Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка
До кваліфікаційної роботи
магістра

на тему: «Прогнозування цитування наукових текстів на основі семантичного аналізу»
за освітньою програмою інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення.

Виконав: студент групи ПЗ2121М: /Ілля КАМЕНЄВ/

Керівник: /Вадим АНДРЮЩЕНКО/

Нормоконтролер: /Світлана ВОЛКОВА/

Консультанти:
Економічний розділ /Микола ГНЕНИЙ/

Засвідчую, що у цій роботі немає запозичень з
праць інших авторів без відповідних посилань
Студент

Дніпро – 2022 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»
Завідувач кафедри
___ Вадим ГОРЯЧКІН

ДИПЛОМНА РОБОТА

на здобуття ОС «магістр»

Галузь знань 12 Інформаційні технології
(шифр) (назва)

Спеціальність 121 Інженерія програмного забезпечення
(код) (повна назва)

Освітньо-професійна програма Інженерія програмного забезпечення
(повна назва)

Тема «прогнозування цитування наукових текстів на основі семантичного аналізу»

Theme «prediction of citations of scientific texts based on semantic analysis»
(theme in English)

Керівник дипломної роботи доцент _____Вадим АНДРЮЩЕНКО
(посада) (підпис) (ПІБ)

Консультант економічного розділу _____ _____
(посада) (підпис) (ПІБ)

Нормоконтролер _____ _____ _____
(посада) (підпис) (ПІБ)

Студент групи ПЗ2121 Каменєв І.Ф.
(номер групи) (підпис) (ПІБ)

Student _____
(Family name)

Дніпро 2022

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

«Computer technologies and system»

(faculty)

«Computer information technology»

(department)

Explanatory Note

to Master's Thesis

_____

(higher education degree)

on the topic: «prediction of citations of scientific texts based on semantic analysis»

according to educational curriculum_____

in the Speciality: «121 Software engineering»

(speciality and its code) «Software engineering 121»

Done by the student of the group «П32121» /Illia Kameniev/

(name, surname)

Scientific Supervisor: /_____/

(position, name, surname)

Normative controller : /_____/

(position, name, surname)

Supervisors

_____ /_____/

(Chapter title heading) (position, name, surname)

_____ /_____/

(Chapter title heading) (position, name, surname)

_____ /_____/

(Chapter title heading) (position, name, surname)

Dnipro – 2022

Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем кафедра Комп’ютерні інформаційні технології

Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

_____ Вадим ГОРЯЧКІН

(підпис)

«___»_____2022р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС магістр
(освітній ступінь)

студента групи _____ _____
(номер групи) (ПІБ)

1 Тема дипломної роботи: прогнозування цитування наукових текстів на основі семантичного аналізу _____

затверджена наказом по університету від «___» _____ 201__ р. № ____.

2 Термін подання студентом закінченої роботи_____

3 Вихідні дані до дипломної роботи _____

_____

_____

_____

4 Зміст пояснювальної записки (перелік питань до розробки)

Теоретичні засади по інформаційним технологіям прогнозування цитування наукових текстів. Застосуваня в освітньому процесі. Опис програмного додатку. Етапи розробки та створення додатку. Дослідження області робои та результати данного дослідження.

5 Перелік демонстраційного матеріалу

Вступ та теоритичні засади;

Існуючі методи прогнозування;

Етапи основної моделі прогнозування;

Огляд аналогів;

Середовище створення продукту;

Вимоги до технічного забезпечення;

Функціонал;

Висновки.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки			
Охорона праці			

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ		
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами		від 70 джерел
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження		
4	Постановка задачі, технічне завдання		30%
5	Техніко-економічні показники		
6	Розробка інструментальних засобів дослідження		
7	Виконання досліджень		60%
8	Оформлення тез доповідей		
9	Оформлення статті у фаховий журнал		
10	Оформлення пояснювальної записки		
11	Розробка демонстраційних матеріалів		100%

Дата видачі завдання «___» _____ 2022р.

Керівник дипломної роботи _____ Вадим АНДРЮЩЕНКО
(підпис) (ПІБ)

Завдання прийняв до виконання _____ _____
(підпис) (ПІБ)

Зміст

ВСТУП

Мовна обробка тексту природною мовою є одним із основних питань розробки інформаційних технологій. Такі розвинені країни, як Європа та США, приділяли значну увагу цьому питанню, про що свідчать величезні кошти, що виділяються на розробку мовного програмного забезпечення. Численні дослідницькі проекти спрямовані на розвиток мовних інформаційних систем. Зі швидким розвитком Інтернету та інших комп'ютерних і комунікаційних технологій ця проблема стала серйознішою.

З середини 1950-х років основні зусилля вчених були спрямовані на розробку математичних алгоритмів і комп'ютерних програм для обробки тексту природною мовою. Для автоматичного аналізу та синтезу тексту створюються різноманітні моделі процесу обробки тексту, а також відповідні алгоритми та структури представлення даних. Традиційно аналіз текстів природної мови подається як низка процесів - морфологічний аналіз, синтаксичний аналіз, семантичний аналіз. Для кожного етапу створюються відповідні моделі та алгоритми. Для семантики тексту – класична семантична мережа Мінського та модель фрейму, для синтаксису речень – граматика Хомського, системна граматика Холлідея, дерево залежностей Гладкі та система компонентів, розширення мережі перетворення; для морфологічного аналізу розроблено багато різних моделей, увага зосереджена на конкретному мовна група.

Найскладніші проблеми при обробці текстів природної мови виникають через такі явища, як полісемія, омофонії тощо, які вносять багатозначність у мову та ускладнюють завдання правильного відображення семантико-синтаксичної структури тексту в його формі. логічне представлення. Усі ці питання розглядаються на рівні семантичного аналізу.

З іншої сторони. Використання ресурсомістких можливостей логічного семантичного аналізу робить програми обробки тексту надто складними та повільними. Логіку в розумінні тексту використовують нечасто – інші механізми використовуються лише тоді, коли є проблеми з логікою, насамперед – пошук за асоціацією, формою чи контекстом.

Великі компанії кожен день намагаються розширити свої можливості ботами, які дають змогу виконувати різноманітні дії – від вибору товарів в інтернеті до безпосереднього їх замовлення без участі людини. Варто зауважити, що боти почали активе співіснування з людьмо зовсім нещодавно, тому актуальність питання щодо семантичного аналізу тексту наразі знаходиться ледь не на своєму апогеї.

В нашому разі необхідно провести дослідження методів пошуку релевантності, семантичного аналізу, прогнозування та навчання нейромереж.

Пошук релевантності – це оцінка понять, які відповідають даному слову та контекстуально близькі до його оточення. Тим часом асоціативний пошук є швидким і економним способом вирішення неоднозначності в текстових інтерпретаціях. Тому частина методу пов’язана з визначенням контекстної асоціації. Для роботи цих методів необхідна бібліотека словників онтології (онтологія), яка містить пов’язані мовні поняття (поняття), зв’язки між поняттями та словами мови, а також зв’язки між поняттями (синоніми, антоніми, надслова, гіпоніми) Інформація. , і інші). З онтологіями використовується ряд алгоритмів, а саме: алгоритми визначення концептів зі слів у тексті, алгоритми відновлення значення мовного індикатора на основі онтології, алгоритми визначення тематичної атрибуції слів і концептів. Текст і текст у цілому, алгоритм визначення змістовної близькості слів і понять, алгоритм узагальнення.

Пошук за формою мовної структури - оцінюйте фрази/речення за їхньою формою (структура синтаксису, регулярні вирази, наявність певних елементів) і створюйте відповідні відповіді на знайдені шаблони. Водночас вчені намагаються побудувати такі шаблони чи розмітку, щоб дозволити охопити якомога більше мовних явищ.

Наразі дуже велика увага приділяється саме підходам до лінгвістичного аналізу та обробки текстів. Значимість та актуальність проблеми зростає з кожним днем розвитку інтернету. Буквально щодня відбуваються інвестиції в цю сферу з боку найбільших компаній сьогодення.

Головною метою та завданням дослідження цієї роботи є вивчення підходів та методів дослідження семантичного аналізу, раціональний підбір підходу для вибору нейронної мережі, яка буде навчатися в процесі в процесі та давати певні прогнозування.

Кожен метод дослідження слугує для найкращого розкриття семантичних харакретистик мовної одиниці. Це дозволяє абсолютно точно зрозуміти всю суть та думку, яка була закладена в об’єкт.

Порівняльний метод абсолютно точно відповідає своїй назві. Адже завдяки порівнянню ми можемо знайти схожість та певні зв’язки. Тобто, чим вище будуть знайдені нами зв’язки, тим більша ймовірність того, що результат аналізу певної одиниці інформації буде давати нам приблизно аналогічний результат, якщо не точний, але з мінімальними розбіжностями.

Для прикладу можемо навести аналіз слів з різних мов, які близькоспоріднені за своїм походженням. Очевидно буде зауваження, що слова, які несуть за собою однакове значення, мають явну схожість між собою. Але в цьому разі важливо зауважити, що при подібному порівнянні треба враховувати низку особливостей кожної мови, від правил написання до фонетики, що значно ускладнює задачу.

На сьогодні всі сучасні методи дослідження семантики так чи інакше дотичні до порівнянна та співставлення, адже це є основні фундаментальні підходи, які можна використовувати.

Наступним методом є метод компонентного аналізу, який можна назвати дочірнім від порівняльного. Але в цьому методі все зводиться до пошуку ознак, які відрізняються між собою, або ж навпаки, поєднуть ті чи інші одиниці.

Диференціальними називають ознаки, які відрізняють інформацію, а інтергральними є об’єднувальні або ж спільні ознаки.

Семантична основа кожного слова будується на основі декількох його значень, а також пунктів, які відкидають інші визначення, які ніяким чином його не стосуються. Таким чином спекстр пошуку буде максимально скерований в необхідному напрямку, завдяки чому буде відкинуто хибні допущення та обрано максимально наближені або точні значення.

Наукова новизна роботи розкривається в аналізі особливостей семантичних конструкцій.

- Наведення детального списку ознак цільової семантики
- Було розглянуто специфік семантики в порівнянні з близькими значеннями або ж призначеннями
- Аналіз випадків семантичної схожості в ході роботи аналізу
- Надано повний аналіз синтаксичних засобів від простого слова до складних речень та більше

Практична цінність та значимість полягає в тому, що отримані практичні та теоритичні засади можна використовувати при вивченні відповідних розділів лінгвістики.

Фрейми використовуються для аналізу складних ситуацій. Вони забезпечують найбільш зручний механізм представлення строго структурованих знань про предметну область або завдання.

Поєднання всіх цих методів із налаштованими статистичними методами може значно спростити та прискорити створення та використання інтелектуальних систем обробки тексту.

Для кожної задачі також наводяться спеціалізовані методи, які мають основне застосування лише у вказаній задачі.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ПО ІНФОРМАЦІЙНИМ ТЕХНОЛОГІЯМ ПРОГНОЗУВАННЯ ЦИТУВАННЯ НАУКОВИХ ТЕКСТІВ. ЗАСТОСУВАННЯ В ОСВІТНЬОМУ ПРОЦЕСІ

1.1 Теоретичні засади

Потреба в дослідженні та розробці автоматизованих довідкових систем зумовлена збільшенням кількості та обсягу електронних документів, які потребують обробки, оскільки ці документи здебільшого мають форму неструктурованого тексту, що складається з природної мови, а більшість програмного забезпечення орієнтоване з обробки структурованих даних. Крім того, в Інтернеті зростає кількість новинних сайтів, де різні сайти пропонують різні інтерпретації однієї події. З поширенням мобільного Інтернету та пристроїв на рівні смартфонів це породило потребу в системі, яка збирає дані з різних джерел, які можуть надавати користувачам короткі, але достатні підсумкові звіти, що висвітлюють поточні новини. Інше використання автоматичного посилання пов’язане із системами підтримки прийняття рішень. Експертам необхідно аналізувати великі обсяги документів для швидкого перегляду, а успішна довідкова система може скоротити час, необхідний для читання. Замінити довідкову систему пошуковою системою неможливо, тому що пошукові системи шукають те, що фахівці вже знають або здогадуються.

Автоматичні системи відліку здебільшого бувають двох типів. Це або системи, вбудовані в якийсь великий продукт, від агрегаторів новин до Microsoft Office Word, із різними рівнями участі користувачів у процесі, або онлайн-системи. Останні здебільшого безкоштовні та пропонують прості алгоритми та низькоякісні збірки.

З огляду на те, що численні цитати мають на меті побудувати анотації з кількох документів, необхідно певним чином зробити можливі зв’язки між текстом та його частинами. Це, у свою чергу, вимагає вирішення проблем індексування та визначення близькості між текстами.

Близькість між текстами – за тематикою, за метою викладу матеріалу (висновок, який пропонує текст), також можливе часткове або повне запозичення тексту. Найбільшою особливістю новинних текстів є запозичення, коли інформаційна організація передруковує цитату з іншої новини.

Окремим завданням є забезпечення читабельності статті. Якщо текст реферату формується шляхом підбору та редагування речень вихідного тексту, то необхідно забезпечити зв’язність тексту.

Зв’язність тексту — це наявність елементів, які дають змогу правильно визначити зв’язки між елементами, що мають зміст у тексті.

Основною причиною порушення зв’язності є те, що змістовні речення, що йдуть після виділених речень, є в оригінальному тексті, а не в анотації. Коли виділені речення зв’язані у зв’язний текст, анотація готова. При багаторазовому цитуванні, окрім зв’язності, необхідно забезпечити мінімальний рівень повторення, оскільки часте й необґрунтоване повторення може зробити абстрактні тексти психологічно неприємними та стилістично невиправданими.

Рівень стиснення – це те, що зберігається в оригінальному тексті. Звичайно, рівень стиснення становить від 30% до 5% вхідного тексту.

Отже, у загальному вигляді завдання абстрагування полягає в наступному: з набору текстів (ймовірно, одного з елементів) побудувати мінімально повторюваний текст, який відображає головний елемент (основну подію) вхідного тексту і легко читати.

Абстрактні завдання зі стиснення фрагмента тексту в один тут не розглядаються, хоча вони дуже актуальні, особливо для агрегаторів новин.

Можливість передбачати педагогічні явища, процеси та об'єкти та ефективно впливати на них з боку освітніх систем та самих учнів та їх педагогів на освітню систему стала однією з найважливіших проблем дидактики XXI ст., у період зростання впливу цифрових технологій на всі сфери людської діяльності. Ведеться пошук переліку факторів (показників та критеріїв їх оцінки), що впливають на освітні результати, що досягаються студентами.

Термін «педагогічне прогнозування» у зарубіжній літературі найчастіше описується авторами з пострадянського простору, а більшість написаних робіт – українською мовою. Іноземні вчені найчастіше розглядають його як частину соціального чи економічного прогнозування [9; 10]. На думку авторів роботи, педагогічна наука має особливості об'єкта прогнозування, хоча соціальне прогнозування справді ширше поняття, але не враховує специфіки дидактики. Педагоги повинні не тільки вміти складати ефективний прогноз, але й вживати екстрених заходів щодо усунення недоліків у виховній та освітній роботі. Деякі вітчизняні автори розглядають прогнозування без застосування сучасних обчислювальних та комп'ютерних методів, що є неприпустимим, на наш погляд, у вік інформаційних технологій та «великих даних» (Big Data – обсяги інформації, які неможливо впорядкувати без спеціальної комп'ютерної обробки). У цілому нині проблема прогнозування актуальна й у ній є низка не остаточно досліджених питань, наприклад, існує так багато технологій прогнозування в дидактиці.

1.2 Методи прогнозування цитування наукових текстів

Допоміжні засоби перераховані, але не детально описані, оскільки їх можна реалізувати кількома способами, і заміна їх еквівалентами не змінює роботу всієї системи.

Зверніться до допоміжних засобів, які використовуються в роботі системи:

- підсистеми морфологічного аналізу (наприклад, [6], або словники, вбудовані в системи аналізу вищого рівня [7]), результатами роботи яких є синтаксичні нотації та парадигми (леми);

- Частина підсистеми аналізу (припускається, що вона реалізує принаймні зв'язок між іменниками та прикметниками, більш потужна [7]);

- підсистема заміни займенників (описана в [9,12]);

- Підсистема семантичного аналізу (опціонально [2,13]);

На цьому задача поділу тексту на слова та визначення морфологічної характеристики слів вважається вирішеною. На основі отриманих морфологічних

даних проведено частковий синтаксичний розбір. Прикметники (прислівники) пов’язані з відповідними іменниками, що є мінімальною вимогою для роботи системи. Щоб замінити займенники повнозначними словами (антецедентами), до яких вони відносяться, використовуйте спочатку морфологічний запис. І лише якщо їх недостатньо, скористайтеся простим семантичним розбором, тобто: серед варіантів виберіть слово, яке за семантичною близькістю найближче до слова в займенниковому контексті. Для визначення семантичної близькості рекомендується використовувати семантичну бібліотеку WordNet (за наявності згоди локалізовані версії українською мовою) та алгоритм пошуку найкоротшої відстані.

Практично в кожному рішенні, яке вони приймають, сьогодні керівники враховують якийсь прогноз. Обґрунтовані прогнози попитів і тенденцій – це вже не предмети розкоші, а необхідність, якщо менеджери хочуть впоратися з сезонністю, різкими змінами рівня попиту, маневрами конкурентного зниження цін, страйками та великими коливаннями економіки. Прогнозування може допомогти їм впоратися з цими проблемами; але це може допомогти їм більше, чим більше вони знають про загальні принципи прогнозування, що воно може, а що не може зробити для них зараз, і які методи відповідають їхнім потребам на даний момент. Тут автори намагаються пояснити менеджерам потенціал прогнозування, приділяючи особливу увагу прогнозуванню збуту продукції Corning Glass Works, оскільки вона дозріває протягом життєвого циклу продукту. Також міститься короткий опис методів прогнозування (рис. 1.1).

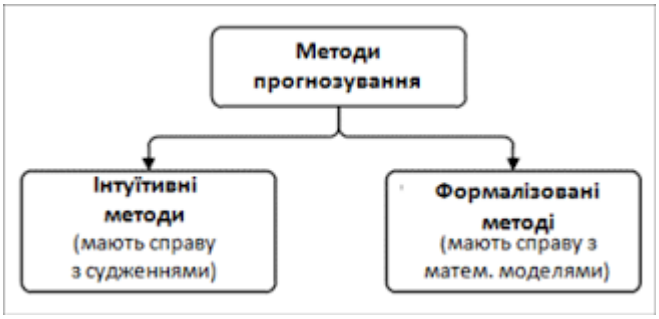


Рис. 1.1 Методи прогнозування

Для вирішення зростаючого різноманіття та складності проблем управлінського прогнозування в останні роки було розроблено багато методик прогнозування. Кожен з них має своє особливе застосування, і потрібно уважно вибирати правильну техніку для конкретного застосування. Менеджер, а також прогнозист відіграють роль у виборі техніки; і чим краще вони розуміють діапазон можливостей прогнозування, тим більша ймовірність, що зусилля компанії з прогнозування принесуть результати.

Вибір методу залежить від багатьох факторів — контексту прогнозу, релевантності та доступності історичних даних, бажаного ступеня точності, періоду часу, який потрібно прогнозувати, вартості/вигоди (або цінності) прогнозу для компанії та час, доступний для проведення аналізу.

Ці фактори необхідно зважувати постійно і на різних рівнях. Загалом, наприклад, синоптик повинен вибрати методику, яка найкраще використовує наявні дані. Якщо прогнозист може легко застосувати одну методику з прийнятною точністю, він або вона не повинні намагатися «золотою пластиною» використовувати більш досконалу техніку, яка пропонує потенційно більшу точність, але яка вимагає неіснуючої інформації або інформації, отримання якої є дорогим. Такий компроміс відносно легко зробити, але інші, як ми побачимо, вимагають значно більше уваги.

Крім того, якщо компанія бажає зробити прогноз щодо певного продукту, вона повинна враховувати етап життєвого циклу продукту, для якого вона робить прогноз. Доступність даних і можливість встановлення взаємозв'язків між факторами безпосередньо залежать від зрілості продукту, і, отже, етап життєвого циклу є головним визначальним фактором методу прогнозування, який буде використовуватися.

Наша мета — представити огляд цієї галузі, обговоривши, як компанія повинна підійти до проблеми прогнозування, описати доступні методи та пояснити, як узгодити метод із проблемою. Однак у міру необхідності ми торкнемося інших продуктів та інших методів прогнозування.

Під технологією прогнозування будемо розуміти сукупність етапів і операцій, послідовне виконання яких дозволить отримувати вірну попереджувальну інформацію про об'єкт, що вивчається.

Основна модель прогнозування складається з наступних етапів

1. Стадія ретроспекції

– Визначення цілей та завдань педагогічного прогнозування.

– Інтернет-аналітика, контент-аналіз

– Передпрогнозна орієнтація. Відстеження результатів підготовчої інформації

– Попередня систематизація зібраної інформації про навчальний процес

– Розробка попередньої програми дослідження

– Неформалізовані методи, кластеризація, візуальна аналітика

– Організаційна аналітика, карти майбутнього та SWOT-аналіз

2. Стадія постановки діагнозу

– Складання попереднього переліку показників моделі за даними різних джерел.

– Збір даних прогнозного фону (економічний, соціологічний, соціально-культурний, політичний та міжнародний, правовий, педагогічний, демографічний, природний, філософський, науково-технічний, організаційний)

– Зведення попереднього переліку показників до виду, який допоможе уявити найкращий вид педагогічного прогнозу

– Аналіз діагностичної інформації

3. Стадія проспектції

– Побудова динамічних рядів прогнозування за кожним показником базової моделі

– Визначення абсолютного оптимуму з умовним абстрагуванням від обмеження прогнозного фону

– Верифікація педагогічного прогнозу та експертиза

– Коригування базової моделі прогнозування

– Побудова часових рядів, аналітика в реальному часі: візуальна та багатовимірна

1.3 Структура наукових текстів

Існує два способи створення тематичних демонстрацій - за допомогою фіксованих і динамічних тем. Індекси з фіксованою темою спираються на словники з фіксованою темою.

Найпростіше складати теми за чітко визначеними термінами: біологія, хімія, інформатика, фінанси, геологія та географія, право, лінгвістика, математика, атомна енергетика, фізика тощо. У цих темах ці ключові слова містять як детальні, так і більш загальні поняття.

Складним є створення тематичних словників для загальнополітичного дискурсу, як і історичних словників. Рекомендується реалістичний професійний словник відповідних місць (регіонів) і часів.

Динамічний індекс використовує динамічно створений набір повних слів, які належать приблизно до однієї теми.

Для завдань з кількома цитатами, які зосереджуються переважно на текстах новин, нерозумно використовувати фіксований список ключових слів. Використовуйте лише список слів, які будуть проігноровані під час аналізу теми тексту.

Значна близькість

Змістова близькість є одним із найважливіших елементів зв’язності тексту. Розглянемо такий фрагмент тексту:

«Розвиток обчислювальної техніки збігся з розвитком техніки, на зміну електронним лампам прийшли напівпровідникові, на їх основі створювалися інтегральні схеми, все більше логічних елементів стали розміщувати на кристалі.

У цій роботі виділяються два типи змістовної близькості: - пов'язана за значенням терміна;

- Пов'язані зі структурою тексту.

Фрази, виділені в тексті жирним шрифтом, представляють поняття, пов’язані зі значенням відповідного слова чи терміна. Перший тип зв’язку базується на знанні того, що два елементи (концепти) у базі знань пов’язані один з одним.

Підкреслені слова позначають поняття, пов’язані з іншими поняттями, оскільки вони розташовані поруч. Цей тип зв'язку встановлюється автором тексту і відображає саме те, що автор хоче повідомити цим текстом.

Зв’язність може бути досягнута або безпосередньо через терміни, або через лінгвістичні індикатори (які в лінгвістиці називаються анафорами). Задача встановлення елемента, на який вказує покажчик мови, називається розв’язуючою анафорою [12].

Словниковий ланцюжок

Лексичні ланцюжки представляють значущі одиниці між будь-якою кількістю сполучників.

По-перше, це лексичний ланцюжок одного і того ж об’єкта. Він включає в себе ті текстові елементи, які відносяться до того самого об'єкта або того самого поняття. Задача пошуку всіх згадок одного і того ж об’єкта називається задачею розв’язання кореференцій [12].

По-друге, це лексичний ланцюжок семантично пов’язаних об’єктів.

Такі ланцюги не обмежують типи елементів і типи зв'язків між ними, якщо вони взаємопов'язані з точки зору автора. Саме таке значення терміна буде використовуватися надалі, оскільки воно визначає не лише тотожність/відмінність між об’єктами, а й тематичну приналежність. Для побудови такого словникового ланцюжка використовується WordNet як джерело знань.

Способи поєднання ланцюжків слів

Лексичні ланцюжки обчислюються шляхом групування послідовних наборів семантично пов’язаних слів. Однотонні слова, синоніми, гіперніми та гіпоніми, мероніми, голоніми — це символи, які дозволяють об’єднати слова в ланцюжок.

Гіпернім — це узагальнююче поняття, дане в онтології (WordNet [8]). Гіпонім — це концепція, яка є заданою специфікацією в онтології (WordNet).

Частка — це поняття, яке представляє «ціле» у відношенні «ціле-частина». Голонім — це поняття, яке представляє «частину» у відношенні «ціле-частина».

Важливо зазначити, що WordNet представляє не слова, а значення (поняття) — поняття, і кожне значення має власний набір слів для його представлення. Такий набір слів називається "синсет" умова групування.

1. Обидва входження синоніма однакові і вживаються в тому самому значенні. (Великий корабель на рейді. Корабель — вітрильник.)

2. Повні синоніми з’являються двічі з однаковим значенням, але різною граматикою, тобто синоніми. (Цей літак летить швидко. Але мій швидше.)

3. Між змістом двох повнозначних слів існує зв’язок гіпернім/гіпонім. (У мене є автомобіль. Це вантажівка.)

4. Вміст повнозначних слів, які зустрічаються двічі, є елементами одного рівня в дереві гіпернімів/гіпонімів і мають спільного предка. (Цей Airbus літає швидко. Проте мій винищувач швидший.)

5. Два входження повнозначних слів позначають частину і ціле - з меронімічним зв'язком. (Дмитро відчиняє двері. Клацає замок.)

При обчисленні лексичних ланцюжків входження повнозначних слів необхідно групувати за наведеними вище правилами, але кожне входження повнозначних слів має належати рівно одному лексичному ланцюжку.

Розглянемо загальний підхід до побудови колекції лексичних ланцюжків для заданого тексту, запропонований Регіною Барзілай.

1. Виберіть слово або фразу (далі – об’єкт) із тексту та відобразіть їх у WordNet.

2. Для кожного предмета знайдіть відповідний ланцюжок і вставте в нього предмет.

3. Якщо ланцюжка не існує, створити новий на основі зазначеного об'єкта.

Як видно з опису методів, чіткої відповіді на деякі питання немає.

1. Як визначити правильний ланцюг? Наприклад, входження іменника може відповідати кільком різним значенням слова, і система повинна визначити, яке входження є таким. Наприклад, «коса» як засіб і «коса» як зачіска.

2. Як забезпечити ясність? Навіть якщо значення слова можна визначити, може виникнути така ситуація, що слово можна ввести в кілька різних лексичних ланцюжків, оскільки слово може бути пов’язане зі словами в різних ланцюжках.

Для коректності вводяться параметри, які об'єднують об'єкт з ланцюжком. Ці параметри включають відстань і напрямок ключових критеріїв групування, визначених у попередньому розділі. Важливо відзначити, що комбінуючи умови групування, можна встановити як завгодно складні зв’язки.

1. Для умови групування 3 накладається обмеження на довжину такого зв’язку при обчисленні від більш детальної концепції до більш загальної концепції, а не навпаки.

2. Для умови групування 4 накладається обмеження на відстань від узагальненого поняття. У нашому випадку - 2.

3. Для умов групування 5. Якщо в шляху є різні типи переходів, накласти обмеження на довжину такого посилання. У нашому випадку - 2.

Важливо відзначити, що якщо в аналізованому тексті два слова розташовані близько одне до одного, вони, швидше за все, будуть пов’язані разом. Це призводить до ще одного параметра асоціації: об’єкти пов’язані, якщо вони знаходяться в суміжних реченнях.

Для різних умов групування варто встановлювати різні допустимі відстані для визначення сусідства.

1. Умови 1-3, відстань може бути до 7 речень.

2. Умови 4-5, відстань не повинна перевищувати 4 речень.

При визначенні запозичення необхідно враховувати обставини, за яких відбулося запозичення. Якщо людина вдасться до плагіату, то, ймовірно, вона переробить текст, щоб приховати факт незаконного запозичення. При цьому необхідно домогтися мінімізації випадків помилкової активації. Якщо газета

передруковує новину, вона зазвичай містить посилання на авторство та, можливо, нову інтерпретацію та переробку тексту.

У тексті розслідування підозрілі фрагменти можуть мати такі спотворення:

– межі підозрілих фрагментів не обов'язково збігаються з межами абзаців або речень;

– підозрілі фрагменти можна змінити шляхом вставки та видалення елементів;

– елементи можуть бути переставлені в підозрілі фрагменти;

– у підозрілих фрагментах можливі синтаксичні зміни, що проявляються у вигляді зміни граматичної форми слів;

– у підозрілих роликах можливі стилістичні зміни, що проявляються у заміні слів синонімами;

– обмеження цитат у підозрілих кліпах можуть не відповідати оригінальним кліпам. Загалом, текст – це спеціальне відображення знань про дійсність поза текстом. Текст повинен мати такі властивості: роздільність, смислову цілісність, зв'язність. При цьому слід пам'ятати, що різні спостерігачі (читачі) побачать різні тексти в одному наборі символів.

Приналежність до тексту означає, що текст поділений на абзаци, згруповані за функцією однорідної одиниці, представленої сюжетом або темою. Порядок абзаців утворює порядок, у якому одна одиниця є похідною від іншої чи інших одиниць. Це, у свою чергу, формує висновок, який автор хоче донести до читача. У свою чергу абзац поділяється на речення — групу кількох споріднених слів, які виражають певний зміст. Речення складаються зі слів і словосполучень, які відображають зміст і зв'язок між ними.

Семантична цілісність досягається, якщо читач може коротко викласти текст і дати йому назву.

Зв'язність тексту залежить від наявності в ньому зв'язків, забезпечених семантичними чи синтаксичними засобами. Семантика стосується раніше описаних згадок, не обов’язково повторюваних форм слів. Це може бути посилання на аспект або якість. Такі згадки створюють «старе і нове» відношення

між елементами пропозиції. Синтаксичні засоби включають використання слів або навіть синтаксичних структур, які не мають власного значення, але вказують на вищезгадані сутності.

Текст можна розглядати як низку тематичних або сюжетних елементів, причому кожен наступний елемент набуває наступного зв'язку з усіма попередніми елементами. Не обмежуючи загальності, ми розглядаємо скорочення слів у тексті до їх нормальної форми, що значно спрощує роботу.

$$T_k = T_k(A(T), R(T)),$$

де

T_k – один текст,

A(T) – набір абзаців,

R(T) - це наступне співвідношення, визначене в абзаці, залежно від того, як і що письменник хоче донести до читача.

Абзац ділиться на набір речень, кожне з яких вводить, роз'яснює або пов'язує якийсь зміст.

$$a = a_i(S(a_i), R(a_i)),$$

де a_i - і-й абзац,

$S(a_i)$ — набір речень в абзаці,

$R(a_i)$ – наступне співвідношення залежить від речення, від думок та ідей, які автор хоче донести до читача. Водночас між реченнями в одному абзаці є змістовні зв'язки та мовні показники.

Крім того, введено структуру речень, щоб нею можна було керувати словами, а не окремими реченнями.

$$s_{i,j} = s_j(W(s_j), R(s_j)),$$

де s_j — j-те речення і-го абзацу, $W(s_j)$ — набір слів у абзаці, а $R(s_j)$ — синтаксис між словами набір відносин.

Це може статися, коли між елементами тексту (наприклад, абзацами) відсутні значущі зв’язки. Він поставляється у формі

$$T = U \cdot T_k,$$

Потім текст для аналізу представляється як набір окремих текстів, кожен із яких має власне значення.

Зауважте, що в задачах із кількома цитуваннями текст насправді є багатотекстовим.

1.4 Огляд аналогічних програм

Існує великий клас веб-додатків, які включають відповіді користувача на параметри. Такі об'єкти називаються рекомендаційними системами. Є два гарних приклади систем рекомендацій:

1. Надайте новинні статті читачам онлайн-газет на основі прогнозів читацького інтересу. [15]

2. Надайте клієнтам онлайн-магазинів пропозиції щодо товарів, які вони могли б захотіти придбати на основі минулої історії покупок та/або пошуку продуктів. [15]

Системи рекомендацій використовують багато різних методів. Ці системи можна розділити на дві великі категорії.

Система на основі вмісту перевіряє атрибути елементів.

Наприклад, якщо користувач Netflix дивиться багато ковбойських фільмів, порекомендуйте фільми, класифіковані як «ковбойські» в базі даних. [15]

Показники подібності між користувачами та/або елементами. Товари, рекомендовані користувачами, - це товари, які подобаються схожим користувачам. [15]

Одних лише технологій недостатньо, є кілька нових алгоритмів, ефективність яких доведена для систем рекомендацій. [15]

При побудові системи рекомендацій на даний момент використовуються три основних алгоритми:

- Алгоритм спільної фільтрації;
- Алгоритми на основі контенту;
- Гібридні алгоритми.

Основною ідеєю алгоритмів спільної фільтрації є побудова рекомендацій для конкретного споживача, маючи деякі дані про його профіль переваг, історію думок, рейтинги) або думки (рейтинги) групи однодумців.

У цьому випадку «схожістю» користувача вважається співвідношення вектора його рейтингу, яке можна розрахувати кількома способами. Можливість використання методів спільної фільтрації передбачає, що інтереси користувачів будуть представлені оцінками, які вони дають об'єктам після покупки, перегляду тощо.

Жодної інформації про самого користувача та того, кого оцінюють, не потрібно.

З іншого боку, алгоритми на основі контенту вимагають наявності інформації про характеристики об’єкта для створення рекомендацій. При цьому для роботи таких алгоритмів показники повинні бути представлені у вигляді структурованого тексту.

На сьогоднішній день існує понад 50 таких Інтернет-ресурсів - систем АМ, які постійно вдосконалюються і трансформуються в Translation Environments (TS) - Translation Environments (TeNT). SP — це третє покоління засобів автоматичного перекладу. У спільному підприємстві основною метою було не лише створення баз даних перекладу та допоміжних інструментів, а й забезпечення максимально зручного інтерфейсу та всіх необхідних допоміжних функцій. Приклад інтерфейсу наведено на рисунку 1.2.

Найвідомішими з спільних підприємств є SDL Trados, Transit, Deja Vu, Wordfast, AIT, MemoQ.

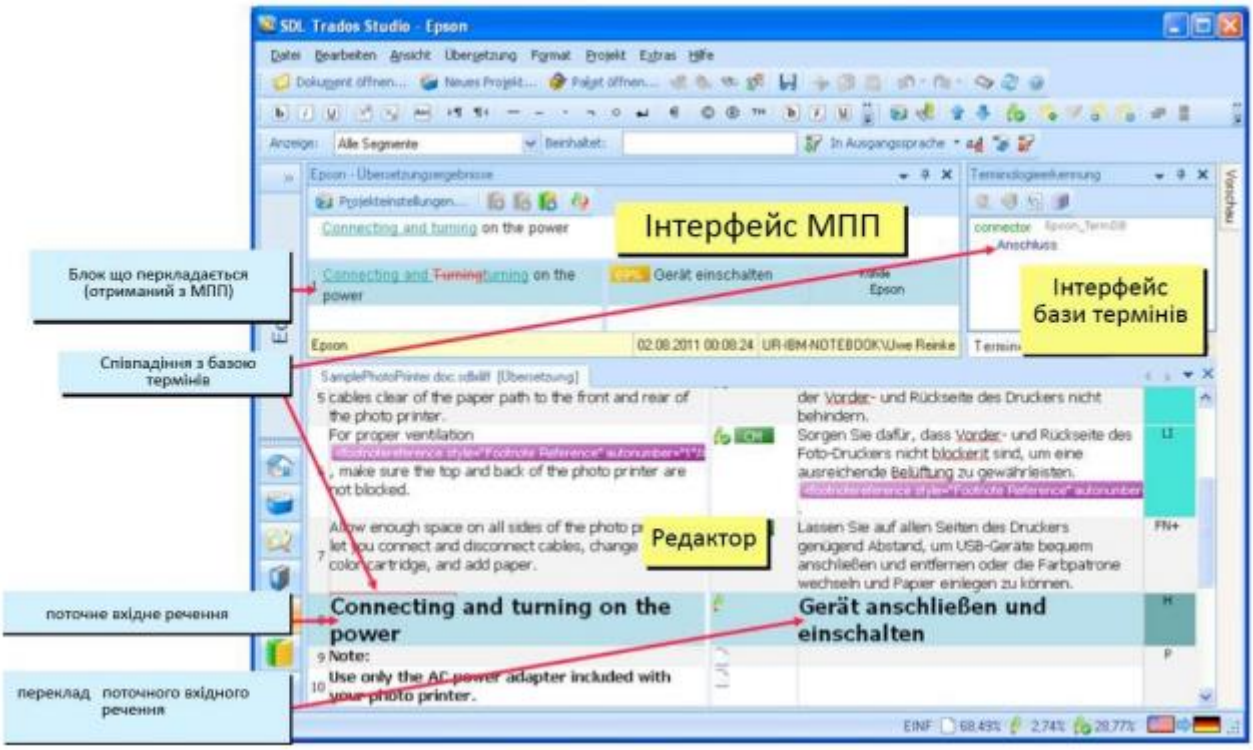


Рисунок 1.2: Інтерфейс користувача SDL Trados Studio Цитата: Уве Рейнке Сучасний стан технології пам’яті перекладів [14]

Веб-технології також з’явилися в окремому випадку, доповнюючи TA та SP машинним перекладом на основі статистичних даних у потужних інструментах, таких як служба Google Translate, що використовує власне програмне забезпечення Google. Сервіс дозволяє автоматично перекладати слова, фрази, текст і веб-сторінки з однієї мови на іншу, підтримуючи понад 80 мов. Перекладайте на 8 мов, включаючи українську, білоруську, каталонську, за допомогою споріднених, але більш поширених проміжних мов. Globefish, gTranslate і UnofficialGoogleTranslate також працюють із Google Translate.

Інші розробники продовжували змішувати моделі, включаючи не тільки англійську участь, як-от португальсько-китайський PCTAssist. Ми почали поєднувати MT і автоматичний переклад із хмарними технологіями, такими як Memsource Cloud. Це зріле середовище перекладу, вперше запущене в закритій бета-версії в 2011 році, включаючи пам’ять перекладів, інтегрований механізм машинного перекладу, керування термінологією та редактор перекладу, як веб- та окрему програму та використання хмарних служб.

1.5 Висновки до розділу 1

В першому розділі описано низку алгоритмів, які разом складають основу для створення систем автоматичного прив’язування або кількох привідних систем. Кожна з них окремо не може забезпечити достатню еталонну якість. Однак, якщо використовувати їх комплексно, більшість проблем можна вирішити. Це дозволяє дуже ефективно будувати статті, добре передавати зміст оригінального тексту і водночас легко читати.

РОЗДІЛ 2. ОПИС ПРОГРАМНОГО ДОДАТКУ

2.1. Опис технічного завдання

Розробити програму, яка реалізує систему яка прогнозує цитування наукових текстів на основі семантичного аналізу. Реалізувати прогнозування різними методами. Розробити можливість взаємодії з користувачем. Надати можливість вибору введення даних вручну, автоматично або зчитуванням із файлу. Також додати можливість зберігати отриманні данні до файлу.

2.2. Опис середовища та системи для створення власного продукту

Оскільки C++ одна із найпоширеніших мов програмування, йому існує велика кількість інтегрованих середовищ розробки з різним функціоналом. Оскільки висока продуктивність є однією з основних вимог до розробки програми/

Важливим критерієм вибору середовища розробки програми є наявність профільника. Як об'єкти дослідження були обрані інтегровані середовища розробки MS Visual Studio, JetBrains Clion, Qt Creator, NetBeans та Eclipse SDK , як найбільш функціональні та поширені.

а) MS Visual Studio

Microsoft Visual Studio - це інтегроване середовище, призначене для розробки програмного забезпечення, що має низку додаткових інструментів.

Продукти MS Visual Studio дозволяють розробляти консольні програми, програми з графічним інтерфейсом, веб-сайти, веб-програми та веб-сервіси для платформ.

Рис. 2.1 –Microsoft Visual Studio

MS Visual Studio містить вбудований інструмент для редагування вихідного коду програми та дозволяє проводити рефакторинг коду. Відладчик, вбудований у MS Visual Studio, має два рівні роботи: рівень налагодження вихідного коду та рівень налагодження машинного коду. Додаткові вбудовані інструменти включають форми GUI та зручний редактор для створення програм з інтерфейсами, містять веб-редактор та редактор класів. Це середовище розробки підтримує можливість створення та підключення плагінів (надбудов), що використовуються для розширення функціональності. MS Visual Studio підтримує системи контролю версій вихідного коду (наприклад, Visual SourceSafe або Subversion), містить безліч інструментів, що дозволяють редагувати та проектувати код предметно-орієнтованими мовами програмування.

б) JetBrains Clion

JetBrains CLion — інтелектуальне інтегроване середовище розробки програмного забезпечення, призначене для розробки додатків мовами C і C++ [28]. JetBrains Clion дозволяє розробляти програмне забезпечення на платформах Windows, OS X та Linux.

Рис. 2.2 - JetBrains CLion

Багатофункціональне середовище розробки - JetBrains CLion включає вбудований налагоджувач, безліч шаблонів коду та інтерфейс для популярних систем управління версіями (таких як Subversion, Git, GitHub, Mercurial, CVS, Perforce та TFS). CLion надає можливості автодоповнення та автоформатування коду, виконує аналіз коду на стрічці з виділенням потенційних проблем та пропонує способи їх усунення, підтримує систему складання для кросплатформових проектів CMake, а також різні рефакторинги коду. JetBrains CLion має великий репозиторій модулів (доповнень) для розширення існуючої функціональності.

в) Creator Qt

Qt Creator – це повністю інтегроване середовище розробки програмного забезпечення, що містить інструменти для проектування та розробки складних програм у різних операційних системах.

Рис. 2.3 - Qt Creator

Qt Creator призначений для розробки програмного забезпечення мовою програмування C++. Також є PyQt для програмування на Python та PHP-Qt для PHP.

Qt Creator інтегрований з кросплатформовими системами автоматизації складання: qmake і CMake, містить редактор коду Qt Designer, що дозволяє проектувати і створювати графічні інтерфейси користувача (GUI) з віджетів Qt. Це середовище містить багато корисних інструментів, таких як підтримка систем контролю версій (Subversion, Git, GitHub, Mercurial та CVS) та емулятор Qt. Qt Creator містить внутрішній налагоджувач для налагодження звичайних програм C++, а також включає можливість підключення мобільних пристроїв до вашого комп'ютера та налагодження програм, що працюють на них.

г) NetBeans

IDE NetBeans – це безкоштовне середовище IDE з відкритим вихідним кодом для розробників програмного забезпечення [30]. NetBeans надає безліч різних інструментів для створення професійного програмного забезпечення мовами Java, C/C++, PHP, Python, JavaScriot, Groovy, Ruby та інших.

Рис. 2.4 - IDE NetBeans

Середовище розробки NetBeans включає такі функції:

– рефакторинг коду;
– профілювання;
– виділення кольором різних синтаксичних конструкцій;
– автоматичне завершення при вході до різних споруд;
– велика кількість шаблонів коду.

У середовищі IDE NetBeans спочатку має бути J2EE SDK або потрібна версія Sun JDK.

NetBeans надає засоби управління системами контролю версій з готовими інтеграціями для Subversion, Mercurial і Git. Редактори середовища IDE та функція перетягування також дозволяють швидко та ефективно розробляти графічні інтерфейси програм.

г) Eclipse SDK

Eclipse SDK – це безкоштовне інтегроване середовище розробки програмного забезпечення з відкритим кодом. Eclipse SDK має велику і розгалужену систему модулів (надбудов), що підключаються, яка забезпечує робоче середовище з такими мовами програмування, як C, C++, PHP, Perl, Python, Ruby, Ada або COBOL.

Рис. 2.5 - Eclipse SDK

Eclipse SDK дозволяє створювати кросплатформне програмне забезпечення для Microsoft Windows, Mac OS X, дистрибутиви Linux та навіть Solaris. Це середовище включає засоби розробки Eclipse Java (JDT), які містять компілятор Java. У програмі використовуються інструменти, що входять до системи Eclipse Rich Client Platform.

Така система допомагає розробникам створювати потужні функціональні користувацькі програми з гарним графічним інтерфейсом на основі CSS (Cascading Style Sheets – каскадні таблиці стилів). Інтегроване середовище розробки.

Eclipse SDK надає Java IDE з усіма інструментами, необхідні для розробки якісних програм.

На основі розглянутих середовищ розробки програмування мовою C++ було створено порівняльну таблицю 1.1.

Параметри	MS Visual Studio	JetBrains Clion	Qt Creator	Eclipse SDK	NetBeans
Редактор користувальницького інтерфейсу	+	-	+	+	+
Вбудований профільувальник	+	-	-	-	-
Вбудований відладчик	+	+	+	+	+

Підтримка C++11	+	+	+	+	+
Встановлене ПЗ на підприємстві, наявність ліцензії	+	-	+	-	-
Опис використання	+	-	+	+	-
Сумісність з колишніми проектами	+	-	+	-	-

Для розробки програми для прогнозування цитування наукових текстів на основі семантичного аналізу вибрали середовище розробки Microsoft Visual Studio. Середовище розробки Microsoft Visual Studio було обрано тому, що він має ряд необхідних для розробки функцій, таких як підтримка профілювання коду мовою програмування C++, налагодження та сумісність з успадкованими проектами.

2.3 Вимоги до технічного забезпечення

Технічна підтримка системи має максимально ефективно використовувати існуючі технічні засоби.

До складу комплексу повинні входити такі технічні засоби:

1) сервер бази даних;

2) персональні комп'ютери (ПК) користувачів.

Мінімальні вимоги до характеристик апаратних компонентів, за яких значення тимчасових параметрів Системи повинні відповідати вимогам, представленим у ТЗ:

для сервера бази даних:

– процесор - 2 x IntelXeon3 ГГц;

– обсяг оперативної пам'яті – 16 ГБ;

– дискова підсистема – 4 x 146 ГБ;

– дисковод компакт-дисків (DVD-ROM);
– мережевий адаптер - 100 Мбіт/с.

для ПК користувача:

– процесор - Intel Pentium 1,5 ГГц;
– обсяг оперативної пам'яті – 256 МБ;
– дискова пам'ять – 40 ГБ;
– мережевий адаптер - 100 Мбіт/с.

2.4 Опис методів

Класифікація — це віднесення певних об’єктів до відомих категорій на основі обраних ознак для визначення подібності чи відмінності між цими об’єктами. Кластеризація — поділ набору об'єктів на кластери (групи). Різниця між цими методами полягає в тому, що під час кластеризації групи (класи) відомі заздалегідь. У випадку кластеризації список груп заздалегідь невідомий, він визначається під час роботи алгоритму. Об’єкти в різних групах повинні бути максимально схожі один на одного і максимально відрізнятися від об’єктів інших груп.

Одними з найпростіших і популярних алгоритмів класифікації є лінійні класифікатори та наївні класифікатори Байєса. Перша ідея полягає в тому, щоб мати кілька площин, які ділять простір на півпростори. Алгоритм погано працює з великою кількістю функцій, і перенавчання в цьому випадку не працює добре.

Байєсовські класифікатори базуються на теоремі Байєса. Варто зазначити, що алгоритм передбачає, що наявність однієї ознаки не буде корелювати з наявністю якоїсь іншої. Іншими словами, функції не залежать одна від одної. Робота алгоритму складається з наступних кроків:

– перетворіть набір даних у частотну таблицю
– створіть таблицю ймовірностей;
– обчислити ймовірності класів за допомогою теореми Байєса.

Алгоритм можна використовувати для сегментації об'єктів на базі.

Різноманітність функцій, тому його можна успішно застосовувати для багатокласової класифікації тексту. Цей метод чудово працює, його швидко та легко виконати. Серед його недоліків слід відзначити неможливість робити прогнози, якщо певна ознака не присутня в навчальних вибірках, і припущення про незалежність ознак, що рідко трапляється в реальності.

Що стосується кластеризації, важливо відразу зазначити, що є алгоритми, які є ієрархічними та плоскими, а також чіткими та нечіткими. Планарний алгоритм просто ділить вибірки на кластери, що не перекриваються. Ієрархічні алгоритми створюють повну вкладену кластерну систему. У випадку різних алгоритмів кожен об’єкт належить лише одному кластеру. Нечіткі алгоритми допускають можливість того, що кожен об'єкт належить до кожного класу з певною ймовірністю. Прикладом алгоритму текстової кластеризації є метод LSA/LSI.

Він базується на принципах факторного аналізу. Використовуючи статистичну інформацію, LSA допомагає впоратися із синонімією (кілька слів можуть мати однакове значення) та омофонією (слово може мати більше одного значення). Ідея алгоритму полягає у виконанні сингулярної декомпозиції (розкладання матриці для приведення її в канонічний вигляд) матриці tf-idf. Цей розподіл допомагає краще фільтрувати шум і знаходити кластери.

Переваги цього підходу включають використання матриць tf-idf, відсутність потреби в навчанні та попередньому налаштуванні з наборів документів і придатність для прихованих залежностей. Недоліками цього методу є повільність алгоритму через важкі обчислення, відсутність імен для отриманих факторів і неможливість обробки пересічних кластерів.

Алгоритми Buckshot і Fractionation є дуже цікавими прикладами кластеризації тексту. У Buckshot береться випадкова вибірка документів, яка потім піддається процесу кластерного пошуку, який досягається шляхом «склеювання» документів, які є найближчими один до одного. Це відбувається до тих пір, поки не буде знайдено вказану кількість центрів. Після цього ідентифіковані центри приписуються документам за принципом присвоєння найближчому центру. Назви кластерів складаються зі слів, які найчастіше

зустрічаються в них. Перевагою цього методу є те, що він працює швидко, не вимагає навчання, використовує матрицю близькості документів, недоліками – низька точність (або зниження швидкості при вдосконаленні алгоритму), необхідність вказувати кількість кластерів, нездатність обробляти кластери, що перекриваються [15].

Одним із найпопулярніших алгоритмів кластеризації є метод k-середніх. Він заснований на ітераційній процедурі для стабілізації центроїдів кластера. Його перевагою є лінійна швидкість роботи та той факт, що метод не вимагає навчання та може накопичувати інформацію для покращення майбутньої точності, якщо це необхідно, недоліком є кількість кластерів, необхідних для введення, і оскільки кластери не можуть перетинатися, що Скажімо, алгоритм зрозумілий

Альтернативою k-середнім є c-середнє [16]. Цей алгоритм дозволяє розділити набір об'єктів на задану кількість нечітких наборів. Його вважають покращеною версією k-середніх, оскільки він дозволяє обчислювати для кожного об’єкта, наскільки він належить до кожного кластера.

Існують інші моделі для нечіткої класифікації: методи Монте-Карло з використанням ланцюгів Маркова (з використанням вибірок із розподілів ймовірностей); MaxMax, яка працює зі структурами даних графів і дозволяє вершині бути в кількох кластерах; WatSet, яка також базується на обробці графів і групуванні. синоніми (вершини). Однак перший алгоритм досить складний, а другий і третій призначені лише для графових структур даних.

Методи нейронних мереж стають все більш поширеними. Вони добре працюють із великою кількістю тексту та є дуже гнучкими, оскільки працюють із різноманітними архітектурами, але мають недолік — потребують перенавчання.

У випадку простих методів (байєсівських класифікаторів, k-середніх) моделі можуть бути не дуже добре «навчені», але їх легко інтерпретувати, що полегшує розуміння виправлення недоліків. Хоча нейронні мережі загалом працюють добре, їх важко інтерпретувати, оскільки вони є «чорним ящиком», який потребує перенавчання.

Крім перерахованих вище методів, існують випадки, коли для конкретних проблем розробляються спеціальні методи.

2.5 Аналіз факторів для прогнозування

Одним із найважливіших факторів, що впливають на прогнозування, є концентрація специфічних термінів, так зване «лексичне навантаження» — чим воно вище, тим важче сприймати текст. Іншим важливим компонентом є довжина речення: довгі речення важче зрозуміти.

Флейш запропонував одну з метрик читабельності. Відповідно до нього оцінка читання визначається середньою довжиною речення (розраховується як кількість слів, поділена на кількість речень) і середньою довжиною слова (вимірюється в складах, яка дорівнює загальній кількості складів у тексті). ділиться на кількість слів). У формулу розрахунку також входять коефіцієнти, обрані дослідниками.

Існує також спрощена формула (спрощена Дженкінсом і Паттерсоном). Враховується кількість однокореневих слів на 100 слів і середня довжина речення в словах. Примітно, що спрощена формула була більш актуальною для розуміння письмового тексту.

Дейл і Челл пропонують дещо інший підхід. Їхній метод використовує словник із 3000 слів, які визначаються як прості. Для використання запропонованого методу вибирається кілька фрагментів тексту по 100 слів, обчислюється середня довжина речення (шляхом ділення кількості слів на кількість речень) і обчислюється відсоток слів, які не увійшли до лексикону. Застосовуючи певні коефіцієнти та виконуючи прості математичні операції, можна виконати аналіз складності тексту, який корелює з розумінням майже так само добре, як модифікована формула Флейша.

Іншу формулу запропонував Р. Ганнінг. За цією формулою складність розраховується як сума середньої довжини речення, помноженої на коефіцієнт 0,4, і відсотка складних слів (з більш ніж двома складами). Згодом Г. Маклафлін

запропонував використовувати квадратний корінь слів, що мають більше двох складів, у текстових фрагментах із тридцяти речень. Цю формулу часто рекомендують використовувати в текстах про охорону здоров’я.

Існують також формули, в яких використовується лише кількість складів у слові (наприклад, формула FORCAST). На жаль, використання таких формул обмежене, і кореляція з труднощами читання не дуже висока. Однак їх можна застосувати до тексту з неповними реченнями.

Цікавим методом оцінки складності є «Оцінка щільності синтаксису Голуба». У методиці велика увага приділяється синтаксичному аналізу речень, особливо наявності складних частин, середній довжині слів у підметовій і підрядній частинах речень, кількості модальних дієслів (метод запропоновано для англійської мови) і форм з «бути» та «мати», кількість прийменників, прислівників місця, часу, герундія та часток. Кожен з цих параметрів множиться на певний коефіцієнт. Оцініть тест, порівнявши підсумкові результати з табличними даними. Звичайно, цей підхід є більш складним, оскільки вимагає аналізу та більш окремих обчислень. Ще одна проблема з цим підходом полягає в тому, що він залежить від мови, якою написаний текст, оскільки мови часто мають різний синтаксис.

Окрім вищезазначених параметрів, багато інших досліджень пропонують такі параметри, як використання активного та пасивного станів, наявність різноманітних вставок у реченнях, зв’язність тексту, вік документа, наявність зображень, схем та графіків. , наявність структури документа тощо.

Варто зазначити, що формула, яка використовується для оцінки розбірливості, дійсно має високий коефіцієнт кореляції зі справжньою розбірливістю, але хороший показник при оцінці формули все одно означає, що текст справді легко читати. По-перше, скорочення довжини слів і речень не завжди покращує сприйняття, а по-друге, існують інші характеристики, які формули зазвичай не враховують, наприклад дизайн і форматування тексту, інтереси та освіченість читача. читач і так далі. Однак наведені вище формули досить добре зарекомендували себе протягом тривалого періоду часу, і доведення

їхнього розумного використання дійсно може дозволити вам проаналізувати складність тексту та допомогти його спростити.

2.6 Висновки до розділу 2

У другому розділі розповідається про технічне завдання. Розповідається про засоби розробки та які необхідні вимоги до технічного забезпечення.

Також розповідається про класи які використовувалися у роботі та методи якими користувалися для реалізації програми.

На практиці реалізація визначених структурних модулів передбачає вплив суб’єктів навчально-виховного процесу, до яких належать інформаційно-освітні ресурси, засоби реалізації навчально-педагогічної взаємодії, інструменти навчально-виховної діяльності, матеріальна база та засоби зв’язку.

Розглядаючи види інформаційно-освітніх ресурсів як об’єктів IEE, слід зазначити, що до них належать бази та бази знань, експертні системи, різноманітні інформаційно-довідково-пошукові системи, електронні бібліотеки та колекції наукових ресурсів, цифрові освітні ресурси, комп’ютерне навчання та контроль. системи.

РОЗДІЛ 3. ЕТАПИ РОЗРОБКИ ТА СТВОРЕННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Опис розроблювального алгоритму

Моделі прогнозування є одним із багатьох інструментів, які використовуються для прогнозування результатів щодо цитування наукових текстів. Існує кілька методів прогнозування, які використовують та надають різний рівень інформації. Від простого до складного, привабливість використання моделей прогнозування полягає в наявності візуального посилання на очікувані результати.

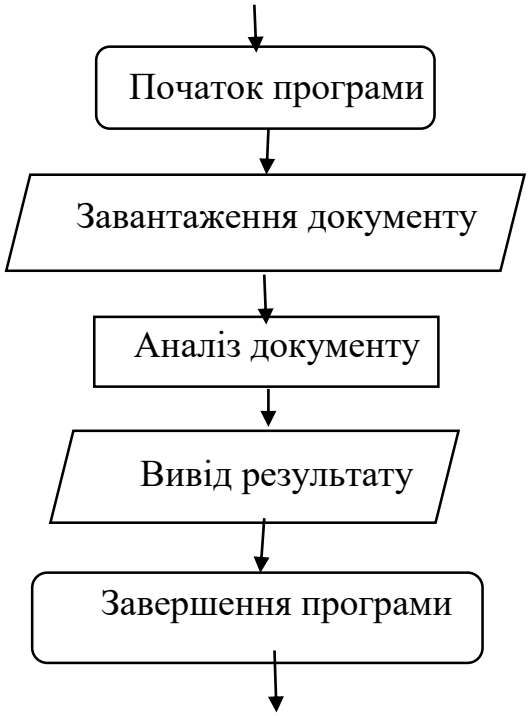
Хоча існує безліч способів прогнозування результатів, існує чотири основних типи моделей або методів, які використовуються для прогнозування дій у майбутньому. Наведено нижче приклади поширених моделей прогнозування:

- Модель часового ряду
- Економетрична модель
- Осудна модель прогнозування
- Метод Delphi.

Часто в галузі технологій використовують методи штучного інтелекту (ШІ) для прогнозування конкретної області зростання. Цей метод прогнозування дає надзвичайно точні результати за допомогою математичних алгоритмів.

Прогнозування релевантних результатів відбувається шляхом групування та інтерпретації даних у поєднанні з інформацією та даними. Більше даних дає більш якісні результати.

Найбільш основним і початковим кроком для аналізу обробки є виділення ключових слів, оскільки без вилучення ключових слів неможливо буде рухатися далі. Оскільки ми знаємо, що в наукових текстах ми отримуємо багато алгоритмів, які можуть допомогти нам у вилученні ключового слова для наших текстових даних. Ми запропоновано наступну схему алгоритму:



3.2 Розробка програми

Етапи створення додатку для прогнозування цитування наукових текстів на основі семантичного аналізу.

1. Генерація ідеї

Багато підприємців-початківців застрягають на першому етапі: створення ідей та мозковий штурм. Це часто відбувається тому, що вони чекають геніальності, щоб виявити ідеальний продукт, який вони повинні продавати. Хоча створення чогось принципово «нового» може бути креативним, багато з найкращих ідей є результатом повторення існуючого продукту.

2. Дослідження

Зважаючи на свою ідею продукту, ви можете відчути бажання перескочити у виробництво, але це може стати помилкою, якщо ви не спершу підтвердите свою ідею.

Перевірка продукту гарантує, що ви створюєте продукт, за який люди будуть платити, і що ви не витрачатимете час, гроші та зусилля на ідею, яка не продається. Є кілька способів підтвердити свої ідеї щодо продуктів, зокрема:

- Обговоріть свою ідею з родиною та друзями
- Надсилання онлайн-опитування, щоб отримати відгук
- Початок краудфандингової кампанії
- Запит на відгуки на таких форумах, як Reddit
- Дослідження ринкового попиту за допомогою Google Trends
- Запуск сторінки «Незабаром», щоб оцінити інтерес за допомогою електронної пошти або попереднього замовлення

3. Планування

Оскільки розробка продукту може швидко стати складною, важливо приділити час для планування, перш ніж почати створювати свій прототип.

Коли ви врешті-решт звертаєтесь до виробників або починаєте шукати матеріали, якщо ви не маєте конкретного уявлення про дизайн вашого продукту та як він буде функціонувати, на наступних кроках легко заблукати.

Найкраще почати планування з намальованого від руки ескізу того, як буде виглядати ваш продукт. Ескіз має бути якомога детальнішим, з етикетками, що пояснюють різні функції та функції.

4. Прототипування

Метою етапу створення прототипу під час розробки продукту є створення готового продукту для використання в якості зразка для масового виробництва.

Малоймовірно, що ви отримаєте готовий продукт за одну спробу — створення прототипу зазвичай включає в себе експерименти з декількома версіями продукту, повільне усунення опцій і вдосконалення, поки ви не будете задоволені кінцевим зразком.

5. Пошук джерел

Коли у вас є прототип продукту, яким ви задоволені, настав час почати збирати матеріали та залучити партнерів, необхідних для виробництва. Це також називають побудовою вашого ланцюга поставок: постачальників, діяльності та ресурсів, необхідних для створення продукту та передачі його в руки клієнта.

6. Калькуляція витрат

Після того, як дослідження, планування, створення прототипів і пошук джерел будуть виконані, ви повинні мати більш чітке уявлення про те, скільки буде коштувати виробництво вашого продукту. Калькуляція витрат — це процес бізнес-аналізу, в якому ви берете всю зібрану на даний момент інформацію та додаєте, яку буде ваша вартість проданих товарів, щоб ви могли визначити роздрібну ціну та валову маржу.

7. Комерціалізація

На цьому етапі у вас є прибутковий і успішний продукт, готовий для світу. Останній крок у цій методології – вивести свій продукт на ринок! На цьому етапі команда розробників передасть кермо маркетингу для запуску продукту.

Почніть зі створення електронної таблиці, у якій кожна додаткова вартість буде розбита як окрема позиція. Це має включати всю сировину, витрати на заводські налаштування, витрати на виробництво та витрати на доставку. Важливо врахувати доставку, імпортні збори та будь-які мита, які вам доведеться сплатити, щоб отримати кінцевий продукт у руки клієнта, оскільки ці збори можуть мати значний вплив на ваші ціни на вартість, залежно від того, де ви виробляєте продукт.

Структура даних – це спосіб збору та організації даних таким чином, щоб ми могли ефективно виконувати операції з цими даними. Структури даних – це відтворення елементів даних у термінах певного зв’язку для кращої організації та зберігання.

Структура даних —спосіб організації та зберігання даних, щоб ефективно виконувати операції. Доступ, вставка, видалення, пошук і сортування даних є одними з основних операцій, які можна виконувати за допомогою структур даних. Не всі структури даних можуть ефективно виконувати ці операції, це призвело до розробки різних структур даних. Скажімо, вам потрібно знайти конкретну книгу в неорганізованій бібліотеці, це завдання займе величезну кількість часу. Подібно до того, як бібліотека організовує свої книги, нам потрібно організувати наші дані так, щоб операції можна було виконувати ефективно.

Для створення структури даних використовується попередньо визначений тип даних, наприклад Integer, Stings, Boolean. Використання типу даних залежить від вимог користувача та типу даних, які вони хочуть зберігати. Тепер давайте зануримося в структури даних, які використовуються в нашому повсякденному програмуванні, а також розглянемо підтримку структур даних для різних мов програмування.

Java: Є кілька класів для масивів, але найвідомішим є клас ArrayList.

C++: - Вектори використовуються для представлення динамічних масивів у C++. Його можна реалізувати за допомогою std::vector.

Python: Python має новий тип даних, відомий як список, як і Boolean і Integers. Тип даних списку можна використовувати для динамічного визначення масивів.

Однозв’язний список

Пов’язаний список — це набір вузлів, які з’єднані за допомогою посилань. Пов’язаний список містить вузол, який зберігає елементи даних і адресу наступного вузла. Перший вузол зазвичай називають головним вузлом, а останній — хвостовим. Покажчик головного вузла вказує на наступний вузол, а вказівник хвостового вузла вказує на Null.

Динаміка цієї структури даних полегшує додавання або видалення вузлів. Щоб додати/видалити вузол, вам просто потрібно відстежити попередній вузол і вузол після нього і відповідно налаштувати покажчики.

Двозв’язаний список

Двозв’язаний список мало чим відрізняється від однозв’язаного списку, єдине, що їх відрізняє, — це вказівник на попередній вузол. Зображення нижче може дати вам коротке уявлення про те, як повинна виглядати структура.

У випадку двозв’язаного списку попередній вказівник головного вузла вказує на Null, а наступний покажчик хвостового — на Null. Попередній покажчик полегшує перехід в будь-якому напрямку. Отже, додавання та видалення вузлів стає дуже простим, все, що вам потрібно зробити, це відстежувати попередній і наступний вузли та відповідно налаштовувати вказівник.

Круговий зв’язаний список

Круговий зв’язаний список – це зв’язаний список, де всі вузли з’єднані, щоб утворити коло. Головного чи хвостового вузла немає. Перевага такого типу зв’язаного списку полягає в тому, що будь-який вузол може бути використаний як відправна точка.

Порівняння між масивами та зв'язаним списком

Список — це відмінний від масиву тип структури даних. Замість того, щоб зберігати дані в шматках пам'яті, масив потребує безперервного простору пам'яті. Якщо додати елемент до масиву, це може змінити розташування пам’яті всього масиву, але дані залишаться неушкодженими. Масив дає вам номер індексу, отже, ви можете отримати до нього прямий або послідовний доступ. Тоді як для доступу до будь-якого члена даних потрібно пройти через весь список.

Для початку роботи програми для цитування наукових текстів на основі семантичного аналізу підключимо наступні бібліотеки

```
import re
import os
import time
import requests
from tqdm import tqdm
from io import StringIO
```

Далі створюємо функції парсингу текстів

```
def download_file(n, url):
    with open(f'files/{n}.pdf', "wb") as f:
        headers = {'User-
Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20
100101 Firefox/98.0'}
        pdf = requests.get(url, headers=headers)
        f.write(pdf.content)

def read_pdf(n):
    rm = PDFResourceManager()
```

```
        f = StringIO()
        converter = TextConverter(rm, f)
        pinter = PDFPageInterpreter(rm, converter)
        with open(f'files/{n}.pdf', 'rb') as pdf:
            for page in PDFPage.get_pages(pdf, caching=True, check_e
xtractable=True):
                pinter.process_page(page)
            text = f.getvalue()
        return text

    def check_references(text, n):
        try:
            sources = re.findall(r'\[\d+]', text)
            if len(sources) > 5:
                return True
        except:
            pass
        return False

    def save_text(text, n):
        text = re.sub(r'(\\[ux][\w\d]+):?', ' ', text)
        text = re.sub(r'\s+', ' ', text)
        with open(f'/content/drive/MyDrive/files/{n}.txt', 'w', enco
ding='utf-8') as f:
            f.write(text)

    def write_log(n):
        with open('/content/drive/MyDrive/files/log.txt', 'a') as f:
            f.write(f'{n}, ')

    def check_correct_parcing(text):
        try:
            re.search(r'[\w\d.,;]{60}', text).group(0)
            return False
        except:
            return True
```

```
    def main():
        with open('pdf_links.txt', 'r') as f:
            pdf_links = f.readlines()
        for i in tqdm(range(847, 1000)):
            link = pdf_links[i]
            try:
                download_file(i+9000, link)
                if os.stat(f'files/{i+9000}.pdf').st_size/1024 < 2:
                    os.remove(f'files/{i+9000}.pdf')
                    break
                text = read_pdf(i+9000).lower()
            except:
                write_log(i)
                os.remove(f'files/{i+9000}.pdf')
                time.sleep(1)
                continue
            if check_correct_parcing(text):
                if check_references(text, i+9000):
                    save_text(text, i+9000)
            os.remove(f'files/{i+9000}.pdf')
            time.sleep(1)
```

Далі були описані декілька методів в класі

```
    # Функція видалення блоку посилань (для запобігання подвійного р
ахунку)

    def del_ref_block(text):
        references = re.findall('references:*\s*\[*1[.\]]*', text)
        if len(references) == 1:
            text = re.sub('references:*\s*\[*1[.\]]*.*', '', text)
        elif len(references) > 1:
            try:
                last_references = re.findall('references:*\s*\[*1\.*
\]*.{5}', text)[-1]
```

```
                last_references = re.sub('\.', '\.', last_references
)
                last_references = re.sub('\[', '\[', last_references
)
                last_references = re.sub('\]', '\]', last_references
)
                del_pattern = last_references + '.*'
                text = re.sub(del_pattern, '', text)
            except:
                last_references = re.findall('.{5}references:*\s*\[*
1[.\]]*', text)[-1]
                del_pattern = last_references[:16] + '.*'
                text = re.sub(del_pattern, '', text)
        else:
            ref_list = re.findall('\[\d+[,\-]*\d*\]', text)
            ref_str = str(ref_list)
            if "'[1]', '[2]', '[3]', '[4]'" in ref_str:
                try:
                    last_one = re.findall('\[1\].{5}', text)[-1]
                    last_one = re.sub('\]', '\]', last_one)
                    last_one = re.sub('\[', '\[', last_one)
                    del_pattern = last_one + '.*'
                    text = re.sub(del_pattern, '', text)
                except:
                    last_one = re.findall('.{5}\[1\]', text)[-1]
                    last_one = re.sub('\]', '\]', last_one)
                    last_one = re.sub('\[', '\[', last_one)

                    del_pattern = last_one + '.*'
                    text = re.sub(del_pattern, '', text)
        return text
```

3.3 Опис функціоналу

Переходячи за посиланням (рис. 3.1)та відкривши файл site.exe

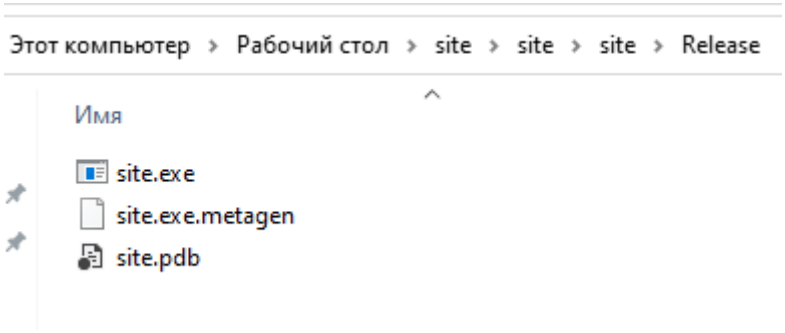


Рис. 3.1 – Розташування програми

Перед нами відкриється розроблена програма та її код на ресурсі Google Collab, що прогнозує цитування наукових текстів на основі семантичного аналізу (рис. 3.2)

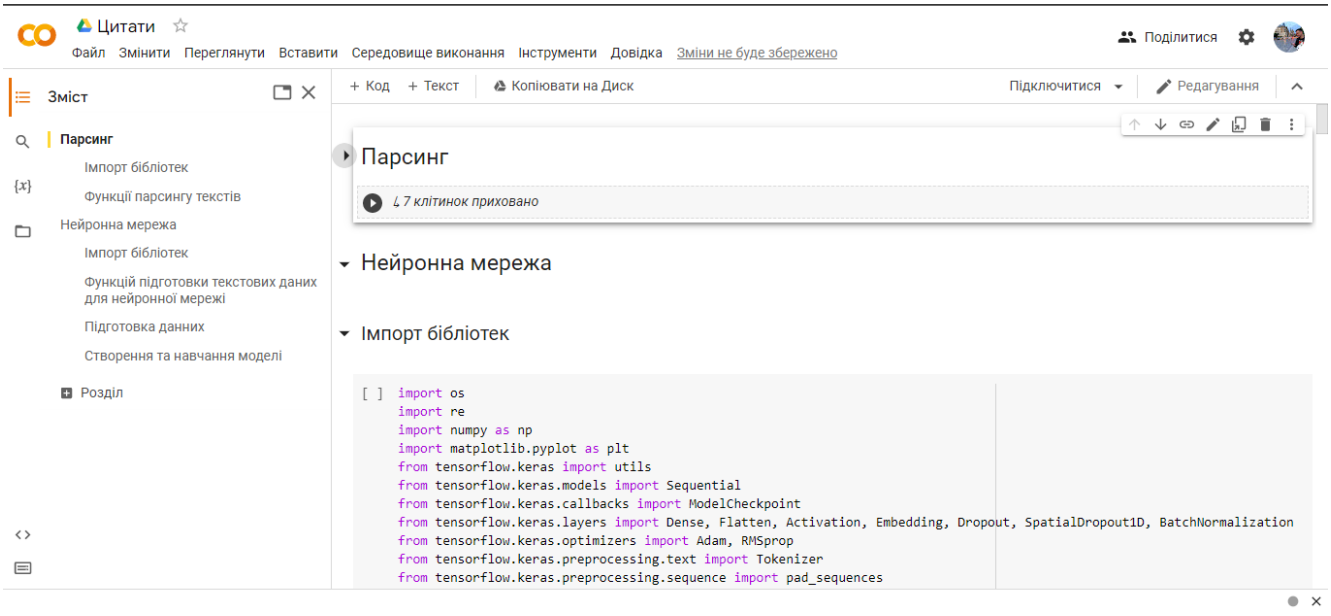


Рис. 3.2 – Стартове вікно програми

На рисунку 3.2 ми можемо спостерігати короткий зміст програми зліва та безпосередньо його частину, на якій відбувається імпорт необхідних бібліотек.

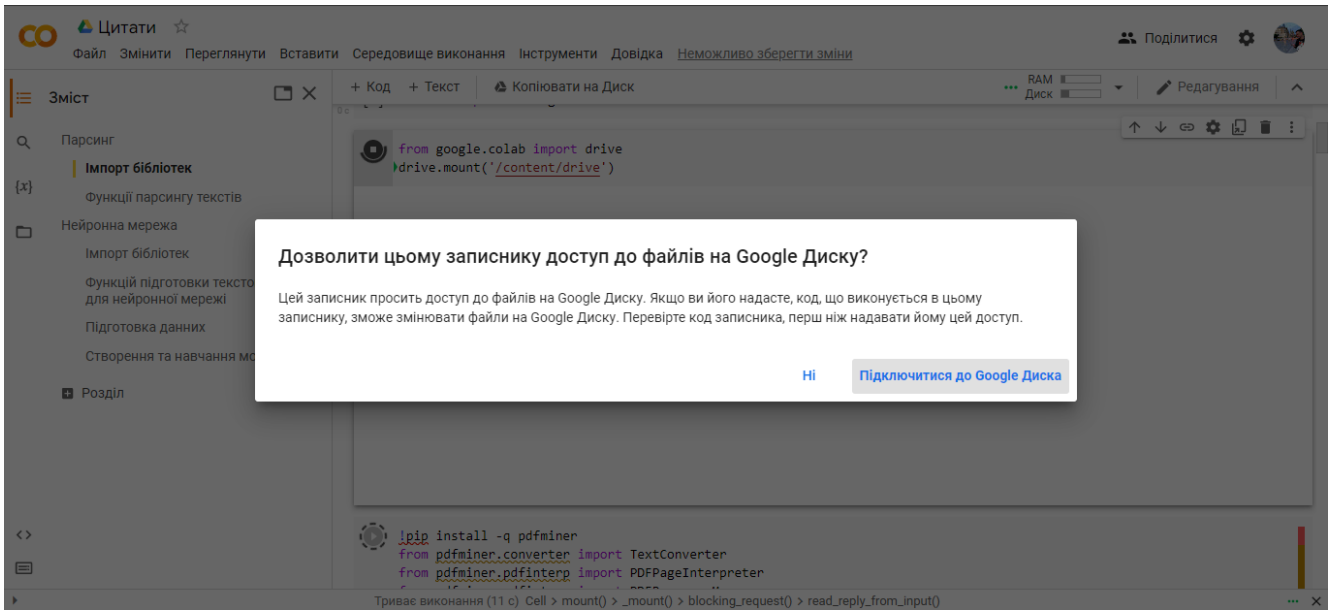


Рис. 3.3 – Запуск коду

На рисунку 3.3 ми можемо бачити запит щодо доступу до файлів на Google Drive. А саме нам необхідний файл pdf_links.txt, в якому зберігаються посилання на необхідні нам статті з ресурсу arxiv.org

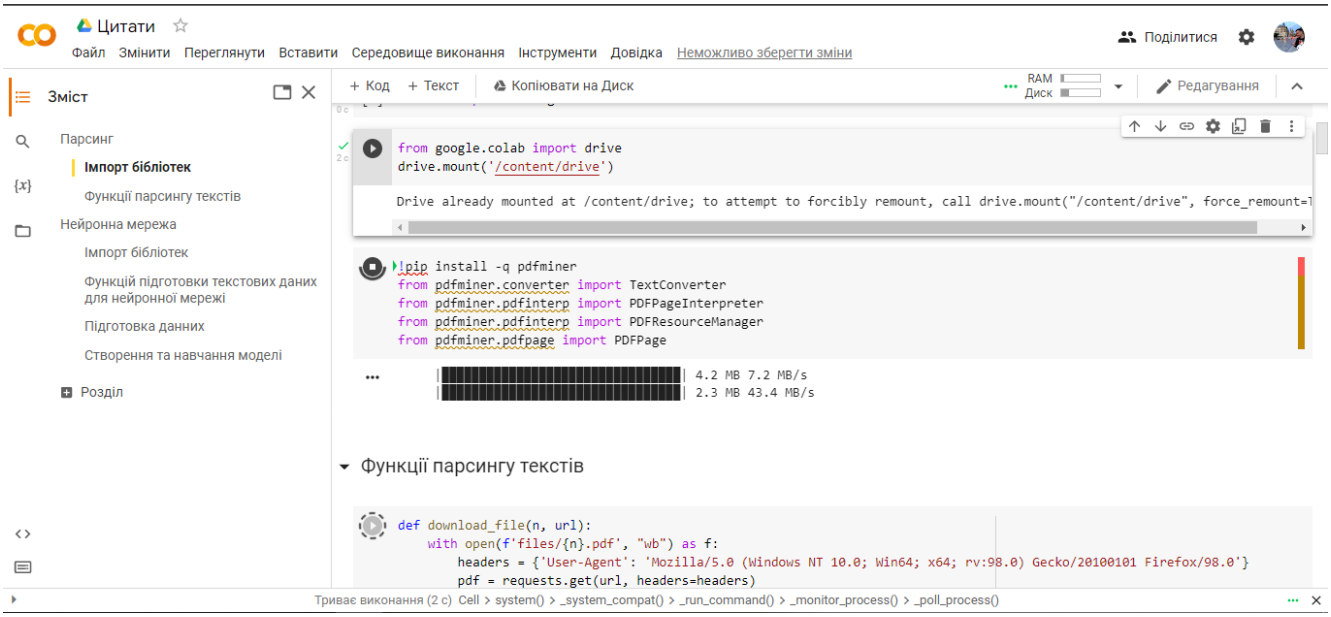


Рис. 3.4 – Виконання коду після надання доступу

Виконання коду займає доволі довгий період. Однією з ключових причин цьому є те, що парсинг відбувається не менше однієї доби, в залежності від ресурсів обчислювальної системи.

Рис. 3.5 – Результат виконання

На цьому рисунку показано результат у вигляді таблиці, на якій відображено фактичні дані, та результат нейронної мережі. Таким чином ми можемо проаналізувати помилку при навчанні.

3.4 Тестування

Під час тестування програми що прогнозує цитування наукових текстів на основі семантичного аналізу, усі виникаючі помилки, одразу ж виправлялися.

Загалом, виникали помилки при синхронізації процесу на пристрої, де буде виконано вхід в інший обліковий запис. Тобто, необхідно буде виконати створення директорії в Google Drive з коректними назвами, аби при виконанні коду не виникло помилок під час парсингу та збереженні опрацьованих данних.

3.5 Висновки до розділу 3

В цьому розділі розповідається про етапи розробки та створення програмного додатку з прогнозування цитування наукових текстів на основі семантичного аналізу. Описується розроблений алгоритм та за якими етапами відбувалося написання програмного коду. Розповідається про розробку структури даних та описується функціонал програми. Також розповідається як відбувалося тестування, та які помилки виникали під час тестування.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ОБЛАСТІ РОБОТИ ТА РЕЗУЛЬТАТИ ДАННОГО ДОСЛІДЖЕННЯ

4.1 Проведення наукового аналізу дослідницької роботи

Додаток для прогнозування цитування наукових текстів на основні семантичного аналізу було створено для підвищення ефективності роботи викладачів.

Ефективність впровадження системи полягає в автоматизації процесу прогнозування цитування наукових текстів.

Зниження витрат на оплату праці зменшить і фінансові витрати, що призведе до загального зростання продуктивності та економії.

Основною метою даного розділу є визначення вартості науково-дослідних робіт, витрат на визначення економічного ефекту від основних і відповідних результатів, отриманих при розв’язанні технічних задач у підсумковій ідентифікаційній роботі для автоматизації процесу прогнозування цитування наукових текстів. При оцінці ефективності прийнятих науково-технічних рішень повинні бути враховані всі необхідні витрати і витрати, що вимагає проведення необхідних розрахунків за певним рішенням.

В ході дослідження області роботи було порівняно бібліотеки мови програмування Python, для машинного навчання та створення нейронних мереж.

Таблиця 4 – порівняння бібліотек Python

#	**Название**	**Язык**	**ОС**	**FC NN**	**CNN**	**AE**	**RBM**
1	DeepLearnToolbox	Matlab	Windows, Linux	+	+	+	+
2	Theano	Python	Windows, Linux, Mac	+	+	+	+

3	Pylearn2	Python	Linux, Vagrant	+	+	+	+
4	Deepnet	Python	Linux	+	+	+	+
5	Deepmat	Matlab	?	+	+	+	+
6	Torch	Lua, C	Linux, Mac OS X, iOS, Android	+	+	+	+
7	Darch	R	Windows, Linux	+	—	+	+
8	Caffe	C++, Python, Matlab	Linux, OS X	+	+	—	—
9	nnForge	C++	Linux	+	+	—	—
10	CXXNET	C++	Linux	+	+	—	—
11	Cuda-convnet	C++	Linux, Windows	+	+	—	—
12	Cuda CNN	Matlab	Linux, Windows	+	+	—	—

Експерименти проводилися на повнозв'язковій та згортковій нейронних мережах наступної структури:

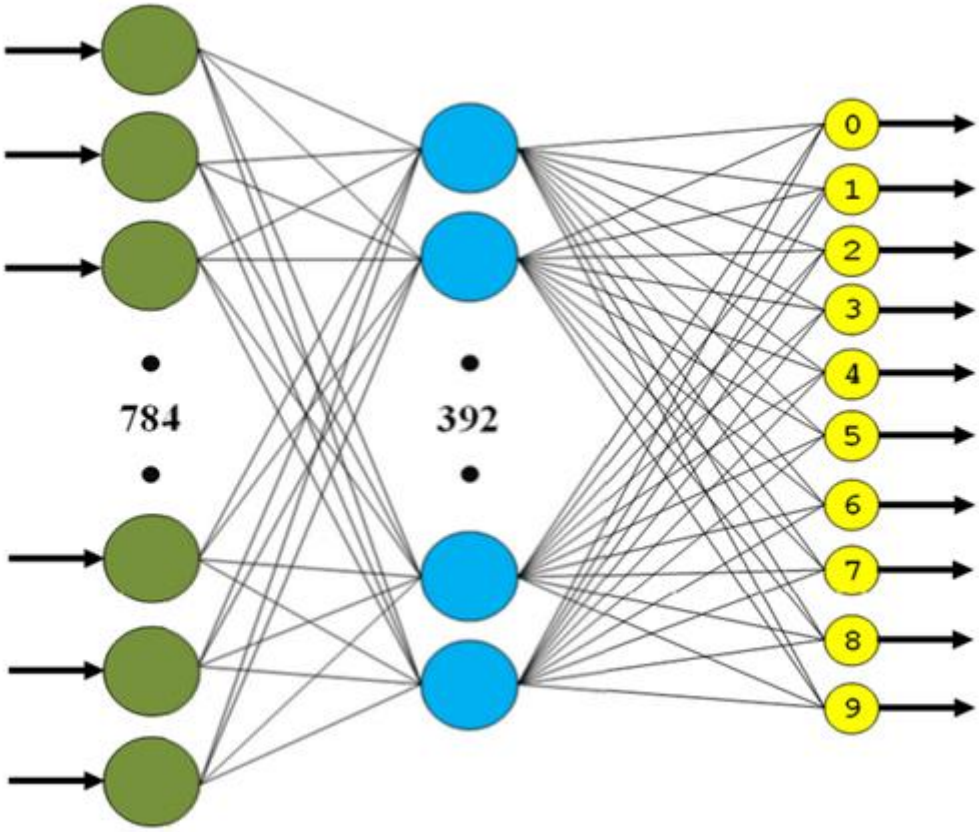


Рисунок 5 - Структура тришарової повної мережі

- 1st layer – FC (in: 784, out: 392, activation: tanh).
- 2d layer – FC (in: 392, out: 196, activation: tanh).
- 3d layer – FC (in: 196, out: 10, activation: softmax).

Наступним об’єктом була згорткова нейронна мережа:

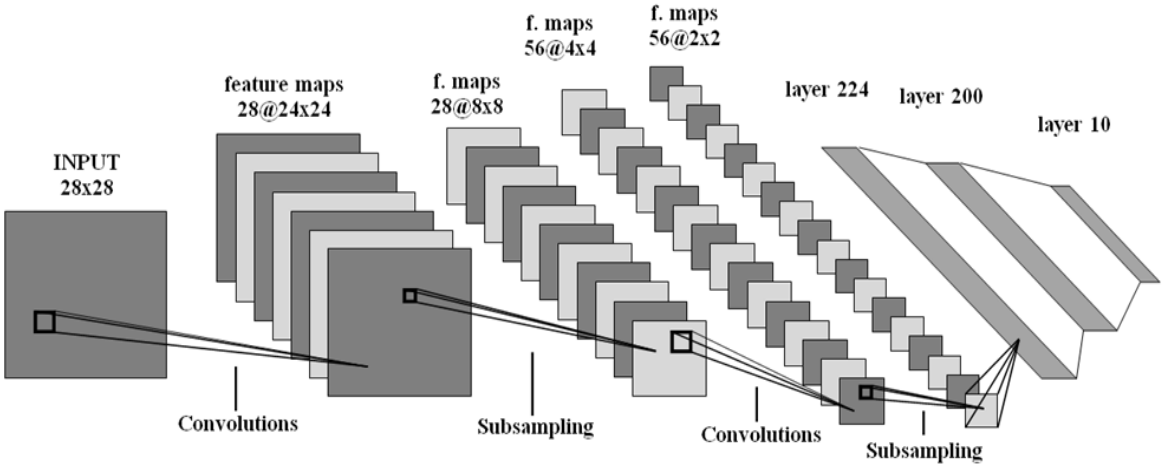
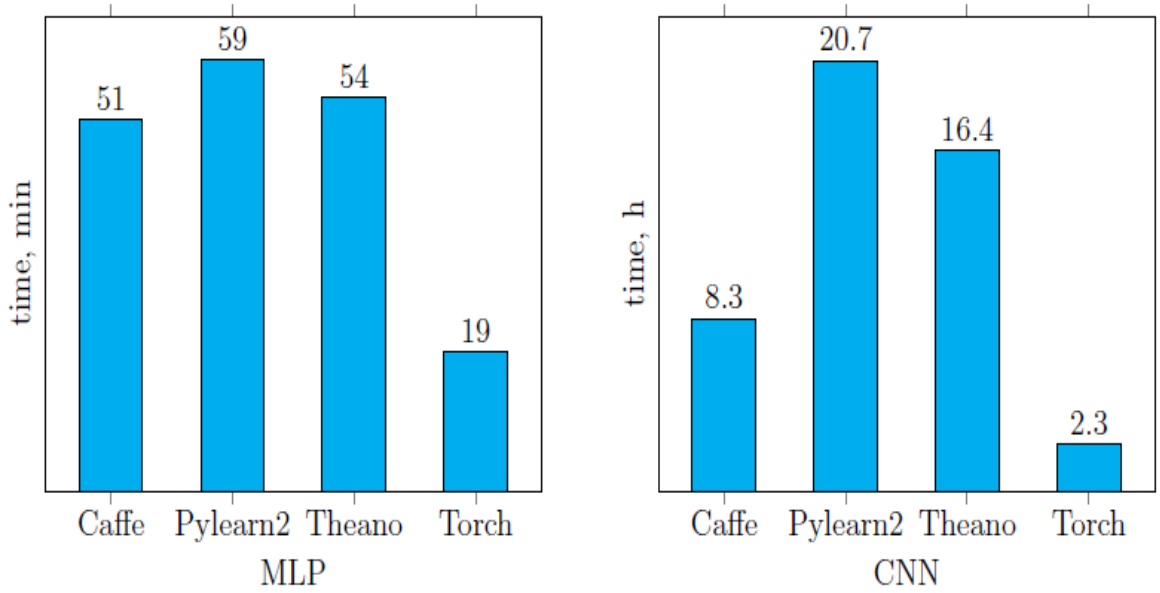


Рисунок 5.1 - Структура згорткової нейронної мережі

- 1st layer – convolution (in filters: 1, out filters: 28, size: 5x5, stride: 1x1).
- 2d layer – max-pooling (size: 3x3, stride: 3x3).

- 3d layer – convolution (in filters: 28, out filters: 56, size: 5x5, stride 1x1).
- 4th layer – max-pooling (size: 2x2, stride: 2x2).
- 5th layer – FC (in: 224, out: 200, activation: tanh).
- 6th layer – FC (in: 200, out: 10, activation: softmax).

Час навчання нейронних мереж, описаних раніше за допомогою чотирьох розглянутих бібліотек, наведено нижче. Легко помітити, що Pylearn2 показує найгіршу продуктивність (як на CPU, так і на GPU) порівняно з іншими бібліотеками. Що ж до інших, час навчання залежить від структури мережі. Найкращий результат серед реалізацій мереж, запущених на CPU, показала бібліотека Torch (причому на CNN вона випередила саму себе, запущену на GPU). Серед GPU-реалізацій найкращий результат (на обох мережах) показала бібліотека Caffe. Загалом від використання Caffe залишилися лише позитивні враження.



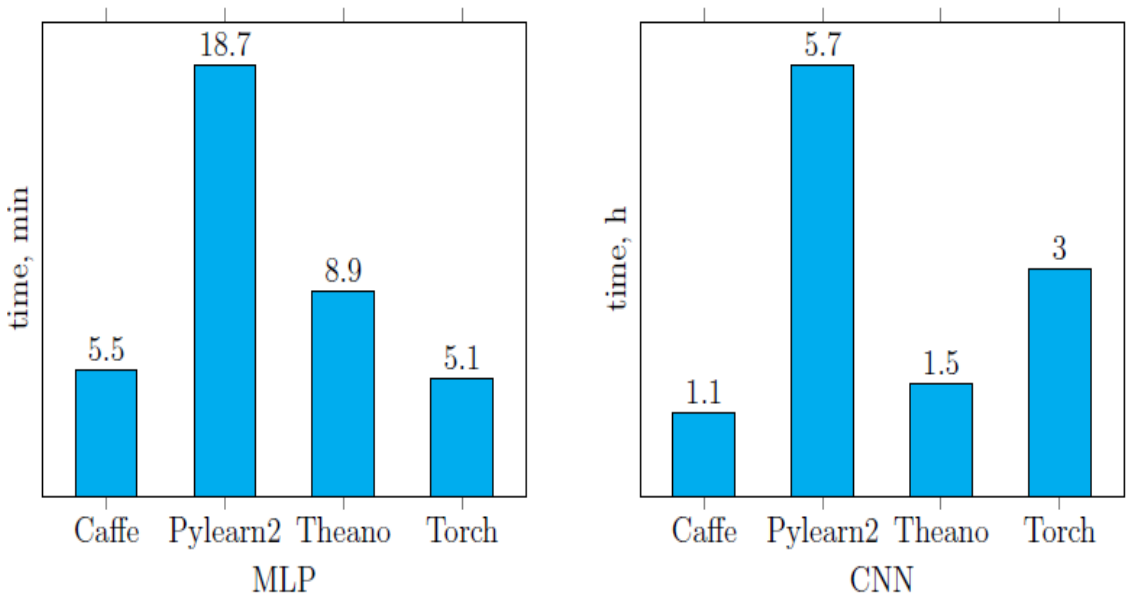


Рисунок 5.2 - Час навчання мереж MLP та CNN

Порівняння вибраних бібліотек

На підставі проведеного дослідження функціоналу бібліотек, а також аналізу продуктивності на прикладі задачі класифікації рукописних цифр дано оцінку кожної з них за шкалою від 1 до 3 за такими критеріями:

- Швидкість навчання відбиває час навчання нейромережевих моделей, розглянутих на етапі проведення експериментів.
- Швидкість класифікації відбиває час класифікації одного зображення.
- Зручність використання — критерій, що дозволяє оцінити час, витрачений вивчення бібліотеки.
- Гнучкість налаштування зв'язків між шарами, налаштування параметрів методів, а також наявність різних способів обробки даних.
- Обсяг функціоналу - наявність реалізації типових методів глибокого навчання (повністю пов'язаних мереж, згорткових нейромереж, автокодувальників, обмежених машин Больцмана, різних методів оптимізації та функцій помилки).
- Наявність та зручність використання документації та навчальних матеріалів

Загальна оцінка була предствалена на таблиці 4.1 - розставимо місця кожній бібліотеці від першого до третього.

Таблиця 4.1

Швидкість навчания	Швидкість класифікац	Зручність	Гнучкість	Функціонал	Документація	Сума
Caffe	1	2	1	3	3	**12**
Pylearn2	3	3	2	3	1	**15**
Torch	2	1	2	2	2	**10**
Theano	2	2	3	1	2	**12**

Комісійні витрати орієнтуються на цінову політику мережі, як показано в таблиці 4.2.

Таблиця 4.2 – Вартість програмного та апаратного забезпечення

Назва	Кількість	Ціна
Apache HTTP Server	1	0
СУБД MySql + phpMyAdmin	1	0

Таблиця 4.2 – Матеріали

Назва	Кількість	Ціна

USB-флешка KINGSTON DataTraveler SE9 16Gb (DTSE9H)	1	450
Диск DVD-R 4.7Gb SlimCase VS	1	25

Вартість електроенергії базується на тривалості періоду розробки програмного забезпечення, кількості кіловат-годин, витрачених на розробку програмного забезпечення, і вартості електроенергії за 1 кіловат-годину. Митний збір для юридичних осіб становить 4,68 грн. за кіловат годину. Витрати відображені в таблиці 4.3.

Таблиця 4.3 - Плата за електроенергію

Елементи системи	Встановлена потужність, кВт	Вартість 1 кіловат за годину (грн.)	робочі години	загальне споживання
Aspire E5-532-C5SZ Сірий	0,057	4,68	398	106,18

Амортизація обладнання оприбутковується в процесі його використання, тобто під час впровадження та створення ПЗ.

Грошова оцінка амортизації — це витрати на амортизацію, що входять до поточної собівартості.

4.2 Розрахунок витрат на розробку проекту

Основна мета розробки техніко-економічного обґрунтування (ТЕО) – дати фінансову оцінку передбачуваних витрат та одержуваного корисного результату, а також оцінити прибутковість проекту і, в кінцевому підсумку,

економічну доцільність його розробки та впровадження додатку для прогнозування цитування наукових текстів на основі семантичного аналізу..

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості [4].

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF, \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Розмір програмного коду було підраховано за допомогою інструменту вбудованного у Visual Studio 2019 Community – Code Metrics Results.

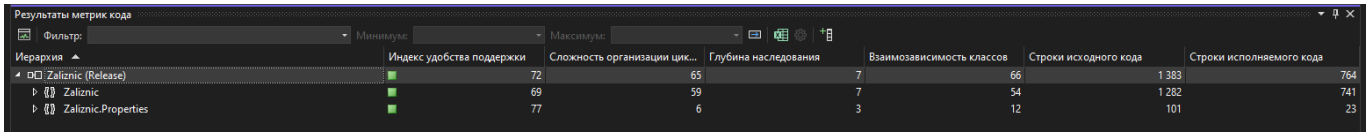


Рисунок 5.1 – Підрахунок розміру програмного коду

Отже розмір програмного коду становить 764 рядки:

$$E = 2{,}4 \cdot 0{,}764^{1{,}05} \cdot 1 = 1{,}81$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1,81 людино-місяці.

Нижче наведені розрахунки вартості розробки «Автоматизована система оцінки схожості програм». Основними статтями витрат прийняті:

– основна заробітна плата;
– відрахування на соціальні потреби;
– накладні витрати;
– витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	
1	інженер-програміст	16450 [2]	1	1,81	33000

Оклад було взято на основі аналізу середньої заробітної плати по ринку на 4 квартал 2022 року.

Описаний в проекті програмний продукт був розроблений одним програмістом в період з 27.08.22 до 22.10.22, що складає 40 днів або приблизно 7 робочих тижнів. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 103 грн/год. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \times N_{\text{тиж}} \times N_{\text{год}}, \quad (5.2)$$

де $N_{\text{чол}}$ – кількість виконавців, чол;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 7 \cdot 40 = 280 \text{ чол/год.}$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5.3)$$

де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, обумовлений від стажу

роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) - ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та

становить:

- для працюючих до 2 років- 0,75;
- від 2 до 3 років 1,0;
- від 3 до 5 років - 1,1-1,2;
- від 5 до 7 років - 1,3-1,4;
- понад 7 років - 1,5-1,6.

В даному випадку KKB приймається 0,75. ОЗП складає:

$$\text{ОЗП} = 280 \cdot 103 \cdot 0{,}75 = 21630 \text{ грн.}$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати (22% [4]):

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%} \tag{5.4}$$

$$C_{\text{соц}} = \frac{21630 \cdot 22\%}{100\%} = 4758\text{грн.}$$

Отримані результати за (5.3) та (5.4) підсумовуються. Вони складають 27450грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$\text{С}_{\text{накл}} = \frac{(\text{ОЗП} + \text{С}_{\text{соц}}) \cdot 40\%}{100\%}; \tag{5.5}$$

$$\text{С}_{\text{накл}} = \frac{(21630 + 4758) \cdot 40\%}{100\%} = 10555 \text{ грн.}$$

Протягом усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються

протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

– вартість витратних матеріалів;
– витрати на ремонт;
– заробітна плата ремонтника;
– оренда приміщення;
– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;
– амортизаційні витрати на персональний комп'ютер і програмне забезпечення;
– в

які визначаються за

ормулою:

р

$$C_{ел} = P \cdot B \cdot T_{розр}, \quad (5.6)$$

– де Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,53 кВт/год;
– B – вартість 1 кВт/годин для непобутових споживачів,
– искладає 1,68 грн [3];
– Tрозр – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}53 \cdot 1{,}68 \cdot 280 = 249{,}3 \text{ грн.}$$

Витрати на витратні матеріали ($C_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп’ютеру. Вартість робочої станції приймається 18 000 грн. [6], термін експлуатації – 5 років. Отже, можна визначити ці витрати за період створення програмного засобу:

$$Свм = Вком \cdot NДNексп \cdot 365 \cdot 10\% 100\%,$$

де $B_{КОМ}$ – вартість персонального комп’ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп’ютеру.

Витрати на витратні матеріали визначаються так:

Витрати на витратні матеріали визначаються так:

$$C_{\text{вм}} = 25000 \cdot \frac{40}{5 \cdot 365} \cdot \frac{10}{100} = 54{,}8 \text{ грн.}$$

Заробітна плата ремонтника ($C_{\text{рем}}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 9000 грн. Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$Срем = СремNKOM \cdot Тміс,$$

де $C_{\text{рем}}^{'}$ – середньомісячна заробітна плата;

N_{KOM} – кількість комп'ютерів на одного ремонтника.

$T_{мес}$ – час розробки програмного продукту, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{90'00}{50} \cdot 1{,}61 = \ 289{,}8 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби (C_{KOM}) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{KOM} = C_{BM} = 54{,}8 \text{ грн.} \qquad (5.9)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = 25000 \cdot \frac{1{,}81}{3 \cdot 12} = 1256{,}9$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння для Windows 5 років та Visual Studio за 2 рік то амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася

операційна система Windows 10, для написання програмного забезпечення - програмне середовище Visual Studio 2019 Community.

$$АКП_w = 13800 \cdot \frac{1{,}81}{5 \cdot 12} = 416{,}3$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2. Додаткові витрати ($C_{дод}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера- програміст, тобто 7875 гривень на місяць.

Оренду приміщень для однієї людини приймемо рівною 3200 гривень на місяць [5]. Тобто за весь період розробки – 4 800 грн. Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{експ} = C_{ел} + C_{ВМ} + C_{рем} + АКП + АПО + C_{ор} + C_{дод}; \quad (5.11)$$

$$C_{експ} = 259{,}2 + 40 + 325{,}8 + 905 + 416{,}3 + 4100 + 7875 = 13921{,}30 \text{ грн}$$

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	13800	http://mtsoft.kiev.ua/product/windows-10-professional	416,3
Visual Studio 2019 Community	0	https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes	0
Всього:	13800		416,3

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	249

Вартість витратних матеріалів	59
Витрати на ремонт	290
Амортизація персонального комп’ютера	1257
Амортизація програмного забезпечення	416
Оренда приміщення	4800
Додаткові витрати	7635
Всього	14706

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (5.12)$$

$$C_{\text{розробки}} = 22500 + 4950 + 9607 + 14051 = 51108 \text{ грн.}$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	
Відрахування на соціальні потреби	
Накладні витрати	
Експлуатаційні витрати	14706
Всього	

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 50978 грн.

4.3 Розрахунок ефективності використання розробленої програми

Час викладачів витрачається на невикористання додатків, розроблених у рамках даної роботи.

Також необхідно враховувати значення середньої вартості робочого часу працівників.

Оскільки програмне забезпечення, що розробляється, покликане полегшити роботу працівників, які також є клієнтами, окремо додаткових витрат на оплату праці не буде.

Опрацювання документів, складання тестів $t_{n1} = 3г$, за допомогою веб-додатку робота зі складання тестів буде меншою, $t_{n2} = 1г$.

Економія часу становитиме:

$$t_{n1} - t_{n2} = 3г - 1г = 2 \text{ години на день.}$$

За один місяць вчитель зекономив час:

$$T_{рм1} = 26 * 2год = 52г.$$

Це означає, що за рахунок економії часу можна збільшити години і вигода в грошовому вираженні складе: $P_{м1} = 168.26 * 52г = 8749,52$ грн. на місяць. Далі розраховуємо термін окупності системи: поточний $T_{ok} = 105141.84 / 8749.52 = 12.01$ місяць.

4.4 Висновок до четвертого розділу

У цьому розділі аналізується розроблене програмне забезпечення з точки зору актуальності та економічної доцільності його впровадження, а не використовуваних засобів. По-перше, визначається складність і час роботи по створенню програмного додатку.

Наступним кроком є розрахунок вартості впровадження та впровадження розробленого веб-додатку. Заключним етапом є розрахунок економічного ефекту від впровадження інформаційної системи.

Проведені розрахунки та метрики дозволяють говорити про доцільність та економічну вигоду з точки зору витрат часу впровадження інформаційної тестової системи.

ВИСНОВКИ

При виконанні магістерської роботи було проаналізовано застосування методу семантичного аналізу в пошуковій системі. З розглянутих методів обрано тематичне моделювання, яке є основою для прогнозування цитування наукових текстів на основі семантичного аналізу. Ці вдосконалені методи застосовуються до прототипу інтелектуальної пошукової системи.

В рамках магістерської роботи були виконані наступні завдання:

- Проаналізовано предметні області;
- розглянути можливість застосування методів семантичного аналізу для прогнозування;
- розглянули методи попереднього опрацювання текстів;
- розроблено математичну модель пошуку наукових текстів;
- оптимізація роботи системи шляхом вибору оптимальних параметрів, кількості тем і застосованих додаткових регуляризаторів для підвищення якості обраних тем;
- проаналізовано роботу програми;
- Порівняно методи прогнозування цитування наукових текстів на основі семантичного аналізу.
- Розроблено прототип інтелектуальної системи.

Прототипування системи з використанням мови програмування C++ та нейронної мережі з використанням WindowsForms. Розроблена система вибирає матеріали з бази даних на основі вхідних тем і документів, а також може прогнозувати цитування текстів на основі семантичного аналізу.

СПИСОК ЛІТЕРАТУРИ

1. Анісімов А.В. Комп'ютерна лінгвістика для всіх: Міфи. Алгоритми. Мова Київ: Наук. думка, 1988. - 223 с.

2. Білоногов Г.Г. Комп'ютерна лінгвістика та перспективні інформаційні технології М.: Російський світ. 2004 - 248 с. Їл. ресурс. Режим доступу: http://www.twirpx.com/file/134393/

3. Волошин В.Г. Комп'ютерна лінгвістика: Навчальний посібник. - Суми: Університетська книга, 2004. -382 с.

4. Марчук Ю.М. Комп'ютерна лінгвістика М: Вид-во Схід-Захід, 2007 р., - 317 с Ел. ресурс. Режим доступу: http://www.twirpx.com/file/398578/

5. Партіко З.В. Прикладна та комп'ютерна лінгвістика, Львів, «Афіша», 2008, - 221 с.

6. Сайт "Автоматична обробка текстів" Ел. ресурс. Режим доступу: http://aot.ru

7. Сайт проекту Link Grammar Ел. ресурс. Режим доступу http://www.link.cs.cmu.edu/link/

8. Сайт проекту WordNet Ел. ресурс. Режим доступу: http://wordnet.princeton.edu/

9. Clark A. Країна комп'ютерних лінгвістичних та природничих мовних процесів /Clark A., Fox C., Lappin S.// Blackwell Publishing, 2010, - 775р. Ел. ресурс. Режим доступу: stp.lingfil.uu.se/~santinim/sais/ClarkEtAl2010 _HandbookNLP.pdf

10. Burger J Дослідження, Tasks and Program Structures до mapmap Research in Question & Answering (Q&A) /John Burger et al.// ел. ресурс. Режим доступу ^ http://www.inf.ed.ac.uk/teaching/courses/tts/papers/qa_roadmap.pdf

11. Jurafsky D. Speech and Language Processing: In Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2nd edition, / Jurafsky D., Martin J. // 2009 Ел. ресурс. Режим доступу: http://www.cse.iitk.ac.in/users/mohit/Speech-and-Language-Processing.pdf

12. Mitkov R. The Oxford handbook of computational linguistics / Oxford University Press, 2003 -786 p. Їл. ресурс. Режим доступу: http://www.google.com.ua/books?hl=uk&lr=&id=yl6AnaKtVAkC&oi=fnd&pg=PP2&dq=5.%09The+Oxford+handbook+of+computational+linguistics

13. Nirenburg S. Ontological semantics / Nirenburg S., Raskin V. // MIT Press, 2004, - 420 p. Їл. ресурс. Режим доступу: http://books.google.com.ua/books/about/Ontological_semantics.html?id=OPek3LpMIigC&redir_esc=y

14. Uwe Reinke State of the Art in Translation Memory Technology Ел. ресурс. Режим доступу http://www.t-c3.org/index.php/t-c3/article/view/25

15. Большакова Е. Автоматическая обработка текстов на естественном языке и анализ данных: учеб. пособие / Е.И. Большакова, К.В. Воронцов, Н.Э. Ефремова, Э. С. Клышинский – М.: Изд-во НИУ ВШЭ, 2017. – 269 с.

16. Кравченко Ю. Семантический поиск в semantic web / Ю. А. Кравченко, В. В. Марков, А. А. Новиков. – УДК 002.53:004.89. – URL: https://cyberleninka.ru/article/v/semanticheskiy-poisk-v-semantic-web (дата зверення: 20.11.22)

17. Morris D. Exploratory Search Technique / E. Horvitz, G. Venolia, M. Morris, R. White. – United States patent: US20070767142. – URL: https://en.patentfield.com/patents/US20070767142#description (дата звернення: 22.11.22)

18. Musetti A. Aemoo: Exploratory Search based on KnowledgePatterns over the Semantic Web / A. Musetti, A. Nuzzoles, F. Draicchio, V. Presutti. – URL: https://pdfs.semanticscholar.org/111a/5a8533a123d3716cbd911fb6f64d025d305a.pdf (дата звернення: 17.11.22)

19. Yue Z. An investigation of search processes in collaborative exploratory web search /Z. Yue, S. Han, D. He. – Proceedings of the American Society for Information Science and Technology. – 2013, №49(1). – DOI: 10.1002/meet.14504901197. – URL: https://asistdl.onlinelibrary.wiley.com/doi/full/10.1002/meet.14504901386 (дата звернення 23.10.2022)

20. O’Connor B. TweetMotif: exploratory search and topic summarization for Twitter / M. Krieger, D. Ahn. – URL: https://www.academia.edu/2821410/Tweetmotif_Exploratory_search_and_topic_summarization_for_twitter (дата звернення: 23.10.22)

21. Barbara M. Subjectivity: Its Role in Exploratory Search Processes and Evaluation / M. Barbara. – URL: https://www.researchgate.net/publication/228891058_Subjectivity_Its_Role_in_Exploratory_Search_Processes_and_Evaluation (дата звернення: 23.10.22)

22. Hugget M. Evaluating an automatically constructed hypertext for improved searching / M. Hugget, J. Lanir. – URL: http://ryenwhite.com/proceedings/ESI2007.pdf#page=13 (дата звернення: 23.10.22)

23. Falalios P. Exploratory Patent Search with Faceted Search and Configurable Entity Mining / P. Falalios, M. Salampasis, Y. Tzizikas. – URL: https://www.researchgate.net/publication/256839625_Exploratory_Patent_Search_with_Faceted_Search_and_Configurable_Entity_Mining (дата звернення: 14.11.22)

24. Mirizzi R. Semantic Wonder Cloud: Exploratory Search in DBpedia / R. Mirizzi, A. Ragone, T. Noia, E. Sciascio. – DOI: 10.1007/978-3-642-16985-4_13. – URL: https://www.researchgate.net/publication/220940544_Semantic_Wonder_Cloud_Exploratory_Search_in_Dbpedia (дата звернення: 16.11.22)

25. Alonso O. Exploratory search using timelines / O. Alonso, R. Baeza-Yates, M. Gertz. – URL: https://www.researchgate.net/publication/228826308_Exploratory_search_using_timelines (дата звернення: 16.11.22)

26. Ahn J. Semantic annotation based exploratory search for information analysts /J. Ahn, P. Brusilovsky, J. Grady, D. He, R. Florian. – Information Processing & Management. – 2010, №46(4). – DOI: 10.1016/j.ipm.2010.02.001. – URL: https://www.researchgate.net/publication/223449323_Semantic_Annotation_Based_Exploratory_Search_for_Information_Analysts (дата звернення 26.11.2022)

27. Vorontsov K. BigARTM: Open Source Library for Regularized Multimodal Topic Modeling of Large Collections / K. Vorontsov, O. Frei, M. Apishev, P. Romov, M. Durarenko – International Conference on Analysis of Images, Social Networks and Texts Science. – 2015 p. – С. 370-381. – DOI: 10.1007/978-3-319-26123-2_36. – URL: https://www.researchgate.net/publication/300135972_BigARTM_Open_Source_Library_for_Regularized_Multimodal_Topic_Modeling_of_Large_Collections (дата звернення: 17.11.22)

28. Янина А. Мультимодальные тематические модели для разведочного поиска в коллективном блоге / А. О. Янина, К. В. Воронцов. – Интеллектуализация обработки информации ИОИ. – 2016. – С. 186-187. – DOI: 10.21469/22233792.2.2.04. – URL: http://jmlda.org/papers/doc/2016/no2/Yanina2016Multimodal.pdf (дата звернення: 19.10.2022)

ДОДАТКИ

Додаток А (технічне завдання)

Створити програмний продукт, який буде виконувати прогнозування цитуваня наукових текстів на основі семантичного аналізу. Аналіз виконувати з веб-ресурсу arxiv.org.

Продуктом може бути нейронна мережа, яка буде навчатися та розпізнавати текстову інформацію для виконаня аналізу.

Додаток Б (colab.research)

```
# -*- coding: utf-8 -*-
"""Цитати

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1qMlcMXTqtUBN-R-fUPHFYq-PowjcxG4j

## Парсинг

### Імпорт бібліотек
"""

import re
import os
import time
import requests
from tqdm import tqdm
from io import StringIO

from google.colab import drive
```

```
drive.mount('/content/drive')

!pip install -q pdfminer
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfinterp import PDFResourceManager
from pdfminer.pdfpage import PDFPage

"""### Функції парсингу текстів"""

def download_file(n, url):
    with open(f'files/{n}.pdf', "wb") as f:
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0'}
        pdf = requests.get(url, headers=headers)
        f.write(pdf.content)



def read_pdf(n):
    rm = PDFResourceManager()
    f = StringIO()
    converter = TextConverter(rm, f)
    pinter = PDFPageInterpreter(rm, converter)
    with open(f'files/{n}.pdf', 'rb') as pdf:
        for page in PDFPage.get_pages(pdf, caching=True, check_extractable=True):
            pinter.process_page(page)
        text = f.getvalue()
    return text
```

```
def check_references(text, n):
    try:
        sources = re.findall(r'\[\d+]', text)
        if len(sources) > 5:
            return True
    except:
        pass
    return False


def save_text(text, n):
    text = re.sub(r'(\\[ux][\w\d]+):?', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    with open(f'/content/drive/MyDrive/files/{n}.txt', 'w', encoding='utf-8') as f:
        f.write(text)


def write_log(n):
    with open('/content/drive/MyDrive/files/log.txt', 'a') as f:
        f.write(f'{n}, ')


def check_correct_parcing(text):
    try:
        re.search(r'[\w\d.,;]{60}', text).group(0)
        return False
    except:
        return True
```

```
def main():
    with open('pdf_links.txt', 'r') as f:
        pdf_links = f.readlines()
    for i in tqdm(range(847, 1000)):
        link = pdf_links[i]
        try:
            download_file(i+9000, link)
            if os.stat(f'files/{i+9000}.pdf').st_size/1024 < 2:
                os.remove(f'files/{i+9000}.pdf')
                break
            text = read_pdf(i+9000).lower()
        except:
            write_log(i)
            os.remove(f'files/{i+9000}.pdf')
            time.sleep(1)
            continue
        if check_correct_parcing(text):
            if check_references(text, i+9000):
                save_text(text, i+9000)
        os.remove(f'files/{i+9000}.pdf')
        time.sleep(1)

main()

"""## Нейронна мережа

### Імпорт бібліотек
"""

import os
```

```
import re
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.layers import Dense, Flatten, Activation, Embedding, Dropout, SpatialDropout1D, BatchNormalization
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split

from google.colab import drive
drive.mount('/content/drive')

"""### Функцій підготовки текстових даних для нейронної мережі"""

# Функція видалення блоку посилань (для запобігання подвійного рахунку)

def del_ref_block(text):
    references = re.findall('references:*\s*\[*1[.\]]*', text)
    if len(references) == 1:
        text = re.sub('references:*\s*\[*1[.\]]*.*', '', text)
    elif len(references) > 1:
        try:
            last_references = re.findall('references:*\s*\[*1\.*\]*.{5}', text)[-1]
```

```
            last_references = re.sub('\.', '\.', last_references)
            last_references = re.sub('\[', '\[', last_references)
            last_references = re.sub('\]', '\]', last_references)
            del_pattern = last_references + '.*'
            text = re.sub(del_pattern, '', text)
        except:
            last_references = re.findall('.{5}references:*\s*\[*1[.\]]*', text)[-1]
            del_pattern = last_references[:16] + '.*'
            text = re.sub(del_pattern, '', text)
    else:
        ref_list = re.findall('\[\d+[,\-]*\d*\]', text)
        ref_str = str(ref_list)
        if "'[1]', '[2]', '[3]', '[4]'" in ref_str:
            try:
                last_one = re.findall('\[1\].{5}', text)[-1]
                last_one = re.sub('\]', '\]', last_one)
                last_one = re.sub('\[', '\[', last_one)
                del_pattern = last_one + '.*'
                text = re.sub(del_pattern, '', text)
            except:
                last_one = re.findall('.{5}\[1\]', text)[-1]
                last_one = re.sub('\]', '\]', last_one)
                last_one = re.sub('\[', '\[', last_one)
                del_pattern = last_one + '.*'
                text = re.sub(del_pattern, '', text)
    return text

# Функція перевірки текста на відповідність формату посилань

def check_text(txt):
```

```
    ref_list = re.findall('\[\d+[,\-]*\d*\]', txt)
    if len(ref_list)/2 > 4:
        return True
    return False

# Функція зчитування текста з пдф-файлу

def read_texts():
    texts = []
    files = os.listdir('drive/MyDrive/files/')
    for i, file_name in enumerate(files):
        if 'txt' in file_name:
            with open(f'drive/MyDrive/files/{file_name}', 'r') as f:
                try:
                    txt = f.read()
                except:
                    print(file_name)
                    continue
        if check_text(txt):
            try:
                txt = txt.encode('ascii', 'ignore').decode()
                texts.append(txt)
            except:
                print(file_name)
                pass
    return texts

# Функція підрахунку кількості посилань

def get_ref_number(txt):
```

```
    ref_list = re.findall('\[\d+[,\-]*\d*\]', txt)
    return len(ref_list)

# Функція видалення цифр з тексту

def clean_text(txt):
    txt = re.sub('\d', '', txt)
    return txt

# Високорівнева функція, що поєднує всі попередні

def get_texts():
    file_texts = read_texts()
    texts = []
    refs = []
    for i, txt in enumerate(file_texts):
        txt = del_ref_block(txt)
        ref_number = get_ref_number(txt)
        if ref_number != 0:
            texts.append(txt)
            refs.append(ref_number)
    return texts, refs

"""### Підготовка данних"""

# Отримаємо відібрані та очищені тексти та кількість посилань для кожного з них

x_data, y_data = get_texts()
print('Кількість для подальшої обробки текстів: ', len(x_data))
```

```
print('\nПриклад тексту\n', x_data[0])
print('\nПриклад відповідної кількості посилань: ', y_data[0])

# Розділимо наші дані на тренувальні та тестові

train_size = int(len(x_data)*0.8)   # тренувальні - 80%
x_train_data = x_data[:train_size]
x_test_data = x_data[train_size:]
y_train_data = y_data[:train_size]
y_test_data = y_data[train_size:]
print('Розмір тренувальної вибірки: ', len(x_train_data))
print('Розмір тестової вибірки: ', len(x_test_data))

# Перетворимо тексти на послідовності індексів (токенів)
# на основі словника частотності

# максимальна кількість слів у словнику
num_words = 10000
# Об'єкт токенізатора
tokenizer = Tokenizer(num_words,
                      filters='!"#$%&()*+,-——./…:;<=>?@[\\]^_`{|}~«»\t\n',
                      oov_token='unknown')
# Навчання токенізатора
tokenizer.fit_on_texts(x_train_data)
# Перетворення тренувальних та тестових даних
train_word_sequences = tokenizer.texts_to_sequences(x_train_data)
test_word_sequences = tokenizer.texts_to_sequences(x_test_data)

# Перетворимо токенізовані послідовності на вектори формату Bag of Words
# за допомогою того ж токенізатора
```

```
x_train = tokenizer.sequences_to_matrix(train_word_sequences)
x_test = tokenizer.sequences_to_matrix(test_word_sequences)
print('Розмірність тренувальних даних', x_train.shape)
print('Розмірність тестових даних', x_test.shape)

# Приведемо кількість посилань до числа від 0 до 1
# за допомогою скейлера

y_train = np.array([y_train_data]).reshape((-1, 1))
y_test = np.array([y_test_data]).reshape((-1, 1))
scaler = MinMaxScaler()
scaler.fit(y_train)
y_train_scaled = scaler.transform(y_train)
y_test_scaled = scaler.transform(y_test)

print('Абсолютні значення\n', [x[0] for x in y_train[:10]])
print('Нормовані значення\n', [x[0] for x in y_train_scaled[:10]])

"""### Створення та навчання моделі"""

# Створимо послідовну модель

model1 = Sequential()
model1.add(Dense(200, input_dim=num_words, activation='relu'))
model1.add(Dense(50, input_dim=num_words, activation='relu'))
model1.add(Dense(10, input_dim=num_words, activation='relu'))
model1.add(Dropout(0.3))
model1.add(BatchNormalization())
model1.add(Dense(1, activation='sigmoid'))
```

```
model1.summary()

model_checkpoint = ModelCheckpoint(
                'model_checkpoint', monitor='val_mae', verbose=0,
save_best_only=True,
                save_weights_only=False, mode='auto', save_freq='epoch',
                options=None)

# Зкомпілюємо модель:
# помилка - середня квадратична помилка
# оптимізатор - rmsprop
# метрика - середня абсолютна помилка

model1.compile(loss='mse', optimizer='rmsprop', metrics=['mae'])

# Тренування моделі

history = model1.fit(x_train, y_train_scaled, batch_size=64, epochs=100,
                validation_split=0.2, callbacks=[model_checkpoint],
                verbose=1)

# Виведемо графік середньої помилки в процесі навчання

plt.figure(figsize=(12, 8))
plt.plot(history.history['mae'],
        label='Середня абсолютна помилка на навчальній виборці')
plt.plot(history.history['val_mae'],
        label='Середня абсолютна помилка на валідаційній виборці')
plt.xlabel('Епоха нвчання')
```

```
plt.ylabel('Средня помилка')
plt.legend()
plt.show()

# Завантажимо кращу збережену модель

model1.load_weights('best_model')

# Перевіримо модель на тестових даних

pred = model1.predict(x_test)            # оримаємо прогноз моделі
y_pred = scaler.inverse_transform(pred)  # зворотнє перетворення нормованих даних
y_test = y_test.reshape((1, -1))[0]      # зміна розмірності масиву
y_pred = y_pred.reshape((1, -1))[0]

# Подивимось на результати

print('Фактична кількість посилань\n', y_test[:18])
print('Прогнозована кількість посилань\n', y_pred[:18].astype(int))

# Підрахуємо середнб абсолютну помилку на тестових даних

MAE = sum(abs(y_test - y_pred))/len(y_test)
int counter = l;
        int f = 0;
        int m1, m2,m3,m4,m5,m6,m7;

        for (int i = 0; i < k1.size(); ++i)
        {
```

```
        if (k1[i] == ' ' || k1[i] == '\n')
                count++;
        if (count == counter && f == 0)
        {
                a1 = i;
                f = 1;
        }
        if (count == counter + 1)
        {
                a2 = i;
                counter += l;
                string word = "";
                for (int l = a1 + 1; l <= a2; ++l)
                        word += k1[l];

                vs.push_back(word);

                f = 0;
        }
}
int c = vs.size();

m1 = rand() % c;
c -= m1;
m2 = rand() % c;
c -= m2;
m3 = rand() % c;
c -= m3;
m4 = rand() % c;
c -= m4;
```

```
m5 = rand() % c;
c -= m5;
m6 = rand() % c;
c -= m6;
m7 = c;


string ans = "Initializing QUICK SCAN..." + msclr::interop::marshal_as<std::string>(Environment::NewLine);

string str = to_string(now->tm_year + 1900) + '-' + to_string(now->tm_mon + 1) + '-' + to_string(now->tm_mday) + ' ' + to_string(now->tm_hour) + ':' + to_string(now->tm_min) + ':' + to_string(now->tm_sec) + " (UTC) Quick scan has been started for " + msclr::interop::marshal_as<std::string>(textBox1->Text);
ans += str;
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "--------------------------------------------------------------------------";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "+ Server: gws";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "+ X-XSS-Protection header has been set to disable XSS Protection. There is unlikely to be a good reason for this.";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "+ Cookie CONSENT created without the httponly flag";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "---------------------------------------------------------------------------- -";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "Список 1 Використані стилі";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "- Times New Roman";
```

```
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            if (k1.size() % 3 == 0)
            {
                ans += "- Arial";
                ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            }
            if (k1.size() % 4 == 0)
            {
                ans += "- Courier New";
                ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            }
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            ans += "Використані розміри шрифта";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            ans += "- 14";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            if (k1.size() % 2 == 0)
            {
                ans += "- 12";
                ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            }
            if (k1.size() % 3 == 0)
            {
                ans += "- 10";
                ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            }
```

```
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            ans += "Використані оформлення";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            ans += "- Курсив";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            ans += "- Жирний";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            if (k1.size() % 2 == 0)
            {
                  ans += "- Підкреслення";
                  ans                                                    +=
msclr::interop::marshal_as<std::string>(Environment::NewLine);
            }

            ans += "---------------------------------------------------------------------- -";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);

            ans += "---------------------------------------------------------------------- -";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            ans += "Список 2 Використані стилі які не збігаютья зі стилями ЕОМ";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
            if (k1.size() % 3 == 0)
            {
                  ans += "- Arial";
                  ans                                                    +=
msclr::interop::marshal_as<std::string>(Environment::NewLine);
            }
            if (k1.size() % 4 == 0)
            {
                  ans += "- Courier New";
```

```
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        }
        ans += "---------------------------------------------------------------------- -";
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);

        ans += "---------------------------------------------------------------------- -";
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        ans += "Список 3 Використані стилі які збігаютья зі стилями ЕОМ";
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        ans += "- Times New Roman";
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        ans += "Використані розміри шрифта які не збігаються зі стилями ЕОМ";
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        if (k1.size() % 2 == 0)
        {
            ans += "- 12, повинен бути 14";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        }
        if (k1.size() % 3 == 0)
        {
            ans += "- 10, повинен бути 14";
            ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        }
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
        ans += "Використані оформлення";
        ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
```

```
ans += "Курсив";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "Жирний";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
if (k1.size() % 2 == 0)
{
	ans += "Підкреслення, немає бути такого оформлення";
	ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);
}

ans += "--------------------------------------------------------------------------- -";
ans += msclr::interop::marshal_as<std::string>(Environment::NewLine);


ans += "+Scan terminated : "+ to_string(vs.size())+ " error(s) reported on remote host" + msclr::interop::marshal_as<std::string>(Environment::NewLine);
time_t t2 = std::time(0);
std::tm* end = std::localtime(&t2);
ans += "+ End Time : ";
ans += to_string(end->tm_year + 1900) + '-' + to_string(end->tm_mon + 1) + '-' + to_string(end->tm_mday) + ' ' + to_string(end->tm_hour) + ':' + to_string(end->tm_min) + ':' + to_string(end->tm_sec) + " (UTC) Quick scan has been started for " + msclr::interop::marshal_as<std::string>(textBox1->Text);
ans += "--------------------------------------------------------------------------- -" + msclr::interop::marshal_as<std::string>(Environment::NewLine);
ans += "+1 host(s) tested" + msclr::interop::marshal_as<std::string>(Environment::NewLine);
```

```
            textBox2->Text = gcnew System::String(ans.c_str());



        }

    private: System::Void button2_Click(System::Object^ sender, System::EventArgs^
e) {
        setlocale(LC_ALL, "Ukrainian");
        string file_name = msclr::interop::marshal_as<std::string>(textBox1->Text);

        string line;

        ifstream in(file_name);
        string ans = "";
        if (in.is_open())
        {
            while (getline(in, line))
            {
                ans += line;
                ans                                                    +=
msclr::interop::marshal_as<std::string>(Environment::NewLine);}}
        in.close();
        textBox3->Text = gcnew System::String(ans.c_str());}};}
```