одному рівні.

Неперервний пропуск-грам має схожу ідею, але мережа працює навпаки. Тобто використовує цільове слово для прогнозування його контексту.

Дослідникі зі Стенфорду у своїй роботі [2] опублікували метод GloVe, основа мета якого сфокуватися на спільних зустрічах слів у всьому корпусі тексту. Його вкладені дані стосуються ймовірності того, що два слова з'являються разом.

У 2016 Facebook представив метод FastText, який покращує роботу Word2Vec, беручи до уваги також частини слів. Такий спосіб дає можливість тренувати вбудовування на менших наборах даних та узагальнення невідомих слів.

1.2 Аналіз публікацій з узагальнення текстів

Серед типових задач NLP наведених у підрозділі 1.1, задача узагальнення текстів виділяється тим, що для її вирішення можливе застосування як

аналітичних, так і синтетичних методів. Наприклад, при вилученні, ключових слів або речень та, відповідно, для генерації на їх основі зв'язного тексту резюме.

Як наслідок, усі алгоритми автоматичного узагальнення тексту (англ. automatic text summarization, ATS) можна класифікувати за двома основними підходами, а саме екстрактивним (англ. extractive) та абстрактним (англ. abstractive) узагальненням [3-5]. Вибір підмножини речень із вхідного тексту на основі певного алгоритму без будь-яких змін у структурі називається вилученням (extraction), тоді як генерування резюме, використовуючи нові слова, фрази й терміни, зберігаючи основну думку, що є у вхідних даних, називається абстракцією. Резюме, створене людиною, можна було б віднести саме до абстрактного підходу. Проте, більшість досліджень пов'язані з екстрактивним узагальненням через складність формування узгодженого та граматичного викладу правильних речень в абстрактному узагальненні.

Далі розглянемо кожен з підходів більш докладно.

Серед екстрактивних підходів найбільш поширеним є метод узагальнення на основі функцій (англ. Feature-Based Summarization). У роботах [6-8] описано, що важливість речення залежить від слів, які найчастіше зустрічаються у документі, тобто алгоритм вимірює частоту слів і фраз у документі та вирішує важливість речення з урахуванням слів у реченні та їх частоти. Це означає, що якщо речення має слова з найбільш високим показником частоти, то це речення у тексті є важливим. Використовуючи такий алгоритм, не беруться до уваги такі загальновживані частини мови, як прийменники, сполучники і т.д.

Екстрактивні алгоритми спочатку створюють проміжне подання (англ. intermediate representation), головне завдання якого – виділити найважливішу інформацію тексту на основі його подання. Існує два основних типи способів подання (англ. representations):

а) подання тем (англ. Topic representations) [9]. Основна увага приділяється темі, представленій у тексті. Існує кілька видів підходів для отримання цього подання:

1) підходи на основі частоти (англ. Frequency Driven Approaches) [10]: у цьому підході словам призначається вага. Якщо слово пов'язане з темою, призначається мітка 1, якщо не пов’язане – 0. Ваги можуть бути безперервними, залежно від реалізації. Дві загальноприйняті техніки подання тем:

- імовірність слів (англ. Word Probability) [11]: цей метод використовує частоту слів, як показник важливості слова. Імовірність слова w задається частотою зустрічей слова f (w), поділену на всі слова у вхідних даних, що має загальну кількість N слів;
- TFIDF. (Частота терміну, інверсія частоти документа (англ. Tern Frequency Inverse Document Frequency) [12, 13]: цей метод розроблений як вдосконалення методу імовірності слова. Тут використовується метод TF-IDF для присвоєння ваги. TFIDF – це метод, який призначає низьку вагу словам, які дуже часто трапляються у більшості документів, за логікою того, що вони є стоп-словами або словами типу "The". В іншому випадку, якщо слово з'являється в документі однозначно з високою частотою, йому надається велика вага.

2) тематичні слова (англ. Topic word Approaches): цей метод обчислює частоти слів і використовує поріг частоти, щоб знайти слово, яке потенційно може описати тему. Він класифікує важливість речення, як функцію кількості тематичних слів, які він містить [14, 15].

б) індикатори показників (англ. Indicator Representations) [11]. Цей тип подання залежить від особливостей речень і класифікує їх на основі ознак. Тобто, тут важливість речення залежить не від слів, які він містить, а безпосередньо від особливостей речення. Для цього типу подання існує два методи:

1) графічні методи (англ. Graph-Based Methods) [16, 17]. Базуються на алгоритмі Page Rank. Він представляє текстові документи, як пов'язані графи. Речення представлені, як вузли графів та ребра між вузлами, або речення є мірою подібності між двома реченнями;

2) методи машинного навчання (англ. Machine-Learning Methods) розглядають проблему узагальнення, як проблему класифікації. Моделі класифікують речення за їхніми особливостями на речення, які або відносяться до резюме, або ні. Для навчання моделей використовують навчальний набір документів та відповідні екстрактивні резюме, зроблені людиною. Зазвичай тут використовуються Naive Bayes [15, 18], дерево рішень [19, 20] та SVM [21].

Абстрактні методи у свою чергу виконують перефразування основного змісту тексту, використовуючи словниковий запас, відмінний від оригінального документа. Розробка такого роду узагальнювачів може бути складною, оскільки їм потрібно генерувати природню мову (англ. Natural Language Generation).

Так, у роботах [22, 23] автори використовують рекурентні автокодувальники з шаром “уваги” (англ. Attentional Encoder-Decoder Recurrent Neural Networks). Така модель призначена для створення вихідної послідовності слів із вхідної послідовності слів. Послідовність на вході у розглянутому випадку є фактичним текстовим документом, а вихід нейронної мережі – скороченим резюме.

Одним з підходів абстрактних методів є мережі кодерів та декодерів (англ. Encoder and Decoder Networks), які поєднуються у автокодувальник.

У роботах [24, 25] використовують LSTM (англ. Long term short memory) модель. Така модель або створює вихідні дані для кожного введення, або створює вектор ознак, який згодом використовується щільними нейронними мережевими шарами для надання класифікації із застосуванням шарів softmax. Наприклад, ми передаємо ціле речення через RNN і використовуємо вектори функцій, які підходять для шарів softmax для отримання кінцевого результату.

Автори роботи [26] запропонував метод Фейсбуку, який складається з трьох енкодерів:

а) енкодер мішок слів (англ. bag-of-words encoder): представляє вхідне речення, і ігнорує взаємозв'язок із сусідніми словами. Декодер приймає закодований вектор або мішок слів;

б) згортковий енкодер (англ. Convolutional encoder): згорткові шари використовуються для створення векторів ознак;

в) енкодер з використанням шарів “уваги” (Attention-based encoder).

У роботах [27, 28] запропоновано метод на основі Мережі генераторів покажчиків (англ. Pointer-Generator Networks) модель використовує механізм покриття, щоб зменшити проблему повторень мережі послідовності до послідовності.

1.3 Бібліотеки Python

Обробка природної мови знаходиться на стику науки про дані і сфери штучного інтелекту. Вона цілком присвячена навчанню машин розуміти людську мову і вилучати сенс з тексту. Оскільки NLP спирається на просунуті обчислювальні навички, розробникам потрібні найкращі доступні інструменти. Ці інструменти повинні допомогти отримати максимальну користь з підходів і алгоритмів NLP для створення сервісів, здатних працювати з природними мовами.

Для вирішення проблем NLP створено безліч бібліотек. Далі розглянемо найбільш корисні та багаті на функціонал.

Бібліотека NLTK підтримує такі задачі, як класифікація, стемінг, маркування, синтаксичний аналіз і семантичне міркування в Python. Це основний інструмент для обробки природної мови та машинного навчання. Сьогодні він служить освітньої базою для розробників Python, які тільки розпочинають вивчення NLP і машинного навчання.

Бібліотека була розроблена Стівеном Бердом і Едвардом Лопера в Пенсильванському університеті. Вона зіграла ключову роль в проривних дослідженнях NLP. NLTK, поряд з іншими бібліотеками та інструментами

Python, тепер використовують в своїх навчальних програмах університети по всьому світу.

Бібліотека досить універсальна, однак її важко використовувати для обробки природної мови. NLTK може бути досить повільною, адже вона повність написана на мові Python та працює ці строками.

SpaCy відносно молода бібліотека, призначена для виробничого використання. Вона набагато доступніше інших NLP-бібліотек Python, таких як NLTK. SpaCy пропонує найшвидший синтаксичний парсер, наявний сьогодні на ринку. Крім того, оскільки інструментарій написаний на мові Cython, бібліотека також дуже швидка й ефективна. На рисунку 1.2 наведено порівняльний графік швидкості роботи бібліотек SpaCy та NLTK.

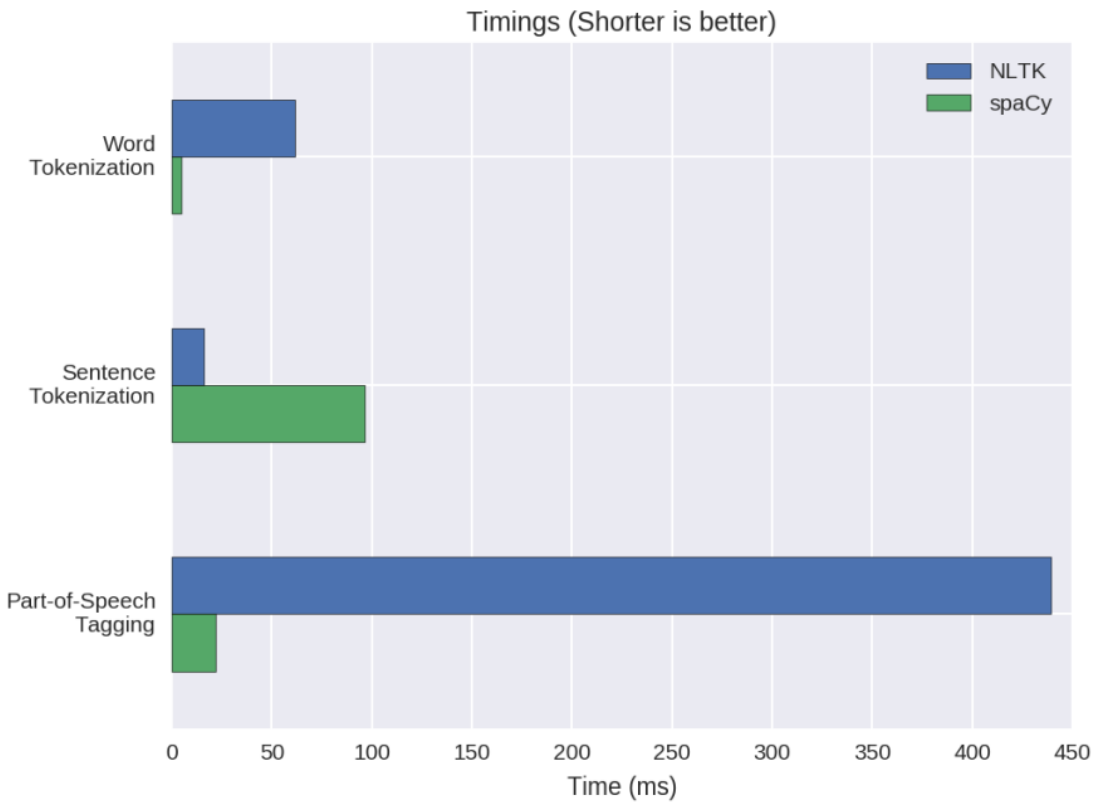

Рисунок 1.2 – Порівняльний графік швидкості роботи бібліотек SpaCy та NLTK

Проте, у порівнянні з іншими бібліотеками, spaCy підтримує найменшу кількість мов – усього сім. Однак, зростаюча популярність машинного навчання,

NLP і spaCy, як ключової бібліотеки, означає, що цей інструмент може незабаром почати підтримувати більше мов програмування.

Менш популярною, проте більш підходящою для новачків є бібліотека TextBlob. Вона надає простий інтерфейс для допомоги в освоєнні більшості основних завдань NLP, таких як аналіз настроїв, POS-маркування або витяг іменних фраз. Вона дуже корисна при проектуванні прототипів. Однак, вона також успадкувала основні недоліки NLTK. Для ефективної допомоги розробникам, що стикаються з вимогами використання NLP Python в виробництві, ця бібліотека занадто повільна.

Ще одною бібліотекою, яку використовують розробники під час вирішення задачі NLP є Gensim. Ця бібліотека Python, яка спеціалізується на виявленні семантичної подібності між двома документами за допомогою векторного просторового моделювання та інструментарію тематичного моделювання. Вона може обробляти великі текстові масиви за допомогою ефективної потокової передачі даних і інкрементних алгоритмів. Важливою перевагою бібліотеки є оптимізація використання пам'яті і швидкість обробки. Все це досягається за допомогою іншої бібліотеки Python – NumPy. Можливості векторного моделювання простору цього інструменту також варті уваги.

1.4 Алгоритм TextRank та LSA

Узагальнення тексту базується на рангах текстових речень із використанням варіації алгоритму TextRank. TextRank – це загальний алгоритм ранжування на основі графів для обробки природної мови. TextRank - це метод автоматичного підведення підсумків. Метою алгоритмів ранжирування на основі графів є вирішення важливості вершини в графі, базуючись на глобальній інформації, рекурсивно взятої з цілого графіка.

Основною ідеєю, реалізованою моделлю ранжування за допомогою графів, є голосування чи рекомендація. Коли одна вершина посилається на іншу, вона в основному голосує за цю вершину. Чим більша кількість голосів, поданих за вершину, тим вища важливість цієї вершини. На рисунку 1.3 наведено схему роботи алгоритму.

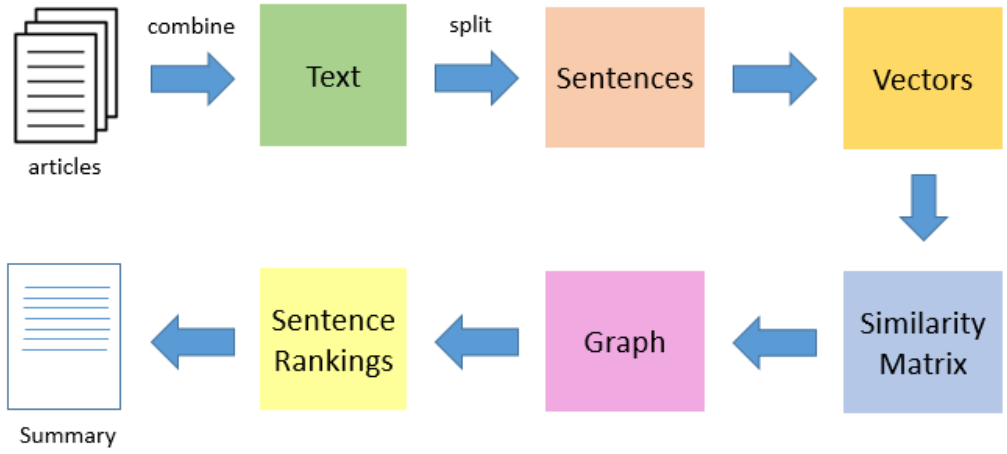

Рисунок 1.3 – Схема роботи алгоритму TextRank

TextRank включає в себе два завдання NLP: завдання вилучення ключових слів та завдання на вилучення речення.

Завдання алгоритму вилучення ключових слів полягає в тому, щоб автоматично визначити в тексті набір термінів, які найкраще описують документ. Такі ключові слова можуть становити корисні записи для побудови автоматичного індексу для колекції документів, може бути використана для класифікації тексту, або може слугувати стислим резюме для даного документа. Більш того, для автоматизації може використовуватися система автоматичної ідентифікації важливих термінів у тексті проблема вилучення термінології та побудови доменних словників. Одним із можливих підходів є використання критерію частоти [5].

Важливим аспектом TextRank є те, що він не вимагає глибоких лінгвістичних знань, а також анотованих корпусів й конкретних мов, що робить його дуже портативним для інших доменів, жанрів чи мови.

TextRank дуже добре підходить для додатків, що включають цілі речення, оскільки дозволяє ранжирувати текстові одиниці, що рекурсивно обчислюється на основі інформації, вилученої з цілого тексту.

Алгоритм LSA (англ. Latent семантич Analysis) заснований на прихованому семантичному аналізі, що дозволяє отримати неявне уявлення текстової семантики на основі спільної зустрічальності слів.

Він полягає в наступному:

а) нехай A – матриця терм-речення, отримана з вхідного документу. Її розмір дорівнює $n \times m$, де n – кількість термів в документі, m – кількість речень.

б) елемент a_{ij} цієї матриці дорівнює частоті терма i в тексті, якщо цей терм зустрічається в реченні j, і 0 – в іншому випадку. До отриманої матриці застосовується сингулярне розкладання:

$$A = U\Sigma V^T ,$$

де $U = [u_{ij}]$ – ортонормована матриця розміру $n \times m$;

$\Sigma = diag\ (\sigma 1, \sigma 2,..., \sigma m)$ – діагональна матриця;

$V = [vij]$ – ортонормована матриця розміру $m \times m$.

Якщо $rank\ (A) = r$, то виконується:

$$\sigma 1 \geq \sigma 2 \geq . . . \geq \sigma r > \sigma r + 1 = \sigma m = . . . = 0 .$$

З точки зору семантики сингулярне розкладання матриці A інтерпретується, як розбиття вихідного документа на r концепцій (тем). Кожен елемент vij матриці V відображає ступінь інформативності пропозиції j по темі i. При цьому значення σi матриці Σ відображає ступінь важливості теми i в вихідному документі.

в) кожному реченню s_k вхідного документа присвоюється вага за формулою:

$$s_k = \sqrt{\sum_{i=1}^{m} v_{ik^2} * \sigma_{i^2}} .$$

Отже, більшої ваги набувають речення, що є найбільш інформативними за однією з тем документа, при цьому враховується і ступінь важливості концепції у документі.

г) значення ваги речень упорядковуються за спаданням, і в резюме включаються речення, що відповідають першим l значенням, де l – бажана кількість речень у резюме.

1.5 Огляд існуючих датасетів

Для того, щоб задовольнити вимоги сучасних підходів на основі даних, було запропоновано кілька великомасштабних наборів даних. Більшість із доступних корпусів походять із галузі новин. У роботі [29] описано датасет Gigaword – це набір статей та відповідних їм назв, який спочатку використовувався для генерації заголовків, але також він був адаптований до узагальнення тексту за допомогою одного речення [30]. Автори роботи [31] застосовували датасет NYT – це збірка статей з журналу New York Times з резюме, які були створені вченими бібліотеки. Цей набір даних в основному використовувався для екстрактивного узагальнення [32] та для визначення важливості фрази у тексті [33].

У роботі [34] описано набір даних CNN / DailyMail, який складається зі статей з резюме, які скомпоновані разом та помічені тегом «@highlights», написаних самими авторами. Він зазвичай використовується як для абстрактних, так і для екстрактивних методів узагальнення. Автори роботи [35] використовували набір даних XSum – це збірник статей, асоційованих резюме,

що складається з одного речення. Цей датасет орієнтований на абстрактні моделі. Набір даних NewsRoom [36] – це різноманітна колекція статей, поданих із 38 основних сайтів Інтернет-новин. Цей набір даних було випущено разом із таблицею лідерів та проведеним тестуванням.

За межами галузі новин було зібрано декілька наборів даних із відкритих форумів та інших порталів, що пропонують структурну інформацію. Так, датасет Reddit TIFU [37] – це колекція дописів, зібраних з Reddit, де користувачі публікують свої щоденні історії, і кожна публікація повинна містити резюме TL; DR (з англ. Too Long; Didn't Read). У роботі [38] наведено інформацію про набір даних WikiHow – це збірник статей з бази знань WikiHow, де кожна стаття містить інструкції щодо виконання процедурних багатоступеневих завдань, що охоплюють різні сфери, зокрема: мистецтво, фінанси, подорожі та здоров'я.

1.6 Огляд методів оцінки систем узагальнення тексту

Тестування систем вручну вимагає багато часу і зусиль, тестування однієї системи може займати близько трьох тисяч годин. Отже, дані підходи вкрай неефективні, і потрібна розробка систем автоматичного оцінювання різноманітних алгоритмів. Розробка системи автоматичного тестування ускладнюється тим, що не існує загального алгоритму оцінки резюме, виходячи з кінцевого набору ознак і правил. Якби існував подібний алгоритм, то не існувало б завдання реферування. Сучасні підходи до оцінювання ґрунтуються на порівнянні отриманого резюме з декількома модельними, вручну створеними резюме. Завдання ділиться на два етапи:

а) дослідження і розробка методів порівняння резюме з модельними резюме;

б) створення набору модельних резюме (корпусів)

Rouge – це система тестування, яка є однією з кращих на сьогоднішній день. Дана система містить багато метрик: "Rouge-N", "Rouge-L", "Rouge-W", "Rouge-S", "Rouge-SU". Всі метрики засновані на ідеї максимального покриття модельних резюме – тестованими і навпаки. При цьому у всіх методах використовуються N-грами. У якості модельних резюме беруться анотації, створені людиною.

Для оцінки якості машинного перекладу також використовують числову метрику BLEU. Основне завдання даного алгоритму полягає в тому, щоб порівнювати N-грами машинного перекладу з N-грамами еталонного перекладу і порахувати кількість збігів. Такі збіги не залежать від порядку слів. Чим їх більше, тим краще пропонований переклад.

Ще однією метрикою є METEOR – метрика для оцінювання якості машинного перекладу. Метрика базується на використанні n-gram та орієнтована на використання статистичної та точної оцінки вихідного тексту. На відміну від метрики BLUE, дана метрика використовує функції співставлення синонімів разом із точною відповідністю слів. Вона була розроблена, щоб вирішити проблеми, які були знайдені в більш популярній метриці BLUE, а також відтворити хорошу кореляцію з оцінкою експертів на рівні словосполучень або речень.

У таблиці 1.1 наведено переваги та недоліки кожної з розглянутих метрик.

Таблиця 1.1 – Переваги та недоліки метрик

Метрика	Переваги	Недоліки
ROUGE	а) простота використання; б) висока швидкість; в) можливість надавати оцінки г) в автоматичному режимі; д) висока кореляція з ручними оцінками.	а) чутлива до довжин порівнюваних документів; б) не враховує зв'язність анотацій.

Продовження таблиці 1.1

BLEU	а) «людська» логіка перекладу, за допомогою зміни форми точності для зіставлення перекладу кандидата з декількома перекладами; б) проста у використанні; в) враховує не тільки точність г) перекладу окремих слів, а й ланцюжків слів (n-грам); д) найкраще працює на рівні великого тексту.	а) об'єктивна тільки для б) статистичних або гібридних систем і для мов з нерозвиненою морфологічної структурою; в) для отримання високої г) оцінки є важливим правильний порядок слів; д) на маленькому обсязі тексту метрика часто дає нульовий результат через відсутність співпадаючих N-грам і не функціонує належним чином.
METEOR	має такі типи показників як прив'язка і синонімічно, а також стандартне точне зіставлення слів	не ставиться завдання розуміння семантики тексту, що і веде до деякої неточності такої оцінки.

Отже, алгоритми Rouge дозволяють досить точно оцінювати якість резюме. Також варто звернути увагу на Rouge-2, він має високу точність при низькій обчислювальної складності, що робить його привабливим при оцінці великої кількості резюме.

2 ПРОЄКТУВАННЯ БІБЛІОТЕКИ УЗАГАЛЬНЕННЯ ТЕКСТУ

У даній главі розглядаються діаграми взаємодії, поведінки, а також структурні діаграми створеного програмного забезпечення та функціональне призначення його компонентів.

2.1 Діаграми поведінки

На рисунку 2.1 наведено можливості, які надаються користувачу при використанні бібліотеки.

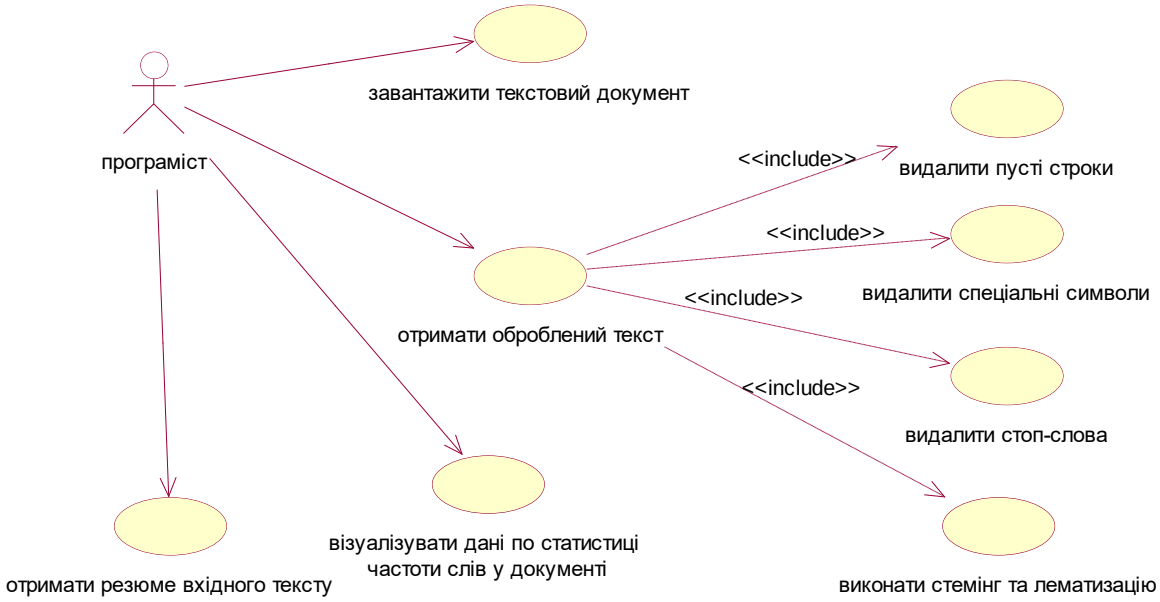

Рисунок 2.1 – Діаграма використання

Бібліотека підтримує узагальнення для двох мов – англійської та російської. Зчитування тексту відбувається з текстового файлу без обмежень на об’єм вхідних даних.

2.2 Діаграми взаємодії

На рисунку 2.2 наведено порядок взаємодії класів та методів між собою.

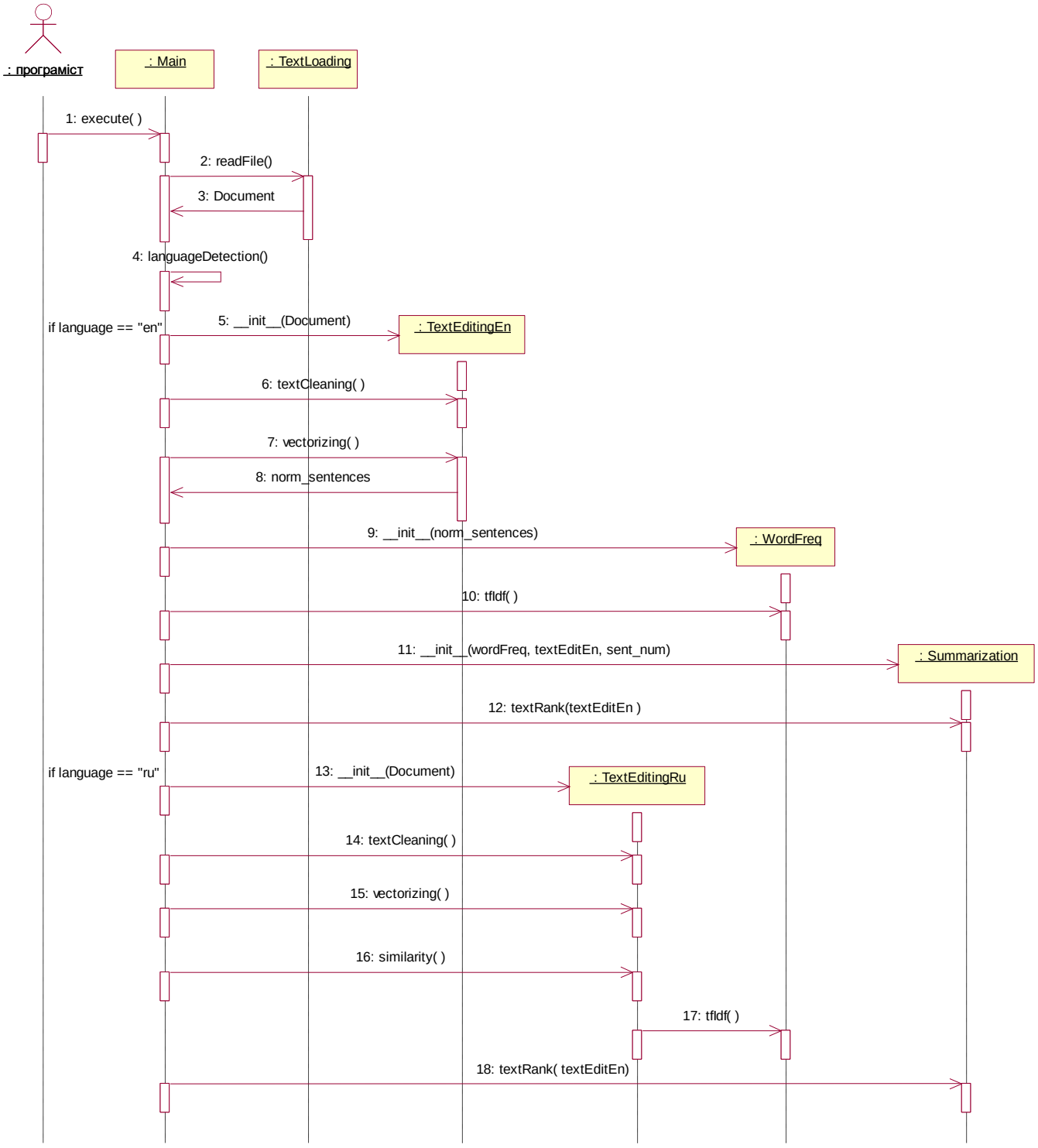

Рисунок 2.2 – Діаграма послідовності

Користувач взаємодіє з класом Main, за допомогою якого викликає основний функціонал бібліотеки через метод execute(), який, у свою чергу, звертається до методів інших класів. Так спочатку виконується завантаження тексту з файлу методом readFile(). Далі програма визначає мову вхідного тексту методом languageDetection(), за результатом якого буде створено об’єкт класу TextEditingEn для роботи з англійським текстом або TextEditingRu – з російським. В обох випадках у відповідних класах буде викликано метод textCleaning() для обробки тексту, а саме: видалення пустих рядків, зайвих пробілів, спеціальних символів, видалення стоп-слів, а також інші методи аналізу й нормалізації тексту. За допомогою методу tfIdf() з классу WordFreq знаходимо частоту використання кожного слова у тексті і передаємо ці дані у метод textRank() для створення резюме.

Для більшого розуміння взаємодії між елементами на рисунку 2.3 наведено діаграму кооперації.

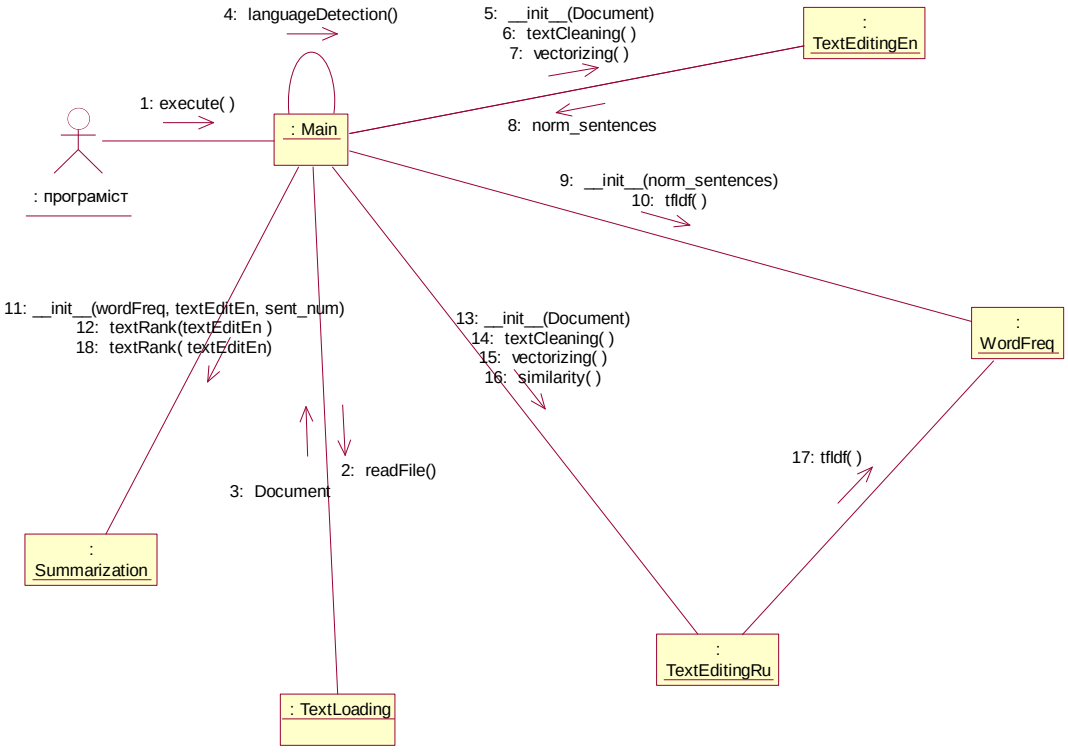

Рисунок 2.3 – Діаграма кооперації

Дане представлення дозволяє зрозуміти, який зв’язок між класами та якими повідомленнями вони обмінюються.

2.3 Структурні діаграми

На рисунку 2.4 наведено діаграму класів.

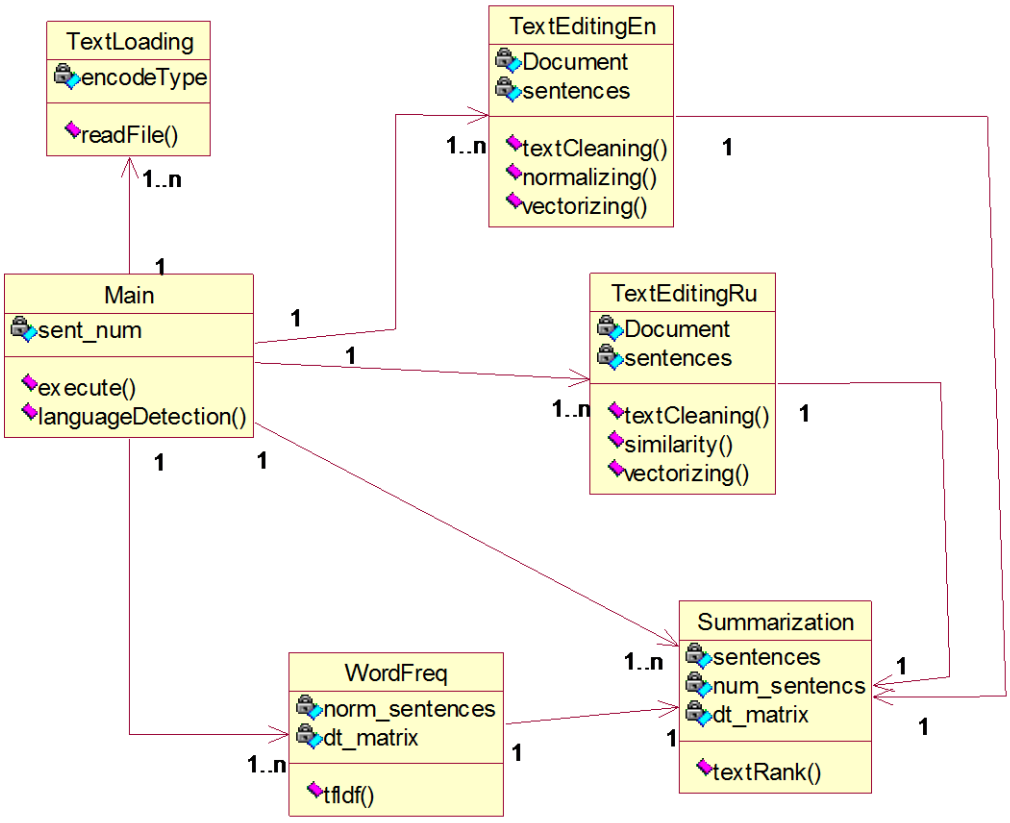

Рисунок 2.4 – Діаграма класів

За допомогою класу TextLoading відбувається завантаження тексту з файлу. Класи TextEditingRu й TextEditingEn містять методи для обробки, аналізу та нормалізації тексту залежно від мови вхідного тексту. У класі WordFreq знаходяться методи для знаходження частоти кожного слова у тексті. Клас

Summarization містить у собі необхідні методи для створення узагальненого тексту. Основним класом для початку роботи з бібліоткою є клас Main.

Для візуалізації структурних компонентів та відношень між ними на рисунку 2.5 наведено діаграму компонентів.

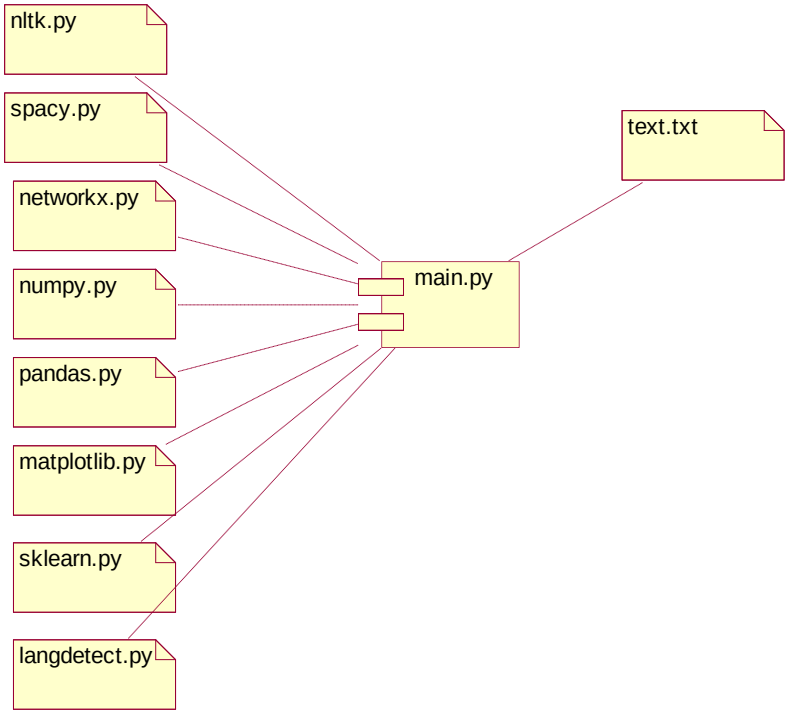

Рисунок 2.5 – Діаграма компонентів

Зліва представлені бібліотеки, які було використано, а справа – вхідні дані у вигляді текстового файлу.

На рисунку 2.6 наведено фізичне розгортання артефактів на вузлах.

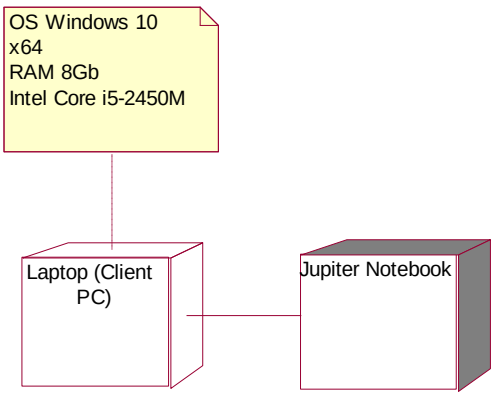

Рисунок 2.6 – Діаграма розгортання

Бібліотека була розроблена на машині з операційною системою Windows 10, з об’ємом пам’яті у 8 Gb та центральним процесором Intel Core i5. Також було використано середовище Jupyter Notebook.

Отже, створене програмне забезпечення має шість класів – клас для завантаження даних, клас для нормалізації тексту, а також класи для визначення важливості кожного слова у тексті та безпосередньо сам клас для знаходження резюме тексту. Для створення високорівневої бібліотеки було використано ряд інших бібліотек, таких як NLtk та Spacy, які надають алгоритми для роботи з текстовими даними.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Підготовка даних

Для тестування та оцінки роботи алгоритму в даній роботі використовується набір даних Multiling 2015. Оскільки цей датасет призначений для великої кількості мов, його ієрархія складається з 38-ми директорій (для кожної підтримуваної мови). Кожна директорія містить у собі 30 документів – наукових статей Вікіпедії, а також 30, створених людиною, резюме. Кожне резюме складається приблизно з 7-8 речень.

Статі не містять у собі зображень чи тегів мови розмітки HTML, а складаються лише з символів, що підтримує кодування UTF-8. На рисунку 3.1 наведено етапи подальшої роботи з даними.

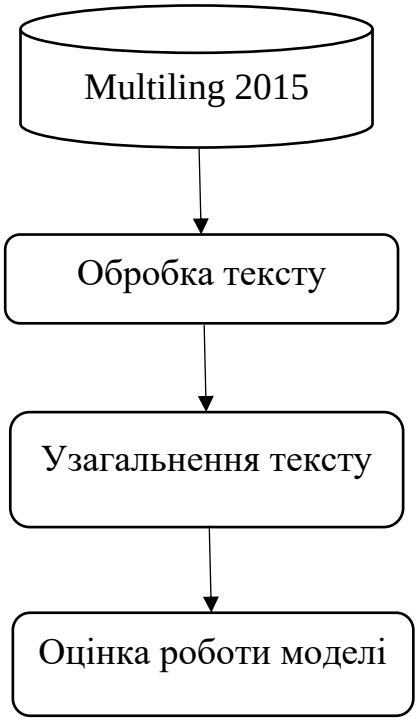

Рисунок 3.1 – Етапи роботи з даними

Статистику набору даних для англійської та російської мови наведено у таблицях 3.1 та 3.2. Для кожної характеристики було розраховано такі параметри:

а) avg – середнє значення;

б) min – мінімальне значення;

в) max – максимальне значення.

Таблиця 3.1 – Статистика повних текстів

Англійська мова:			
Характеристика	avg	min	max
Кількість речень	205	83	318
Кількість унікальних слів	3152	2103	5021
Російська мова:			
Характеристика	avg	min	max
Кількість речень	224	101	324
Кількість унікальних слів	3892	2527	6051

Таблиця 3.2 – Статистика створених резюме

Англійська мова:			
Характеристика	avg	min	max
Кількість речень	16	12	18
Кількість унікальних слів	290	270	346
Російська мова:			
Характеристика	avg	min	max
Кількість речень	13	11	15
Кількість унікальних слів	192	179	201

Слід зазначити деякі особливості використовуваного набору даних:

а) зразкові резюме мають невелику довжину: 13-16 речень, коли вхідні тексти мають середню довжину 200 речень. Таким чином, стиснення вихідного тексту, в середньому, проводиться в 15.5 разів;

б) резюме не є екстракційними, тобто речення в них перефразовані, а деякі слова замінені синонімами;

в) набір вхідних текстів різноманітний: в ньому присутні і зовсім короткі тексти (довжиною 2000 слів), і досить довгі (довжиною понад 6000 слів).

3.2 Опис створеної моделі

Першим етапом є визначення мови вхідного тексту, який було зчитано з файлу. Ця дія виконується за допомогою бібліотеки langdetect методом detect().

Оскільки програма підтримує узагальнення для двох мов, вона має два різних класи для обробки тексту. Проте спільними методами є видалення пустих рядків, зайвих пробілів і токенізація речень та слів. На рисунку 3.2 наведено приклад тексту після видалення зайвих рядків, символів та переходу до нижнього регістру.

```
LANGUAGE: en

at a modern-day nursing home, an elderly man, duke, reads a romantic story from his notebook to a fellow patient. in 1940, at a
carnival in seabrook island, south carolina, poor lumber mill worker noah calhoun sees 17-year-old heiress allison "allie" hami
lton, who is spending the summer in town with her parents. he pursues her, and they begin a summer romance. one evening, he tak
es her to the abandoned windsor plantation that he intends to buy and restore for them. while there, they attempt to have sex f
or the first time, but are interrupted by noah's friend fin with the news that allie's parents have the police looking for her.
when allie and noah return to her parents' mansion, it is revealed that due to their difference in social class, allie's parent
s, particularly her mother anne, never approved of their relationship and forbid her from seeing him. overhearing allie's mothe
r’s insults, which include going as far as to call him trash, noah walks out and allie chases after him. after revealing to all
ie that he does not think their relationship will work, an argument ensues, leading allie to break up with noah in the heat of
the moment. the next morning, anne announces that the family is returning home to charleston that same day. allie attempts to f
ind noah at the lumber mill, but he is out delivering a load, so she asks fin to tell noah that she loves him. when noah receiv
es the message he rushes to allie's home, only to find the house gated up and empty. noah writes a letter to allie every day fo
r a year, but it is later revealed that allie's mother had been intercepting the letters so that they never reach allie. after
365 letters, noah gives up and stops writing. while apart, noah enlists with fin to fight in world war ii, where fin is killed
in battle. meanwhile, allie attends college and also volunteers as a nurse's aide in a hospital for wounded soldiers, where she
meets captain lon hammond jr., a young lawyer who comes from old southern money. after a few years of being together, the two b
ecome engaged, to the delight of allie's parents. when noah returns from the war, his father reveals that he has sold their hom
e so that noah can buy the windsor plantation. while visiting charleston to get his building plans approved, noah glimpses alli
e from a bus and chases after her, only to witness allie and lon kissing at a restaurant. it is then that he convinces himself
```

Рисунок 3.2 – Приклад тексту після початкової обробки

Для виконання даної дії було створено метод для обох класів, який наведено на рисунку 3.3.

```
# blank lines removing
def textCleaning(self):
    self.DOCUMENT = re.sub(r'\n|\r', ' ', self.DOCUMENT)
    self.DOCUMENT = re.sub(r' +', ' ', self.DOCUMENT)
    self.DOCUMENT = self.DOCUMENT.strip()
    self.DOCUMENT = self.DOCUMENT.lower()
```

Рисунок 3.3 – Метод початкової обробки

Наступним етапом є видалення спеціальних символів, токенізація на масив слів, видалення стоп-слів та стемінг.

Іноді одних слів в тексті більше, ніж інших, і вони можуть зустрічатися майже в кожному реченні й не нести інформативного навантаження. Такі слова є «шумом» для подальшої роботи з текстом. На рисунку 3.4 наведено приклад стоп-слів для обох мов, які повинні бути видалені.

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
lf', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'th
ose', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing',
'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'aga
inst', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
y', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than',
'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 'v
e', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "has
n't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "sha
n't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

а) стоп-слова англійської мови

```
['и', 'в', 'во', 'не', 'что', 'он', 'на', 'я', 'с', 'со', 'как', 'а', 'то', 'все', 'она', 'так', 'его', 'но', 'да', 'ты', 'к',
'у', 'же', 'вы', 'за', 'бы', 'по', 'только', 'ее', 'мне', 'было', 'вот', 'от', 'меня', 'еще', 'нет', 'о', 'из', 'ему', 'тепер
ь', 'когда', 'даже', 'ну', 'вдруг', 'ли', 'если', 'уже', 'или', 'ни', 'быть', 'был', 'него', 'до', 'вас', 'нибудь', 'опять', 'у
ж', 'вам', 'ведь', 'там', 'потом', 'себя', 'ничего', 'ей', 'может', 'они', 'тут', 'где', 'есть', 'надо', 'ней', 'для', 'мы', 'т
ебя', 'их', 'чем', 'была', 'сам', 'чтоб', 'без', 'будто', 'чего', 'раз', 'тоже', 'себе', 'под', 'будет', 'ж', 'тогда', 'кто',
'этот', 'того', 'потому', 'этого', 'какой', 'совсем', 'ним', 'здесь', 'этом', 'один', 'почти', 'мой', 'тем', 'чтобы', 'нее', 'с
ейчас', 'были', 'куда', 'зачем', 'всех', 'никогда', 'можно', 'при', 'наконец', 'два', 'об', 'другой', 'хоть', 'после', 'над',
'больше', 'тот', 'через', 'эти', 'нас', 'про', 'всего', 'них', 'какая', 'много', 'разве', 'три', 'эту', 'моя', 'впрочем', 'хоро
шо', 'свою', 'этой', 'перед', 'иногда', 'лучше', 'чуть', 'том', 'нельзя', 'такой', 'им', 'более', 'всегда', 'конечно', 'всю',
'между']
```

б) стоп-слова російської мови

Рисунок 3.4 – Приклад стоп-слів для обох мов

Оскільки даний набір є списком, то до нього можна додати додаткові слова, які будуть інформативними для конкретного випадку або, навпаки, видалити необхідні.

Обидві вибрані мови мають багату морфологічну структурою. Так, наприклад, слово «програма» й «програмою» мають однакове значення, але різну форму. Тому для машинного навчання краще привести їх до однієї форми для зменшення розмірності. Для цього використовується метод стемінгу – це процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс. В Python-бібліотеці NLTK для цього є Snowball Stemmer, який підтримує російську мову й англійську мови.

На рисунку 3.5 наведено приклад стемінгу для російської мови.

```
LANGUAGE: ru

[{'кажд', 'социальн', 'истор', 'дом', 'мужчин', 'пожил', 'ден', 'женщин', 'девушк', 'чита', 'престарел', 'юнош', 'сло', 'разн',
'отношен'}, {'симпатичн', 'богат', 'бедн', 'красавиц', 'обаятельн', 'аристократ', 'но', 'элл', 'доч', 'прост', 'парен', 'сем',
'весел'}, {'парк', 'встрет', 'перв', 'аттракцион'}, {'отказыва', 'но', 'влюбля', 'памят', 'е', 'приглаша', 'свидан', 'элл'},
{'отчая', 'уныва', 'красавиц', 'пыта', 'но', 'сердц', 'завоева'}, {'взаимн', 'чувств', 'молод', 'человек', 'влюб', 'внезапн',
'понима', 'неожида', 'начина', 'элл', 'просыпа'}, {'дни', 'прекрасн', 'незабыва', 'провод', 'вмест', 'влюблен'}, {'хотел', 'до
м', 'котор', 'дел', 'но', 'мечт', 'полност', 'сво', 'реконструирова', 'элл', 'куп', 'старин'}, {'знаком', 'родител', 'приход',
'момент', 'нужн'}, {'родител', 'бедн', 'объясн', 'эт', 'но', 'одобря', 'прост', 'отец', 'е', 'сем', 'ум', 'элл'}, {'кажд', 'защ
ищ', 'ссор', 'родител', 'слыш', 'но', 'е', 'комнат', 'элл', 'ожида', 'слов'}, {'остав', 'пробл', 'вывод', 'приход', 'он', 'е',
'принесет'}, {'тяжел', 'разр', 'ссор', 'друг', 'сердц', 'люб', 'нем', 'когд', 'расста', 'искрен', 'об', 'элл', 'очен', 'пережив
а', 'выход'}, {'ден', 'но', 'на', 'узна', 'переезжа', 'сем', 'след', 'элл'}, {'дом', 'мест', 'он', 'едет', 'пуст', 'срыва', 'лю
бим'}, {'кажд', 'пис', 'получа', 'ден', 'котор', 'эт', 'мат', 'прячет', 'но', 'пишет', 'письм', 'прощен', 'говор', 'приедет',
'е', 'прос', 'элл'}, {'надежд', 'спуст', 'последн', 'но', 'год', 'отправ', 'письм', 'утрат'}, {'воева', 'нача', 'войн', 'друг',
'вскор', 'но', 'втор', 'вмест', 'миров', 'ушл'}, {'ранен', 'работа', 'лон', 'младш', 'вниман', 'обраща', 'воен', 'однажд', 'вв
с', 'хэммонд', 'пилот', 'не', 'тяжел', 'госпитал', 'элл', 'медсестр'}, {'предложен', 'посл', 'дела', 'встреча', 'лон', 'больни
ц', 'начина', 'элл', 'выход'}, {'вспомина', 'родител', 'богат', 'сих', 'соглаша', 'девушк', 'нов', 'избранник', 'лет', 'но', 'р
ад', 'пор', 'е', 'прошл', 'несмотр', 'чем', 'элл', 'очен'}, {'возвраща', 'войн', 'фронт', 'но', 'заканчива'}, {'мужчин', 'смею
т', 'друг', 'вид', 'однажд', 'беж', 'держат', 'улиц', 'элл', 'рук'}, {'надежд', 'но', 'мечт', 'поздн', 'сво', 'осуществля', 'по
нима', '200', 'реставрир', 'стар', 'дом', 'потеря', 'приедет', 'ран', 'элл', 'любим', 'летн', 'сюд', 'полност'}]
```

Рисунок 3.5 – Приклад стемінгу для російської мови

Програмна реалізація, описаної обробки тексту, зображена на рисунку 3.6.

```
def normalizing(self, doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
    # tokenize document
    tokens = nltk.word_tokenize(doc)
    # filter stopwords out of document
    stop_words = nltk.corpus.stopwords.words('english')
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc
```

а) метод для англійської мови

```
sentences = nltk.sent_tokenize(self.DOCUMENT)
tokenizer = nltk.RegexpTokenizer(r'\w+')
# ------
tokens = nltk.word_tokenize(self.DOCUMENT)
# filter stopwords out of document
stop_words = nltk.corpus.stopwords.words('russian')
filtered_tokens = [token for token in tokens if token not in stop_words]
cleanSentences = TreebankWordDetokenizer().detokenize(filtered_tokens)
tokenizedCleanSnts = nltk.sent_tokenize(cleanSentences)
# ------
lmtzr = RussianStemmer()
words = [set(lmtzr.stem(word) for word in tokenizer.tokenize(sentence.lower()))
         for sentence in tokenizedCleanSnts]
```

б) метод для російської мови

Рисунок 3.6 – Програмна реалізація обробки тексту

Далі потрібно векторизувати нормалізовані речення, використовуючи статистичний показник, що використовується для оцінки важливості слів у контексті документа – технологію функцій TF-IDF схеми. Частота слова (англ. term frequency) – це відношення числа входжень обраного слова до загальної кількості слів документа, а обернена частота документа (англ. inverse document frequency) – інверсія частоти, з якою слово зустрічається в документах колекції. Таким чином, показник TF-IDF – це добуток двох множників: TF та IDF і який розраховується за формулою:

$$tf - idf(t, d, D) = \frac{n_t}{\sum k * n_k} * log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

де n_t – число входжень слова в документ;

$\sum k * n_k$ – загальна кількість слів в документі;

$|D|$ – кількість документів в колекції;

$|\{d_i \in D \mid t \in d_i\}|$ – кількість документів, в яких зустрічається слово d_i.

На рисунку 3.7 зображено результат виконання векторайзеру, а саме матрицю, що відображає вагу кожного слова у тексті.

	0	1	2	3	4	5	6	7	8	9	...	\
abandoned	0.0	0.00	0.0	0.35	0.00	0.00	0.00	0.00	0.00	0.00	...	
actually	0.0	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	
admission	0.0	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	
ago	0.0	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	
aide	0.0	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	
allie	0.0	0.10	0.0	0.00	0.00	0.10	0.12	0.25	0.00	0.13	...	
allies	0.0	0.00	0.0	0.00	0.19	0.16	0.19	0.00	0.00	0.00	...	
allison	0.0	0.24	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	
also	0.0	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	
anne	0.0	0.00	0.0	0.00	0.00	0.18	0.00	0.00	0.27	0.00	...	

	27	28	29	30	31	32	33	34	35	36
abandoned	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00
actually	0.00	0.57	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00
admission	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00
ago	0.00	0.00	0.24	0.0	0.0	0.00	0.00	0.00	0.0	0.00
aide	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00
allie	0.14	0.12	0.10	0.0	0.0	0.11	0.11	0.00	0.0	0.09
allies	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.23	0.0	0.00
allison	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00
also	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00
anne	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.0	0.00

Рисунок 3.7 – Візуалізація ваги кожного слова у тексті

Для реалізації було використано метод TfidfVectorizer() з бібліотеки SkLearn. Програмний код наведено на рисунку 3.8.

```
class WordFrequency():

    def __init__(self, norm_sentences):
        self.norm_sentences = norm_sentences
        self.dt_matrix = 0

    def tfIdf(self):
        # vectorizing normalized sentences using the TF-IDF feature engineering scheme
        tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
        self.dt_matrix = tv.fit_transform(self.norm_sentences)
        self.dt_matrix = self.dt_matrix.toarray()

        vocab = tv.get_feature_names()
        td_matrix = self.dt_matrix.T
        print(td_matrix.shape)
        print(pd.DataFrame(np.round(td_matrix, 2), index=vocab).head(10))
```

Рисунок 3.8 – Програмний код векторизації

Модифікацією оригінального алгоритму TextRank в даній роботі є те, що ребра, які з’єднують вершини графу, мають коефіцієнт ваги, який показує на скільки важливий зв’язок між певними вершинами. Формула для розрахування рангу вершини виглядає так:

$$TR(V_i) = (1-d) + d * \sum_{V_i \in In(V_i)} * \frac{w_{ij} TR(V_j)}{\sum_{V_k \in Out(V_i)} w_{jk}},$$

де d – коефіцієнт демпування, який має значення від 0 до 1;

TR – зважена оцінка PageRank для вершини;

$In(V_i)$ – кількість слів (ребер), які входять до вершини;

$Out(V_i)$ – кількість слів (ребер), яка виходить з вершини.

Більш докладно принцип роботи алгоритму TextRank наведено у першому розділі.

Для створення графу подібності необхідно спочатку побудувати подібну матрицю шляхом множення матриці частот на її транспоновану матрицю. На рисунку 3.9 наведено програмний код, а також результат його виконання.

```
similarity_matrix = np.matmul(self.dt_matrix, self.dt_matrix.T)
print(similarity_matrix.shape)
np.round(similarity_matrix, 3)
```

а) програмна реалізація

```
[10 rows x 37 columns]
(37, 37)
[[1.         0.         0.         ... 0.         0.         0.        ]
 [0.         1.         0.10049375 ... 0.01058996 0.         0.0151643 ]
 [0.         0.10049375 1.         ... 0.         0.         0.        ]
 ...
 [0.         0.01058996 0.         ... 1.         0.         0.01000024]
 [0.         0.         0.         ... 0.         1.         0.        ]
 [0.         0.0151643  0.         ... 0.01000024 0.         1.        ]]
```

б) матриця подібності

Рисунок 3.9 – Програмний код створення матриці подібності й приклад результату

Використовуючи речення у якості вершин графу, а матрицю подібності, як вагу ребер між кожною парою речень, потрібно створити граф, який буде вхідним параметром методу Pagerank() з бібліотеки networkx. На рисунку 3.10 наведено приклад створеного графу подібності.

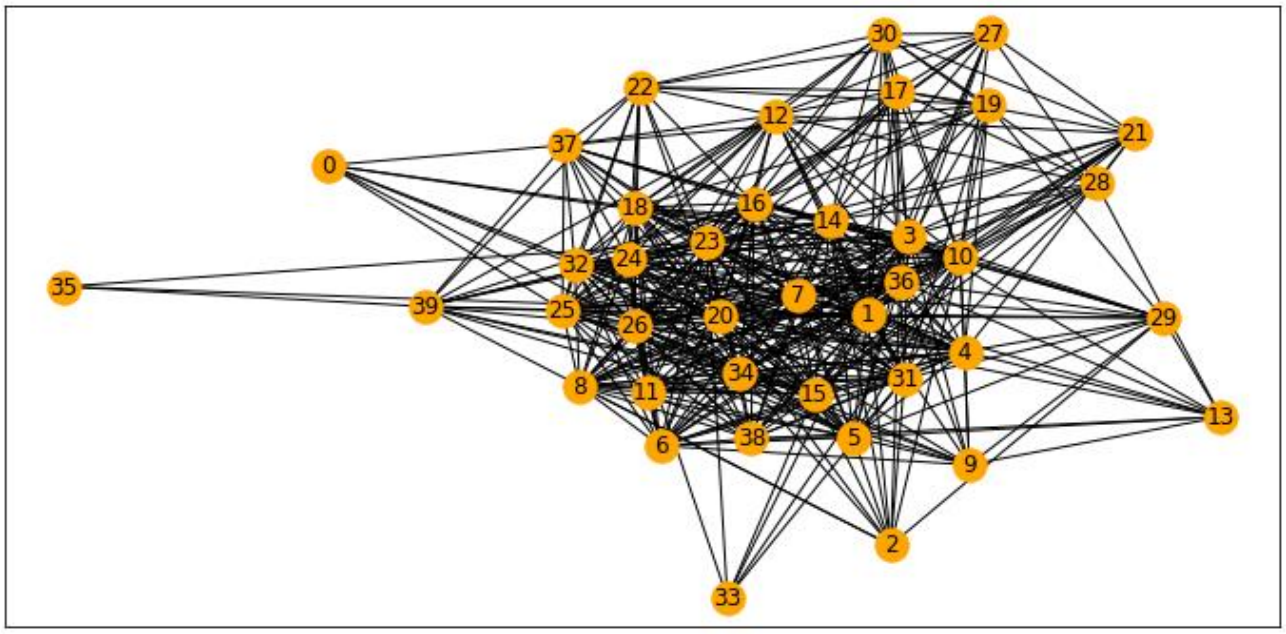

Рисунок 3.10 – Приклад графу подібності

Такий граф відображає на скільки сильно деякі речення з’єднані одне з одним. На рисунку 3.11 наведено програмну реалізацію створення графу подібності.

```
similarity_graph = networkx.from_numpy_array(similarity_matrix)
plt.figure(figsize=(12, 6))
networkx.draw_networkx(similarity_graph, node_color='yellow')
```

Рисунок 3.11 – Код для створення графу подібності

Останнім етапом є знаходження оцінок для кожного речення, на основі яких буде створено резюме для вхідного тексту. На рисунку 3.12 зображено програмний код методу обчислення важливості кожного речення, а також приклад результату його виконання.

```
scores = networkx.pagerank(similarity_graph)
ranked_sentences = sorted(((score, index) for index, score in scores.items()), reverse=True)
ranked_sentences[:10]
print(ranked_sentences[:10])

top_sentence_indices = [ranked_sentences[index][1]
                for index in range(self.num_sentences)]
top_sentence_indices.sort()

print('\n'.join(np.array(self.sentences)[top_sentence_indices]))
```

а) код методу обчислення важливості кожного речення

```
(0.03709346654181642, 25)
(0.034978780987031395, 9)
(0.034411855525991304, 11)
(0.034312503920075316, 5)
(0.03374634528011231, 16)
(0.031518005533105804, 26)
(0.03050086326282201, 27)
(0.02991534503649904, 10)
(0.02931462724808119, 17)
(0.02873607573752273, 29)
```

б) отримані оцінки для речень

Рисунок 3.12 – Програмний код обчислення важливості кожного речення та приклад результату

3.3 Тестування функціоналу бібліотеки

В процесі тестування буде застосовано ad-hoc тестування, з причини відсутності строгої специфікації. На першому етапі, в ході функціонального тестування, планується виявити помилки, засновані на взаємодії функціоналів шляхом реалізації нетривіальних сценаріїв. На другому етапі буде проведено ряд тестів, що визначають обробку виняткових ситуацій.

У таблиці 3.3 наведено функціонал для тестування та його результати виконання.

Таблиця 3.3 – Результати тестування функціоналу

Функція	Результат
Зчитування документу з файлу	Пройдено
Визначення мови вхідних даних	Пройдено
Обробка текстових даних	Пройдено
Обчислення частоти слів	Пройдено
Обчислення матриці подібності	Пройдено
Побудова графа подібності	Пройдено
Візуалізація результатів	Пройдено

Можливі тестові сценарії для функції «Зчитування документу з файлу»:

а) вказано коректний шлях до текстового файлу. Очікуваний результат: усі дані будуть завантажені до строкової змінної

б) вказано некоректний шлях до текстового файлу. Очікуваний результат: користувач отримає повідомлення про помилку.

Можливі тестові сценарії для функції «Визначення мови вхідних даних»:

а) мовою вхідних даних є англійська або російська. Очікуваний результат: буде створено резюме для відного тексту.

б) мовою вхідних даних не є англійська або російська. Очікуваний результат: користувачу буде виведено повідомлення про підтримувані мови.

Представлені тестові сценарії було пройдено та, засновуючись на їх результатах, було проведено обробку виключень.

3.4 Результати та обчислювальні експерименти

У даному підрозділі проводиться порівняння алгоритмів узагальнення тексту, а саме алгоритму TextRank та LSA. У таблицях 3.4 – 3.6 представлена

різниця між оцінками кожного з алгоритмів, які було застосовано для створення резюме на англійській мові. Оцінка була проведена за допомогою метрики ROUGE. Кожна метрика була обчислена для трьох випадків:

а) basic – обчислення на автоматичному й зразковому резюме в їх первісному вигляді;

б) stem – з використанням стемінгу;

в) stem, stop-words – з використанням стемінгу й видаленням стоп-слів.

Таблиця 3.4 – Оцінка алгоритму TextRank

	TextRank		
	basic	stem	stem, stop-words
ROUGE-1	0,47	0,49	0,42
ROUGE-2	0,18	0,19	0,17
ROUGE-L	0,30	0,32	0,31

Таблиця 3.5 – Оцінка алгоритму LSA

	LSA		
	basic	stem	stem, stop-words
ROUGE-1	0,33	0,38	0,35
ROUGE-2	0,16	0,14	0,15
ROUGE-L	0,30	0,28	0,30

З наведених табличних даних видно, що найбільша різниця в оцінках спостерігається при оцінці з видаленням стоп-слів і проведенням стемінгу, а саме ця оцінка найбільш об'єктивна з точки зору людини.

У таблиці 3.6 наведено порівняння алгоритмів по метрикам, розрахованим на резюме, що були сформовані за допомогою стемінгу й видалення стоп-слів.

Таблиця 3.6 – Оцінка алгоритмів

	TextRank	LSA
ROUGE-1	0,49	0,41
ROUGE-2	0,20	0,20
ROUGE-L	0,33	0,32

З отриманих оцінок видно, що алгоритм TextRank, хоч і незначно, але перевищує алгоритм LSA за метриками ROUGE-1 та ROUGE-L. За метрикою ROUGE-2 алгоритми TextRank і LSA показали однакові результати. Таким чином, алгоритм TextRank перевершує алгоритм LSA за використовуваними метриками.

У таблицях 3.7 – 3.9 наведено представлена різниця між оцінками кожного з алгоритмів, які було застосовано для створення резюме на російській мові.

Таблиця 3.7 – Оцінка алгоритму TextRank

	TextRank		
	basic	stem	stem, stop-words
ROUGE-1	0,35	0,37	0,31
ROUGE-2	0,13	0,14	0,11
ROUGE-L	0,22	0,25	0,22

Таблиця 3.8 – Оцінка алгоритму LSA

	LSA		
	basic	stem	stem, stop-words
ROUGE-1	0,24	0,28	0,21
ROUGE-2	0,15	0,12	0,16
ROUGE-L	0,29	0,30	0,28

Таблиця 3.9 – Оцінка алгоритмів

	TextRank	LSA
ROUGE-1	0,38	0,35
ROUGE-2	0,18	0,18
ROUGE-L	0,32	0,29

З наведених даних видно, що значення отриманих оцінок нижче, ніж значення оцінок, що були розраховані для англійських текстів. Це пояснюється рядом причин:

а) російська мова дуже різноманітна, в ній присутня велика кількість синонімічних слів і виразів, а метрики, за якими оцінювалися алгоритми не враховують можливу синонімію слів;

б) високий ступінь стиснення тексту в використовуваному для оцінки наборі даних. Тому, при формуванні рефератів, які значно коротше вихідного тексту, великий ризик втрати важливої інформації з тексту;

в) у використовуваному наборі даних, деяка інформація зі зразкових резюме може не повторюватись у тексті. Таким чином, алгоритм не має можливості згенерувати реферат, що містить частину інформації зі зразкового резюме, що знижує значення оцінок.

На рисунках 3.12-3.13 наведено приклади роботи алгоритмів для російської мови. Для демонстрації узагальнення російського тексту було обрано опис сюжету кінофільму «Іронія долі». Вхідний текст містить 40 речень, який було скорочено до 16-ти. Повний текст з описом наведено у додатку А.

36-летний хирург Женя Лукашин, закоренелый холостяк, живущий вдвоём с матерью, намеревается встретить Новый год со своей невестой Галей в квартире, неда вно полученной по адресу: 3-я улица Строителей, дом 25, квартира 12.
Путём логических умозаключений они приходят к выводу, что это могут быть либ о Павел, либо Женя.
По невероятному совпадению в Ленинграде по этому адресу находится точно тако й же типовой дом, как и в Москве, к типовому замку вполне подходит ключ от м осковской квартиры, типовая мебель ничем не отличается от московской, а в кв артире — так же после новоселья — царит беспорядок.
Когда, наконец, Женя понимает, что находится в Ленинграде, то приходит в отч аяние: Галя может неверно истолковать его внезапное исчезновение.
В это время появляется жених Нади — Ипполит Георгиевич, с которым она собира лась встретить праздник.
Женя уходит, но, сообразив, что денег на обратный билет у него нет, возвраща ется, чтобы попросить взаймы денег на билет до Москвы.
Сочувствуя незадачливому гостю, Надя разрешает ему позвонить с её домашнего телефона: сначала в аэропорт — узнать, когда первый самолёт в Москву; затем — чтобы объясниться со своей невестой, но та, услышав про Ленинград, бросает трубку.
Но, опять застав Женю, снова затевает новую сцену, Ипполит и Женя дерутся др уг с другом, за что Надя выгоняет из дома обоих.
Однако Женя, найдя предлог, снова возвращается, и Надя предлагает ему остать ся. Женя опять звонит в аэропорт, узнать расписание рейсов в Москву, объясня я Наде, что он не хочет торопиться с возвращением в Москву.
Надя, надеясь поскорее отправить Женю домой, едет на такси на Московский вок зал и покупает ему билет на поезд до Москвы, но Женя выбрасывает билет через балкон. Мокрый Ипполит уходит из Надиной квартиры.
Слова Ипполита производят впечатление на хозяйку дома — она вслух замечает, что будущего у них с Женей быть не может, а всё, что произошло этой новогодн ей ночью, было заблуждением.
Женя сообщает ей, что он по чужой ошибке, вместо Павлика, улетел в Ленингра и там встретил другую женщину. Уставший после невероятной, суматошной ночи, Женя снова мирно спит, на этот раз уже в своей квартире. Проснувшись, он не сразу понимает, что видит любимую женщину наяву: Надя приехала, чтобы навсег да остаться с ним.

Рисунок 3.12 – Робота алгоритму LSA

36-летний хирург Женя Лукашин, закоренелый холостяк, живущий вдвоём с матерью, намеревается встретить Новый год со своей невестой Галей в квартире, неда вно полученной по адресу: 3-я улица Строителей, дом 25, квартира 12. В знак серьёзности своих нынешних намерений Женя даёт девушке ключи от квартиры. По давно сложившейся традиции Женя со своими друзьями Павлом, Сашей и Мишей пе ред встречей Нового года идут в баню. Отмечая предстоящую Женину женитьбу, о ни изрядно выпивают и отправляются в аэропорт провожать Павла, который долже н лететь в Ленинград к своей жене, задержавшейся там в командировке. Однако в аэропорту Павлик и Женя засыпают, а нетрезвые Саша и Миша не могут вспомни ть, кто из компании должен полететь. Путём логических умозаключений они прих одят к выводу, что это могут быть либо Павел, либо Женя. Сонный и не протрез вевший Женя сопротивляется: он никак не может взять в толк, почему постороння я женщина прогоняет его из его же собственной квартиры. Когда, наконец, Жен я понимает, что находится в Ленинграде, то приходит в отчаяние: Галя может н еверно истолковать его внезапное исчезновение. В это время появляется жених Нади — Ипполит Георгиевич, с которым она собиралась встретить праздник. Женя уходит, но, сообразив, что денег на обратный билет у него нет, возвращается , чтобы попросить взаймы денег на билет до Москвы. Но, опять застав Женю, сн ова затевает новую сцену, Ипполит и Женя дерутся друг с другом, за что Надя выгоняет из дома обоих. Однако Женя, найдя предлог, снова возвращается, и На дя предлагает ему остаться. Женя опять звонит в аэропорт, узнать расписание рейсов в Москву, объясняя Наде, что он не хочет торопиться с возвращением в Москву. Надя, надеясь поскорее отправить Женю домой, едет на такси на Москов ский вокзал и покупает ему билет на поезд до Москвы, но Женя выбрасывает бил ет через балкон. Женя самолётом возвращается в Москву. Женя сообщает ей, что он по чужой ошибке, вместо Павлика, улетел в Ленинград и там встретил другу ю женщину. Уставший после невероятной, суматошной ночи, Женя снова мирно спи т, на этот раз уже в своей квартире. Проснувшись, он не сразу понимает, что в идит любимую женщину наяву: Надя приехала, чтобы навсегда остаться с ним.

Рисунок 3.13 – Робота алгоритму TextRank

На рисунках 3.14-3.15 наведено приклади роботи алгоритмів для англійської мови. Для демонстрації узагальнення англійського тексту було обрано опис сюжету кінофільму «Фокус». Вхідний текст містить 41 речення, який було скорочено до 12-ти. Повний текст з описом наведено у додатку Б.

```
Seasoned con-man Nicky Spurgeon (Will Smith) goes to an upscale restaurant,
where an inexperienced grifter, Jess Barrett (Margot Robbie), seduces him an
d then pretends they've been caught by her jealous husband.
To win it back, Nicky asks Tse to pick any player on or off the field and sa
ys that Jess will guess the number picked.
Nicky explains to Jess how Tse had been programmed to pick 55 since he arriv
ed, with subtle, subconscious prompts throughout his day.
Jess cries as her limo drives off, leaving Nicky to climb into another waiti
ng car.
At a pre-race party, Nicky runs into Jess, who is now Garriga's girlfriend.
After faking heavy drinking upon seeing Jess, Nicky has a convincing fight w
ith Garriga in public and after being thrown out, is recruited by McEwen to
provide the component.
Nicky begins pursuing Jess again, and they eventually rekindle their relatio
nship.
Garriga is convinced that Jess had something to do with Nicky gaining access
to EXR and begins to suffocate the gagged Jess by holding her nose.
However, Jess then reveals that she was only trying to seduce Garriga in ord
er to steal his valuable watch and to make Nicky jealous.
Nicky promises to come clean in order to spare Jess's life but Owens shoots
him in the chest, causing a horrified Garriga to leave.
Owens then reveals himself to be Nicky's father, Bucky and assures Jess that
he avoided any major arteries.
```

Рисунок 3.14 – Робота алгоритму LSA

```
Nicky follows Jess and convinces her to have a drink with him.
Jess follows Nicky to New Orleans, successful in persuading Nicky to take he
r under his wing, where she is also introduced to Nicky's crew as well, incl
uding the obese and profane Farhad (Adrian Martinez) and fellow con-man Hors
t (Brennan Brown).
To win it back, Nicky asks Tse to pick any player on or off the field and sa
ys that Jess will guess the number picked.
Nicky will pretend to be a disgruntled technician on Garriga's team willing
to sell Garriga's custom fuel use algorithm EXR.
At a pre-race party, Nicky runs into Jess, who is now Garriga's girlfriend.
Nicky begins pursuing Jess again, and they eventually rekindle their relatio
nship.
Garriga is convinced that Jess had something to do with Nicky gaining acces
to EXR and begins to suffocate the gagged Jess by holding her nose.
In order to save Jess, Nicky explains that he gained access to EXR by tricki
ng Jess into believing he still had feelings for her.
However, Jess then reveals that she was only trying to seduce Garriga in ord
er to steal his valuable watch and to make Nicky jealous.
Owens then reveals himself to be Nicky's father, Bucky and assures Jess that
he avoided any major arteries.
Bucky drives Nicky and Jess to the hospital to treat Nicky's punctured lung
and departs with Nicky's money as a reminder of the consequences of losing f
ocus.
After he leaves, Nicky notices that Jess snatched Garriga's watch before he
left the warehouse, and a smiling Nicky and Jess then go into the hospital t
ogether.
```

Рисунок 3.15 – Робота алгоритму TextRank

Дані приклади ілюструють, що алгоритм TextRank генерує реферати, найбільш близькі за змістом до зразкового, ніж алгоритм LSA.

ВИСНОВКИ

В ході роботи проведено детальний аналіз попередніх досліджень, виявлено переваги та недоліки вже існуючих методів та моделей, описано способи обробки тексту алгоритмами з бібліотеки NLtk, а також були вивчені основні підходи до екстракційного реферування текстів і способи оцінки їх якості.

Для виконання поставленої мети було спроектовано та реалізовано модулі роботи алгоритму екстракційного реферування TextRank, модуль попередньої обробки текстів, модуль збору статистики з тестового набору даних, модулі оцінки алгоритмів й інші допоміжні модулі.

Створена бібліотека дозволяє отримувати резюме з вказаною кількістю речень для російського та англійських текстів. Також є можливість завантажити текст для його подальшої обробки.

Перспективи подальшого розвитку роботи пов'язані з вдосконаленням розробленої моделі шляхом впровадження абстрактних методів для генерування нових речень, базуючись на змісті вхідного тексту, на більш людський манер.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *International Conference on Learning Representations*. 2013. P. 23-36.

2. Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. 2019. P. 56-89.

3. Pradeepika Verma, Anshul Verma. A Review on Text Summarization Techniques // Journal of Scientific Research Institute of Science. 2020. P. 251-257.

4. C. Aggarwal. Machine Learning for Text // Text Summarization. 2018 P. 122-126.

5. D. Sarkar. Text Summarization // Text Analytics with Python. 2016. P. 435-450.

6. Kushal Bafna, Durga Toshniwal. Feature Based Summarization of Customers' Reviews of Online Products. *17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems – KES2013.* Procedia Computer Science.2013. P. 142 – 151.

7. M. Hu and B. Liu. Mining and summarizing customer reviews. *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD).* Seattle, USA, 2004. P. 113-126.

8. B. Pang and L. Lee. Opinion Mining and Sentiment Analysis // Foundations and Trends in Information Retrieval. Jan. 2008. P. 1-135.

9. Taeho Jo. Text Mining, Studies in Big Data. Springer International Publishing AG, part of Springer Nature, 2019. 213 p.

10. Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, Krys Kochut. Text Summarization Techniques: A Brief Survey // International Journal of Advanced Computer Science and Applications. July 8 P. 397-405.

11. Ani Nenkova, Kathleen McKeown. A survey of text summarization techniques // Journal of Emerging Technologies in Web Intelligence. 2014. P. 56-89.

12. E. Filatova and V. Hatzivassiloglou. A formal model for information selection in multi-sentence text extraction. *In Proceedings of the International Conference on Computational Linguistic*. 2004. P. 397–403.

13. E. Hovy and C.-Y. Lin. Automated text summarization in summarist // Advances in Automatic Text Summarization. 1999. P. 82– 94.

14. Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, Krys Kochut. Text Summarization Techniques: A Brief Survey // International Journal of Advanced Computer Science and Applications. July 8 P. 397-405.

15. Matthew Mayo. Approaches to Text Summarization: An Overview. 2013.

16. Khushboo S. Thakkar, Rajiv Vasantrao Dharaskar, Manoj Chandak. Graph-Based Algorithms for Text Summarization. 2010.

17. Mozhgan Nasr Azadani Nasser Ghadiri Ensieh Davoodijam. Graph-based biomedical text summarization: An itemset mining and sentence clustering approach // Journal of Biomedical Informatics. 2018. Pages 42-58.

18. Chintan Shah, Anjali Jivani. An Automatic Text Summarization on Naive Bayes Classifier Using Latent Semantic Analysis // Data, Engineering and Applications. 2019. P. 171-180.

19. Djoko Budiyanto Setyohadi. Summarizing Indonesian text Automatically by using sentence scoring and decision tree. *2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE).* 2012. P. 56-63.

20. Joel Larocca Neto Alex A. Freitas Celso A. A. Kaestner. Automatic Text Summarization using a Machine Learning Approach. 2015. P. 13-27.

21. Nadira Begum, Mohamed Fattah. Automatic text summarization using support vector machine // International journal of innovative computing, information & control. 2015. P. 89-95.

22. Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, Bing Xiang. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *Conference on Computational Natural Language Learning (CoNLL).* 2016. P. 15-26.

23. Sumit Chopra, Michael Auli. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016. P. 93-98.

24. Shengli Song, Haitao Huang, Tongxiao Ruan. Abstractive text summarization using LSTM-CNN based deep learning // Multimedia Tools and Applications. 2019. P. 857-875.

25. Bowen Zhou, Ramesh Nallapati, Cicero Nogueira Dos Santos. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *20th SIGNLL Conference on Computational Natural Language Learning*. 2016. P. 280-290.

26. Alexander M Rush, Sumit Chopra, Jason Weston. A neural attention model for abstractive sentence summarization. *Computation and Language (cs.CL); Artificial Intelligence (cs.AI).* 2013. P. 47-54.

27. Abigail See, Peter J. Liu, Christopher D. Manning Get To The Point: Summarization with Pointer-Generator Networks. *Computation and Language*. 2017. P. 56-85.

28. Freek Boutkan, Jorn Ranzijn, David Rau, Eelco van der Wel. Point-less: More Abstractive Summarization with Pointer-Generator Networks. 2019. P. 56-62.

29. Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. 2019.

30. David Graff and C Cieri. English gigaword, linguistic data consortium. 2015.

31. Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. *The 2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, San Diego California. USA, June 12-17, 2016. P. 25-35.

32. Evan Sandhaus. The New York Times annotated corpus. Linguistic Data Consortium, Philadelphia. 2008. P. 45-52.

33. Kai Hong and Ani Nenkova. Improving the estimation of word importance for news multidocument summarization. *In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014. P. 65-78.

34. Yinfei Yang and Ani Nenkova. Detecting information-dense texts in multiple news domains. *Twenty-Eighth AAAI Conference on Artificial Intelligence*. July 27 -31, 2014. P. 56-69.

35. Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. 2014. P. 65-96.

36. Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium. 2018. P. 65-89.

37. Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* New Orleans, Louisiana, USA, June 2018.

38. Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of reddit posts with multi-level memory networks. 2016. P. 98-87.

ДОДАТОК А

Оригінальний вхідний текст для російської мови

Действие происходит в Москве, в новостройке. 36-летний хирург Женя Лукашин, закоренелый холостяк, живущий вдвоём с матерью, намеревается встретить Новый год со своей невестой Галей в квартире, недавно полученной по адресу: 3-я улица Строителей, дом 25, квартира 12. Собираясь в новогоднюю ночь сделать Гале предложение, он рассказывает ей, что как-то уже делал предложение другой женщине, но, испугавшись предстоящих перемен, сбежал в Ленинград.

По давно сложившейся традиции Женя со своими друзьями Павлом, Сашей и Мишей перед встречей Нового года идут в баню. Отмечая предстоящую Женину женитьбу, они изрядно выпивают и отправляются в аэропорт провожать Павла, который должен лететь в Ленинград к своей жене, задержавшейся там в командировке. Однако в аэропорту Павлик и Женя засыпают, а нетрезвые Саша и Миша не могут вспомнить, кто из компании должен полететь. Путём логических умозаключений они приходят к выводу, что это могут быть либо Павел, либо Женя. Не рискнув бросать монетку и основываясь на разговорах о свадьбе, они сажают в самолёт Женю, предположив, что в Ленинграде его ждёт невеста.

Проснувшись только в ленинградском аэропорту, он берёт такси и называет свой московский адрес. По невероятному совпадению в Ленинграде по этому адресу находится точно такой же типовой дом, как и в Москве, к типовому замку вполне подходит ключ от московской квартиры, типовая мебель ничем не отличается от московской, а в квартире — так же после новоселья — царит беспорядок. И ничего не подозревающий Женя мирно засыпает.

В квартиру возвращается хозяйка — Надя Шевелёва, незамужняя 34-летняя учительница русского языка и литературы. С ужасом она обнаруживает на своей тахте спящего незнакомца и пытается разбудить. Сонный и не протрезвевший Женя сопротивляется: он никак не может взять в толк, почему посторонняя женщина прогоняет его из его же собственной квартиры. Когда, наконец, Женя понимает, что находится в Ленинграде, то приходит в отчаяние: Галя может неверно истолковать его внезапное исчезновение. В это время появляется жених Нади — Ипполит Георгиевич, с которым она собиралась встретить праздник. Обнаружив в квартире постороннего, он устраивает невесте сцену ревности. Женя уходит, но, сообразив, что денег на обратный билет у него нет, возвращается, чтобы попросить взаймы денег на билет до Москвы. Сочувствуя незадачливому гостю, Надя разрешает ему позвонить с её домашнего телефона: сначала в аэропорт — узнать, когда первый самолёт в Москву; затем — чтобы объясниться со своей невестой, но та, услышав про Ленинград, бросает трубку.

Вынужденно находясь в одной квартире, друзья по несчастью постепенно начинают проникаться друг к другу симпатией. Ипполит возвращается к Наде, извиняется перед ней за свою ревность. Но, опять застав Женю, снова затевает новую сцену, Ипполит и Женя дерутся друг с другом, за что Надя выгоняет из дома обоих. Женя опять звонит в аэропорт, узнать расписание рейсов в Москву, объясняя Наде, что он не хочет торопиться с возвращением в Москву. Надя, надеясь поскорее отправить Женю домой, едет на такси на Московский вокзал и покупает ему билет на поезд до Москвы, но Женя выбрасывает билет через балкон. Некоторое время спустя в квартиру вваливается пьяный Ипполит. Хозяйка квартиры приходит в ужас. Незваный гость поражает своим сумасбродным поведением (принимает душ в пальто) и при этом логичными и близкими к истине рассуждениями о произошедшем этим вечером и ночью. Мокрый Ипполит уходит из Надиной квартиры.

Слова Ипполита производят впечатление на хозяйку дома — она вслух замечает, что будущего у них с Женей быть не может, а всё, что произошло этой новогодней ночью, было заблуждением. Женя самолётом возвращается в Москву. Женя сообщает ей, что он по чужой ошибке, вместо Павлика, улетел в Ленинград и там встретил другую женщину. Но он не надеется соединиться с любимой и останется, как и прежде, холостяком. Уставший после невероятной, суматошной ночи, Женя снова мирно спит, на этот раз уже в своей квартире. Женская рука открывает дверь: это Надя, прилетевшая в Москву. Она садится рядом со спящим Женей и с улыбкой смотрит на него. Проснувшись, он не сразу понимает, что видит любимую женщину наяву: Надя приехала, чтобы навсегда остаться с ним.

ДОДАТОК Б

Оригінальний вхідний текст для англійської мови

Seasoned con-man Nicky Spurgeon (Will Smith) goes to an upscale restaurant, where an inexperienced grifter, Jess Barrett (Margot Robbie), seduces him and then pretends they've been caught by her jealous husband. When the deception fails, Nicky advises them never to lose focus when faced with unexpected situations. Nicky follows Jess and convinces her to have a drink with him. Over drinks, he tells her the story of how his father killed his grandfather in a stand-off, explaining the tactic called a "Toledo Panic Button" which means that you shoot your partner to show you're loyal.

Jess follows Nicky to New Orleans, successful in persuading Nicky to take her under his wing, where she is also introduced to Nicky's crew as well, including the obese and profane Farhad (Adrian Martinez) and fellow con-man Horst (Brennan Brown). She picks a few pockets as a test, and soon Nicky and Jess develop a romantic relationship, upsetting Nicky, who was taught by his father to never become emotionally involved with anyone in their line of business. At the 17th Associated Football Franchise of America Championship Game at the Mercedes-Benz Superdome, Nicky gets into a round of increasingly extravagant bets with gambler Liyuan Tse (B.D. Wong), eventually losing all of the money the crew has earned.

To win it back, Nicky asks Tse to pick any player on or off the field and says that Jess will guess the number picked. A distraught Jess scans the field and notices Farhad wearing jersey number 55 and realizes it is another con. They take Tse for millions of dollars. Nicky explains to Jess how Tse had been programmed to pick 55 since he arrived, with subtle, subconscious prompts throughout his day. Afterward, Nicky, wary of his growing emotional involvement, leaves Jess by the side of the road with her cut. He instructs the driver to take her to the airport. Jess cries as her limo drives off, leaving Nicky to climb into another waiting car.

Three years later, Nicky is in Buenos Aires, working for billionaire motorsport team owner Rafael Garriga (Rodrigo Santoro). Garriga needs to beat a team headed by Australian businessman McEwen (Robert Taylor) to win the championship. Nicky will pretend to be a disgruntled technician on Garriga's team willing to sell Garriga's custom fuel use algorithm EXR. Instead, he will sell McEwen a bogus version which will slow their car down during the race. At a pre-race party, Nicky runs into Jess, who is now Garriga's girlfriend. After faking heavy drinking upon seeing Jess, Nicky has a convincing fight with Garriga in public and after being thrown out, is recruited by McEwen to provide the component.

Nicky begins pursuing Jess again, and they eventually rekindle their relationship. The head of Garriga's security entourage, Owens (Gerald McRaney), is suspicious and narrowly misses catching the two together. Nicky delivers the component to McEwen for three million euros but also sells it to the other teams for similar amounts.

Nicky and Jess attempt to return to the United States together. However, they are caught by Garriga's men and taken to Garriga's garage. Jess is tied up and her mouth is taped shut whilst Nicky is given a beating. Nicky has actually sold the real EXR to all of the various teams. Garriga is convinced that Jess had something to do with Nicky gaining access to EXR and begins to suffocate the gagged Jess by holding her nose. In order to save Jess, Nicky explains that he gained access to EXR by tricking Jess into believing he still had feelings for her. That the necklace he had given to Jess was equipped to secretly record Garriga's password and login information. He explains that Jess was conned and knew nothing about this. However, Jess then reveals that she was only trying to seduce Garriga in order to steal his valuable watch and to make Nicky jealous.

Nicky promises to come clean in order to spare Jess's life but Owens shoots him in the chest, causing a horrified Garriga to leave. Owens then reveals himself to be Nicky's father, Bucky and assures Jess that he avoided any major arteries. He simply employed the "Toledo Panic Button." Bucky then tapes up Nicky's wounds and draws excess blood out of his son's chest with a metal plunger so that he can breathe again. They flee the garage in Garriga's vehicle.

Bucky drives Nicky and Jess to the hospital to treat Nicky's punctured lung and departs with Nicky's money as a reminder of the consequences of losing focus. After he leaves, Nicky notices that Jess snatched Garriga's watch before he left the warehouse, and a smiling Nicky and Jess then go into the hospital together.