Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(повна назва)

Розробка та дослідження методів автоматизованого аналізу текстів в Web-додатках
(тема)

Виконав:
студент 2 курсу, групи СШІм-18-1
Демченко А. Є.
(прізвище, ініціали)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва освітньої програми)

Керівник проф. Філатов В. О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо -наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2019 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Демченко Андрію Євгенійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методів автоматизованого аналізу текстів в Web-додатках

затверджена наказом по університету від «4» листопада 2019 р. №1623 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «18» грудня 2019 р.

3. Вихідні дані до роботи Науково-технічні публікації та дані інтернет джерел з тематики атестаційної роботи, методичні вказівки до виконання атестаційної роботи магістра

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної області, збір даних, типи даних, методи збору даних, обробка даних, сегментування даних, лексимізація даних, маркування даних, стемінг даних, очистка даних, аналіз даних, алгоритм TextRank, алгоритм Luhn, алгоритм LSA, порівняння алгоритмів аналізу даних, постановка задачі, теоретичне дослідження, аналіз лексичних зв'язків, правила пошуку зв'язків, об'єднуючі слова, використання займенників, повторювані іменники, взаємини між словами, алгоритми пошуку лексичних зв'язків, оцінка алгоритму, алгоритм TextRank, побудова графа, обчислення ваги вершин графа, стиснення хешем, підрахунок відстані Левенштейна, подібність по Левенштейну, обчислення ваги ребер графа, підрахунок рангу пропозиції, експериментальне дослідження, мова програмування Python, технології розробки інтерфейсу користувача, HTML, CSS, JS, Vue, технологія розробки серверу Tornado, технологія розробки баз даних SQLite,

технології хмарного обчислення Amazon Web Services

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Рисунок 2.1 – Схема роботи алгоритму, Рисунок 2.2 – Виваженим, не направлений, повний граф, Рисунок 2.3 – Граф з підрахованою вагою вершин, Рисунок 2.4 – Граф з підрахованою вагою ребер, Рисунок 3.1 – Приклад програмного коду на мові програмування Python, Рисунок 3.2 – Приклад простого HTML-документа, Рисунок 3.3 – Приклад простого CSS-документа, Рисунок 3.4 – Приклад програмного коду на мові програмування JavaScript, Рисунок 3.5 – Приклад програмного коду з використанням фреймворку Vue, Рисунок 3.6 – Приклад програмного коду з використанням фреймворку Tornado, Рисунок 3.7 – Приклад програмного коду з використанням фреймворку SQLite

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Філатов В. О.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	4.11.19-5.11.19	виконано
2	Аналіз предметної області	6.11.19-10.11.19	виконано
3	Постановка завдання та узгодження з керівником	10.11.19-12.11.19	виконано
4	Дослідження методів автоматизованого аналізу текстів	12.11.19-25.11.19	виконано
5	Розробка алгоритму	26.11.19-4.12.19	виконано
6	Розробка програмного забезпечення	5.12.19-10.12.19	виконано
7	Написання пояснювальної записки	11.12.19-15.12.19	виконано
8	Попередній захист	16.12.19	виконано
9	Захист перед ЕК	18.12.19	

Дата видачі завдання 4 листопада 2019 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Філатов В. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 90 с., 11 рис., 3 табл., 2 дод., 21 джерел.

АНАЛІЗ ДАНИХ, АНАЛІЗ ТЕКСТІВ, ОБРОБКА ТЕКСТІВ, РАНЖУВАННЯ ТЕКСТІВ, СКОРОЧЕННЯ ТЕКСТІВ, WEB-ДОДАТКИ.

Об'єктом дослідження даної атестаційної роботи є сучасні методи автоматизованого аналізу даних у Web-додатках.

Предметом дослідження даної атестаційної роботи є сучасні методи автоматизованого аналізу текстів у Web-додатках.

Метою даної атестаційної роботи є розробка алгоритмів автоматизованого аналізу текстів у Web-додатках.

Методами дослідження даної атестаційної роботи є спеціалізована література та консультації з експертами у сфері автоматизованого аналізу текстів у Web-додатках.

РЕФЕРАТ

Пояснительная записка: 90 с., 11 рис., 3 табл., 2 прил., 21 источников.

АНАЛИЗ ДАННЫХ, АНАЛИЗ ТЕКСТОВ, ОБРАБОТКА ТЕКСТОВ, РАНЖИРОВАНИЕ ТЕКСТОВ, СОКРАЩЕНИЕ ТЕКСТОВ, WEB-ПРИЛОЖЕНИЯ.

Объектом исследования данной аттестационной работы являются современные методы автоматизированного анализа данных в Web-приложениях.

Предметом исследования данной аттестационной работы являются современные методы автоматизированного анализа текстов в Web-приложениях.

Целью данной аттестационной работы является разработка алгоритмов автоматизированного анализа текстов в Web-приложениях.

Методами исследования данной аттестационной работы является специализированная литература и консультации с экспертами в области автоматизированного анализа текстов в Web-приложениях.

ABSTRACT

Explanatory note: 90 p., 11 fig., 3 tabl., 2 ann., 21 sources.

DATA ANALYSIS, TEXT ANALYSIS, TEXT PROCESSING, TEXT RANKING, TEXT SUMMARIZATION, WEB APPLICATIONS.

The object of this certification work is the modern methods of automated data analysis in Web applications.

The subject of this certification work is modern methods of automated text analysis in Web applications.

The purpose of this certification work is to develop algorithms for automated text analysis in Web applications.

The research methods for this certification work are specialized literature and consultations with experts in the field of automated text analysis in Web applications.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

ACE – набір текстових даних, з маркованими семантичними зв’язками між реченнями та словами у них, створений для роботи і обробки природних мовних запитів;

ANSI – символи, що включаються в текст і використовуються для зміни форматування, кольору, та інших налаштувань виводу тексту на текстовий термінал;

ASCII – система кодів, у якій числа від 0 до 127 включно поставлені у відповідність літерам, цифрам і символам пунктуації. Це 7-бітний код, де восьмий біт часто використовується для забезпечення відповідності чи додаткових символів;

AWS – платформа хмарних обчислень, що дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп'ютерів, який завжди доступний через Інтернет;

CSS – це спеціальна мова стилю сторінок, що використовується для опису їх зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних;

DOM – специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами, що визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленням документів у вигляді дерева;

DNS – ієрархічна розподілена система перетворення імені хоста, комп'ютера або іншого мережевого пристрою в IP-адресу, де кожен комп'ютер в Інтернеті має свою власну унікальну адресу — число, яке складається з чотирьох або шістнадцяти байтів;

EC2 – простий веб-інтерфейс сервісу дозволяє отримати доступ до обчислювальних потужностей і повний контроль над обчислювальними ресурсами, а також доступне середовище для роботи;

HTML – стандартна мова розмітки для створення Web-додатків, що описує структуру веб-сторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа;

HTTP – протокол передачі даних, що використовується в комп'ютерних мережах, де основним призначенням протоколу HTTP є передача Web-додатків, хоча за допомогою нього успішно передаються і інші файли, які пов'язані Web-додатками;

ISO – міжнародна організація, що займається випуском стандартів для сприяння розвитку стандартизації та суміжних видів діяльності у світі з метою забезпечення розвитку співробітництва в інтелектуальній, науково-технічної та економічної областях;

JS – динамічна, об'єктно-орієнтована прототипна мова програмування, що найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки;

JSX – є розширенням до синтаксису мови JS, що забезпечує спосіб структурування візуалізації компонентів за допомогою синтаксису, знайомого багатьом розробникам;

LD – міра відмінності двох послідовностей символів, що обчислюється як мінімальна кількість операцій вставки, видалення і заміни, необхідних для перетворення одної послідовності в іншу;

LSA – метод обробки інформації природною мовою, зокрема, дистрибутивної семантики, що дозволяє аналізувати взаємозв'язок між

набором документів і термінами, які в них зустрічаються, шляхом створення набору понять;

SNS – сервіс посилки повідомлень між мережевими компонентами і зовнішніми компонентами, який забезпечує масову доставку повідомлень в складній мережі і являє собою єдину шину, яка розсилає повідомлення найрізноманітніших пристроїв і платформ;

SVD – один з важливих методів розкладу матриці з дійсними або комплексними числами, що є узагальненням власного розкладу матриці невід'ємно визначеної нормальної матриці, наприклад, симетричної матриці з додатними власними значеннями на матрицю розміру як узагальнення полярного розкладу;

SQL – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними базами даних, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних;

SQS – розподілена служба черги повідомлень, що підтримує програмне відправлення повідомлень через програми веб-сервісів як спосіб спілкування через Інтернет і призначена для забезпечення високо масштабованої черги повідомлень;

VPC – комерційна послуга хмарних обчислень, яка надає користувачам віртуальну приватну хмару, через віртуальну приватну мережу де замовник може призначити IP-номери на свій вибір із однієї чи декількох підмереж;

URL – стандартизована адреса певного ресурсу, такого як документ, чи зображення в інтернеті, створений для використання у мережі Інтернет. Включає в себе назву протоколу доступу і шлях до ресурсу, формат якого залежить від схеми доступу.

ВСТУП

Прогрес технічної думки людини завжди найбільш швидко відображався саме в сфері інформаційних технологій. Способи збору, зберігання, систематизації, розповсюдження інформації проходять червоною ниткою через всю історію людства. Прориви будь то в сфері технічних, або гуманітарних наук, так чи інакше, відгукувалися на інформаційних технологіях. Пройдений людством цивілізаційний шлях, це низка послідовних кроків удосконалення способів зберігання і передачі даних. Саме еволюційний становлення нашого виду кинуло людей в обійми аналогового сприйняття оточуючого їх простору, що багато в чому і вирішило долю нашого технологічного становлення.

За тисячі років еволюції людство перейшло від глиняних табличок та інших аналогових форматів до хмарних сховищ та цифрових технологій обробки даних. Проте будь-який прогрес тягне за собою і приховані загрози, що потрібно вирішувати. Так, у XXI столітті людство за один тиждень створює та обробляє більше інформації, ніж у XX за десять років. Окрім проблем зі зберіганням щодня зростаючої кількості даних, також існує проблема їх обробки. Саме цим і займається інтелектуальних аналіз даних.

Інтелектуальних аналіз даних є найважливішим засобом отримання нових знань не тільки в галузі природничих та технічних наук, а й в економіці, соціології, політиці, психології, літературознавстві і в інших галузях. Ці дослідження дають критерії оцінки обґрунтованості та прийнятності на практиці будь-яких теорій і теоретичних припущень. Інтелектуальних аналіз даних спрямований, як правило, на побудову математичної моделі досліджуваного об'єкта або явища, а також на отримання відповіді на питання: «Чи достовірні наявні дані в межах

необхідної точності або допусків?». Сама ж математична модель в залежності від цілей (дослідження, управління, контроль) може бути використана для різних цілей: для предметно-смислового аналізу об'єкта чи явища, прогнозування їх стану в різних умовах функціонування, управління ними в конкретних ситуаціях, оптимізації окремих параметрів, а також для вирішення якихось інших специфічних завдань. Кінцевою метою будь-якої обробки даних є висунення гіпотез про клас і структурі математичної моделі досліджуваного явища, визначення складу і обсягу додаткових вимірів, вибір можливих методів подальшої статистичної обробки та аналіз виконання основних передумов, що лежать в їх основі.

Слід зазначити, що в залежності від кінцевих цілей дослідження, складності досліджуваного явища і рівня апріорної інформації про нього, обсяг задач, які виконуються в ході попередньої обробки, може істотно змінюватися. Те ж саме можна сказати і про співвідношення цілей і завдань, які вирішуються при попередній обробці і на наступних етапах аналізу, спрямованих на побудову моделі явища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Інтелектуальний аналіз даних – це напрямок інформаційних технологій, що охоплює всю область проблем, пов'язаних з отриманням знань з масивів даних. В основі інтелектуального аналізу даних лежить ідея активного застосування математичних методів, таких як оптимізація, генетичні алгоритми, розпізнавання образів, статистика.

Основу методів складають всілякі методи класифікації, моделювання і прогнозування, засновані на застосуванні дерев рішень, штучних нейронних мереж, генетичних алгоритмів, еволюційного програмування, асоціативної пам'яті, нечіткої логіки. Сюди також відносять статистичні методи: дескриптивний аналіз, кореляційний і регресійний аналіз, факторний аналіз, дисперсійний аналіз, компонентний аналіз, дискримінантний аналіз, аналіз часових рядів, аналіз виживаності, аналіз зв'язків. Такі методи, проте, припускають деякі апріорні уявлення про аналізованих даних, що виникає певна розбіжність з цілями виявлення раніше невідомих нетривіальних і практично корисних знань.

Одне з найважливіших призначень методів інтелектуального аналізу даних полягає в наочному поданні результатів обчислень, що дозволяє використовувати інструментарій людьми, які не мають спеціальної математичної підготовки.

1.1 Збір даних

Збір даних – це процес пошуку та отримання інформації про цікаві змінні у встановленому систематизованому порядку, який дає змогу відповідати на заявлені дослідницькі запитання, перевіряти гіпотези та оцінювати результати. Компонент збору даних у дослідженнях є загальним

для всіх галузей дослідження, включаючи фізичні та суспільні науки, гуманітарні науки, бізнес тощо. Хоча методи відрізняються за дисципліною, акцент на забезпеченні точного та чесного збору залишається однаковим. Мета всього збору даних – отримати якісні дані, які потім дозволяють побудувати переконливу та достовірну відповідь на поставлені питання. Незалежно від галузі дослідження або переваги визначення даних (кількісних, якісних), точний збір даних має важливе значення для підтримки цілісності досліджень. Як вибір відповідних інструментів збору даних (існуючих, модифікованих, чи нещодавно розроблених), так і чітко окреслені інструкції щодо їх правильного використання знижують ймовірність виникнення помилок. Збір даних – один з найважливіших етапів проведення дослідження, це дуже вибаглива робота, яка потребує ретельного планування, наполегливої роботи, терпіння, наполегливості та багато іншого, щоб успішно виконати завдання.

1.1.1 Типи даних

Існує кілька основних типів даних, і важливо знати, як можна працювати з кожним із них, щоб мати можливість займатися збором даних у формі, яка найкраще задовольняє ваші потреби. Існує багато класифікацій типів даних, але ми зупинимося на таких рівнях вимірювання, як якісні та кількісні.

Якісні дані мають здебільшого не числовий характер, і зазвичай мають описовий чи номінальний характер. Це означає, що зібрані дані представлені у формі слів і речень. Часто такі дані фіксують почуття, емоції чи суб'єктивне сприйняття чогось. Якісні підходи спрямовані на те, щоб вирішити питання «як» і «чому» і, як правило, використовувати

неструктуровані методи збору даних для повного вивчення теми. Якісні запитання мають відкритий характер. Якісні методи включають фокус-групи, групові дискусії та інтерв'ю. Якісні підходи хороші для подальшого вивчення наслідків та непередбачуваних наслідків програми. Однак вони дорогі та трудомісткі для впровадження. Крім того, результати не можуть бути узагальнені для учасників поза програмою і є показовими лише для залученої групи. Якісні методи збору даних відіграють важливу роль в оцінці впливу, надаючи інформацію, корисну для розуміння процесів, що стоять за спостережуваними результатами, та оцінки змін у сприйнятті людей їхнього самопочуття. Крім того, якісні методи можуть бути використані для поліпшення якості кількісних оцінок на основі опитування, сприяючи формуванню гіпотези оцінювання.

Кількісні дані мають чисельний характер і можуть бути обчислені математично. Кількісний показник даних використовує різні шкали, які можна класифікувати як номінальну шкалу, порядкову шкалу, інтервальну шкалу та шкалу відношення. Часто такі дані включають вимірювання чогось. Вони використовують систематизований стандартизований підхід і використовують методи, такі як опитування та задавання питань. Переваги в кількісному відношенні мають перевагу в тому, що вони дешевші для впровадження, стандартизовані, щоб порівняння можна було легко зробити, і розмір ефекту зазвичай можна виміряти. Однак кількісні підходи обмежені у своїх можливостях для дослідження та пояснення подібності та несподіваних відмінностей. Важливо зазначити, що для однорангових програм кількісний підхід до збору даних часто виявляється важким для впровадження агентств, оскільки відсутність необхідних ресурсів для забезпечення суворого виконання опитувань, а часто спостерігаються низькі показники участі та втрати для подальшого спостереження. Кількісні методи збору даних покладаються на випадкові вибірки та

структуровані інструменти збору даних, які підходять до різноманітного досвіду в заздалегідь визначені категорії відповідей. Вони дають результати, які легко узагальнити та порівняти. Якщо наміром є узагальнення від учасників дослідження до більшої сукупності, дослідник використовуватиме вибіркові ймовірності для відбору учасників.

1.1.2 Методи збору даних

Дані, зібрані з перших рук, відомі як первинні дані. Первинні дані ще не опубліковані і є більш надійними, достовірними та об'єктивними. Первинні дані не були змінені, тому їх обґрунтованість більша, ніж вторинних даних. У статистичних опитуваннях необхідно отримувати інформацію з первинних джерел та працювати над первинними даними. Наприклад, статистичні записи жіночого населення в країні не можуть базуватися на газетних, журнальних та інших друкованих джерелах. Дослідження може проводитися без вторинних даних, але дослідження, засноване лише на вторинних даних, є найменш надійним і може мати упередження, оскільки вторинні дані були вже маніпульовані людиною. По-перше деякі з таких джерел є старими, по-друге, вони містять обмежену інформацію, тому вони можуть бути оманливими та упередженими.

Джерела первинних даних обмежені, і часом стає важко отримати дані з первинного джерела через дефіцит населення чи відсутність співпраці. Далі наведено деякі джерела первинних даних.

Експерименти потребують штучного або природного середовища, в якому слід проводити логічне дослідження для збору даних. Експерименти більше підходять для медицини, психологічних досліджень, харчування та

для інших наукових досліджень. В експериментах експериментатору слід тримати контроль над впливом будь-якої сторонньої змінної на результати. Опитування є найбільш поширеним методом у соціальних науках, менеджменті, маркетингу та психології певною мірою. Опитування можна проводити різними методами.

Анкета це найбільш часто використовуваний метод опитування. Це перелік питань або відкритих, або закритих, на які респонденти дають відповіді. Анкетування можна проводити по телефону, пошті, в громадській місцевості або в інституті, електронною поштою або факсом та іншими методами.

Інтерв'ю являє собою особиста розмова з респондентом. В інтерв'ю основна проблема виникає, коли респондент навмисно приховує інформацію, інакше вона є глибоким джерелом інформації. Інтерв'юер може не лише записувати твердження, які говорить респондент, але він також може спостерігати мову тіла, вирази та інші реакції на питання. Це дає можливість інтерв'юеру легко робити висновки.

Спостереження можна проводити, повідомляючи людині, яка спостерігає, що за ним спостерігається або не повідомляючи про це. Спостереження можна проводити і в природних умовах, і в штучно створеному середовищі.

Дані, зібрані з джерела, які вже були опубліковані в будь-якій формі, називаються вторинними даними. Огляд літератури в будь-якому дослідженні базується на вторинних даних. Його збирає хтось інший з якоюсь іншою метою. Наприклад, дані перепису, які використовуються для аналізу впливу освіти на вибір кар'єри та заробіток. Загальні джерела вторинних даних для суспільствознавства включають переписи, організаційні записи та дані, зібрані за допомогою якісних методологій або

якісних досліджень. Вторинні дані є важливими, оскільки неможливо провести нове опитування, яке може адекватно фіксувати минулі зміни.

Вторинні дані можуть бути менш достовірними, але їх значення все ще існує. Іноді важко отримати первинні дані і в цих випадках отримання інформації з вторинних джерел простіше. Іноді первинних даних не існує і доводиться обмежуватися дослідженнями з вторинними даними. Іноді є первинні дані, але респонденти не бажають їх розкривати і в такому випадку, вторинні дані можуть бути достатніми. Очевидною перевагою використання вторинних даних є те, що значна частина необхідних робіт вже проведена. Наприклад, огляди літератури, дослідження конкретних випадків, можливо, було проведено, опубліковані тексти та статистика вже могли бути використані в іншому місці, також було використано просування засобів масової інформації та особисті контакти. Це багатство роботи означає, що вторинні дані, як правило, мають заздалегідь встановлену ступінь достовірності та надійності, які не потрібно переглядати досліднику, який повторно використовує такі дані. Крім того, вторинні дані також можуть бути корисними при розробці дослідження наступних первинних досліджень і можуть забезпечити базу, з якою можна порівняти зібрані результати первинних даних. Тому завжди розумно починати будь-яку дослідницьку діяльність з огляду вторинних даних.

1.2 Обробка даних

Вельми часто отримані дані, які використовуються в подальшому для обробки алгоритмами інтелектуального аналізу даних для вирішення прикладних завдань мають невисоку якість. Очищення даних у статистичній вибірці – одна з найбільш актуальних завдань аналізу. На її виконання витрачається до 80% всього часу, відведеного на проект. Деякі

брудні дані взагалі не піддаються автоматичному очищенню через різнорідність і специфічність інформації. Крім того, потрібно відзначити, що відсутні універсальні способи усунення помилок даних вибірки і кожна ситуація, виходячи з її мінливості, вимагає застосування свого методу або сукупності методів.

Однією з головних проблем аналізу текстів є велика кількість слів у документі. Якщо кожне з цих слів аналізувати, то час пошуку нових знань різко зросте і навряд чи буде задовольняти вимогам користувачів. У той же час очевидно, що не всі слова в тексті несуть корисну інформацію. Крім того, в силу гнучкості природних мов формально різні слова насправді означають однакові поняття. Таким чином, видалення неінформативних слів, а також приведення близьких за змістом слів до єдиної формі значно скорочують час аналізу текстів. Усунення описаних проблем виконується на етапі попередньої обробки тексту.

1.2.1 Сегментація даних

Сегментація тексту – це процес поділу письмового тексту на змістові одиниці, такі як слова, речення чи теми. Термін застосовується як до психічних процесів, які використовуються людиною при читанні тексту, так і до штучних процесів, реалізованих у комп’ютерах, що є предметом природного опрацювання мови. Проблема нетривіальна, оскільки, хоча деякі писемні мови мають явні межові маркери слова, такі як пробіли слів у письмовій англійській мові та відмітні початкові, медіальні та заключні літери форми арабської мови, такі сигнали іноді неоднозначні та відсутні у всіх письмових мовах.

Тематичний аналіз складається з двох основних завдань: ідентифікація теми та сегментація тексту. Хоча перший – це проста

класифікація конкретного тексту, останній випадок передбачає, що документ може містити кілька тем, і завданням комп'ютеризованої сегментації тексту може бути автоматичне виявлення цих тем і сегментація тексту відповідно. Межі теми можуть бути видні з назв розділів та абзаців. В інших випадках потрібно використовувати методи, подібні до тих, що використовуються в класифікації документів.

Сегментування тексту на теми або дискурсні звороти може бути корисним у деяких природних завданнях обробки: воно може значно покращити пошук інформації чи розпізнавання мови. Він також необхідний у системах виявлення та відстеження тем та проблемах узагальнення тексту.

Сегментація тексту – це проблема поділу рядка писемної мови на складові речення. В англійській та деяких інших мовах, використовуючи розділові знаки, особливо символ повної зупинки. Однак навіть в англійській мові ця проблема не є тривіальною через використання символу повної зупинки для скорочень, які можуть або не можуть також закінчувати речення. При обробці простого тексту таблиці скорочень, що містять періоди, можуть запобігти неправильному призначенню меж речення. Як і у випадку сегментації слів, не всі письмові мови містять розділові знаки, які корисні для розпізнавання меж речення.

Сегментація слів – це проблема поділу рядка писемної мови на її складові слова. В англійській та багатьох інших мовах, що використовують певну форму латинського алфавіту, простір є гарним наближенням розділювача слів, хоча це поняття має обмеження через мінливість, з якою мови емічно розглядають колокації та сполуки. Багато англійських складених іменників написано по-різному з відповідною варіацією того, чи вважають мовці їх як іменникові фрази чи одиничні іменники. Існують тенденції у встановленні норм, наприклад, що відкриті сполуки часто

мають тенденцію врешті затвердіти за допомогою широко розповсюдженої конвенції, але зміни залишаються системними. Навпаки, німецькі складені іменники демонструють меншу кількість орфографічних змін, при цьому затвердіння є сильнішою нормою.

Однак еквівалент символу простору слова не знайдений у всіх написаних сценаріях, і без нього сегментація слова є складною проблемою. Мови, які не мають тривіального процесу сегментації слів, включають китайську, японську, де речення, але не слова, розмежовані, тайська та лаоська, де словосполучення та речення, але не слова, розмежовані, та в'єтнамська, де склади, але не слова, розмежовані.

Якщо абзаци можна розглядати, як структурні одиниці об'єкта, тоді пропозиції можна вважати смисловими одиницями. Подібно до абзацу, виражає єдину ідею, пропозицію містить закінчену думку, яку ми б хотіли сформулювати і висловити.

Процес сегментації можна описати таким чином: уявіть курсор, який рухається по тексту, проходячи один символ за раз. Для кожної позиції курсора в заданому порядку застосовуються правила, що складаються з шаблонів «До» і «Після», які перевіряють, чи підходить шаблон «До» до тексту зліва і шаблон «Після» до тексту праворуч від курсора. Якщо будь-яка з правил спрацьовує, то або курсор переходить до наступного символу без початку нового сегмента або в поточній позиції курсору створюється новий сегмент.

Правило Break Rule розділяє вихідний текст на сегменти. Наприклад, пропозиція «Чи варто було це робити? Не впевнений.» має бути розділене на два сегменти. Тобто, потрібно визначити правило розриву для символу «?», за яким слідує пробіл і слово з великої літери.

Правило Exception Rule визначає, в якій частині тексту не повинна відбуватися сегментація. Незважаючи на точку, словосполучення «Mrs.

Dalloway» не потрібно розділяти на два сегменти, тому потрібно визначити правило-виняток для рядка «Mrs» з точкою справа.

1.2.2 Лексемізація даних

Ми визначили пропозиції, як смислові одиниці, а абзаци, як структурні одиниці документа, але також варто відзначити і лексеми – синтаксичні одиниці мови, що кодують семантичну інформацію у вигляді послідовності символів. Лексемізація – процес вилучення лексем по пробільним символам і знаків пунктуації. Як і виділення пропозицій, коректне визначення лексем не завжди просте завдання.

1.2.3 Маркування даних

Отримавши доступ до лексем всередині пропозицій в абзацах, потрібно позначити кожну лексему відповідної їй частиною мови. Частини мови вказують на роль слова в контексті пропозицій. В англійській мові, як і в багатьох інших одне і те ж слово може грати різні ролі і хотілося б мати можливість розрізняти їх. Маркування частинами мови полягає в додаванні до кожної лексеми відповідного тега, що несе інформацію про визначення слова і його ролі в поточному контексті.

1.2.4 Стемінг даних

Стемінг – це процес знаходження основи слова для заданого вихідного слова. Основа слова не обов'язково збігається з морфологічним коренем слова. Стемінг застосовується в пошукових системах для

розширення пошукового запиту користувача і є частиною процесу нормалізації тексту.

Простий стемер шукає флективну форму в таблиці пошуку. Переваги цього підходу полягає в його простоті, швидкості, а також легкості обробки винятків. До недоліків можна віднести те, що все флективні форми повинні бути явно перераховані в таблиці: нові або незнайомі слова не будуть оброблятися, навіть якщо вони є правильними, а також проблемою є те, що таблиця пошуку може бути дуже великою. Для мов з простої морфологією на зразок англійської розміри таблиць невеликі, але для сильно флективних таблиця може мати сотні можливих флективних форм для кожного кореня.

Таблиці пошуку, використовувані в стемері, як правило, генеруються в напівавтоматичному режимі. Наприклад, для англійського слова «run» автоматично будуть згенеровані форми «running», «runs», «runned» і «runly». Останні дві форми є допустимими конструкціями, але вони навряд чи з'являться в звичайному тексті англійською мовою.

Алгоритми усічення закінчень набагато ефективніше, ніж алгоритми повного перебору. Для розробки таких алгоритмів потрібен програміст, який досить добре розбирається в лінгвістиці, зокрема морфології, а також вміє кодувати «правила усічення». Алгоритми усічення закінчень неефективні для виняткових ситуацій (наприклад, «ran» і «run»). Рішення, отримані алгоритмами усічення закінчень, обмежуються тими частинами мови, які мають добре відомі закінчення і суфікси з деякими винятками. Це є серйозним обмеженням, так як не всі частини мови мають добре сформульований набір правил. Алгоритми усічення префікса також можуть бути реалізовані. Однак не у всіх мовах слова мають приставки і суфікси.

Алгоритми усічення закінчень можуть відрізнятися в результатах за цілою низкою причин. Однією з таких причин є особливість алгоритму: має бути слово на виході алгоритму реальним словом в даному мовою. Деякі підходи не вимагають наявності слова в відповідної мовної лексики. Крім того, деякі алгоритми ведуть базу даних всіх відомих морфологічних коренів, які існують як реальні слова. Дані алгоритми перевіряють наявність терма в базі даних для прийняття рішення. Як правило, якщо терм не знайдений, виконуються альтернативні дії. Дані альтернативні дії можуть використовувати дещо інші критерії для прийняття рішення. Неіснуючий терм може послужити застосування альтернативних правил усічення.

Може бути так, що два або більше правила усічення застосовуються для одного і того ж вхідного терма, що створює невизначеність щодо того, яке правило застосувати. Алгоритм може визначити пріоритет виконання таких правил (за допомогою людини або стохастичним способом). Або алгоритм може відхилити одне з правил, якщо воно призводить до неіснуючого терму, тоді як інша – ні. Наприклад, для англійського терма «friendlies» алгоритм може визначити суфікс «ies», застосувати відповідне правило і отримати результат «friendl». Терм «friendl», швидше за все, не буде знайдений у лексиконі і, отже, дане правило буде відкинуто.

Одним з поліпшень алгоритмів усічення закінчень є використання підстановки суфіксів і закінчень. Подібно правилом усічення, правило підстановки замінює суфікс або закінчення альтернативним. Наприклад, може існувати правило, яке замінює «ies» на «y». Оскільки правила усічення призводять до неіснуючого терму в лексиконі, то правила підстановки вирішують цю проблему. У цьому прикладі, «friendlies» перетвориться в «friendly» замість «friendl».

Зазвичай застосування даних правил носить циклічний або рекурсивний характер. Після першого застосування правила підстановки для цього прикладу алгоритм виробляє вибір наступного правила для терма «friendly», в результаті якого буде виявлено і визнано правило усічення суфікса «ly». Таким чином, терм «friendlies» за допомогою правила підстановки стає термо «friendly», який після застосування правила усічення, стає термо «friend».

Даний приклад допомагає продемонструвати різницю між методом, заснованим на правилах, і методом «грубої сили». За допомогою повного перебору алгоритм буде шукати терм «friendlies» в наборі з сотні тисяч флективних словоформ і в ідеалі знайде відповідну основу «friend». У методі, заснованому на правилах, відбувається послідовне виконання правил, в результаті чого виходить те саме рішення. Швидше за все, метод на основі правил буде швидше.

1.2.5 Очистка даних

Стоп-слова – це слова, які відфільтровані перед обробкою природних мовних даних. Не існує єдиного універсального списку стоп-слів, використовуваних усіма засобами обробки природних мов, і справді не всі інструменти навіть використовують такий список. Деякі інструменти уникають видалення стоп-слів для підтримки пошуку фрази.

Будь-який набір слів може бути обраний як стоп-слова для заданої мети. Для деяких пошукових систем це декілька найпоширеніших коротких функціональних слів, таких як, є, на, який і далі. У цьому випадку зупинка слів може спричинити проблеми під час пошуку фраз, що включають їх, особливо в назвах. Інші пошукові системи видаляють деякі найпоширеніші слова, включаючи лексичні слова, наприклад «хочу», із

запиту з метою підвищення продуктивності. У оптимізації пошукових систем стоп-слова представляють інтерес, оскільки не корисно ставити фрази, включаючи їх.

1.3 Аналіз даних

Узагальнення тексту стосується техніки скорочення довгих фрагментів тексту. Наміром є створення цілісного та вільного резюме, що містить лише основні моменти, викладені в документі. Автоматичне узагальнення тексту є поширеною проблемою в машинному навчанні та обробці природних мов.

При великій кількості даних, що циркулюють у цифровому просторі, виникає потреба розробити алгоритми інтелектуального аналізу даних, які дозволять автоматично скорочувати більш довгі тексти та доставляти точні зведення, які можуть вільно передавати призначені повідомлення.

Крім того, застосування узагальнення тексту скорочує час читання, прискорює процес дослідження інформації та збільшує кількість інформації, яка може вміститися в певній місцевості.

Є два основні типи узагальнення тексту: метод узагальнення тексту і метод абстрагування тексту.

Методика узагальнення тексту екстрактивного тексту передбачає витягнення ключових фраз із вихідного документа та їх поєднання для створення резюме. Видобуток проводиться відповідно до визначеної метрики без внесення змін до текстів.

Під час узагальнення на основі видобутку підмножина слів, які представляють найважливіші моменти, витягується з фрагмента тексту та

поєднується для створення резюме. Подумайте про це як підсвічування, яка вибирає основну інформацію з вихідного тексту.

У машинному навчанні екстрактивне узагальнення зазвичай передбачає зважування суттєвих розділів речень та використання результатів для створення підсумків.

Різні типи алгоритмів і методів можна використовувати для вимірювання ваги речень, а потім класифікувати їх відповідно до їх релевантності та схожості один з одним і надалі з'єднувати їх для створення підсумків.

Техніка абстрагування тягне за собою перефразовування та скорочення частин вихідного документа. Якщо абстрактне застосування застосовується для узагальнення тексту в проблемах глибокого навчання, воно може подолати граматичні невідповідності екстрактивного методу.

Під час узагальнення на основі абстракції застосовуються передові методи глибокого навчання, щоб перефразовувати та скорочувати оригінальний документ, як і люди. Подумайте про це як ручку, яка створює нове речення, яке може не входити до вихідного документа.

Оскільки абстрактні алгоритми машинного навчання можуть генерувати нові фрази та речення, які представляють найважливішу інформацію з вихідного тексту, вони можуть допомогти у подоланні граматичних неточностей методів вилучення.

Незважаючи на те, що абстрагування краще підходить під час узагальнення тексту, розробка його алгоритмів вимагає складних методів глибокого навчання та складного моделювання мови.

Для отримання правдоподібних результатів підходи до узагальнення на основі абстракції повинні вирішувати широкий спектр проблем ынтелектуального аналызу даних, таких як генерація природних мов, семантичне представлення та перестановка висновку.

Таким чином, підходи до узагальнення тексту все ще широко популярні. У цій статті ми зупинимося саме на цих методах.

1.3.1 Алгоритм TextRank

Алгоритми ранжування на основі графів – це, по суті, спосіб вирішити важливість вершини в межах графіка на основі про інформацію, отриману із структури графіків. У цьому розділі ми розглянемо два алгоритми ранжування на основі графіків, які раніше виявилися успішними у ряді проблем ранжирування. Ці алгоритми можуть бути адаптовані до ненаправлених або зважених графіків, які особливо корисні в контексті текстових додатків для ранжування.

Щоб застосувати TextRank, спочатку нам потрібно побудувати графік, пов’язаний з текстом, де вершини графа є репрезентативними для одиниць, що підлягають ранжируванню. Для завдання вилучення речень метою є ранжування цілих речень, а тому до графа додається вершина для кожного речення в тексті.

Графік сильно пов’язаний із вагою, пов'язаною з кожним краєм, що вказує на міцність зв’язків, встановлених між різними парами речень у тексті. Тому текст представлений у вигляді зваженого графіка. Після запуску алгоритму ранжирування на графіку речення сортуються у зворотному порядку їх оцінки, а речення, що виходять з рейтингу, вибираються для включення до зведення. Речення з найвищим рангом підбираються для включення в конспект.

Узагальнення на основі видобутку зазвичай дає підсумки, які читати дещо неприваблиро. У тексті бракує потоку, оскільки витягнуті частини, зазвичай речення, взяті з різних частин оригінального тексту. Наприклад, це може призвести до дуже раптових зрушень теми. Ідея представленого

способу вилучення речень, які утворюють шлях, коли кожне речення є аналогічним попередньому, полягає в тому, що отримані підсумки, сподіваємось, мають кращий потік. Однак цю якість досить важко оцінити. Оскільки підсумки все ще є витягами, високої якості резюме все ж не слід очікувати.

Коли текст повинен бути узагальнений, його спочатку розбивають на речення та слова. Речення стають вузлами графіка. Подібні один до одного вироки мають між собою перевагу. Тут подібність просто означає накладення слів, хоча можна застосовувати й інші заходи. Таким чином, якщо два речення мають принаймні одне спільне слово, між ними буде ребро. Звичайно, багато слів неоднозначні, а наявність відповідного слова не гарантує подібності. Оскільки всі речення походять з одного документа, а слова, як правило, менш неоднозначні в одному тексті, ця проблема дещо пом'якшується.

Усі речення також мають ребро до наступного речення. На ребра надаються витрати (або ваги). Чим більше подібність двох речень, тим менша вартість ребра. Чим далі розділені речення в оригінальному тексті, тим вище вартість ребра. Для сприяння включенню «цікавих» речень, усі речення, які вважаються відповідними документу за класичними методами узагальнення, знижують витрати на всі ребра, що ведуть до них.

Оскільки подібність ґрунтується на кількості спільних слів між двома реченнями, довгі речення мають більший шанс бути подібними до інших речень. Використання довгих речень часто добре з точки зору гладкості тексту. Узагальнення з багатьма короткими реченнями мають більший шанс на різкі зміни, оскільки є більше розривів речень.

Коли графік побудований, підсумок створюється шляхом взяття найкоротшого шляху, який починається з першого речення оригінального тексту та закінчується останнім реченням. Оскільки оригінальний текст

також починається і закінчується в цих положеннях, це дасть плавний, але коротший набір речень між цими двома пунктами.

N найкоротших шляхів знаходять просто запустившись у вузол старту і додавши всі шляхи довжиною один до черги пріоритету, де значенням пріоритету є загальна вартість шляху. Потім вивчається найдешевший шлях, і якщо він не закінчується на кінцевому вузлі, усі шляхи, що починаються з цього шляху та містять ще одне ребро, також додаються до черги пріоритетів. Шляхи з петлями відкидаються. Щоразу, коли в кінцевому вузлі закінчується поточний найкоротший шлях, інший найкоротший шлях знайдено, і пошук продовжується до тих пір, поки не буде знайдено N найкоротших шляхів.

1.3.2 Алгоритм Luhn

Алгоритми на основі особливостей використовуються для аннотацій тексту та побудовані на основі пошуку специфічних особливостей у тексті. Це може бути весь текст, абзац, речення чи навіть словосполучення. Такий тип алгоритму підходить для вузьких доменних областей та допомагає визначати та підкреслювати схожі риси у різних текстах. Однією з таких областей є анотація відгуків на сайтах з різними товарами.

Алгоритм Luhn був вперше запропонований у документі 1958 року, написаному Гансом Пітером Луном. Як було сказано раніше, алгоритм Luhn заснований на тому, що люди – це створіння звички, і вони будуть повторювати ключові слова в документі. Що ще важливіше, він вважає, що ключові слова, якими користується автор, чітко визначені і являють собою єдине поняття. Навіть якщо автор намагається використовувати розумні синоніми для свого ключового слова, вони з часом закінчаться і перейдуть

до використання найкращого слова, яке визначає поняття, яке буде ключовим словом, яке найбільше повторюється.

Закінчивши із думкою про те, що автор буде повторюватись із використанням обмеженої кількості ключових слів для передачі значення, ми можемо почати ранжувати речення на основі частоти та близькості ключових слів у реченні. Для визначення ваги речення ми спочатку шукаємо значущі слова у реченні, потім беремо підмножину слів у реченні, причому перше і останнє слово в підмножині є значущим словом. Підмножина закривається, коли перед наступним використанням значущого слова присутні чотири або п’ять незначних слів. У межах підмножини ми зараз підраховуємо кількість присутніх значущого слова, а потім ділимо на кількість загальних слів у підмножині. Це число буде вагою, наданим цьому реченню. Якщо дане речення є досить довгим, щоб містити кілька таких підмножин значущих слів, ми просто приймаємо більш високий бал підмножини як вагу речення. Щоб створити автоматичне вилучення, нам потрібно взяти найвищі x пропозиції, де x – це визначена користувачем кількість речень за підсумковою довжиною та повернення речень у тому порядку, в якому вони з’являються вперше. Окрім того, що ви можете взяти речення з найвищим рейтингом, також можна розбити текст на абзаци і взяти найвищі у пропозиції кожного абзацу, де у x поділено на кількість абзаців. Ми могли б використовувати цю систему, оскільки абзаци – це логічні поділи інформації, визначені автором тексту.

1.3.3 Алгоритм LSA

Основним аспектом виявлення відповідної інформації є з'ясування того, про який текст йдеться. Документ, як правило, містить різноманітну

інформацію, орієнтовану на основну тему, і охоплює різні аспекти основної теми. Так само людські резюме, як правило, охоплюють різні теми оригінального вихідного тексту, щоб збільшити інформативний зміст резюме. Різні підходи використовували функції, засновані на виявленні тем для побудови загальних або орієнтованих на запит резюме. Часто тематичні особливості покладаються на визначення та зважування важливих ключових слів або створення підписів тем. Речення оцінюються комбінаціями балів ключових слів або обчисленням подібності між реченнями та запитами. Однак добре відомо, що відповідність термінів має серйозні недоліки через амбівалентність слів та відмінності у використанні слова та особистому стилі між авторами. Це особливо важливо для автоматичного підбиття підсумків, оскільки резюме, вироблене людьми, може суттєво відрізнятися, потенційно не поділяючи дуже багато термінів.

LSA – це алгебраїчно-статистичний метод, який витягує приховані смислові структури слів і речень. Це непідконтрольний підхід, який не потребує ані підготовки, ані зовнішніх знань. LSA використовує контекст вхідного документа та витягує інформацію, таку як слова, які вживаються разом, а які загальні слова бачать у різних реченнях. Висока кількість поширених слів серед речень свідчить про те, що речення є семантично спорідненими. Значення речення визначається за допомогою слів, які він містить, а значення слів визначаються за допомогою речень, які містять слова. Сингулярне значення декомпозиції, алгебраїчний метод, використовується для з'ясування взаємозв'язків між реченнями та словами. Окрім того, що має можливість моделювати зв’язки між словами та реченнями, SVD має можливість зменшення шуму, що сприяє підвищенню точності.

Вхідний документ повинен бути представлений таким чином, що дозволяє комп'ютеру зрозуміти і виконати обчислення на ньому. Це

подання, як правило, є матричним поданням, де стовпці – це речення, а рядки – слова чи фрази. Клітини використовуються для відображення значення слів у реченнях. Для заповнення значень комірок можна використовувати різні підходи. Оскільки всі слова розглядаються не у всіх реченнях, більшість часу створена матриця є розрідженою.

Спосіб створення вхідної матриці дуже важливий для узагальнення, оскільки він впливає на отримані матриці, обчислені за SVD. Як вже було сказано, SVD є складним алгоритмом, і його складність збільшується з розміром вхідної матриці, що погіршує продуктивність. Щоб зменшити розмір матриці, рядки матриці, тобто слова, можна зменшити за допомогою підходів, таких як видалення стоп-слів, використовуючи лише корені слів, використовуючи фрази замість слів тощо. Також значення комірок матриці можуть змінювати результати SVD. Існують різні підходи до заповнення значень комірок.

LSA має кілька обмежень. Перший полягає в тому, що він не використовує інформацію про порядок слів, синтаксичні відносини та морфології. Цей вид інформації може бути необхідним для з’ясування значення слів та текстів. Друге обмеження полягає в тому, що воно не використовує світових знань, а лише ту інформацію, яка існує у вхідному документі. Третє обмеження пов'язане з продуктивністю алгоритму. При більших і неоднорідних даних продуктивність різко знижується. Зниження продуктивності викликано SVD, що є дуже складним алгоритмом.

1.4 Порівняння алгоритмів аналізу даних

Однією з найважливіших функцій у методах інтелектуального аналізу текстових даних є можливість проведення високоякісного семантичного аналізу слів, речень, а також зв’язків, які можуть бути скриті

всередині тексту між словами та реченнями. Це дає змогу складати кращі анотації, що передають суть тексту при цьому зберігаючи його контекст.

Окрім цього, особливо важливою є можливість алгоритмів адаптуватися під умови того, что текст може бути не підготовленим для подальшої роботи, тобто не мати первинної обробки, що додасть проблем у подальшій роботі.

Слід зазначити, что швидкодія та багатомовність є також одним з важливих критеріїв якісної роботі з текстовими даними, адже вони можуть сягати великих об’ємів та розмірності.

Далі, у таблиці 1.1 ми порівняємо основні типи алгоритмів по цим критеріям, щоб дізнатися, який з них є найбільш придатним для якісного аналізу і анотації тексту.

Табл. 1.1 – Порівняння різних алгоритмів інтелектуального аналізу тексту

Особливість	TextRank	Luhn	LSA
Аналіз слів	Має	Має	Має
Аналіз речень	Має	Має	Має
Аналіз зв’язків між словами	Має	Немає	Має
Аналіз зв’язків між реченнями	Має	Немає	Немає
Адаптивність	Має	Немає	Немає
Багатомовність	Має	Має	Має
Швидкодія	Середня	Висока	Низька

1.5 Постановка задачі

Інтелектуальних аналіз даних охоплює дуже обширну область знань про отримання додаткової інформації при маніпуляціях з різноманітними даними, серед яких особливе місце посідає обробка тексту.

Під час проведення аналізу предметної області було виявлено три основні типи алгоритмів для анотації тексту, серед яких було обрано три найбільш перспективних у даній області. Після проведення порівняння по основним критеріям виявилося, що TextRank є найбільш адаптованим алгоритмом для анотації тексту, у той час як Luhn орієнтований на пошук ключових слів у тексті, а LSA – на виявлення семантичних залежностей між словами для пошуку тематики тексту.

2 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

В рамках даного дослідження було проведено аналіз даних, а зокрема лексичні і семантичні зв'язки між словами і реченнями в тексті для виділення чітких правил скорочення тексту за допомогою алгоритму TextRank.

2.1 Аналіз лексичних зв'язків

Зв'язок речень і слів в тексті робить його осмисленим. Це досягається за допомогою семантичних особливостей мови, таких як займенники («ми», «він», «вона»), вказівні займенники («цей», «той», «ці»), лексичні відносини в тексті і словосполучення, що посилаються на один і той же об'єкт. Так, згідно з Вольфану Дресслера, зв'язок в тексті можна визначити як речення, які продовжують або взаємно доповнюють один одного. Таким чином текст наповнений змістом в результаті об'єднання сенсу всіх речень протягом усього часу його прочитання.

Зв'язки речень в тексті видно, безпосередньо, при прочитанні тексту, як єдиного цілого, в той час як кожне взяте окремо речення може втрачати частину сенсу.

2.1.1 Правила пошуку зв'язків

При аналізі документа потрібно визначити зв'язок між суміжними реченнями і зрозуміти, яке з них продовжує попереднє. Був визначений наступний набір правил для знаходження зв'язків між пропозиціями:

1) наявність об'єднуючих слів («таким чином», «як результат», «також»), які вказують на зв'язок поточного речення з попереднім. Коли такі речення знайдені, вони формують лексичну зв'язок, який не повинен бути розірваний для збереження сенсу речення і тексту в цілому;

2) наявність займенників в поточному реченні, які посилаються на іменник в попередньому. Коли такі речення знайдені, вони формують лексичну зв'язок, який не повинен бути розірваний для збереження сенсу речення і тексту в цілому;

3) повторне використання іменників у тексті говорить про наявність зв'язку між реченнями. Коли такі речення знайдені, вони формують лексичну зв'язок, який не повинен бути розірваний для збереження сенсу речення і тексту в цілому;

4) онтологічний зв'язок між словами в реченнях можуть використовуватися семантичних зв'язків між реченнями. Поява пов'язаних слів говорить про те, що ми виявили лексичний зв'язок, який повинен бути збережений.

2.1.2 Об'єднуючі слова

ACE Copus був використаний для вивчення лексичних шаблонів для зв'язків між реченнями. При проведенні аналізу було обрано набір ключових слів, поява яких в реченні однозначно стверджувала б його лексичний зв'язок з попереднім. Наявність таких слів як «однак», «крім того», «відповідно», «отже», «до речі», «таким чином», «але», «проте», «одного разу», «більш того» на початку поточного речення однозначно говорить про його зв'язки з попереднім. Також наявність цих слів в середині поточного речення також може говорити про зв'язок з попереднім реченням. Розглянемо приклади далі:

1) Діма запізнився на зустріч. Проте вона пройшла успішно;

2) Міша і Катя посварилися. Зрештою їм довелося розлучитися;

3) таксист потрапив в аварію. Однак ніхто не постраждав.

2.1.3 Використання займенників

В даному підході було проаналізовано що наявність займенники в поточному реченні може говорити про наявність його зв'язку з попереднім реченням. Якщо в реченні фігурують такі займенники як «він», «вона», «воно», «вони», «його», «її», «їх» це може говорити про наявність іменника до якого вони відносяться як в поточному, так і в попередньому реченні. Однак не всі займенники можуть однозначно говорити про зв'язок між реченнями. Такі займенники як «сам», «сама» часто не створюють зв'язку. Розглянемо приклади далі:

1) Максим хороший хлопець. Він не раз мене виручав;

2) кажуть Toyota хороша машина. Вона пройшла всі тести на безпеку;

3) Вова любить азартні ігри. Сам того не знаючи підриває економіку країни.

2.1.4 Повторювані іменники

Два сусідніх речення можна вважати пов'язаними, якщо в них є повторювані іменники. Розглянемо наступні приклади:

1) Кирило дуже розумний. Кирило склав іспити на п'ятірки;

2) машина є хорошим способом пересування. Машина може набирати швидкість до 230 км / ч.

2.1.5 Взаємини між словами

Крім усього потрібно враховувати зв'язку між словами, як всередині одного речення, так і між ними. Такі поняття як синоніми, антоніми та інші можуть бути використані для семантичного зв'язку між реченнями. Розглянемо кілька прикладів:

1) у машини, на відміну від мотоцикла, чотири колеса. Колеса дають їй велику стійкість на дорозі;

2) у новій версії телефону Samsung поліпшили камеру. Камера здатна знімати відео в темряві.

2.2 Алгоритми пошуку лексичних зв'язків

Даний алгоритм буде виконуватися в кілька етапів. Спочатку, кожне з наведених вище правил буде застосовуватися по черзі для отримання семантичних зв'язків в тексті. Далі семантично пов'язані речення будуть об'єднані разом. Схема представлена на рисунку 2.1.

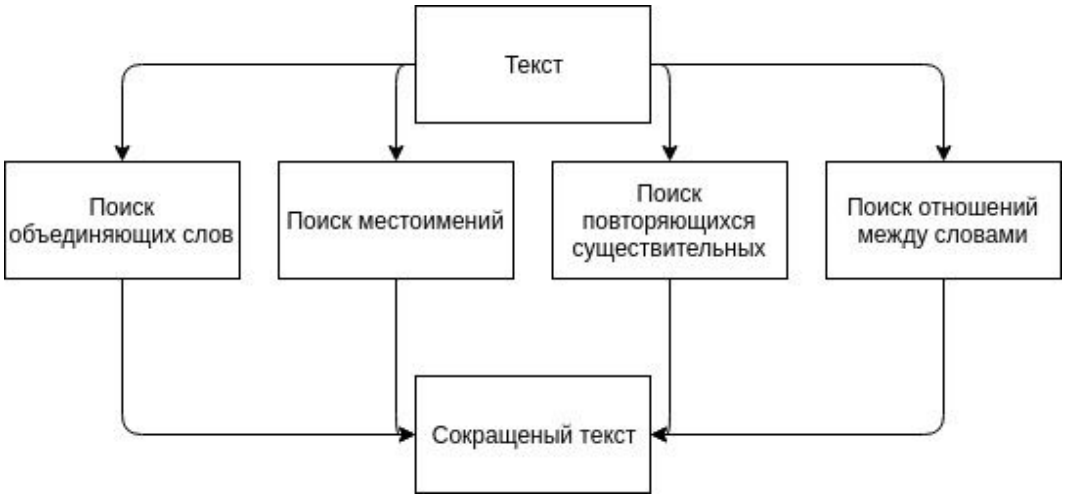

Рис. 2.1 – Схема роботи алгоритму

2.3 Оцінка алгоритму

Один із способів оцінки роботи алгоритму полягає в порівнянні з судженнями людини, що прочитав текст, або ж авторів тексту, які можуть однозначно стверджувати, наскільки точно відпрацював алгоритм і, які важливі частини він упустив.

Порівнювати можна не тільки з вручну зробленим скороченням тексту, як це робиться на сайті, але і з іншими алгоритмами скорочення тексту.

2.4 Алгоритм TextRank

Пропонований алгоритм ранжування тексту на основі графів складається з трьох етапів:

1) аналіз слів у тексті;
2) побудова графа і обчислення ваги вершин на основі зв'язків;
3) ранжування результатів отриманих на етапі 1 і 2.

2.4.1 Побудова графа

Нехай G (V, E) буде виваженим, не направленим, повним графом, де V буде набором вершин графа, а E буде набором його ребер.

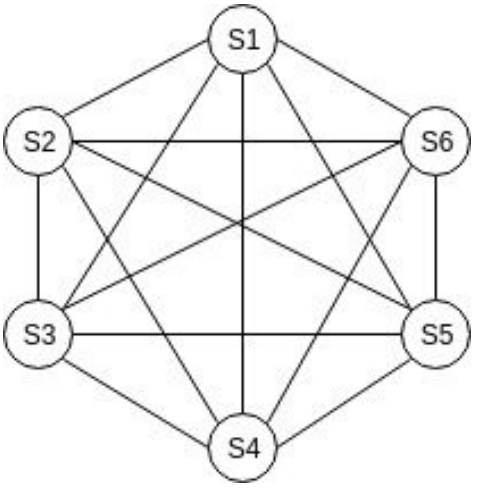

Рис. 2.2 – Виваженим, не направлений, повний граф

На малюнку 2.2 зображений граф G, де кожна вершина графа являє собою речення S з документа. Кожна пропозиція має відношення до кожної пропозиції в тексті.

2.4.2 Обчислення ваги вершин графа

Нехай усі речення в документі $S = \{s_i \mid 1 \le i \le n\}$, де n – це кількість речень в *S*. Вага речення *(SW)* для кожного речення буде розрахована за середньою вагою схожості слів у ньому.

Для речення $s_i = \{w_j \mid 1 \le j \le m_i\}$, де m_i – це номер слова у реченні s_i, $(1 \le i \le n)$ схожі ваги *(AW)* для слова w_j рахуються по наступній формулі:

$$AW(w_j) = \frac{\sum_{\forall w_k \in S} IsEqual(w_j, w_k)}{WC(S)}, \quad (2.1)$$

де *S* це набір усіх речень у документі, $WC(S)$ – це всі слова у *S*, а функція IsEqual() повертає значення 1, якщо слова однакові та 0, якщо слова різні.

Далі ми знаходимо вагу речення $SW(s_i)$ для кожного речення s_i $(1 \le i \le n)$ по заданій формулі:

$$SW(s_i) = \frac{1}{m_i} \sum_{\forall w_j \in S_i} AW(w_j), \quad (2.2)$$

Врешті-решт ми отримаємо граф зображений на малюнку 2.3, де у кожної вершини присутня вага:

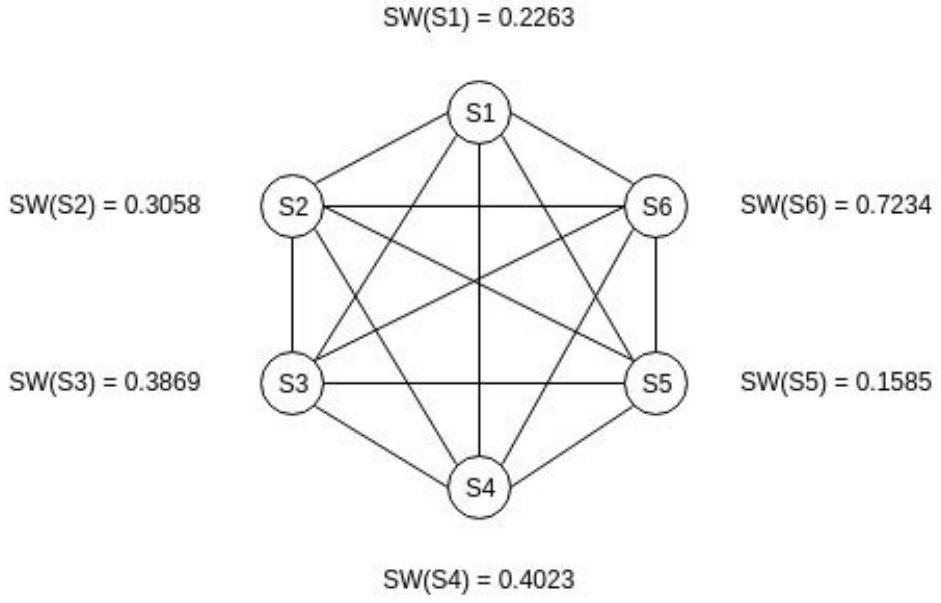

Рис. 2.3 – Граф з підрахованою вагою вершин

Наведений вище графік був побудований на основі тексту, зазначеного в таблиці нижче.

2.4.3 Стиснення хешем

Швидка хеш-функція стиснення над словом *w* працює за наступною формулою:

$$H(w) = (c_1 a^{k-1} + c_2 a^{k-2} + c_3 a^{k-3} + ... + c_k a^0) \; mod \; p \;\; , (2.3)$$

де $w = \{c_1, \; c_2, c_3 \; ... \; c_n\}$, це впорядкована множина ASCII символів алфавіту в *w* і *k* це кількість алфавітів у *w*. Вибір $a = 2$ допускає і множення в рівнянні, операції двоїчного сдвигу, що суттєво скорочує підрахунок хешу. Встановлюємо $p = 26$ для англійської мови, щоб врахувати всі букви у алфавіті.

Підрахунок *H(w)* для кожного речення s_i для отримання хеша речення:

$$H(s_i) = \{H(w_1), H(w_2) \; ... \; H(w_{m_i})\} \;, (2.4)$$

Далі перетворимо кожен елемент у множині $H(s_i)$ назад у ASCII, щоб отримати еквівалентну множину:

$$\widehat{H}(\widehat{s_i}) = \{H(\widehat{c_1}), H(\widehat{c_2}) \; ... \; H(\widehat{c_{m_i}})\} \;, (2.5)$$

Після цього ми об'єднуємо елементи множини $\widehat{H}(\widehat{s_i})$ щоб отримати строку $\widehat{s_i}$, де $\widehat{s_i}$ – це стисле уявлення речення s_i. Операція хешування застосовується для зменшення обчислювальної складності в обчисленнях ваги вершин графа шляхом стиснення речення і в той же час зберігаючи їх структурні особливості, такі як частоту потрапляння слова в реченні або позицію слова в реченні.

2.4.4 Підрахунок відстані Левенштейна

Відстань Левенштейна *(LD)* між двома рядками *string1* і *string2* це метрика, яка використовується для підрахунку операцій, необхідних для перетворення *string1* у *string2* чи навпаки, де набором можливи операцій над символом є:

1) вставка;

2) видалення;

3) заміщення.

Алгоритм підрахунку відстані Левенштейна можно продемонструвати на наступному прикладі:

1) *LD(ROLL, ROLE)* – 1 операція по заміщенню літери у слові;

2) *LD(SATURDAY, SUNDAY)* – 2 операції по видаленню літери, 1 операція по заміщенню літери.

2.4.5 Подібність по Левенштейну

Розглянемо речення *string1* і *string2*, де ls_1 – це довжина речення *string1*, а ls_2 – це довжина речення *string2,* а $MaxLen = max(ls_1, ls_2)$. Тоді подібність по Левенштейну *(LSW)* буде дорівнювати різниці між $MaxLen$ і *LD,* поділеній на $MaxLen$. *LSW* буде знаходитися в інтервалі від 0 до 1, де 0 означає повну невідповідність, а 1 – повну відповідність. У будь-яких інших випадках *LSW* буде дорівнювати $0 \le LSW \ge 1$. Далі розглянемо приклади, щоб зрозуміти принцип роботи:

1) LSW (ABC, ABC) = 1;

2) LSW (ABC, XYZ) =0;

3) LSW (ABCD, EFD) =0.25.

Таким чином, щоб знайти знайти подібність по Левенштейну *(LSW)* спочатку необхідно знайти дистанцію Левенштейна *(LD)* по наступній формулі:

$$LSW(\widehat{s_i},\ \widehat{s_j}) = \frac{MaxLen(\widehat{s_i}, \widehat{s_j}) - LD(\widehat{s_i}, \widehat{s_j})}{MaxLen(\widehat{s_i}, \widehat{s_j})} \quad , (2.6)$$

де $\widehat{s_i}$ і $\widehat{s_j}$ об’єднанні рядки з формулы (2.5).

2.4.6 Обчислення ваги ребер графа

Нехай $S = \{s_i \mid 1 \le i \le n\}$ це множина всіх речень у документі, де n – цу кількість речень у документі S, а $s_i = \{w_j \mid 1 \le j \le m\}$, де m – це кількість слів у реченні s_i.

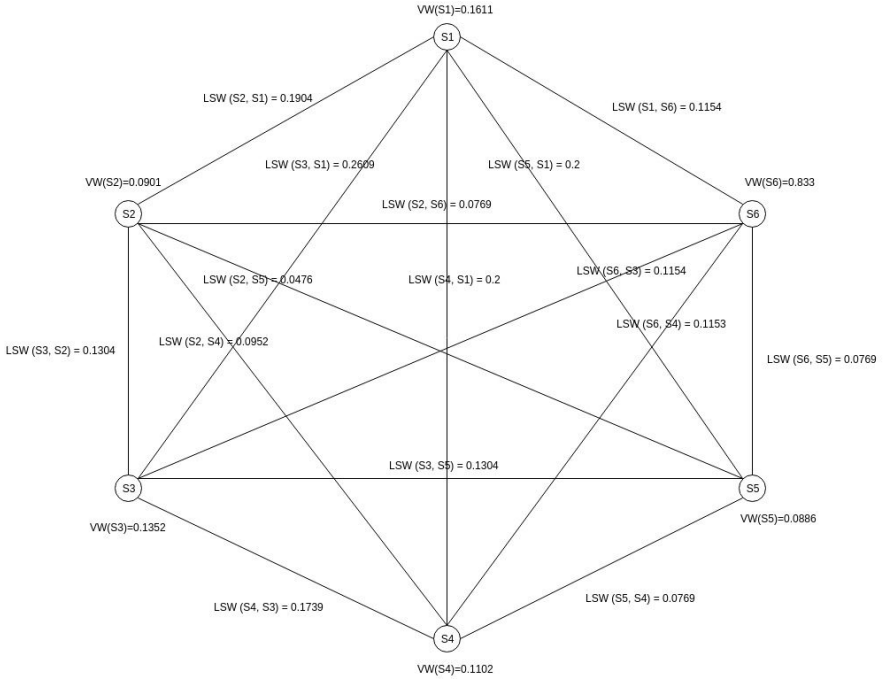

Рис. 2.4 – Граф з підрахованою вагою ребер

$\forall s_i \in S$, знайдемо $\widehat{H}(\widehat{s_i}) = \{H(\widehat{c_1}), H(\widehat{c_2}) \ldots H(\widehat{c_{m_i}})\}$, користуючись формулами (2.3) і (2.4). Потім об'єднаємо елементи у $\widehat{H}(\widehat{s_i})$, щоб отримати $\widehat{s_i}$, де $\widehat{s_i}$ — це хешоване уявлення s_i.

Кожен рядок $\widehat{s_i}$ представлений як вершина графа на малюнку 2.4, де документ представлений як $\widehat{S} = \{\widehat{s_i} \mid 1 \le i \le n\}$. Для ребер графа на малюнку 2.5 мы знаходимо *LSW* по формулі (2.6), після чого шукаємо вагу для кожної вершини $\widehat{s_i}$; $1 \le l \le n$ по формулі:

$$VW(\widehat{s_i}) = \frac{1}{n} \sum_{\forall \widehat{s_i} \neq \widehat{s_i} \in \widehat{S}} LSW(\widehat{s_l}, \widehat{s_i}) \quad , (2.7)$$

2.4.7 Підрахунок рангу пропозиції

Ранг речення s_i; $1 \le l \le n$ рахується по наступній формулі:

$$Rank(s_i) = \frac{SW(s_i) + VW(s_i)}{2}; \ 1 \le i \le n \, , (2.8)$$

де $SW(s_i)$ – результат, отриманий по формулі (2.2), а $VW(s_i)$ – це результат, отриманий по формулі (2.7).

$SW(s_i)$ зберігає спорідненість речення в порядку слів і використовується для визначення значущості речення при суміщенні речення в тексті. $VW(s_i)$ допомагає в загальному ранжируванні, визначаючи найбільші загальні підпослідовності, привласнюючи йому ваги з використанням *LSW*. Крім того, оскільки об'єкти представлені у вигляді рядків, повторні входження зважуються за допомогою *LSW*, тим самим надаючи відповідну позицію ранжирування.

Далі речення сортируються згідно їх рангу і вибирається деяка кількість речень, що будуть використовуватися для скорочення тексту. Як тільки речення для скорочення обрані, вони сортируються у порядку, якому вони стояли у оригінальному тексті і алгоритм завершує роботу.

Таким чином, у декілька простих кроків реалізується алгоритм інтелектуального скорочення текстів на основі графів для Web-додатків, що було оптимізовано за допомогою алгоритмів хешування, не дивлячись на те, що алгоритми хешування зазвичай використовуються для шифрування даних у мережі Інтернет.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

У рамках проведення експериментального дослідження було вирішено розробити Web-додаток, що здатен взаємодіяти з іншими Web-додатками за допомогою прямих запросів через URL, потрібно розглянути деякі технології, а саме мови програмування, технології розробки інтерфейсу користувача, технології розробки серверної частини програм, а також методи зберігання даних та деякі можливості у роботі з хмарними технологіями.

3.1 Мова програмування Python

Python – це універсальна, сучасна мова програмування високого рівня, до переваг якої відносять високу продуктивність програмних рішень і структурованість, дуже простий і зрозумілий код. Синтаксис Python максимально полегшений, що дозволяє вивчити його за порівняно короткий час. Ядро Python має дуже зручну структуру, а широкий перелік вбудованих бібліотек дозволяє застосовувати значний набір корисних функцій і можливостей. Python як мова програмування може використовуватися для написання прикладних програм, а також розробки Web-додатків.

Python може підтримувати широкий перелік стилів розробки додатків, в тому числі, дуже зручний для роботи з об’єктно-орієнтований підхід, а також функціональне програмування. Приклад коду можна побачити на малюнку 3.1.

```
from .pagerank_weighted import pagerank_weighted_scipy as _pagerank
from .preprocessing.textcleaner import clean_text_by_sentences as _clean_text_by_sentences
from .commons import build_graph as _build_graph
from .commons import remove_unreachable_nodes as _remove_unreachable_nodes


def _set_graph_edge_weights(graph):
    for sentence_1 in graph.nodes():
        for sentence_2 in graph.nodes():

            edge = (sentence_1, sentence_2)
            if sentence_1 != sentence_2 and not graph.has_edge(edge):
                similarity = _get_similarity(sentence_1, sentence_2)
                if similarity != 0:
                    graph.add_edge(edge, similarity)

    # Handles the case in which all similarities are zero.
    # The resultant summary will consist of random sentences.
    if all(graph.edge_weight(edge) == 0 for edge in graph.edges()):
        _create_valid_graph(graph)
```

Рис. 3.1 – Приклад програмного коду на мові програмування Python

Один з найпопулярніших інтерпретаторів мови Python – CPython, написаний на мові програмування C. Поширюється це середовище розробки безкоштовно по вільній ліцензії. Інтерпретатор підтримує більшість популярних платформ.

Python активно розвивається. Приблизно раз в 2 роки виходять оновлення. Важливою особливістю мови є відсутність таких стандартів кодування як ANSI, ISO і деяких інших, вони працюють завдяки інтерпретатору.

Python підтримує практично всі поширені операційні системи. Він може прекрасно працювати на кишенькових комп'ютерах, так і на великих серверах. У разі, якщо платформа значно застаріває, вона виключається з підтримки ядра. Наприклад, версії мови, починаючи від 2.6, вже не

працюють з платформами Windows 95 та Windows 98. У разі необхідності можна скористатися більш старими версіями, відмовившись від застосування сучасних інструментів мови. І тоді додаток буде працювати в тому числі з цими ОС. Для старих версій періодично виходять патчі. Мова програмування Python також може підтримувати роботу з віртуальною машиною Java.

Як правило, програми Python запускаються повільніше, ніж програми Java, але на розробку також потрібно набагато менше часу. Програми Python, як правило, в 3-5 разів коротші, ніж еквівалентні програми Java. Цю різницю можна пояснити вбудованими типами даних високого рівня Python та його динамічним набором тексту. Наприклад, програміст Python не витрачає часу, декларуючи типи аргументів чи змінних, а потужний поліморфний список та типи словників Python, для яких багата синтаксична підтримка вбудована прямо в мову, знаходять застосування майже в кожній програмі Python.

Через набрання часу виконання, час роботи Python повинен працювати важче, ніж у Java. Наприклад, оцінюючи вираз a + b, він повинен спочатку оглянути об’єкти a і b, щоб з’ясувати їх тип, який невідомий під час компіляції. Потім він викликає відповідну операцію додавання, яка може бути перевантаженим визначеним користувачем методом. Java, з іншого боку, може виконувати ефективне додавання цілого чи плаваючої точки, але вимагає змінних оголошень для a і b, і не допускає перевантаження оператора + для випадків визначених користувачем класів.

З цих причин Python набагато краще підходить як мова високого рівня, тоді як Java краще характеризувати як мову низького рівня реалізації. Насправді вони разом складають відмінне поєднання. Компоненти можна розробляти на Java та комбінувати для формування

додатків у Python. Python також може використовуватися для прототипу компонентів, поки їх конструкція не може бути «загартованою» в реалізації Java. Для підтримки такого типу розробки розробляється реалізація Python, написана на Java, яка дозволяє викликати код Python з Java та навпаки. У цій реалізації вихідний код Python переводиться на байт-код Java (за допомогою бібліотеки запущеного часу для підтримки динамічної семантики Python).

Майже все, що сказано для Java, стосується і C ++, тим більше: там, де Python код, як правило, в 3-5 разів коротший, ніж еквівалентний код Java, він часто в 5-10 разів коротший, ніж еквівалентний код C ++! Анекдотичні дані свідчать про те, що один програміст Python може закінчити за два місяці те, що два C ++ програмісти не можуть виконати за рік. Python світить як мова клею, використовується для комбінування компонентів, написаних на C ++.

Об'єктна підмножина Python приблизно еквівалентна JavaScript. Як і JavaScript (і на відміну від Java), Python підтримує стиль програмування, який використовує прості функції та змінні без участі у визначеннях класу. Однак для JavaScript це все. Python, з іншого боку, підтримує написання набагато більших програм та краще використання коду за допомогою справжнього об'єктно-орієнтованого стилю програмування, де класи та успадкування відіграють важливу роль.

Python та Perl походять із подібного походження (сценарії Unix, які обоє давно переросли), і спорт мають багато подібних особливостей, але мають різну філософію. Perl наголошує на підтримці поширених завдань, орієнтованих на додатки, наприклад за допомогою вбудованих регулярних виразів, сканування файлів та функцій створення звітів. Python наголошує на підтримці загальних методологій програмування, таких як проектування структури даних та об'єктно-орієнтоване програмування, і заохочує

програмістів писати читабельний (і таким чином ремонтопридатний) код, надаючи елегантні, але не надто криптовані позначення. Як наслідок, Python наближається до Perl, але рідко обіграє його в початковій області застосування; однак Python має можливість застосувати далеко поза ніш Perl.

Таким чином, мова програмування Python має багато переваг у порівнянні з іншими мовами програмування та прекрасно підійде для розробки складних математичних моделей, алгоритмів, бо дозволяє зробити це швидко і просто.

3.2 Технології розробки інтерфейсу користувача

Інтерфейс користувача – одна з різновидів інтерфейсів, яка є совокупностью засобів і методів взаємодії користувача з різноманітними пристроями. У нашому випадку інтерфейсом користувача буде виступати Web-додаток у мережі Інтернет. Далі, розглянемо основні технології, що забезпечують взаємодію користувача з серверною частиною Web-додатка, а саме HTML, CSS, JS та Vue.

3.2.1 HTML

HTML розшифровується як мова розмітки гіпертексту, і це найбільш поширена мова для написання веб-сторінок. Гіпертекст посилається на те, як веб-сторінки пов'язані між собою. Таким чином, посилання, доступне на веб-сторінці, називається гіпертекстом.

Як випливає з назви, HTML – це мова розмітки, що означає, що ви використовуєте HTML, щоб просто розмітити текстовий документ тегами, які вказують веб-браузеру, як структурувати його для відображення.

Спочатку HTML був розроблений з метою визначення структури документів, таких як заголовки, параграфи, списки тощо, щоб полегшити обмін науковою інформацією між дослідниками. Зараз HTML широко використовується для форматування веб-сторінок за допомогою різних тегів, доступних мовою HTML.

Як було сказано раніше, HTML є мовою розмітки та використовує різні теги для форматування вмісту. Ці теги укладені в кутові дужки. За винятком кількох тегів, більшість тегів мають відповідні теги закриття. Наприклад, у <html> є його тег </html>, а у <body> є тег </body> та ін. Нижче на малюнку 3.2 наведено приклад простого HTML-документа.

```
<html>
<head>
<title>Introduction to HTML5</title>

<meta charset='utf-8'>
<script src='slides.js'></script>
<script type="text/javascript" src="jquery.js"></script>
<script>
$(document).ready(function() {
        $("#today").text(new Date().toDateString());
})
</script>
</head>

<style>
```

Рис. 3.2 – Приклад простого HTML-документа

Щоб вивчити HTML, потрібно буде вивчити різні теги та зрозуміти, як вони поводяться, форматуючи текстовий документ. Вивчити HTML дуже просто, оскільки потрібно навчитися використовувати різні теги, щоб відформатувати текст або зображення, щоб зробити красиву веб-сторінку.

3.2.2 CSS

CSS, що в перекладі означає каскадні таблиці стилів використовують для стилізації та компоновки Web-додатків, наприклад, для розміщення шрифтів, кольорів, розмірів та інтервалу, що містять, розділяє його на кілька стільниць або додає анімації та інші декоративні елементи.

CSS використовує створені Web-додатки для визначення кольорів, шрифтів, розміщення окремих блоків та інших аспектів, що представляють зовнішні види цих веб-сторінок. Основні цільові розробки CSS представляють відокремлені описи логічної структури Web-додатку, що виробляється з підтримкою HTML або іншими мовами розміщення, від опису представлених найменувань веб-сторінок, що тепер виробляється з допомогою формального мови CSS. Таке розділення може збільшити доступність документа, представити більшу гнучкість і можливість управління його представити, а також змінити складність і повторити свою ефективність у відповідному вмісті.

Крім того, CSS може представити один і той самий документ у різних стилях або методах виводу, таких як екранне представлення, друковане представлення, представлення за допомогою голосу або при використанні пристроїв, використовуючи шрифт Брайля.

Правила CSS пишуться на формальній мові CSS. Правила можуть розташовуватися як в самому Web-додатку, зовнішній вигляд якого вони описують, так і в зовнішніх файлах, що мають формат CSS. Формат CSS –

це текстовий файл, в якому міститься перелік правил CSS і коментарів до них. Приклад простого CSS-документа наведено на малюнку 3.3.

```
html,
body   {
  margin: 0;
  padding: 0;
  line-height: 1.5;
  font-size: 16px;
}

/* styling our overall page */
body     {
  background: #FFF;
  font-family: Arial, Helvetica, sans-serif;
}
```

Рис. 3.3 – Приклад простого CSS-документа

Кожне правило CSS з файлу має дві основні частини – селектор і блок оголошень. Селектор, розташований в лівій частині правила визначає, на які частини документа поширюється правило. Блок оголошень розташовується в правій частині правила. Він міститься в фігурні дужки, і, в свою чергу, складається з одного або більше оголошень. Кожне оголошення є поєднанням властивості CSS і значення. Селектори можуть групуватися в одному рядку через кому. В такому випадку властивість застосовується до кожного з них.

Основна відмінність між класами елементів і ідентифікаторами елементів в тому, що ідентифікатор призначений для одного елемента, тоді як клас зазвичай привласнюють відразу декільком. Проте, сучасні браузери, як правило, коректно відображають множинні елементи з однаковим ідентифікатором. Також відмінність в тому, що можуть

існувати множинні класи (коли клас елемента складається з декількох слів, розділених пробілами).

В CSS, крім класів, що задаються автором сторінки, існує також обмежений набір так званих псевдокласів, що описують вид гіперпосилань з певним станом в документі, вид елемента, на якому знаходиться фокус введення, а також вид елементів, які є першими дочірніми елементами інших елементів. Також в CSS існує чотири так званих псевдоелемента: перша буква, перший рядок, застосування спеціальних стилів до і після елементу.

3.2.3 JS

JavaScript – мультіпарадігмова мову програмування. Підтримує об'єктно-орієнтована, імперативний і функціональний стилі. Є реалізацією мови EcmaScript.

JavaScript використовується в клієнтської частини веб-додатків: клієнт-серверних програм, в якому клієнтом є браузер, а сервером - веб-сервер, що мають розподілену між сервером і клієнтом логіку. Обмін інформацією в веб-додатках відбувається по мережі. Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому Web-додатки є кроссплатформенную сервісами. Основними архітектурними рисами мови програмування JavaScript є наступне:

1) динамічна типізація;
2) автоматичне керування пам'яттю;
3) прототипне програмування;
4) функції як об'єкти першого класу.

JavaScript є об'єктно-орієнтованою мовою, але що використовується в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, – функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання – що додає мові додаткову гнучкість. Приклад програмного коду JavaScript наведено на малюнку 3.4.

```
import { addFontGaegu, addFontIndieFlower } from './utils/addFonts';
import { csv, tsv, json } from 'd3-fetch';
import { mouse, select, selectAll } from 'd3-selection';
import { arc, pie } from 'd3-shape';
import rough from 'roughjs/dist/rough.umd';
import { colors } from './utils/colors';
import { addLegend } from './utils/addLegend';

const roughCeiling = roughness => {
  let roughVal = roughness > 30 ? 30 : roughness;
  return roughVal;
};
```

Рис. 3.4 – Приклад програмного коду на мові програмування JavaScript

В JavaScript доступ до отладчиков стає особливо корисним при розробці великих нетривіальних програм через відмінності в реалізаціях різних браузерів (зокрема, щодо об'єктної моделі документа). Корисно мати доступ до отладчику для кожного з браузерів, в яких буде працювати веб-додаток.

Більшість фреймворків автоматизованого тестування JavaScript-коду припускають запуск тестів в браузері. Це здійснюється за допомогою

HTML-сторінки, що є контекстом тестування, яка, в свою чергу завантажує все необхідне для здійснення тестування. Альтернативою є запуск тестів з командного рядка. В цьому випадку використовуються оточення, відмінні від браузера. Одним з перших інструментів такого роду є Crosscheck, що дозволяє тестувати код, емулюючи поведінку різних браузерів. Інший приклад фреймворка автоматизованого тестування JavaScript-коду, що не використовує браузер для запуску тестів – бібліотека env.js, що містить емуляцію оточення браузера.

Головна проблема систем тестування, що не використовують браузери, в тому, що вони використовують емуляції, а не реальні оточення, в яких виконується код. Це призводить до того, що успішне проходження тестів не гарантує того, що код коректно відпрацює в браузері. Проблемою систем тестування, що використовують браузер, є складність роботи з ними, необхідність здійснення рутинних неавтоматизованих дій. Для вирішення цього JsTestDriver, фреймворк автоматизованого тестування, що розробляється Google, використовує сервер, який взаємодіє з браузерами для здійснення тестування. Подібним чином поводиться Selenium Remote Control, що входить у фреймворк автоматизованого тестування Selenium: він включає в себе сервер, який запускає і завершальний браузери і діючий як HTTP-проксі для запитів до них. Крім того, в Selenium міститься Selenium Grid, що дозволяє здійснювати одночасне тестування JavaScript-коду на різних комп'ютерах з різними оточеннями, зменшуючи час виконання тестів.

Негативна властивість, яким може володіти фреймворк автоматизованого тестування JavaScript-коду – наявність залежностей. Це створює ризик відмови в роботі тестованого коду, успішно проходить тести, в середовищі з відсутністю цих залежностей. Наприклад, початкова версія JsUnitTest, фреймворка, створеного і використовувався для

тестування бібліотеки Prototype, залежала від самої Prototype, що змінює властивості об'єктів в глобальному контексті. Включення в бібліотеку JavaScript інструменту тестування – поширена практика.

3.2.4 Vue

Vue – фреймворк для JavaScript з відкритим вихідним кодом для створення користувацьких інтерфейсів. Легко інтегрується в проекти з використанням інших JavaScript бібліотек. Може функціонувати як веб-фреймворк для розробки односторінкових додатків в реактивному стилі.

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно пов'язувати рендеринг DOM з основними екземплярами даних в Vue. Все Vue шаблони валідні HTML, і можуть бути розпарсені браузерами і HTML парсером. Усередині Vue компілює шаблони в рендеринге функції віртуального DOM. У поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для рендерінгу і застосувати мінімальну кількість маніпуляцій з DOM, коли стан Web-додатка зміниться. Приклад використання Vue наведено на малюнку 3.5.

```
Vue.config.productionTip = false;
Vue.filter("date", DateFilter);
Vue.filter("error", ErrorFilter);

ApiService.init();

// Ensure we checked auth before each page load.
router.beforeEach((to, from, next) =>
  Promise.all([store.dispatch(CHECK_AUTH)]).then(next)
);

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount("#app");
```

Рис. 3.5 – Приклад програмного коду з використанням фреймворку Vue

В Vue можна використовувати синтаксис шаблонів або безпосередньо писати рендеринг функції використовуючи JSX. Для того, щоб це зробити просто замінити шаблон на рендерінгову функцію. Рендерінгова функція відкриває можливості для потужних патернів заснованих на компонентах – наприклад, нова транзитна система тепер повністю заснована на компонентах, використовує рендеринг функції всередині.

Одна з найвиразніших особливостей Vue – це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об’єкти. Це робить управління станами дуже простим і інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без необхідності робити що-небудь додатково. Кожен компонент стежить за своїми реактивними залежностями при рендеринга, тому система знає точно коли має відбуватися ре-рендеринг і які компоненти потрібно ре-рендерить.

Vue сам по собі не включає роутинг, і є vue-router пакет, який вирішує це питання. Він підтримує зв'язування вкладення шляхів з

вкладеними компонентами і пропонує деталізований контроль над переходами. Vue дозволяє створення додатків за допомогою компонентів. Якщо додати vue-router до цього, все що потрібно зробити це зв'язати ваші компоненти з роута і дозвольте vue-router вирішувати де їх рендерить.

3.3 Технологія розробки серверу Tornado

Tornado – фреймворк для створення веб-серверів написаний на мові програмування Python. Tornado був створений для забезпечення високої продуктивності і є одним з веб-серверів, здатних витримувати завдання конфігурації і обслуговування високопродуктивного сервера, здатного обслуговувати понад 10 тисяч з'єднань одночасно. Слід розуміти, що при порівнянні продуктивності мова йде про так звані легкі запити. Бо тривала обробка запиту (наприклад, через взаємодії з сервером баз даних), зводить переваги Торнадо нанівець. Нижче на малюнку 3.6 наведено приклад програмного коду з використанням фреймворку Tornado.

```
class BaseHandler(tornado.web.RequestHandler):
    def row_to_obj(self, row, cur):
        """Convert a SQL row to an object supporting dict
        obj = tornado.util.ObjectDict()
        for val, desc in zip(row, cur.description):
            obj[desc.name] = val
        return obj

    async def execute(self, stmt, *args):
        """Execute a SQL statement.

        Must be called with ``await self.execute(...)``
        """
        with (await self.application.db.cursor()) as cur:
            await cur.execute(stmt, args)
```

Рис. 3.6 – Приклад програмного коду з використанням фреймворку Tornado

У таблиці 3.1 наведено порівняння продуктивності Tornado з іншими фреймворками для написання веб-серверів на мові програмування Python.

Табл. 3.1 – Порівняння продуктивності різних фреймворків

Фреймворк	Запитів в секунду
Tornado	8353
Django	2223
Flask	2066

3.4 Технологія розробки баз даних SQLite

SQLite – компактна, вбудована система управління базами даних з відкритим вихідним кодом. Слово вбудована означає, що SQLite не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а являє собою бібліотеку, з якої програма компонується, і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується, а функції досягаються в тому числі за рахунок створення файлу журналу. Нижче на малюнку 3.7 наведений програмний код з використанням фреймворку SQLite на мові програмування Python.

```
def initialize_database():
    cursor.execute('''CREATE TABLE category (category_id INTEGER PRIMARY KEY AUTOINCREMENT,
                                             name TEXT,
                                             description TEXT)''')

    cursor.execute('''CREATE TABLE article (article_id INTEGER PRIMARY KEY AUTOINCREMENT,
                                            category_id INTEGER,
                                            title TEXT,
                                            text TEXT,
                                            date DATE,
                                            time TIME,
                                            FOREIGN KEY (category_id) REFERENCES category (category_id))''')

    cursor.execute('''CREATE TABLE keyword (keyword_id INTEGER PRIMARY KEY AUTOINCREMENT,
                                            article_id INTEGER,
                                            keyword TEXT,
                                            FOREIGN KEY (article_id) REFERENCES article (article_id))''')

    connection.commit()
```

Рис. 3.7 – Приклад програмного коду з використанням фреймворку SQLite

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті поставки йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина є кроссплатформенной утилітою командного рядка. SQLite можливо використовувати як на вбудованих системах, так і на виділених машинах з великими масивами даних.

SQLite підтримує динамічне типування даних. Кожне значення в будь-якому полі будь-якого запису може бути будь-якого з типів, незалежно від типу, зазначеного при оголошенні полів таблиці. Зазначений при оголошенні поля тип зберігається для довідки в його вихідному написанні, і використовується в якості основи для вибору переваг при виконанні неявних перетворень типів на підставі схожості цієї назви типу

на що-небудь, знайоме SQLite. В цей алгоритм зашитий великий перелік практикуються в інших системах управління базами даних варіантів назв типів даних. Якщо безпечного перетворення записується значення в бажаний тип не виходить, SQLite записує значення в його початковому вигляді. Для отримання значень з бази є ряд функцій для кожного з типів, і якщо тип значення, що зберігається не відповідає запитуваній, воно теж, по можливості, перетворюється.

Більш ранні версії SQLite були спроектовані без будь-яких обмежень, єдиною умовою було те, щоб база даних вміщувалася в пам'яті, в якій всі обчислення проводилися за допомогою 32-розрядних цілих чисел. Це створювало певні проблеми. Через те, що верхні межі були визначені і відповідно належним чином протестовані, часто виявлялися помилки при використанні SQLite в досить екстремальних умовах. Тому в нових версіях SQLite були введені межі, які тепер перевіряються разом із загальним набором тестів.

3.5 Технології хмарного обчислення Amazon Web Services

Amazon Web Services – комерційна публічна хмара, яка підтримується і розвивається компанією Amazon. Надає передплатникам послуги як по інфраструктурної моделі (віртуальні сервери, ресурси зберігання), так і платформного рівня (хмарні бази даних, хмарне сполучна програмне забезпечення, хмарні обчислення, засоби розробки). В значній мірі вплинула на формування концепції хмарних обчислень в цілому, і визначило основні напрямки розвитку публічної моделі розгортання.

Хмару розміщено в кількох географічно рознесених центрах обробки даних, що об'єднуються в групи за географічною близькістю, так звані регіони, всередині регіону реалізується кілька зон доступу, всередині яких

забезпечується висока доступність розміщених сервісів. Станом на 2019 рік діють 60 зон доступності в 20 регіонах. Передплатники можуть вибирати регіон і зону доступності, а також надається можливість організовувати реплікацію даних і перенесення додатків між зонами доступності.

Ключова інфраструктурна послуга – служба оренди віртуальних серверів EC2. Передплатникам надаються віртуальні машини, що працюють на гіпервізора Xen, доступний вибір різних по обчислювальної потужності машин, а також машин з доступом до спеціалізованого устаткування. EC2 тісно інтегрована з іншими інфраструктурними послугами хмари, перш за все – Elastic File System, що забезпечує інтегровану до віртуальних машин файлову систему, Elastic Block Store, які надають приєднуються до віртуальних машин томи як блокові пристрої, і S3, що забезпечує хмарне файлове сховище великого обсягу.

Серед інших інфраструктурних послуг є Route 53 – керований хмарний DNS, VPC – засіб створення в рамках ізольованій VPN групи хмарних сервісів, Elastic Load Balancing – балансувальник трафіку між віртуальними машинами, служба Glacier забезпечує довгострокове зберігання даних, а CloudFront – мережа доставки контенту. Ряд послуг забезпечують автоматизоване управління інфраструктурою, що розміщується в AWS, серед таких – CloudFormation, OpsWorks, CloudWatch.

Серед послуг класу сполучного програмного забезпечення – брокер повідомлень Amazon Kinesis (близький за можливостями Apache Kafka), служба черг SQS і служба повідомлень SNS.

3.6 Результати дослідження

У ході аналізу предметної області та теоретичного дослідження було вирішено провести експериментальне дослідження для того, щоб оцінити якість роботи кожного алгоритму не тільки маючи знання у математичній базі алгоритмів, а на реальних прикладах текстових даних.

Для проведення експериментального дослідження було розроблено Web-додаток, що приймав на вхід URL посилання на інший Web-додаток, що містив певну статтю на англійській мові. Після отримання тексту статті чи документа, Web-додаток аналізує його за допомогою алгоритмів TextRank, Luhn, LSA та видає три анотації. Приклади наведені у таблиці 3.2 нижче.

Табл. 3.2 – Порівняння анотацій різних алгоритмів

Алгоритм	Текст
Оригінал	A human embryo that was frozen 24 years ago has now become a baby girl. Emma Wren Gibson was born in Knoxville, Tennessee from an embryo frozen in 1992. The mother, Tina Gibson, at 25, is only a year older than the embryo. It is the longest known frozen embryo that has successfully become a baby. The Gibsons are unable to have children of their own and, in the past, have taken care of several other children. They can decide to store them for later use, dispose of them or donate them for scientific research. Many parents who have leftover embryos give them to special centers where they can be used for others. Doctors claim them frozen embryos can develop just as well as fresh ones. The dangerous part, however, is the thawing process. Only about 75% of all frozen embryos survive it. Health experts think that there may be up to a million frozen embryos in the United States. For those who can't have babies, using a frozen embryo from a donation center is similar to adoption, only that the baby grows inside the adoptive mother.

Продовження таблиці 3.2

1	2
TextRank	A human embryo that was frozen 24 years ago has now become a baby girl. Emma Wren Gibson was born in Knoxville, Tennessee from an embryo frozen in 1992. It is the longest known frozen embryo that has successfully become a baby. For those who can't have babies, using a frozen embryo from a donation center is similar to adoption, only that the baby grows inside the adoptive mother.
Luhn	A human embryo that was frozen 24 years ago has now become a baby girl. Doctors claim them frozen embryos can develop just as well as fresh ones. The dangerous part, however, is the thawing process. Only about 75% of all frozen embryos survive it. For those who can't have babies, using a frozen embryo from a donation center is similar to adoption, only that the baby grows inside the adoptive mother.
LSA	A human embryo that was frozen 24 years ago has now become a baby girl. Emma Wren Gibson was born in Knoxville, Tennessee from an embryo frozen in 1992. The dangerous part, however, is the thawing process. Only about 75% of all frozen embryos survive it. Health experts think that there may be up to a million frozen embryos in the United States.
Оригінал	The World Health Organisation has added the term "gaming disorder" to its International Classification of Diseases. It refers to people who are addicted to video and other games and cannot stop. It is the first update in the WHO's catalog in almost three decades. According to the WHO, gaming becomes a disorder if you are unable to control how long you play and when to stop. When that happens, it gets control of your life, influences everyday situations and affects your daily routine. WHO officials say that excessive gaming is a serious disorder that must be closely watched. In order for a person to be regarded as having a gaming disorder, the behavior must be going on for at least one year, either constantly or in phases. On one side studies have shown that playing video games may help with problems like depression and dementia. However, gaming is highly addictive and many people play for a longer time than is healthy. As a result, people get fired for not going to work or miss school classes for a longer period of time.

Продовження таблиці 3.2

1	2
TextRank	The World Health Organisation has added the term "gaming disorder" to its International Classification of Diseases. It refers to people who are addicted to video and other games and cannot stop. According to the WHO, gaming becomes a disorder if you are unable to control how long you play and when to stop. In order for a person to be regarded as having a gaming disorder, the behavior must be going on for at least one year, either constantly or in phases.
Luhn	The World Health Organisation has added the term "gaming disorder" to its International Classification of Diseases. It refers to people who are addicted to video and other games and cannot stop. When that happens, it gets control of your life, influences everyday situations and affects your daily routine. However, gaming is highly addictive and many people play for a longer time than is healthy.
LSA	The World Health Organisation has added the term "gaming disorder" to its International Classification of Diseases. In order for a person to be regarded as having a gaming disorder, the behavior must be going on for at least one year, either constantly or in phases. On one side studies have shown that playing video games may help with problems like depression and dementia. As a result, people get fired for not going to work or miss school classes for a longer period of time.
Оригінал	According to a new publication by American neurologists, regular exercise can improve your memory and thinking skills. We know that exercise has proven to be good for your heart and overall fitness, but now doctors say that it can actually help you remember things. As we get older most of us have problems with memory, language and thinking about certain things. This is called mild cognitive impairment (MCI). In most cases, such problems don't influence our everyday life but we realize them. Exercising may slow down the rate of MCI and reduce the risk of getting dementia at a later phase in life. Unlike people with dementia, those with mild cognitive impairment can cope with their regular routine, like getting dressed or preparing meals. However, they may have trouble remembering dates, appointments and where they left their keys. This may be the first step to Alzheimer's

Продовження таблиці 3.2

1	2
	disease or dementia. While there is no medicine and a dietary way to fight against memory loss, neurologists encourage people to do some form of aerobic exercise, like walking, running or cycling or swimming for a total of 2 .5 hours a week. They recommend exercising just so fast that you don't sweat and can talk to others. More than 6% of all people around the world have a mild form of cognitive impairment.
TextRank	According to a new publication by American neurologists, regular exercise can improve your memory and thinking skills. Exercising may slow down the rate of MCI and reduce the risk of getting dementia at a later phase in life. Unlike people with dementia, those with mild cognitive impairment can cope with their regular routine, like getting dressed or preparing meals. While there is no medicine and a dietary way to fight against memory loss, neurologists encourage people to do some form of aerobic exercise, like walking, running or cycling or swimming for a total of 2.5 hours a week.
Luhn	According to a new publication by American neurologists, regular exercise can improve your memory and thinking skills. This is called mild cognitive impairment (MCI). In most cases, such problems don't influence our everyday life but we realize them. Unlike people with dementia, those with mild cognitive impairment can cope with their regular routine, like getting dressed or preparing meals. While there is no medicine and a dietary way to fight against memory loss, neurologists encourage people to do some form of aerobic exercise, like walking, running or cycling or swimming for a total of 2 .5 hours a week.
LSA	This is called mild cognitive impairment (MCI). In most cases, such problems don't influence our everyday life but we realize them. Unlike people with dementia, those with mild cognitive impairment can cope with their regular routine, like getting dressed or preparing meals. While there is no medicine and a dietary way to fight against memory loss, neurologists encourage people to do some form of aerobic exercise, like walking, running or cycling or swimming for a total of 2 .5 hours a week. Jogging and other forms of aerobic exercise help you with your memory.

Оскільки оцінка якості анотації тексту є проблемною зоною у вирішенні питання якості роботи алгоритму, бо є суб’єктивною точкою зору експерта у даній області, будемо намагатися виділити певні критерії під час прийняття рішення.

Як можемо бачити у наведеній вище таблиці усі алгоритми анотації тексту працюють з достатньо високою точністю, якісно передаючи сутність оригінального тексту. Однак, слід зауважити, що у текста також є поняття єдності мислення, чи гладкості, яке означає можливість читати текст не втрачаючи при цьому логічного ланцюжка. Саме те завдання є одним з найбільших викликів у вирішенні задачі скорочення тексту. З цією задачею добре справилися лише два з наведених алгоритмів: TextRank та LSA. Проте TextRank зберігає гладкість тексту і намагається охопити якомога більше інформації ніж LSA, що і робить цей алгоритм одним із найкращих у інтелектуальному аналізі даних для вирішення задачі анотації тексту та багатьох інших проблем у цій області.

ВИСНОВКИ

Метою даної роботи була порівняльна характеристика різних за принципом роботи алгоритмів аналізу даних з метою анотації тексту, які б забезпечили швидку та якісну анотацію тексту, зберігаючи при цьому його сутність та сенс з точки зору кінцевого користувача. Для досягнення поставленої задачі були поставлені певні завдання для проведення аналізу предметної області, а саме:

1) аналіз потрібних типів даних;

2) аналіз існуючих шляхів збору даних;

3) аналіз можливостей попередньої обробки даних;

4) аналіз алгоритмів інтелектуального аналізу текстових даних.

Як результат проведення аналізу предметної області було з’ясовано як працюють алгоритми різного характеру у рамках даної предметної області, а також проведено аналіз теоретичних можливостей алгоритмів TextRank, Luhn та LSA для виконання задачі текстової анотації у Web-додатках.

Після проведення аналізу предметної області було ініційовано теоретичне дослідження одного з наведених вище алгоритмів для детального розгляду його функціонування та можливостей покращення швидкодії його роботи. На цьому етапі виконання роботи були розглянуті такі важливі питання як:

1) аналіз важливих лексичних зв’язків;

2) способи пошуку лексичних зв’язків;

3) алгоритм анотації тексту на основі графів TextRank;

4) критерії оцінювання алгоритмів анотації тексту.

Як результат проведення теоретичного дослідження у рамках предметної області було виявлено усі переваги та недоліки у роботі

алгоритмів на основі графів. До основних переваг було виділено можливість знаходити взаємозв’язки між словами, реченнями, а особливо між різними словами у рамках різних речень, що у подальшому дало можливість для якісного скорочення тексту. Серед основних недоліків алгоритму можно вважати його середню швидкість роботи. Через те, що алгоритм представляє дані у формі графа для обчислення кожної вершини, як речення, і кожного ребра, як умовних зв’язків між реченнями, що може значно погіршити продуктивність. Задля вирішення даної проблемі було вирішено інтегрувати хешування слів і речень у даний алгоритм, що позитивно вплинуло на швидкодію алгоритму у цілому.

Після проведення теоретичного дослідження було вирішено поставити експериментальне дослідження та розробити усі три алгоритми для проведення глибинного аналізу результатів роботи і визначення якості скорочення тексту за допомогою алгоритмів. У рамках проведеного експериментального дослідження були теоретично розглянуті та практично застосовані такі речі як:

1) інтерпретована мова програмування Python;

2) технології розробки користувацького інтерфейсу HTML, CSS, JS, Vue для Web-додатків;

3) технологія розробки серверної частини Tornado для Web-додатків;

4) система управління базами даних SQLite;

5) технології хмарного обчислення Amazon Web Services.

Як результат проведеного експериментального дослідження було розроблено Web-додаток, що у якості вхідних даних отримує URL-посилання на інший Web-додаток, що містить певну статтю, після чого отримує її зміст та відправляє до кожного з трьох розглянутих алгоритмів інтелектуального аналізу даних.

Як виявилося після проведеного експериментального дослідження алгоритм TextRank впорався з поставленою задачею анотації тексту краще за інші алгоритми, завдяки побудові графу на основі попереднього семантичного аналізу зв’язків між словами, реченнями та словосполуками у різних реченнях, що дало змогу побудувати високоякісну анотацію тексту, зберігаючи його сутність та сенс.

Не дивлячись на значне підвищення продуктивності у роботі з даним алгоритмом існують додаткові шляхи для розвитку у цьому напрямку. Так, одним із можливих варіантів покращення швидкодії алгоритму може бути багатопоточність обчислення вершин і ребер, побудованого графа, де кожна вершина графа буде обчислюватися на окремому ядрі процесора, що дасть змогу суттєво підняти алгоритм на новий рівень швидкості роботи не тільки з невеликими статтями, а також з великими об’ємами даних.

Щодо застосування алгоритма, то у рамках даної роботи ми розглянули лише один із можливих варіантів використання алгоритмів даного типу. Окрім, безумовно корисної анотації текстів, алгоритм автоматизованого інтелектуального аналізу текстів також здатен на багато інших задач, при умові мінімальної адаптації. Так, алгоритми на основі графу здатні ранжувати не тільки речення у тексті, а і документи поміж собою, що вже являє собою реалізацію інтелектуальної пошукової системи. На додачу до можливості створення пошукової системі на основі таких алгоритмів це ж дає змогу і для пошукової оптимізації для Web-додатків у мережі Інтернет, що є вкрай важливим для прогресу у сфері інформаційних технологій.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. Brin and L. Page. 1998. The anatomy of a large scale hypertextual Web search engine.
2. Erkan and D. Radev. 2004. Lexpagerank: Prestige in multi document text summarization.
3. Lin and E.H. Hovy. From Single to Multi document Summarization: A Prototype System and its Evaluation.
4. Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization.
5. Mihalcea and P. Tarau. 2004. TextRank - bringing order into texts.
6. Kleinberg. 1999. Authoritative sources in a hyperlinked environment.
7. Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics.
8. Lin and E.H. Hovy. 2003. The potential and limitations of sentence extraction for summarization.
9. Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation.
10. Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K. J. Introduction to WordNet.
11. Morris, J., Hirst, G. Lexical cohesion computed by thesaural relations as an indicator of the structure of text.
12. Radev, H. Y. Jing, M. Stys and D. Tam. Centroid-based summarization of multiple documents.
13. Robert de Beaugrande and Wolfgang Dressler. Introduction to Text Linguistics.
14. Silber, H. G., McCoy, K. F. Efficient text summarization using lexical chains.

15. Turney. 1999. Learning to extract keyphrases from text. Technical report, National Research Council, Institute for Information Technologies.
16. Касаткина Н.В. Методы хранения и обработки нечетких данных в среде реляционных систем / Н. В. Касаткина, С. С. Танянский, В. А. Филатов // Автоматика. Автоматизація. Електротехнічні комплекси та системи. ХНТУ. Херсон. 2009. випуск 2 (24). С. 84-90.
17. Пономаренко Л.А. Модель системы управления информационными ресурсами на основе технологии программных агентов / Л. А. Пономаренко, В. А. Филатов // Проблеми інформатизації та керування: Збірник наукових праць: Київ: НАУ, 2004. Вип.9. С. 8-16.
18. Filatov V., Radchenko V. Reengineering relational database on analysis functional dependent attribute // Proceedings of the X Intern. Scient. and Techn. Conf. «Computer Science & Information Technologies» (CSIT'2015), 14-17 sept. 2015. Lviv, Ukraine. P. 85-88.
19. Филатов В.А., Кривоносов В.А., Козырь О.Ф. Адаптивные автономные сценарии в задачах управления информационными ресурсами предприятия. *Инженерный вестник Дона*. 2013. № 3. URL: http://ivdon.ru/magazine/archive/n3y2013/1779 (дата звернення: 10.11.19).
20. Филатов В.А. Мультиагентные технологии интеграции гетерогенных информационных систем и распределенных баз данных: дис. докт. техн. наук: 05.13.06. Харьков: Харьков. нац. ун-т радиоэлектроники. 2004. 341 с.
21. Filatov V. Fuzzy models presentation and realization by means of relational systems // Econtechmod: an international quarterly journal on economics in technology, new technologies and modelling processes. Lublin; Rzeszow, 2014. Vol.(3), № 3. P. 99-102.