Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

(повна назва факультету)

Кафедра програмної інженерії

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

(назва освітнього ступеня)

на тему: Розробка системи для класифікації текстової інформації по категоріях

Виконав: студент IV курсу, групи СП-41

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Бабинець К.М.

(підпис) (прізвище та ініціали)

Керівник Пастух О.А.

(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю.М.

(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М.Р.

(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

Тернопіль

2024

АНОТАЦІЯ

У кваліфікаційній роботі на здобуття освітнього ступеня бакалавр виконано критичний огляд та кретичний аналіз методів, моделей, алгоритмів та програмного забезпечення для обробки та аналізу текстової інформації. Сформульовано актуальні завдання, які у подальшому вирішені у роботі щодо класифікації (категоризації) текстової інформації у вигляді завдання бінарної класифікації. Спроектовано та сконструйовано програмне забезпечення у вигляді програмної системи для бінарної класифікації текстової інформації, оцінено якість роботи програмної системи шляхом обчислення різних видів метрик для класифікації даних. Загальна архітектура обчислювального процесу представляє собою конвеєр різного роду операцій спершу для попередньої їх обробки (стемінг, лемантизація, видалення стоп-слів), а потім для основної їх обробки (бінарної класифікації на основі логістичної регресії). Програмно реалізовано можливість збереження основного програмного об’єкта – навченого алгоритму машинного навчання логістичної регресії у файловий компонент, який може у подальшому використовуватися сторонніми програмами. Для структуризації слабоструктурованих даних, якими є текстова інформація використано модель мішка слів. Для оцінювання точності використані метрики звичайної точності, f1-метрика та roc-auc метрика. Для наочності сприйняття та загалом розуміння величини похибки (або оберненої величини до неї - точності) бінарної класифікації програмно обчислені та візуалізовані нормована та не нормована матриці помилок, величина значень елементів головної діагоналі якої вказують на високу якість роботи алгоритму машинного навчання логістичної регресії, а величина елементів не головної діагоналі вказує на низьку якість.

ANNOTATION

In the qualifying work for obtaining a bachelor's degree, a critical review and critical analysis of methods, models, algorithms, and software for processing and analyzing textual information was performed. Actual tasks are formulated, which are subsequently solved in the work on the classification (categorization) of textual information in the form of a binary classification task. Software in the form of a software system for binary classification of text information was designed and constructed, the quality of the software system was evaluated by calculating various types of metrics for data classification. The general architecture of the computational process is a pipeline of various operations, first for their preliminary processing (stemming, lemmantization, removal of stop words), and then for their main processing (binary classification based on logistic regression). Software has implemented the possibility of saving the main software object - the trained logistic regression machine learning algorithm into a file component that can be used by third-party programs in the future. A bag of words model was used to structure loosely structured data, which is textual information. To evaluate the accuracy, the usual accuracy metrics, f1-metric and roc-auc metric were used. For clarity of perception and general understanding of the magnitude of the error (or the inverse of it - accuracy) of the binary classification, normalized and non-normalized error matrices were calculated and visualized by software, the magnitude of the values of the elements of the main diagonal of which indicate the high quality of the logistic regression machine learning algorithm, and the magnitude of the elements off the main diagonal indicates low quality.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ..... 7
ВСТУП..... 8
РОЗДІЛ 1. ОГЛЯД І АНАЛІЗ ПРИКЛАДНИХ МЕТОДІВ, ПРИКЛАДНИХ МОДЕЛЕЙ, АЛГОРИТМІВ ТА ПРИКЛАДНИХ ПРОГРАМНИХ ІНСТРУМЕНТІВ ДЛЯ ПРИКЛАДНОЇ ОБЛАСТІ ДАНОЇ РОБОТИ..... 9
1.1. Критичний огляд та критичний аналіз прикладної області..... 9
1.2. Завдання, які вирішуються при обробці текстової інформації..... 11
1.3. Алгоритми для класифікації текстової інформації та гіперпараметри для керування ними..... 13
1.4. Відбір інформативних ознак для текстової класифікації..... 14
1.5. Мета, завдання кваліфікаційної роботи бакалавр..... 16
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРИЦИЗІЙНОЇ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ КЛАСИФІ-КАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ ПО КАТЕГОРІЯХ..... 18
2.1. Проектування архітектури прицизійної системи для класифікації текстової інформації по категоріях..... 18
2.2. Деталювання процесу проектування прицизійної системи для класифікац. текстової інформац. по категоріях..... 19
2.2.1. Проектування програмної компоненти для читання даних пандас.. 19
2.2.2. Апріорне опрацювання інформації у форматі тексту..... 22
2.3. Проектування трансформаційної структуризації текстової інформації..... 31
2.4. Проектування наступних програмних компонент системи..... 33
РОЗДІЛ 3. ПРАКТИЧНЕ КОНСТРУЮВАННЯ ПРИЦИЗІЙНОЇ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ ПО КАТЕГОРІЯХ..... 38

3.1. Ввід вхідних даних у програмну прицизійну систему класифікації текстової інформації..... 38
3.2. Селекція корпусу малого об’єму текстових документів..... 41
3.3. Операція стемінг корпусу малого об’єму текстових документів..... 43
3.4. Процес токенізації корпусу малого об’єму текстових документів..... 44
3.5. Перехресна перевірка класифікатора логістичної регресії..... 46
3.6. Розшеплення даних на навчальні та тестові сукупності..... 48
3.7. Навчання та прийняття рішень класифікатором..... 49
3.8. Оцінювання метрик якості..... 50
3.9. Візуалізація нормованої та не нормованої матриць помилок..... 51
3.10. Запис моделі у файл..... 54
РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ..... 55
4.1. Ризик як кількісна оцінка небезпек..... 55
4.2. Навчання працюючих і інструктажі з охорони праці..... 58
ВИСНОВКИ..... 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 62
ДОДАТКИ..... 64
Додаток А Технічне завдання..... 64
Додаток Б. Тези доповіді на конференції..... 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТД – корпус текстових документів

МК – мітки класів

СД – слабоструктуровані дані

ТІ – текстова інформація

ПС – програмна система

ПБ – програмний блок

АПБ – архітектура програмного блоку

АС – архітектура системи

ВСТУП

Одним із головних напрямів обробки інформації є обробка вербальної інформації, зокрема, текстової інформації. Саме текстову інформацію досить часто використовують для представлення різних аспектів життя. Наприклад, часто текстовою інформацією представляються історичні дані, формулюються юридичні норми, висловлюються логічні доведення, тощо. Загалом, ми існуємо у семіотичному світі – світі в якому інформація обробляється, зберігається та передається за допомогою тих чи інших знаків. Набір знаків довільної природи утворює абстрактну абетку на основі якої будуються повідомлення (текстові дані із знаків довільної природи).

Якщо розглянути приклади сучасних сфер бізнесу, діяльність якого суттєво залежить від швидкості, якості, рентабельності обробки текстов. інформац., то можна навести такі приклади: розробк. спам. фільтрів; категорізац. потоку у часі інформаційн. новин текстовог. формату для медійних компаній; пошук інформац.; побудов. чат-ботів; анал. тональност. текстів; переклад текстів; одержання знань із текст.; розпізнаванн. текстів; тематичн. аналіз текст.; перетворення неструктурован. текстов. інформації у структурован. дані; текстов. анотація, тощо.

У даній робот. розглядається завдання класифікац. текстов. інформації (корпусу текстов. документ. – набір книг) по категоріях, зокрема, розробка програм. сист., яка би визначала (класифікувал.) книги для дітей та книги для дорослих. Дані були взяті з відкритої платформ. Kaggle.

РОЗДІЛ 1. ОГЛЯД І АНАЛ. ПРИКЛАДНИХ МЕТОДІВ, ПРИКЛАДНИХ МОДЕЛЕЙ, АЛГОРИТМІВ ТА ПРИКЛАДНИХ ПРОГРАМНИХ ІНСТРУМЕНТІВ ДЛЯ ПРИКЛАДНОЇ ОБЛАСТІ ДАНОЇ РОБОТИ

1.1 Критичний огляд та критичний аналіз прикладної області

У сьогоденні одним із актуальних напрямів для якого розробляються інтелектуалізовані програмні прицизійні системи – це є обробка вербально-текстової інформації. Саме текстову інформацію досить часто використовують для представлення різних аспектів життєдіяльності людини. Наприклад, часто у вигляді текстової інформації представляються історичні дані, формулюються юридичні норми, висловлюються логічні доведення, тощо. Загалом, ми існуємо у семіотичному світі – світі в якому інформація обробляється, зберігаєтьс. та передається за допомогою тих чи інших знаків. Набір знаків довільної природи утворює абстрактну абетку на основі якої будуються повідомлення (текстові дані із знаків довільної природи).

Опрацювання текстової інформації є не що інше, як опрацювання послідовності знаків з довільної абстрактної абетки. Завдання класифікації послідовностей знаків довільної природи – це кластеризація цієї сукупності чи наборів послідовностей абстрактних знаків. Завдання чистої кластеризації або ще називається навчання без вчителя це розбиття на підгрупи без апріорі заданих міток. Але у випадку, що розглядається тут в роботі розбиття на підгрупи розглядається на основі міток, аналог навчання з вчителем.

Опрацювання інформації у текстовому виді є надзвичайним важливим напрямом, оскільки ми люди більшість інформації з навколишнього світу дізнаємося з інформацією яка подається на основі знаків – інформаційних текстів. Але особливо цікавими моментами є випадки, що виявлені в останні роки в експериментах в галузі нейронаук, а саме, дослідження на основі функціональної магніто-резонансної

томогр. Встановлено, що людський мозок знаки довільної абстракції та природи картує по поверхні кори мозку, накшталт, як міста картовані на політичній карті світу. Поверхня кортексу картує слова по типу того, як наведено на рисунку, який наведений нище.

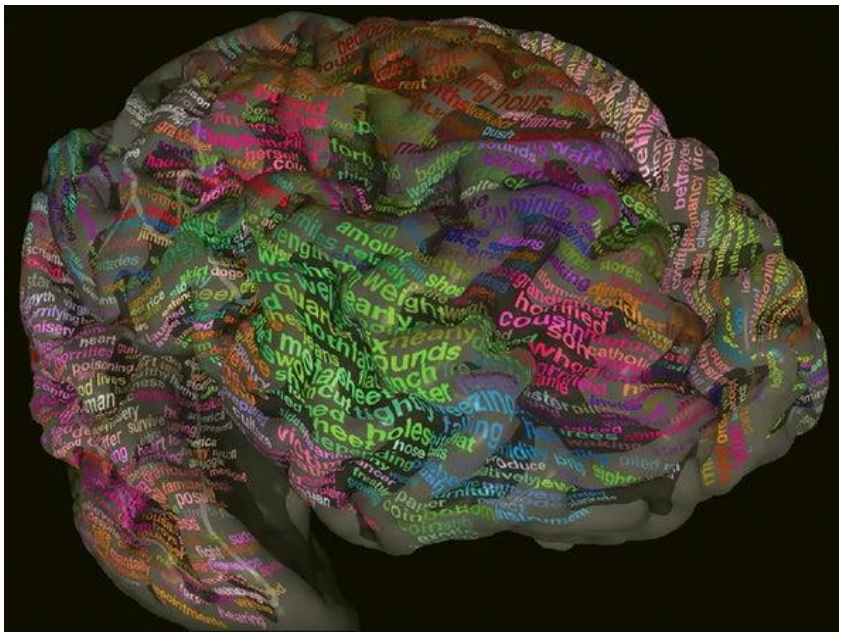

Рис.1.1 – Вид картування слів кортексом головного мозку

Така природа картування кортексом головного мозку знаків довільної природи відкриває великі можливості до розгляду різних завдань обробки природної інформації у формі тексту. З рисунку видно, що знаки-слова утворюють поверхневий простір, а значить можна цей простір розбивати на підпростори, тобто на кластери, що означає на класи. У такому виді слова утворюють на поверхні кортексу кластерну структуру. Визначаючи словники слів для кожного окремого кластера слів можна на основі цих кластерних словників класифікувати різні набори інформації у формі текстів. Наприклад можна розглядати імовірності появи представників слів кластерів кортексу у тестовій інформаціх документу і вже по вазі слів із різних кластерів слів визначати до якого саме класу – кластеру належить той чи інший документ із набору або так званого корпусу документів. Корисно розглянути

збільшений масштаб картованих кортексом слів на кластери, оскільки видно що подібні за лексико-семантичними значеннями слова також кластеризуються і локалізуються на поверхні кори мозку. Для наочності розуміння нище наведено рисунок на якому видно ці структури.

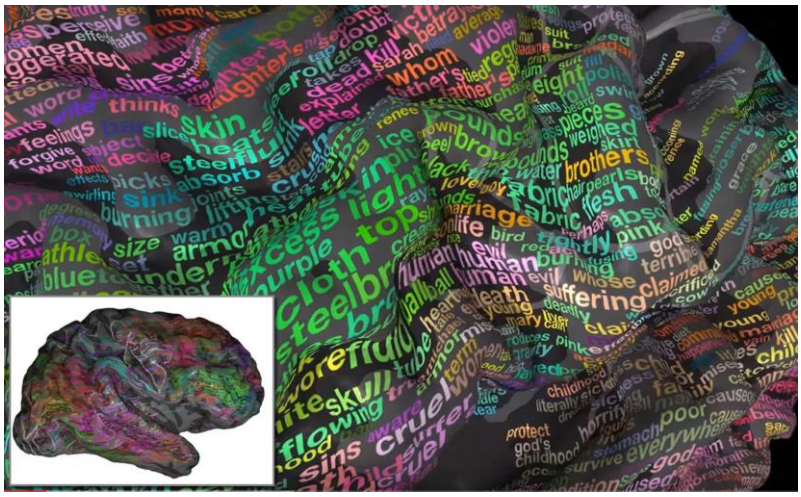

Рис.1.2 – Збільшений масштаб картованих слів на поверхні кортексу

Такого роду картування слів тієї чи іншої природної мови дає змогу вирішувати широке коло прикладних завдань для потреб бізнесу. Тому у наступному підрозділі перейдемо до розгляду переліку важливих прикладних завдань для потреб практики, які можна вирішувати шляхом класифікації інформації у формі тексту.

1.2 Завдання, які вирішуються при обробці текстової інформації

Загалом розглядається широки спектр прикладних завдань, які йдуть з різних реальних бізнес доменів. Розгляньмо деякі базові такого роду завдання. Поряд з цим

зазначимо, що ці прикладні завдання також зустрічаються в машинному навч. і галузь носить назву опрацювання природної мови. Рогляд виконаємо переліком даних завдань ьа коротким їх описом по їх суті, що вони означають і цим самим формується представлення про даного роду завдання. Перейдемо до цих завдань. Серед цих завдань, можна виділити наступні завдання.

- пошук інформації у формі тексту в різного роду наборах повідомлень, накшталт, постів у соціальних мережах, електронних листах, тощо;

- розробка ботів для різних месенджерів, мова йде про ботів для діалогів у текстових чатах, щоб вони могли підтримувати діалоговий формат роботи;

- аналіз психологічного стану людей на основі вимірювання їх емоційних станів, так звані завднання по класифікації емоційно вжитих слів і на їх основі вирішувати завдання сентімент аналізу – класифікації тональності інформації на основі текстів;

- завдання перекладів з однієї довільної мови на іншу мову робиться на основі інформації, що реалізована текстовими знаками, навіть якщо апріорі вона була виражена звуковими сигналами;

- побудова патернів між даними, тобто екстракція знань із інформації, яка представлена у формі текстів;

- розпізнавання інформаціх у формі текстових повідомлень, тобто її класифікація та й загалом регресійне оцінювання;

- аналіз інформації у текстовій формі по темах та й загалом по мета темах, сукупність яких може вибудовувати довільну ієрархічну структуру;

- трансформація слабоструктурованої інформації якою є інформація у текстовій формі до структурованого виду, накшталт, різних алгебраїчних структур;

- скорочення різної фломи обємних текстових повідомлень, наприклад, для зменшення затрат памяті на їх зберігання, скорішу організацію пошуку у наборі

інформації в текстовій формі так звана текстова анотація, або як ще називається текстова сумаризація інформації, і вирішення даного завдання можна організовувати в різні способи, тобто на основі різних методів та алгоритмів;

- визначення частин мови, тобто визначення іменників, прикметників, іменникових фраз, прикметникових фраз, дієслів та на їх основі дієслівних фраз, вирішення завдання визначення іменованих сутностей у різного роду текстах і на основі них визначати різного роду мутності про які ведеться мова у текстових повідомленнях.

Перелік різного роду завдань для різних бізнес доменів можна продовжувати довго, але оскільки обєм даної роботи є лімітованим, то далі зупинятися на інших прикладах не будемо.

1.3 Алгоритми для класифікації текстової інформації та гіперпараметри для керування ними

Вирішення різного роду прикладних завдань з практики, які наведені у попередньому підрозділі не можливе без вибору тих чи інших методів та алгоритмів на яких будуються програмні прицизійні системи. Тому у даному підрозділі, на основі еврістичної інформації буде наведено ряд класифікаторів, що дають високоефективні результати для класифікації інформації у формі тексту.

Еврістика вказує на наступні класифікатори та їх гіперпараметри, які дають хороші практичні результати:

- дерева рішень (гіперпараметри дерев рішень: мінімальне число об’єктів даних у листак дерев рішень, максимальна висота дерева рішень);

- рандомний ліс із гіперпараметрами – це мінімальне число об’єктів даних у листках дерев рандомного лісу, максимальна висота дерев рандомного лісу, кількість дерев рішень – величина рандомного лісу;

- алгоритм логістичної регресії із гіперпараметрами - це константа для регуляризації, класовий вид представника – регуляризатора, кількість повторення ітерацій в організації даного алгоритму;

- баєсівський класифікатор у найпростішому виді, тобто із гаусівськими функціями правдоподібностями;

- опорні вектори із гіперпараметрами – константою регуляризації фіксованого виду, класовий вид представника-регуляризатора, форма поверхні.

Практика показує, що можна підсилити алгоритмічному рівні вирішення завдань класифікації шляхом надбудови різних мета алгоритмічних структур, тобто пробувати зробити різного роду ансамблювання алгоритмів, накшталт, простого голосування вище наведених алгоритмів або зваженого голосування вище наведених алгоритмів. Також можна організовувати на мета рівні різного виду блендінги та стекінги алгоритмів.

1.4 Відбір інформативних ознак для текстової класифікації

Розглядаючи у попередніх підрозділах наведені прикладні завдання для бізнесу та алгоритми високого рівня якості вирішення прикладних завдань також не можливо досягнути. Полтрібно ще детальніше і конструктивніше на більш високому рівні вирішувати завдання із блоку інженерії ознак, тобто потрібно в процесі попередньої обробки даних правильно ввибирати методи для екстракції

інофрмативних даних та добувати в принципі нові дані та мета дані для якісного їх опрацювання, тобто досягати високої точності та стійкості роботи у програмних прицизійних системах для опрацювання інформації в текстовій формі. Нище наводяться різного роду підходи:

- вибір інформативних атрибудів чи ознак можна здійснювати на основі статистичного тесту пірсона, який визначає значимість атрибутів на основі лінійної кореляції між вхідною інформацією у форматі тексту та вихідними даними, наприклад, міткою класу, однак у цього підходу є як переваги так і недоліки, а саме, до переваг відносяться наочність інтерпритації лінійного коефіцієнта кореляції сили звіязку, але є й недоліки, які полягають в тому, що даниї кореляційний показник не реагує на нелінійні патерни, які можуть бути між текстовою інформацією і міткою класу, а також він не враховує каузального ефекту;

- відбір інформативних ознак чи то атрибутів на основі статистичного тесту Фішера. Даний тест дає змогу виявляти не лінійні залежності між вхідною інформаціжю у форматі тексту та цільовою інформацією, накшталт, вихідних даник у вигляді мітки класів чи регресійної змінної, що не можна зробити на основі попередньо описаного підходу в даному переліку. Але і Фішерівський підхід, тобто тест не в змозі забезпечити лише сильні сторони для проектування та конструювання прицизійних систем, оскільки він є обчислювально важким, а тому зменшує рентабельність розробки, також він не вирішує питання каузальності, тому розглядаються і інші підходи;

- із статистичної теорії перевірки статистичних гіпотех відомо, що альфа рівень менший тим більш інформативною є ознака чи то атрибут і його потрібно включати до вхідних даних. разом з цим альфа рівень вказує на величину похибки прийняття рішень, коли спрацювала перша статистична гіпотеза, тому і тут не все так просто, як хотілося би у цьому підході;

- має місце використання підходу на сонові логістичної регресії з використанням різного роду регуляризаторів, як з простору гільбертового

метричного простору так і з інших метричних функціональних просторів. Вага кожної із ознак у рівнянні визначає на інформаційному рівні значимість та важливість тієї чи іншої ознаки-атрибута, тобто змінної, які слід чи не слід включати до вхідних даних перед тим, як їх подати на класифікатор;

- також тут можна використовувати підхід на основі рандомного лісу, зокрема використовувати його гіперпараметр oob_score=True для селекції та екстракції існуючих ознак-атрибутів до сукупності вхідних даних. Даний підхід дає хороші результати коли ознаки-атрибути вхідних даних є не гомогенними, тобто різні по інтерпритації.

1.5 Мета, завдання кваліфікаційної роботи бакалавр

Для успішного виконання роботи потрібно правильно організувати написання технічного завдання, правильно спроектувати програмну прицизійну систему, а також правильно у практичному розумінні її розробити, тобто сконструювати. Це можливо за умови, якщо у роботі апріорі правильно і конструктивно сформулювати мету роботи, а потім виходячи із формулювання мети сформулювати прикладні завдання, які необхідно вирішити у даній роботі. Для початку конструктивно розглянемо формалізацію мети кваліфікаційної роботи, що тут розглядається.

Мета кваліфікаційної бакалаврської праці (роботи): спроектувати та сконструювати систему для класифікації текстової інформації по категоріях.

Для досягнення формалізованої мети у кваліфікаційній бакалаврській праці потрібно конструктивно підійти до її сегментації на конкретні прикладні завдання для даної праці. Для конструктивності краще розглянути формулювання прикладних

завдань по пунктах. Тому нище практично-прикладні завдання формалізуватимуться по пунктах.

Практично-прикладні завдання кваліфікаційної бакалаврської праці:

1. Спроектуват. та практичн. реалізуватр процес попередньої обробки інформації у текстовому форматі.

2. Спроектуват. та практичн. реалізуват. процес класифікації (категоризації) інформації у форматі тексту.

3. Спроектуват. та практичн. реалізуват. процес тестування розроблювальної прицизійної системи.

4. Спроектуват. та практичн. реалізуват. процес візуалізації та виведення вихідних даних.

5. Спроектуват. та практичн. реалізуват. в цілому програмну прицизійну систему у вигляді проекту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРИЦИЗІЙНОЇ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ ПО КАТЕГОРІЯХ

2.1 Проектування архітектури прицизійної системи для класифікації текстової інформації по категоріях

Проектування архітектури програмної прицизійної системи для класифікац. текстової інформац. по категоріях розпочнемо з розгляду рисунку, що наведен. нище. Першим блоком в архітектурі є модуль попередньої обробки інформації у текстовій формі, накшталт, видалення стоп-слів, стемінгування тексту, витягування окремих уніграмних та біграмних токенів для вхідного масиву розмірністю два.

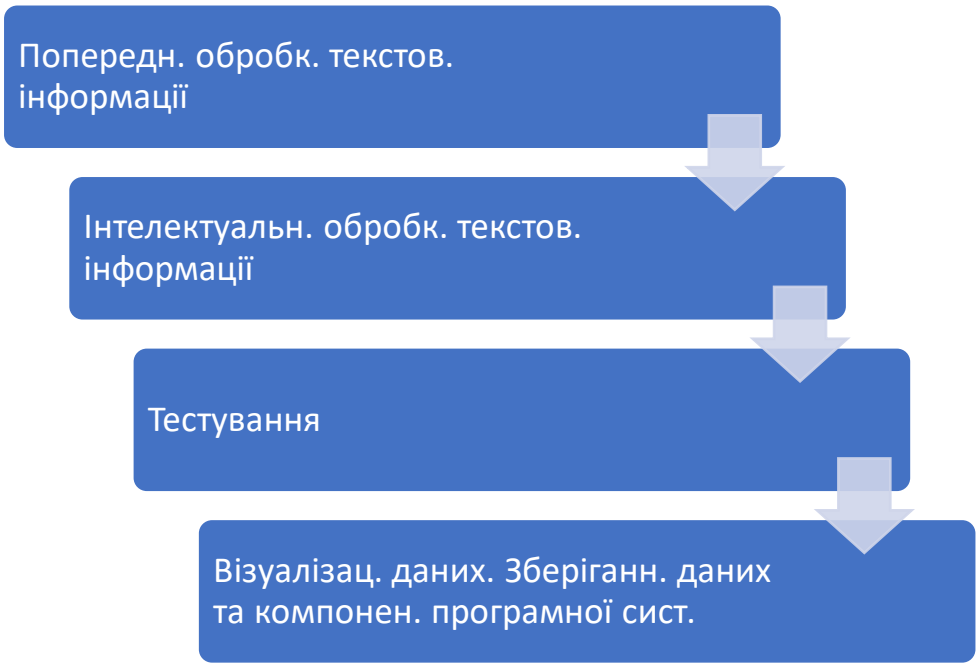

Рис.2.1 – Архітектур. систем. для класифікації текстов. інформац. по категоріях

Другим блоком є інтелектуалізавана обробка тексту, накшталт, використання алгоритму машин. навчан. у вигляді логістичної регресії для бінарної класифікації,

а саме визначення до якої категорії належить книга, до категорії для дітей чи до категорії для дорослихє. Потім йде блок тестування в якому обчислюються різні метрики якості оцінки класифікатора логістичної регресії, накшталт, точність, повнота, площа під кривою. І останнім є блок візуалізації матриць помилок, які дають сприйняття похибки в межах класів; візуалізація даних та зберігання окремих програмних компонент, наприклад, класифікаційний алгоритм – логістична регресія, щоб його могли використовувати сторонні програми.

2.2 Деталювання процесу проектування прицизійної системи для класифікації текстової інформації по категоріях

Перейдемо від узагальненої архітектури програмної прицизійної системи до процесів її деталювання, щоб всі етапи були наочніше та конструктивніше зрозумілими та змістовнішими при процесі прикладного конструювання системи для класифікації текстової інформації по категоріях.

2.2.1 Проектування програмної компоненти для читання даних пандас

Першим програмним компонентом, який розглдатиметься буде компонент бібліотеки пандас, а саме, це метод для читання даних із файлу даних. Використанням програмн. інструмента read_csv. Даний програмн. інструмент після

читання файлу даних повертає на виході фрем даних, накшталт двовимірного масиву, який має назви колонок у вигляді вхідних змінних, а також має назви рядків, що представляють собою моменти часу із кроком дискретизації один місяць і які представляють собою їх рядкові індекси. Даний програмний інструмент read_csv бібліотеки пайтон має наступні атрибути, що наведені нище:

```
pandas.read_csv(
filepath_or_buffer, *, sep=_NoDefault.no_default,
delimiter=None, header='infer',
names=_NoDefault.no_default,
index_col=None, usecols=None,
dtype=None, engine=None,
converters=None, true_values=None,
false_values=None,
skipinitialspace=False,
skiprows=None, skipfooter=0,
nrows=None, na_values=None,
keep_default_na=True,
na_filter=True,
verbose=_NoDefault.no_default,
skip_blank_lines=True, parse_dates=None,
infer_datetime_format=_NoDefault.no_default,
keep_date_col=_NoDefault.no_default,
date_parser=_NoDefault.no_default,
date_format=None, dayfirst=False,
cache_dates=True, iterator=False,
chunksize=None, compression='infer',
thousands=None, decimal='.',
lineterminator=None, quotechar='"',
quoting=0, doublequote=True,
escapechar=None, comment=None,
encoding=None, encoding_errors='strict',
dialect=None, on_bad_lines='error',
delim_whitespace=_NoDefault.no_default,
low_memory=True, memory_map=False,
float_precision=None, storage_options=None,
dtype_backend=_NoDefault.no_default
)
```

У результаті одержуємо данф у форматф пандасівського фрейму, що наведені на рисунку, який наведений нище. Як видно з цього рисунку одна змінна представляє інформацію у текстовому форматі і знаходиться у другій колонці пандасівського фрейму, а у першій колонці цього ж пандасівського фрейму знаходиться мітка категорії-класу для кого призначеня ці інформації в текстовому форматі, що є книжичками для дитячого читача чи для дорослого читача. Цей пандасівський фрейм можна перевести у обєкт даних бібліотеки numpy. Розглянемо виж пандасівських вхідних даних на рисунку нище.

Index	Category	Text
0	children	the house Jim says he rum ; and as …
1	children	has lived rough and I'll raise Cain…
2	children	Now boy he said take me in to the c…
3	children	thanks to you big hulking chicken-h…
4	children	the empty chest; and the next we ha…
5	children	stood irresolute on the road You ha…
6	children	WE rode hard all the way till we dr…
7	children	same as the tattoo mark Billy Bones…
8	children	IT was longer than the squire imagi…
9	children	treasure Long John Silver unearthed…
10	children	We are so grateful to you for havin…
11	children	I am told said the third man that i…
12	children	to find the one paved with yellow b…
13	children	take away the pole I shall be great…
14	children	Won't you tell me a story while we …
15	children	5 The Rescue of the Tin Woodman Whe…
16	children	to a place where the trees and bran…
17	children	All this time Dorothy and her compa…
18	children	I am going to the Great Oz to ask h…
19	children	measuring the distance carefully in…

Рис.2.2 – Частковий вид фрейм. вхід. даних – текстових документів із мітками категорій (класів)

Упершу чергу розгляньмо тип кодування категоріальної змінної, якою є змінна, що представляє мітки класів. Для цього використовується клас із скітлерну

class sklearn.preprocessing.**LabelEncoder**

та його метод

fit_transform(*y*)

Архіважливим моментом є виконання попередньої прицизійного опрацювання даних, що є вхідними. Розглянемо нище конвеєк цього процесу попереднього опрацювання інформації.

2.2.2 Апріорне опрацювання інформації у форматі тексту

Апріорне опрацювання інформації у форматі тексту є надзвичайно важливим у роботі програмної прицизійної системи для класифікації по категоріях, оскільки вона суттєво впливає на точність та стійкість роботи прицизійної системи. Конвеєрний процес апріорного опрацювання інформації у форматі тексту переводить текст природною мовою у формат, зручний для подальшої роботи. Апріорне опрацювання інформації у форматі тексту складається з різних етапів, які можуть відрізнятися залежно від завдання та реалізації. Нижче наведено один можливий набір кроків:

- видалення дублікатів-рядків із даних у форматі пандасівського фрейму – процес дедублікації стрічок;

- трансформація усіх знаків в один розмір регістра, тобто трансформація літер у інформації в форматі тексту в нижній регістр або верхній регістр;

- видалення знаків із апріорі заданої абетки довільної інтерпритації абетки-словника, тобто видалення на основі заданого набору символів;

- процес токенізації на різних рівнях, тобто токенізація на рівні слів чи токенізація на рівні речень, а можлива і токенізація на рівні символів. Окрім різнорівневої токенізації диференціюється токенізація на уніграмний вид, біграмний вид, тощо. На практиці зазвичай використовується уніргамна токенізація та до неї інколи додається біграмна токенізація;

- усунення стоп-слів із інформації у форматі тексту для довільної мови;

- стемінгування інформації у форматі тексту – приведення слів-токенів до їх кореня і цим зменшення затрат обчислювальних ресурсів;

- лематизація інформації у формі текту, тобто трансформація токенів-слів до нормальної форми, яка є для кожної частини мови своя.

Для кращого представлення у конвеєрі апріорного апрацювання інформації у форматі тексту можна розглянути деталізованіший вербальний опис та візуалізацію картинкою. Першою операцією апріорного опрацьовуючого конвеєра операцій буде операція – видалення однакивих стрічок – дедублікація, видаляє надмірні копії стрічок і суттєво зменшує запити до ємності памяті. Візуалізація наведена на наступному рисунку.

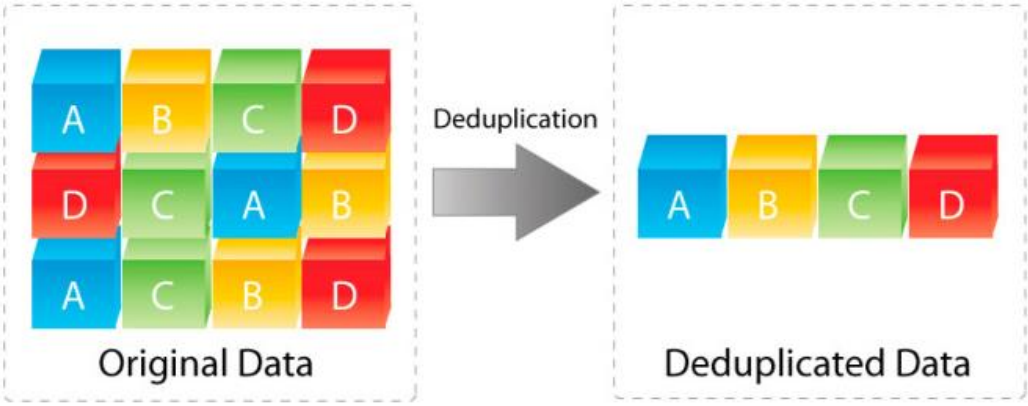

Рис.2.3 – Візуалізація процедури дедублікації стрічок

Наступною операцією у апріорному опрацьовуючому конвеєрі інформації у форматі тексту є операція трансформація усіх знаків в один із розмірів регістра, тобто перетворення всіх знаків у формат нижнього регістра або у формат верхнього регістра це часто зображується нрисунком наступного виду, який наведено далі по тексту.

Find Something	FIND SOMETHING	find something
Frequently Apps	FREQUENTLY APPS	frequently apps
Applications Closed	APPLICATIONS CLOSED	applications closed
Longer Than	LONGER THAN	longer than
Switch Off	SWITCH OFF	switch off
Commonly Used	COMMONLY USED	commonly used
Flashlight Open	FLASHLIGHT OPEN	flashlight open

Рис.2.4 – Візуалізація трансформації знаків абстрактної абетки у один регістр

Далі у конвеєрі апріорного опрацювання інформації у форматі тексту йде операція видалення знаків із довільного апріорного набору, що зображено на рисунку, який наведено нище для візуалізації даної операції.

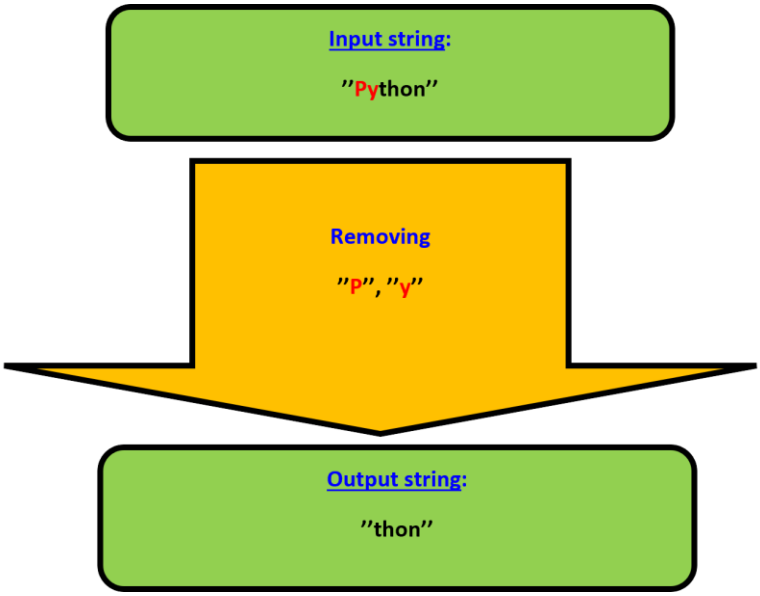

Рис.2.5 – Усунення знаків абстрактної абетки

Важливою операцією у конвеєрі апріорного опрацювання інформації у форматі тексту є правопис слів. Правопис слів також дає змогу часткомо зменшити витрати обчислювальних ресурсів. Правопис слів реалізується на основі підрахування відстані Левенштейна і словника слів. Слід сказати, що існує ціла низка пайтонівських бібліотек для реалізації правопису на онсові відстані Левенштейна. На рисунку, що наведений далі наведено візуалізацію прикладу виконання пайтонівською бібліотеко призначеної для правопису.

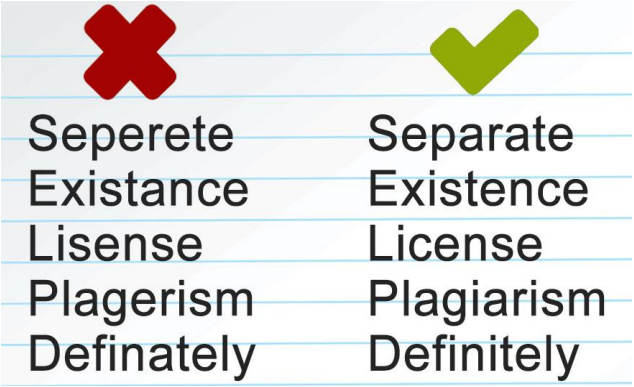

Рис.2.6 – Правопис на основі пайтонівської бібліотеки

Наступною операцією у конвеєрі апріорного опрацювання інформації у форматі тексту є операція токенізація. Для проектування програмної компоненти варто розглянути ряд наступних рисунків, що візуалізують на різних рівнях процес токенізації, фрагментації тексту на різного виду токени.

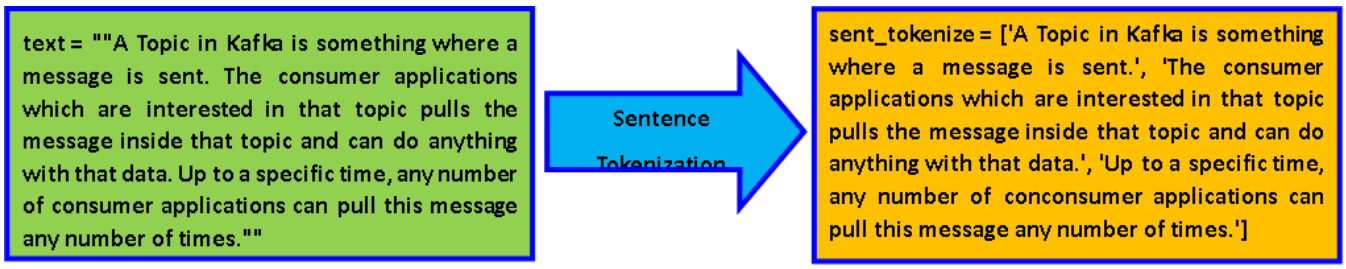

Рис.2.7 – Токенізація-фрагментація на рівні окремих висловлювань

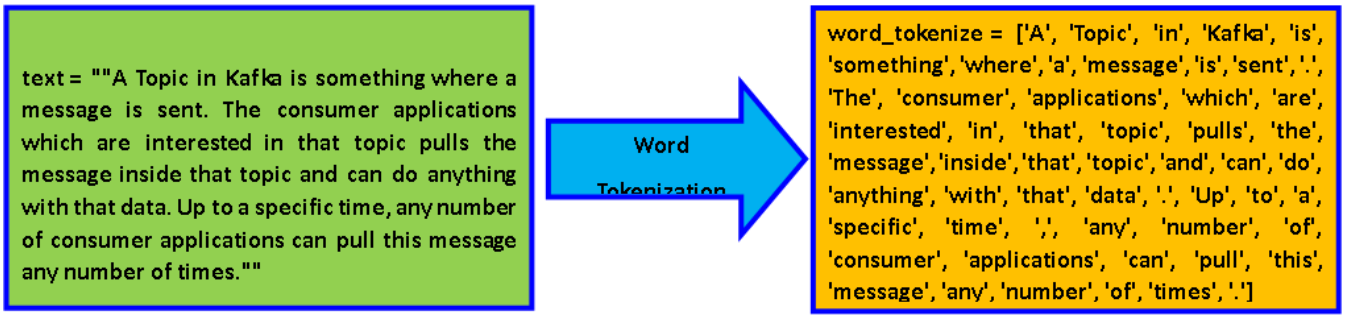

Рис.2.8 – Токенізація-фрагментація на рівні окремих слів

Для прикладу розгляньмо типовий етап токенізації-фрагментації слів, який фрагментує речення на окремі слова-токени, усуває пунктуаційні знаки і трансформує усі літери-знаки на форму малого регістру, як наведено на слідоючому рисунку, що наведений далі по тексту.

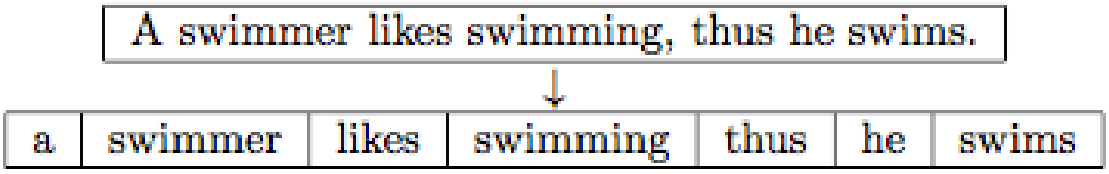

Рис.2.9 – Токенізація-фрагментація на рівні слів

Усунення стоп-слів довільної мови. Стоп-слова довільної мови – це є ті слова, які зустрічаються з високою частотою у інформаційному повідомленні представленому у форматі тексту. Тому такого роду слова вважаються не інформативними або слабоінформативними і немає нічого проблемного у тому аби їх видалити. До таких слів чи символів відносяться, наприклад, "та", "або", тощо. У пайтонівських бібліотеках які призначені для опрацювання інформації у форматі тексту містяться словники такого роду спів чи символів для різних мов і після операції токенізації-фрагментації проходить перевірка окремих токенів зі словниками цих бібліотек для відповідних природних мов. При проектуванні програмних особливо прицизійних систем для класифікації текстів можна

програмним шляхом доповнювати до словників, що містяться у пайтонівських бібліотеках свої слова чи символи і цим самим нарощувати – збагатчувати токени-фрагменти новими малоінформативними знаками.

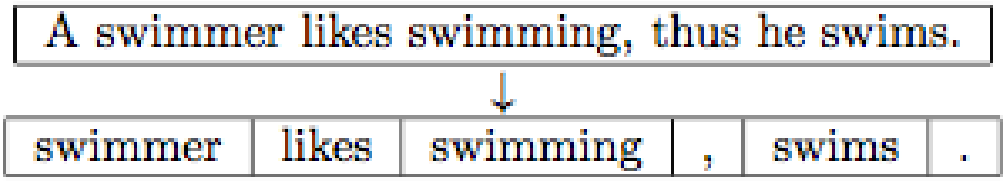

Рис.2.10 – Токенізація-фрагментація

Наступною операцією у конвеєрі апріорного опрацювання інформації у форматі тексту є стемінгування тексту. Стемінгування тексту - це технологія відшукання кореня-основи слова для кожного токена-слова. Стемінгування тексту може виконуватися до прикладу методом вилучення змінних часток слова: часток суфіксів, часток префіксів, часток закінчень слів, тощо. Технологія стемінгування базується на правилах морфології тієї чи іншої мови. Дуже часто пайтонівські бібліотеки стемінгування виконують з значною кількістю помилок. Приклад стемінгування наведено на наступному рисунку, що йде по тексту.

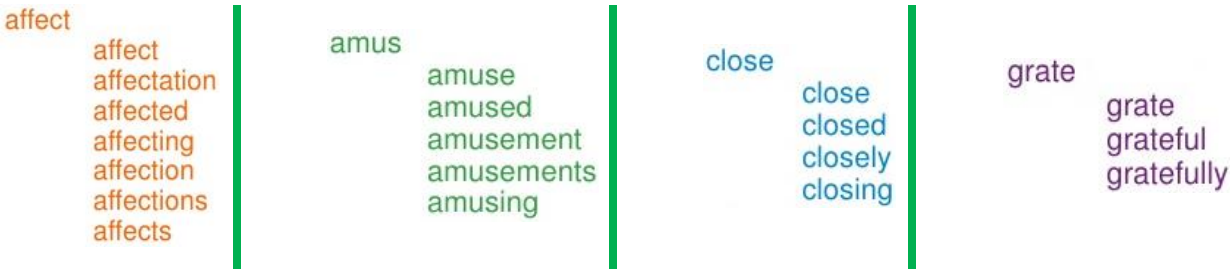

Рис.2.11 – Стемінгування інформації у форматі тексту, зокрема слів

Проектування операції стемінгування тексту у роботі програмної прицизійної системи організовується на основі класу бібліотеки дженсім

Class `gensim.parsing.porter.PorterStemmer`

та його методу

`stem_documents(x)`

Наступною, як правило, операцією у конвеєрі априорного опрацювання інформації у форматі тексту є використання операції лематизації. Лематизація представляє собою трансформацію слів інформації у форматі тексту до нормальної форми. Наприклад, для іменників нормальна форма слова означає трансформацію слова в називний відмінок, число – однину та чоловічий рід; для дієслова – це трансформація в форму інфінітива; і так далі для всіх слів інших частин мови. Це робиться для того аби зменшити затрати обчислювальних ресурсів та здешевити розробку даної прицизійної системи. Додатково загостримо увагу, що лематизація працює для слів різних частин мови: іменників, прикметників, дієслів, тощо. Нище на рисунку по тексту наведено приклад, яким чином виглядає процес лематизації.

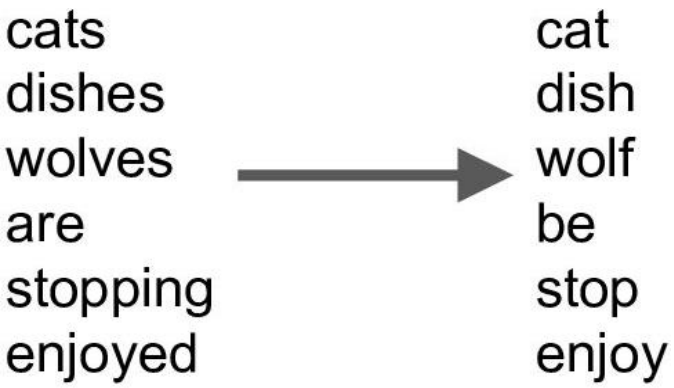

Рис.2.12 – Приклад процесу лематизації

Для того, щоб більш наочно бачити різницю між процесами стемінгом та лематизацією, розглянемо наступний рисунок з якого явно це видно.

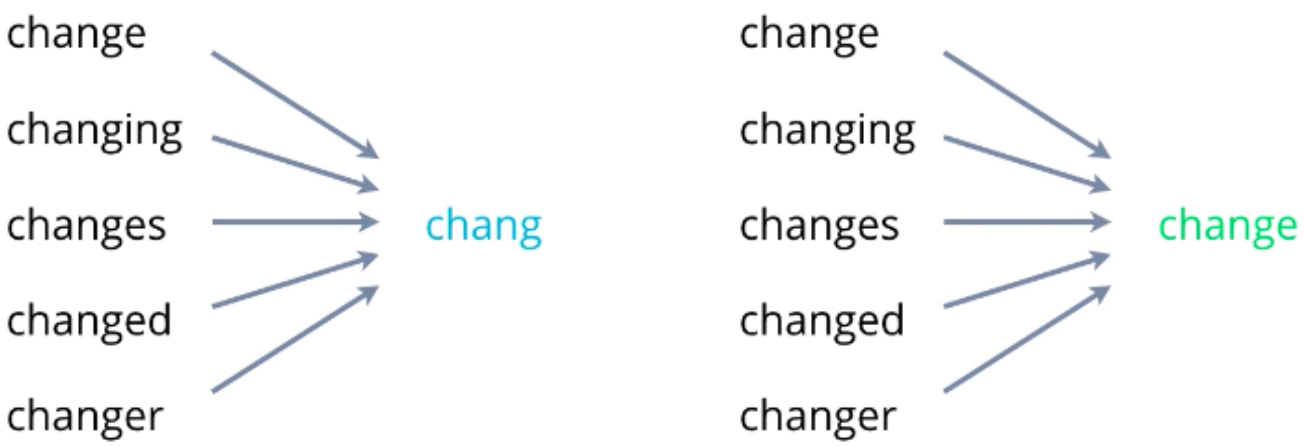

Рис.2.13 – Приклади стемінгування та лематизації

Зазначимо, що саме в українській мові нормальним формами для різних частин мови є конкретні дифініції, а саме, для іменників – називн. відмінок, однин, для прикметників — називн. відмінок, однин, чоловіч. рід, для дієслів і дієприкметників - дієслово у формі інфінітиву.

Повернемося до процесу токенізації і детальніше розглянемо тлумачення грамності токенів. N-грами – це лексеми, які можуть бути означені типу лексеми, а точніше послідовність токенів – це послідовність лексем кількістю n елементів. Найтривіальнишим прикладом є поняття

уніграми (1-грами), коли кожен токен – це виключно одне слово або літера, чи довільний абстрактний знак. Селекція оптимальної кількості n залежить від тагатьох факторів і характеристик конкретної мови і від фіксованої програми. Інколи до словника включають не всі унікальні лексеми, а використовується деяка їх сукупність. Нище на рисунку наведено приклади 1-грами, 2-нрами (біграми), 3-грами (триграми) і загалом n-грами. Звісно, що такого роду моделі представлення слабоструктурованих текстових даних не є ідеальною трансформацією до структурованого виду, але і цей вид формальної моделі структуризації тексту дає доволі часто багато якісних результатів при вирішенні різного роду прикладних

завдань практики, накшталт, тональність текстових повідомлень – постів у соціальних мережах, тощо.

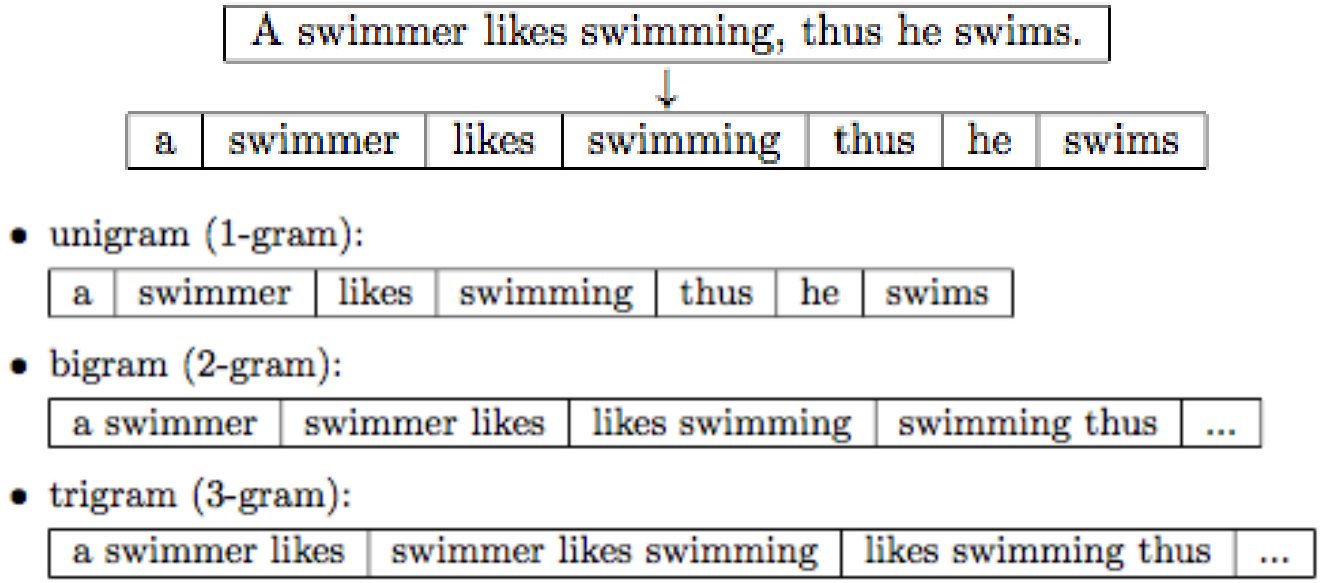

Рис.2.14 – Приклади грамності токенів

Для представлення символьного рівні різної грамності токенів зручно розглянути рисунок, який наведено нище і це демонструє явно (уніграми (1 символ), біграми (2 символи), триграми (3 символи).

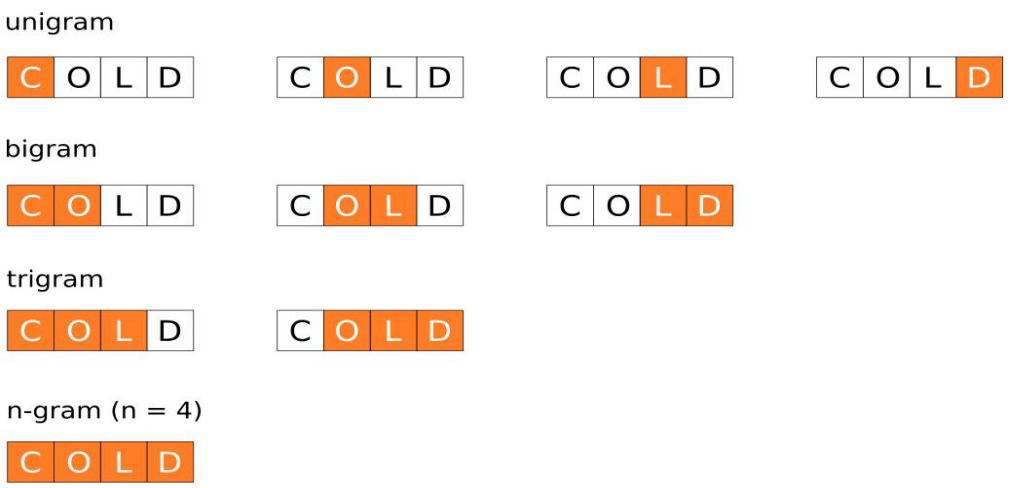

Рис.2.15 – Символьні токени

2.3 Проектування трансформаційної структуризації текстової інформації

При проектуванні програмної прицизійної системи для класифікації інформації у формі тексту важливим моментом є модель представленн вхідної текстової інформації. У дані роботі використовується модель представлення тексту, яка називається сумка слів або мішок слів. На рисунку нище наведено масив у вигляді матриці документ-терам матриці.

	it	is	puppy	cat	pen	a	this
it is a puppy	1	1	1	0	0	1	0
it is a kitten	1	1	0	0	0	1	0
it is a cat	1	1	0	1	0	1	0
that is a dog and this is a pen	0	2	0	0	1	2	1
it is a matrix	1	1	0	0	0	1	0

Рис.2.16 – Масив вигляді матриці документ-терм

Нище наведена декартова система координат представлення кожного окремого документа корпусу у вигляді вектора у просторі розмірністю в унікальний словник.

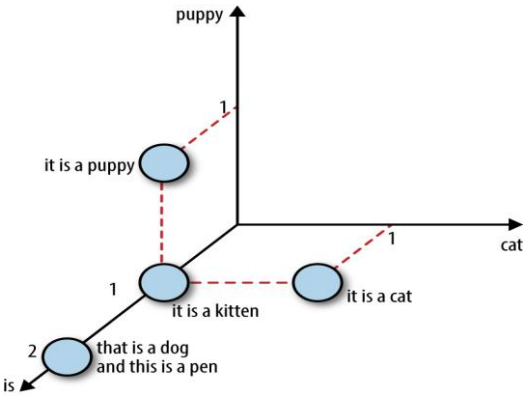

Рис.2.17 – Декартова система координат представлення документів корпусу

Проте, досить часто, коли використовуються ті чи інші програмні інструменти вони на виході повертають не формат даних узгоджений із входом, а повертають транспонований масив, накшталт, того, що наведений на рисунку нище.

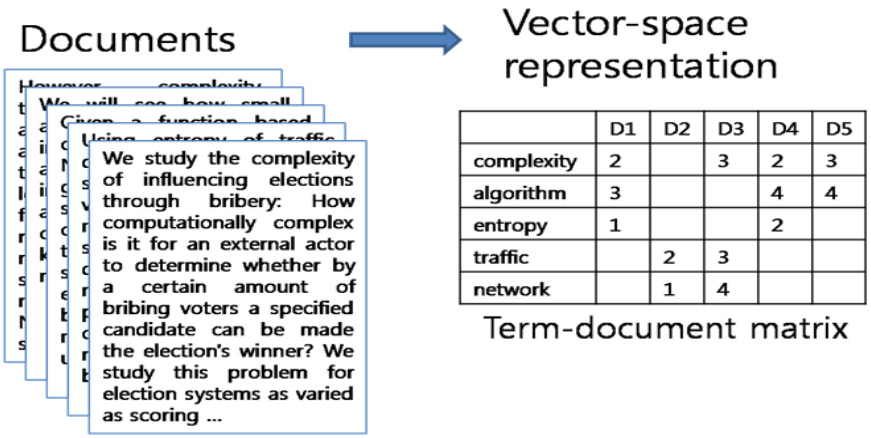

Рис.2.18 – масив у вигляді матриці терм-документ

Коли вираховуються значення для імовірностей входження токенів у документи, то застосовують різного роду математичні трансформації, наприклад, використовують, щоб зосередити увагу алгоритма на значимих словах ьрансформацію TF-IDF. Дана трансформація обчислюється із послідовного застосування таких математичних перетворень:

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

де n_t - це є кількість появи слова у t документі, а у знаменнику цього дробу - загальна кількість слів у цьому документі. Зворотня частота фіксованого документу

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|}$$

$|D|$ - число документів у сукупності, $|\{d_i \in D | t \in d_i\}|$ - число документів у сукупності D , у яких наявний терм t (коли $n_t \neq 0$).

Ну і врешті саме перетворення TF-IDF, тобто частота терму є оберненою до частоти документу:

$$tf - idf(t, d, D) = tf(t, d) \times idf(t, D).$$

Більшу вагу мають терми з високою частотою появи у документі текстової інформації та з малою частотою появи терму в інших докуметнах сукупності.

2.4 Проектування наступних програмних компонент системи

Проектування обчислювального процесу для добування ознак з корпусу текстових документів – токенізації виконується на основі класу

```
class sklearn.feature_extraction.text.TfidfVectorizer(
                                    *,
                                    input='content',
                                    encoding='utf-8',
                                    decode_error='strict',
                                    strip_accents=None,
                                    lowercase=True,
                                    preprocessor=None,
                                    tokenizer=None,
                                    analyzer='word',
                                    stop_words=None,
                                    token_pattern='(?u)\b\w\w+\b',
                                    ngram_range=(1, 1),
                                    max_df=1.0, min_df=1,
                                    max_features=None,
                                    vocabulary=None,
                                    binary=False,
                                    dtype=<class 'numpy.float64'>,
                                    norm='l2',
                                    use_idf=True,
                                    smooth_idf=True,
                                    sublinear_tf=False)
```

Формування навчального словника усіх лексем у не оброблених слабоструктурованих даних - текстових документів виконується на основі методу підгонки або апроксимації:

```
fit(x)
```

перетворення текстової інформації – текстових документів у матрицю термінів документа проектується на основі методу:

```
transform(x)
```

Класифікатор корпусу текстових документів по категоріям реалізуватиметься на класі:

```
class sklearn.linear_model.LogisticRegression(
                                penalty='l2',
                                *,
                                dual=False,
                                tol=0.0001,
                                C=1.0,
                                fit_intercept=True,
                                intercept_scaling=1,
                                class_weight=None,
                                random_state=None,
                                solver='lbfgs',
                                max_iter=100,
                                multi_class='auto',
                                verbose=0,
                                warm_start=False,
                                n_jobs=None,
                                l1_ratio=None)
```

та його методів, що реалізують різний функціонал. Наприклад, метод

```
fit(x)
```

реалізує навчання класифікатора на корпусі текстових документів. Настпуний метод – це є метод прийняття рішень класифікатором:

```
predict(x)
```

який реалізує процес чіткого прийняття рішень класифікатором. А також метод

```
predict_proba(x)
```

який реалізує процес м’якого прийняття рішень класифікатором.

Навчання та тестування, зокрема, перехресна перевірка реалізовується на основі класу бібліотеки скітлерн:

```
sklearn.model_selection.cross_val_score(
                                estimator,
                                X,
                                y=None,
                                *,
                                groups=None,
                                scoring=None,
                                cv=None,
                                n_jobs=None,
                                verbose=0,
                                fit_params=None,
                                params=None,
                                pre_dispatch='2*n_jobs’,
                                error_score=nan)
```

У роботі розглядаються різні метрики для оцінювання якості програмних прицизійних систем на основі наведеного вище класу і проектування обчислювального процесу їх значення виконується на основі програмних інструментів бібліотеки скітлерн. Дана пайтонівська функція дає змогу оцінювати якість роботи системи з різних метричних сторін в цьому і полягає її велика цінність, саме тому вона використовується у проектуванні системи, що розглядається у цій роботі бакалавра. Загалом потрібно відзначити велику цінність пайтонівської бібліотеки скітлерн у розробці такого роду систем. Одже, код має вид такий:

```
sklearn.metrics.accuracy_score(
y_true,
y_pred,
*,
normalize=True,
sample_weight=None)
```

```
sklearn.metrics.f1_score(
y_true,
y_pred,
*,
labels=None,
pos_label=1,
average='binary',
sample_weight=None,
zero_division='warn')

sklearn.metrics.roc_auc_score(
y_true,
y_score,
*,
average='macro',
sample_weight=None,
max_fpr=None,
multi_class='raise',
labels=None)
```

Реалізація процесу розщеплення вхідних даних на навчальну вибірку та тестову вибірку проектується на основі функції бібліотеки скітлерн:

```
sklearn.model_selection.train_test_split(
*arrays,
test_size=None,
train_size=None,
random_state=None,
shuffle=True,
stratify=None)
```

Для наочності відображення процесу оцінювання точності використовується функція бібліотеки скітлерн для обчислення функції помилки у вигляді матриці помилок, що дає змогу бачити точність роботи проектованої системи в межах класів:

```
sklearn.metrics.confusion_matrix(
y_true, y_pred,
*,
labels=None,
sample_weight=None,
normalize=None)
```

а для її візуалізації проектування використовує програм. інструмен. – функцію бібліотеки скітлерн. Цей інструмен дає змогу явно візуалізувати дану матрицю:

```
sklearn.metrics.plot_confusion_matrix(
estimator,
X,
y_true,
*,
labels=None,
sample_weight=None,
normalize=None,
display_labels=None,
include_values=True,
xticks_rotation='horizontal’,
values_format=None,
cmap='viridis’,
ax=None,
colorbar=True)
```

На цьому завершується проект проектування процесу розробки програмної прицизійної системи для класифікації на основі апріорного конвеєра інформації у форматі тексту та переходиться до наступного роздулі роботи – це практичне конструювання розглядуваної у праці системи.

РОЗДІЛ 3. ПРАКТИЧНЕ КОНСТРУЮВАННЯ ПРИЦИЗІЙНОЇ СИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ ПО КАТЕГОРІЯХ

3.1 Ввід вхідних даних у програмну прицизійну систему класифікації текстової інформації

Перед початком опису практичного конструювання програм. системи для класифікації текстової інформації по категоріях апріорі виконується імпортування всіх необхідних бібліотек та фреймворків із дистрибутиву анаконди, що призначена для технології програмування пайтон. До основних фреймворків та бібліотек, що використовуються, а тому імпортуються у програмний скрипт відносяться такі: пандас – це бібліотека загалом для різного роду маніпуляцій даними, накшталт, датафрейми, серії даних та інші. Проте у даній роботі вона використовується для введення вхідних даних з файлу із розширенням csv. Наступною бібліотекою, яка використовується у скриптах прикладного проекту є бібліотека також для різного роду маніпуляцій з даними, але вже для маніпуляцій з даними типу звичайних багатовимірних масивів. Особливо важливою бібліотекою для імпортування у скрипт є бібліотека для обробки природньої текстової інформації є дженсім. Ця бібліотека використовується для ситуації стемінгу, тобто тоді коли потрібно привести всі текстові документи корпусу до їх коренів. Це потрібно для того аби зменшити кількість колонок у вхідному двовимірному масиві даних – документ-терм матриці, зменшити кількість колонок. Це дає змогу зменшити затрати обчислювальних ресурсів: памяті, процесорного часу, а значить здишевити розроблювану програ. сист. Тому ця бібліотека дженсім імпортується у скрипт разом із класом стемер. Особливо у даній розроблювальні роботі є використання бібліотеки скітлерн, багато її окремих модулів, класів та різного роду методів. Фактично ця бібліотека скітлерн є домінуючою у скриптах конструйованої прогрм. сист. Окремим записом імпортуються такі наступні класи з бібліотеки скітлерн – це

кодувальник міток, щоб перетворити цільову змінну у цілочисельну бінарну змінну, клас для токенізації текстової інформації на уніграмні та біграмні токени, що формують словник унікальних слів для утворення документ-терм матриці. Окрім того цей клас видаляє стоп-слова і багато інших операцій. З бібліотеки скітлерн імпортується у скрипт алгоритм машинного навчання, як програмний клас – це логістична регресія, оскільки вона з еврістичних практичних даних дає стабільно високої точності результати багатокласової класифікації, а особливо при класифікації на два класи. До того ж цей клас – логістична регресія має багато гіперпараметрів для його тюнингу, тобто налаштуванню.

Наступним етапом є імпортування функції перехресної перевірки алгоритму логістична регресія із значеннями три фолди. На основі цієї функції – перехресної перевірки забезпечується факт незалежності роботи логістичної регресії від розбиття, тому можна говорити про обєктивність оцінювання метрик точності, накшталт, точність, повнота та площа під кривою рок кривої. Окрім цього функція перехресної перевірки дає можливість оцінити стійкість роботи алгоритму логістичної регресії на основі оцінювання середньоквадратичного відхилення тієї чи іншої метрики якості на фолдах. Функція перехресної перевірки має керуючий гіперпараметр у вигляді стрічкової змінної, яка вказує на значення метрики якості: точності, повноти, кривої під криволінійною трапецією – рок кривою.

На наступному етапі відбувається імпортування фун-ції розщеплення на навчальну та тестову сукупності. У цієї функції розщеплення на навчальну та тестову сукупності є керуючий гіперпарамет. у вигляді вказання частки величини обсягу тестової сукупності. Також дана функція розщеплення має змінні у вигляді вхідних і цільової змінних.

Окремим важливим моментом є імпортування функцій бібліотеки скітлерн – функції точності, функції повноти, функції площі під кривою криволінійної трапеції. Саме ці функції у своїй сукупності – метрики якості оцінюють з різних сторін якість обробки даних алгоритмом логістичної регресії. Ці вище названі функції – метрики

якості у своїй сукупності можна розглядати, як компоненти вектора якості. Це дає змогу обчислити інтегральний показник якості роботи алгоритму логістичної регресії, наприклад, обчисливши норму цього вектора якості, зокрема, наприклад, його модуль в евклідовому просторі. Звісно, що інтегральний показник оцінки якості роботи алгоритму логістичної регресії можна оцінити, як норму вектора якості у довільному іншому виді.

Для візуалізації та наочності представлення точності обробки даних алгоритмом логістична регресія використовуються нормована та не нормована матриці помилок. Ці матриці помилок дають змогу візуально побачити похибки роботи алгоритму логістичн. регрес. в межах окремих класів. І цей вид візуалізації дає змогу бачити диференціальну картину прицизійності. Саме функції – нормована та не нормована матриці із бібліотеки скітлерн дають змогу це організувати. Тому ці функції також імпортуються у пайтонівський скрипт загального проекту розроблювальної системи.

Останньою бібліотекою для імпортування у пайтонівський скрипт є бібліотека джобліб. Ця бібліотека джобліб використовується саме для зберігання машинн. алгоритм. логістична регресія у окремий файл. І тепер цей файл із збереженим попередньо навченою логістичн. регрес. може використовуватися довільними іншими скриптами пайтона і зокрема скриптами розроблювального проекту.

Нище наведено реалізацію імпортування всіх вище перелічених фреймворків, бібліот. та й загалом програмн. компонент. тому розглянемо цей імпорт за допомогою базових функц. пайтона.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from gensim.parsing.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score, accuracy_score
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.externals import joblib
```

Зазначимо, що вхідними даними для спроектованої програмної системи є набір книг, тобто, набір текстових документів – масив стрічкової змінної. файл в якому знаходяться вхідні дані для класифікації – це є текстовий файл з розширенням csv. Щоб прочитати цей файл вхідних даних в нище наведеному фрагменті програмного коду на основі використання бібліотеки Pand. З програмного коду видно, що роздільником даних є знак коми, а роздільником між цілою частиною числа та дробовою частиною числа є по замовчуванню крапка. Метод використовується з позначенням:

```
read_csv
```

Розгляньмо програмну реалізацію вводу – читання вхідних даних.

```
"""Read dataframe"""
df = pd.read_csv("./BOOK-EXCERPTS/BOOK-
EXCERPTS.csv",
                                delimiter=",",
                                 header=0)
```

3.2 Селекція корпусу малого об’єму текстових документів

Розглянемо перші кроки попередньої обробки текстової інформації. Для цього у провіднику змінних переглядаємо містиме змінної типу фрейму даних на вміст змінних та їх тип. З візуалізації прочитаних даних з файлу видно, що першою колонкою фрейму даних є змінна стручкового типу, що представляє собою набір стрічкових даних. саме ці дані і являють собою сукупність документів з текстовою інформацією, тобто представляють собою корпус текстов. документів. Тому можна сформувати вид вхідних даних у вигляді змінної стрічкового типу – це і є кор. текст. документів. Де досягається шляхо написання біля мітки змінної фрейму даних у

квадратних дужках назви змінної, що єявляє сукупність стрічкових значень. Назва змінної записується у лапках, оскільки в них записується назва змінної. В прочитаній змінній це є перша колонка.

Друга колонка прочитаного файлу із набором даних – це є цільова змінна, яка інтерпритується, як категорія до якої належать дані стрічкової змінної у фіксованому рядку. Категорій у цьому завданні є дві – це, перша категорія – книжки для дитячого читача, а друга категорія – це книжки для дорослого читача. Однак, як можна бачити у провіднику даних тип даних цільової змінної є також стрічковим, тому ці стрічкові значення потрібно закодувати у числа, зокрема нуль та один. Для цього з бібліотеки скітлерн береться клас кодування мітки і за допомогою його методу підгонка та трансформація стрічкові значення цільової змінної закодовуються у цілочисильний тип даних із значеннями відповідно нуль та один. Клас кодування мітки апріорі формує словник унікальних значень цільової змінної, а потім їх занумеровує. Оскільки в даних міститься два унікальні стрічкові значення цільової змінної, то клас кодувальник мітки перетворить їх у значення нуль та один. Тому з означення того, що твірна значень – обсяг словника унікальних слів цільової змінної має два елементи, то у даній роботі завдання набуває виду завдання бінарної класифікації. Клас кодування мітки та його метод підгонка із трансформацією застосовуються до фрейму прочитаних даних із наступним синтаксисом, біля змінної, що символізує фрейм даних у квадратних дужках записується назва змінної у лапках, оскільки назва є стрічковою змінною. Цим самим ми одержуємо вихідні дані для подальших дій, зокрема, для пожальшого навчання алгоритм. логістичної регресії та її тестування. Програмна реалізація вище описаних процедур наведена нище. Для кожної дії - операції наведені детальні коментарі для наочності розумінні прогр. коду і зручності подальшого використання.

```
"""Splitting independent 'x' and dependent 'y'  variables of dataframe 'df'"""
x = df['Text']
#Initialization of encoder
labelencoder = LabelEncoder()
#Encoding target variable
y = labelencoder.fit_transform(df['Category'])
```

Розгляньмо наступні етапи коду, які використовуються у скриптах розроблювального проекту, що представляє собою систему для класифік. Текстов. інформац. по категоріях.

3.3 Операція стемінг корпусу малого об’єму текстових документів

У цьому підрозділі буде описано процедуру стемінгу корпус. текстов. документів. У програмі використовується бібліотека NLTK. Зокрема, використовується її клас ПостерСтемер для стемінгу текстових документів із корпусу. Тому у програмному коді після імпортування класу ПортерСтемер виконується його ініціалізація. Проте, текстові документи із корпусу представляються обєктоп пандаса Серіас, тому щоб застосувати клас ПортерСмер апріорі змінну, що представляє текстов. документи із корпусу потрібно перетворити у список стручкової змінної і лише тоді можна застосувати клас ПортерСтемер. За допомогою стандартної пайтонівської функції Ліст обєкт даних бібілотеки Пандас – а саме Серіал перетворюється у список стрічкової змінної. Ось так одержується список документ. текстової інформації, що представляє собою корпус.

Наступним рядком коду є використання класу ПостерСтемера на основі його методу СтемінгДокументів. Даний метод, СтемінгДокументів повертає список токенів після операції стемінг. Нагадаймо, що операція стемінг переводить слова-токени документів до кореня слів. Це дає цілий ряд переваг. Наприклад, першою перевагою є те, що зменшується загальна кількість токенів, тому як наслідок у подальшому затрачатиметься менше обчислювальних ресурсів: менше ресурсів памяті, менще ресурсів процесорного часу, а значить це здешевлює розробку програм. системи, а значить підвищує її рентабельність. Для наочності, наглядності

розуміння нище наведено прокграмна реалізація фрагменту пайтонівського скрипта, який реалізує процедуру стемінгу на основі класу ПортерСтемінг бібліотеки NLTK.

```
"""Stemming"""
#Initialization
stem = PorterStemmer()
#Converting to list
x = list(x)
#Stem documents
x = stem.stem_documents(x)
```

Наступним кроком у роботі є програмування етапу процесу токенізації набору текстової інформації у вигляді документів текстов. інформації, що представляють собою корпус даних.

3.4 Процес токенізації корпусу малого об’єму текстових документів

Токенізація набору текстової інформації, що представлена документами корпусу вхідних даних є одни із важливих кроків у попередній обробці даних, оскільки вона реалізує багато важливих операцій попереднього опрацювання даних. Особливо це важливо у фрагменті конструювання коду пайтонівського скрипта, оскільки в один рядок коду реалізовується цілий пул операцій, які суттєво впливають на якість роботи цілої програм. системи.

Токенізація може бути різною по набору токенів. Наприклад, може бути уніграмна токенізація, де токенами є окремі слова з текстової інформації. Може бути біграмна токенізація, коли токенами є пари слів текстов. Інформації, такаго роду біграмна токенізація дає змогу у подальшому алгоритму виловлювати усталені ідіоми, які складаються з пари слів і мають у своїй парі єдину семантине значення.

Також може бути триграмна токенізація, яка аналогічно до вище наведених токенізацій представляє собою токени, що містять усі варіанти трійок слів, а тому така триграмна токенізація може врахувати усталені ідіоми, що складаються з трійок слів і кожна трійка має єдине семантичне значення. Відповідно можна уявити собі подальші види, загалом, н-грамних токенізацій, які можуть враховувати в рамка одного токена ідіоми, що складаються з більшої кількості слів. Як правило на практиці використовують найчастіше не більше ніж два види токенізації – уніграмну токенізацію та біграмну токенізацію. Тому у даному кодовому рішенні також використано уніграмну токенізацію та біграмну токенізацію на рівні слів. Заради інформативності відзначимо, що у роботі для подальшого розвитку розроблювальної системи можна буде використати токенізації не лише на рівні слів, але і токенізації: уніграмну токенізацію та біграмну токенізацію на рівні символів. Такого роду символьний рівень токенізації дає змогу виловлювати машині емоційний рівень подачі текстової інформації і він може бути значимим фактором для категоризації книжок відповідно для дітей чи дорослих. Але у даній роботі автором було прийняте рішення використовувати лише уніграмну токенізацію та біграмну токенізацію на рівні слів.

Клас Токенізація із бібліотеки скітлерн також підтримує багато природних мов, зокрема і мову, яка наявна в текстовій інформації документів із корпусу. Наявна мова є англійською мовою. Вибір англійської мови у класі Токенізація із бібліотеки скітлерн досягається завдяки керуючому гіперпараметру цього класу, а саме, завдяки керуючому гіперпараметру стоп-слова. Значення цього гіперпараметру є стрічковою змінною, яка задає англійську мову. Окрім цього, цей же гіперпараметр керує відключенням або включенням стоп-слів англійської мови. Його значення за замовчуванням виконує функцію не відключення стоп-слів. Але автором даної роботи було прийняте рішення підключити цей гіперпараметр, щоб машина – клас токенізація видалив стоп-слова з текстової інформації, оскільки вони не несуть значимої семантичної інформації і лише творюють надлишкові лишні затрати ресурсів памяті, затрати процесорного часу та й загалом знижують точність у

подальшому категоризації текстової інформації, що міститься у документах із корпусу. Тому даний керуючий гіперпараметр був активізований і увімкнений в стрічкове значення, що вказує на англійську мову.

Далі у класі Токенізація був використаний метод підгонка, або його ще називають навчанням, оскільки вище була описана процедура ініціалізації класу Токенізація. Після того використовувався метод трансформації, який вже безпосередньо перетворив список текстової інформації у вигляді документів із корпусу в список токенів.

Після того використовувалася операція перетворення вихідних даних після класу Токенізація у звичайний масив даних, як об’єкта бібілотеки numpy.

Нище наведено кодову реалізацію вище описаного.

```
"""Feature extraction from text"""
#Initialization
tfidf = TfidfVectorizer(stop_words='english')
#Learning dictionary of all tokens in the raw documents
tfidf.fit(x)
#Transform documents to document-term matrix
x = tfidf.transform(x)
x = x.toarray()
```

Далі по тексту піде мова про ініціалізацію логістичної регресії та її перехресної перевірки.

3.5 Перехресна перевірка класифікатора логістичної регресії

Цей підрозділ присвячений конструювання фрагменту пайтонівського скрипта, який реалізує наступні кроки.

Першим кроком у цьому фрагменті коду виконується ініціалізація алгоритму маш. навчан. логістичної регресії із бібліотеки скітлерн із гіперпараметрами тим, що є по замовчуванню. Фрагмент даної ділянки коду наведено нище.

```
"""Model cross-validation"""
#Initialization
lg = LogisticRegression()
```

Далі опишемо кодування процесу перехресної перевірки. Тому як буде видно у наведеному нище фрагменті пайтонівського коду – перехресна перевірка реалізовується на основі функції бібліотеки скітлерн, а саме модуля селекція моделей – функція ПерехреснаПеревірка. Дана функція має два аргументи у вигляді вхідних даних, які є текстовою інформацією документів із корпусу, що апріорі були піддані процесу токенізації класом Токенізація та вихідних даних, якими є цільова змінна, що представляє собою категорії – клас дитячих книжок та клас книжок для дорослих. Функція ПерехресноїПеревірки має ряд керуючих гіперпараметрів, а саме є такі настіпні гіперпараметри. Гіперпараметр, що задає кількість фолдів у процедурі перехресної перевірки. Якщо його взяти великим значенням, то буде більш обєктивне значення результату перевірки, але при цьому будуть значні затрати обчислювальних ресурсів: процесор. часу та памяті. Якщо взяти значення гіперпараметру, який відповідає за перехресну перевірку малим значенням, тоді не можливо гарантувати, що якість роботи алгоритму логіст. регресія не залежить від розбиття даних на навчальну та тестову тукупності. Тому у цій роботі було взято значення кількості фолдів по замовчуванню, а воно дорівнює три. Тому процедура перехресної перевірки робилася три рази і на основі них обчислювалося середнє значення того чи іншого виду точності.

Наступним гіперпараметром який використовувася у цьому класі ПерехресноїПеревірки – це є стрічкова змінна, що вказує на назву метрики точності, яка буде обчислюватися цим класом ПерехресноїПеревірки. Нагадаємо, що з попереднього розділу при проектуванні системи для категоризації книжок для дітей та книжок для дорослих використовувалися три метрики якості, а саме, метрика

якості – точність, точність з повнотою та точність на основі площі під кривою рок. Можна реалізувати інтегральну метрику, як інтегральний опис точності роботи системи. Нище наведено фрагменти пайтонівського коду вище описаних речей.

```
#Compute the 'accuracy' by cross-validation
accuracy = cross_val_score(lg, x, y, cv=3, scoring='accuracy').mean()
#Compute the 'f1-score' by cross-validation
f1 = cross_val_score(lg, x, y, cv=3, scoring='f1_weighted').mean()
#Compute the 'roc-auc' by cross-validation
roc_auc = cross_val_score(lg, x, y, cv=3, scoring='roc_auc_ovr_weighted').mean()
```

У подальшому підрозділі даного розділу розглянемо процедуру розщеплення даних текстової інформації на навчальну сукупність та тестову сукупність.

3.6 Розшеплення даних на навчальну та тестову сукупності

Розгляньмо наступний фрагмент програм. коду для процесу розщеплення вхідних даних на навчальну та тестову сукупності. Це досягається за допомогою функції пайтонівської бібілотеки скітлерн та її модуля селекція моделей у який входить цілий ряд інших пайтонівських функцій для розбиття даних спостережень.

У цієї функції розщеплення на навчальну та тестову сукупності є керуючий гіперпарамет. у вигляді вказання частки величини обсягу тестової сукупності. Також дана функція розщеплення має змінні у вигляді вхідних і цільової змінних. Реалізація у програм. коді має наступний вигляд.

```
"""Splitting on training sample and test sample"""
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

Також дана функція має гіперпараметр, який дає змогу керувати обсягом тестової сукупності даних.

Згідно вище наведеног фрагменту пайтонівського коду видно, що обсяг тестової сукупності складає двадцять пять відсотків. Звісно ним можна було би керувати, але, оскільки еврістичні дані вказують на таке значення, то автором було прийняте рішення не змінювати гіперпараметр обсягу тестової сукупності даних.

3.7 Навчання та прийняття рішень класифікатором

Після ініціалізації алгоритму логіст. регрес. з пайтонівської бібліотеки скітлерн далі перейдемо до процедури конструювання процесів навчанн та прийняття рішень. І першим розглянемо кодове конструювання процесу навчання. Для цього використовується через крапку від мітки класифікатора метод пігонки, який на практиці реалізує процедуру навчання логістич. регресії.

Процес прийняття рішень кодово конструюється за допомогою двох різних методів, першим є метож прогнозування, що визначає на даних апроксимовані значення міток класів, якими є значення цільової змінної. Це є так звана чітка класифікація класифікатора. Проте нище наведеному кодовому фрагменті також представлено так звану нечітку класифікацію, тобто мяку клисифікацію. Мяка класифікація конструюється на основі використання методу прогнозування імовірності. Даний метод мякої класифікації повертає мітки класів для цільової змінної вихідних даних, тобто категорію до якої належить книжка до дитячої категорії си до дорослої категорії. Також слід відмітити, що даний метод повертає двовимірний масив у якого друга колонка представляє знаяення імовірностей для відповідних класів. Цими значеннями імовірностей і визначається муть мякої класифікації. Таким чином чітка класифікація є частковим видом мякої класифікації на основі того, що машина повертала мітку класу по максимому імовірності. Тобто

використовувала максимум правдоподібності. У цій роботі можна бачити, як система реалізовує чітку та нечітку (мяку) класифікації.

Нище наведений фрагмент пайтонівської реалізації в коді вище описаних речей із коментарями.

```
"""Training and testing model. Compute f1-score and
area under the receiver operating characteristic curve (ROC AUC)"""
#Initialization
lg = LogisticRegression()
#Training
lg.fit(x_train, y_train)
#Prediction
y_pred = lg.predict(x_test)
#Predict class probabilities
y_pred_prob = lg.predict_proba(x_test)
```

У наступному підрозділі даного розділу буде описано програмне конструювання на практиці процедури оцінювання якості роботи класифікатора вже без перехресної перевірки.

3.8 Оцінювання метрик якості

У цьому підрозділі даного третього розділу розглядається програмне конструювання процесу оцінювання якості роботи системи на основі декількох метрик якості. Розглядаються настпуні метрики якості для оцінбвання. Перша метрика якості це точність, друга метрика це точність з повнотою, третя метрика якості – це площа криволінійної трапеції під рок кривою, яка характеризує точність.

Нище наведено фрагмент пайтонівського кодування вище наведених метрик точності, які характеризують якість вирішення завдання бінарної класифікації.

```
#Compute the 'accuracy'
accuracy_score = accuracy_score(y_test, y_pred)
#Compute the 'f1-score'
f1_score = f1_score(y_test, y_pred, average='weighted')
#Compute the 'roc-auc'
roc_auc_score = roc_auc_score(y_test, y_pred)
```

У наступному підрозділі даного третього розділу буде наведено процеси візуалізації нормованої та не нормованої матриць помилок.

3.9 Візуалізація нормованої та не нормованої матриць помилок

Програмна реалізація процесу візуалізації нормованої матриці помилок та не нормованої матриці помилок є важливою складовою у роботі даної системи, оскільки ці матриці дають можливість візуалізувати точність в межах класів класифікації. Елементи головної діагоналі вказують на якість роботи логістич. регресі., а елементи не головної діагоналі візуалізують помилки класифікатора.

Нище наведений фрагмент коду для практичної реалізації процесу обчислення елементів не нормованої та нормованої матриць помилок класифікації.

```
#Compute confusion matrix to evaluate the accuracy of a classification
conf_matrix = confusion_matrix(y_test, y_pred)
#Compute normalized confusion matrix to evaluate the accuracy of a classification
norm_conf_matrix = confusion_matrix(y_test, y_pred, normalize='all')

"""Metrics"""
print("Dataframe of metrics:",
      pd.DataFrame([accuracy,accuracy_score,f1,f1_score,roc_auc,roc_auc_score],
                   index=["accuracy","accuracy_score","f1","f1_score",
                          "roc_auc","roc_auc_score"]))
```

А тут реалізація візуалізації не нормованої матриці помилок.

```
"""Visualization of confusion matrix"""
#Plot non-normalized confusion matrix
disp = plot_confusion_matrix(lg, x_test, y_test,

display_labels=list(labelencoder.inverse_transform(np.unique(y))),
                              xticks_rotation='vertical')
disp.ax_.set_title('Confusion matrix, without normalization’)
```

Безпосередні вигляд не нормованої матриці помилок класифікації має вид, що наведений на рисунку нище.

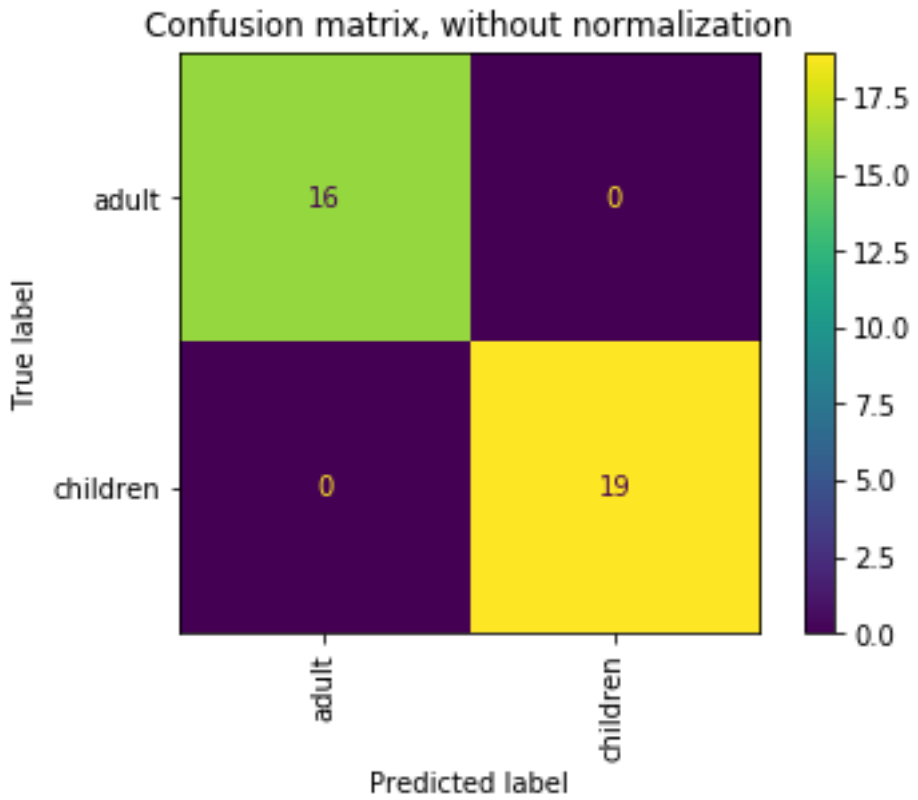

Рис.3.1 – Візуалізац. не нормован. матриці помилок

Нище розглядається фрагментпайтонівського коду для візуалізації нормованої матриці помилок. Зручність використання нормованої матриці помилок полягає в тому, що можна бачити у відсотках точність в межах класів класифікації. В тому і полягає цінність нормованої матриці помилок. Розгляньмо вище наведений

програмний пайтонівський код для реалізації нормованої матриці помилок. Аналогічно елементи головної діагоналі вказують на точність, а наявність не нульових елементів не головної діагоналі вказують на похибку причому чим їх значення більші тим гірше працює класифікатор.

```
"""Visualization of confusion matrix"""
#Plot normalized confusion matrix
disp = plot_confusion_matrix(lg, x_test, y_test, normalize='all',

display_labels=list(labelencoder.inverse_transform(np.unique(y))),
                            xticks_rotation='vertical')
disp.ax_.set_title('Normalized confusion matrix')
```

Нище наведено фізуалізацію нормованої матриці точності для бінарної класифікації, яку виконує алгоритм логіс. Регресія.

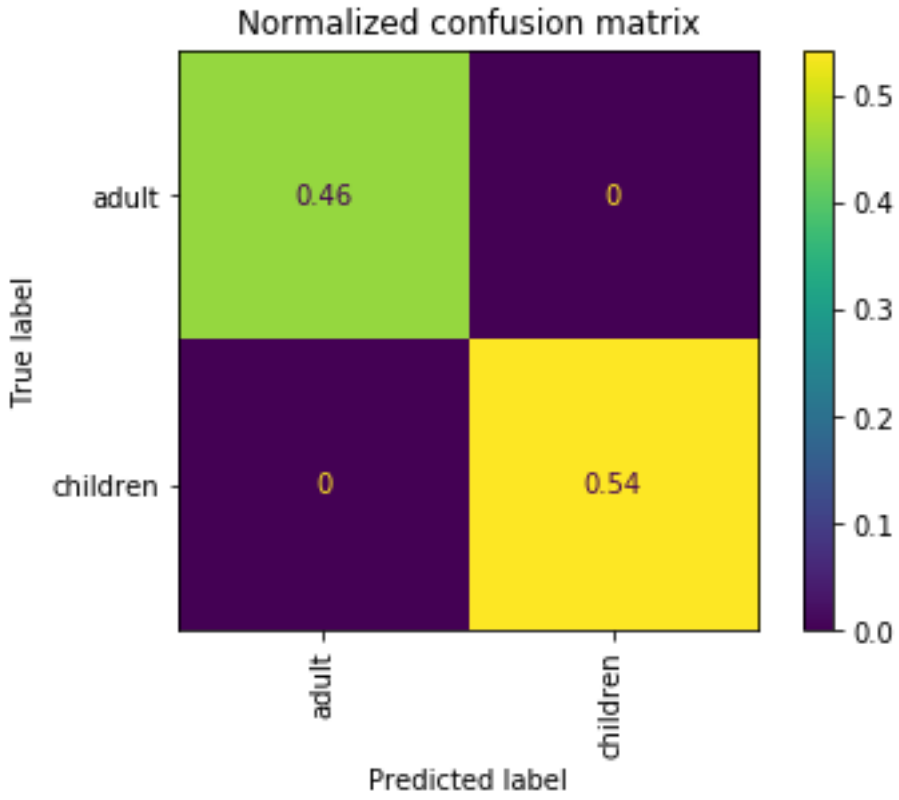

Рис.3.2 – Візуалізац. нормован. матриці помилок

У наступному підрозділі даного третього розділу розглядатиметься процес запису навченої моделі у файл, щоб нею могли користатися сторонні програми.

3.10 Запис моделі у файл

Даний підрозділ третього розділу є дуже важливим, оскільки він дає змогу запису попередньо навченої логіс. регресії у файл, а значить її зможуть використовувати у подальшому, як інші скрипти даного проекту так і загалом сторонні програми.

Нище наведено практичну реалізацію процесу запису логіс. регрес. у файл за допомогою пайтонівської бібліотеки джобліб.

```
"""Saving trained model. Loading trained model.
Decision making by trained model"""
#Saving trained model
joblib.dump(tfidf, 'tfidf.pkl')
#Loading trained model
tfidf = joblib.load('tfidf.pkl')
```

Таким чином розділ конструювання системи для класифікації текст. інформац. по категоріях, а саме книги для діте чи дорослих є завершиною з точки зору практичного конструювання.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ризик як кількісна оцінка небезпек

Ризик – це кількісна характеристика оцінки ступеня небезпеки. Ризик є критерієм реалізації небезпеки. Нескінченно малий (“нульовий”) ризик свідчить про відсутність реальної небезпеки в системі, і навпаки: чим вищий ризик, тим вища реальність впливу небезпеки. Величина ризику визначається як відношення кількості подій з небажаними наслідками до максимально можливого їх за конкретний період часу [12, 13]. Наведена формула дозволяє розрахувати величину загального та групового ризику. При оцінці загального ризику величина N визначає максимальну кількість усіх подій, а при оцінці групового ризику – максимальну кількість подій у конкретній групі. Характерним прикладом визначення загального ризику може служити розрахунок числового значення загального ризику побутового травматизму зі смертельними наслідками. Сучасні вчені висувають концепцію прийнятного ризику, суть якої полягає у напрямі до такої безпеки, яку суспільство може прийняти (дозволити) у даний період часу [12, 13]. Нині з’ясувалося, що не можна досягти абсолютної безпеки; суспільство може собі дозволити лише економічно виправданий рівень безпеки. Введено поняття індивідуального ризику загибелі людини. Максимально сприятливий рівень індивідуального ризику: на 1 млн. населення гине 1 людина на рік (до цього рівня ризику треба прагнути). Забезпечити нульовий ризик неможливо. Набула поширення концепція прийнятого ризику, тобто такий ризик суспільство може прийняти (дозволити) у даний час і котрий може бути економічно виправданий. Прийнятий ризик вміщує технічні, економічні, соціальні та політичні аспекти, являючи собою компроміс між рівнем безпеки та можливостями її досягнення [12, 13]. У деяких країнах, наприклад, у Голландії, рівні прийнятого ризику встановлені у законодавчому порядку. Основним питанням безпеки життєдіяльності є питання підвищення рівня безпеки, тобто

зниження імовірності ризику до припустимого рівня. Це можливо досягти кількома шляхами [12, 13]:

– повна або часткова відмова від робіт, операцій та систем, які мають високий ступінь небезпеки; – заміна небезпечних операцій іншими, менш небезпечними;

– удосконалення технічних систем та об’єктів; – розробка та використання спеціальних засобів захисту; – заходи організаційно-управлінського характеру, в тому числі контроль за рівнем безпеки, навчання людей з питань безпеки, стимулювання безпечної роботи та поведінки; – ліквідація наслідків аварій та катастроф з наступним їх аналізом. Як правило, для підвищення рівня безпеки завжди використовується комплекс цих заходів та засобів. Для того щоб надати перевагу конкретним засобам та заходам або певному їх комплексу, порівнюють витрати на ці заходи та засоби і рівень зменшення шкоди, який очікується в результаті їх запровадження. Такий підхід до зменшення ризику небезпеки називається управлінням ризиком.

Враховуючі наявність в сучасній літературі достатньо значної кількості визначень та трактувань щодо ризиків і пов’язаних з ними понять, розглянемо лише основні, найбільш поширені види існуючих на даний час види класифікацій (ідентифікацій) ризиків.

За джерелом ризику: техногенний ризик; природний ризик.

За видом джерела ризику: внутрішній ризик, пов’язаний із функціонуванням організації; зовнішній ризик, пов’язаний із функціонуванням організації; зовнішній ризик пов’язаний із зовнішнім середовищем і не залежний від функціонування організації; ризик, пов’язаний з помилками людини (так званий людський фактор).

За характером нанесеного збитку: економічний; екологічний; соціальний.

За розміром нанесеного збитку: припустимий; граничний (критичний); катастрофічний.

За рівнем небезпеки: безумовно прийнятний; прийнятний; неприйнятний.

За часом впливу: короткостроковий; середньостроковий; lовгостроковий.

За частотою впливу: разовий; періодичний; постійний.

За рівнем впливу: локальний; глобальний.

За масштабом впливу: індивідуальний; колективний.

За сприйняттям людей: добровільний; примусовий.

4.2 Навчання працюючих і інструктажі з охорони праці

Нормативна база: Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці, затверджене наказом Держнаглядохоронпраці від 26.01.2005 № 15.

Інструктажі з питань охорони праці на підприємстві мають проходити усі працівники під час прийняття на роботу і в процесі трудової діяльності, а також учні, курсанти, слухачі та студенти під час трудового і професійного навчання. Інструктаж, навчання та перевірку знань з охорони праці зобов’язаний організувати роботодавець за власний рахунок.

Перевірку знань працівників з питань охорони праці на підприємстві проводить комісія, склад якої затверджує своїм наказом керівник.

Законодавством заборонено допускати до робіт працівників, у т. ч. посадових осіб, які не пройшли навчання, інструктаж і перевірку знань з питань охорони праці.

Відповідальність за організацію і проведення інструктажів несе роботодавець.

Види інструктажів з охорони праці.

За характером і часом проведення розрізняють вступний, первинний, повторний, позаплановий та цільовий інструктажі, кожен з яких проводиться певним категоріям працівників та за певних умов.

Вступний інструктаж з охорони праці.

Цей вид інструктажу з охорони праці на підприємстві проводять: усім працівникам, яких беруть на постійну або тимчасову роботу, незалежно від їх освіти, стажу роботи та посади; працівникам інших організацій, які прибули на підприємство і беруть безпосередню участь у виробничому процесі або виконують інші роботи для підприємства; учням та студентам, які проходитимуть на підприємстві трудове або професійне навчання; учасникам екскурсії на підприємство.

Первинний інструктаж з охорони праці.

Такий вид інструктажу з питань охорони праці проводять з працівниками: новоприйнятими на постійну чи тимчасову роботу; відрядженими з іншого підприємства; яких перевели з іншого структурного підрозділу підприємства; які виконуватимуть нову роботу; з учнями, курсантами, слухачами та студентами навчальних закладів: до початку трудового або професійного навчання; перед виконанням кожного навчального завдання, пов’язаного з використанням різних механізмів, інструментів, матеріалів тощо.

Повторний інструктаж з охорони праці.

Терміни проведення повторного інструктажу встановлюються НПАОП, які діють у галузі, або роботодавцем з урахуванням конкретних умов праці, проте не рідше: одного разу на три місяці – для робіт з підвищеною небезпекою; одного разу на шість місяців – для інших робіт.

Позаплановий інструктаж з охорони праці.

Цей вид інструктажу з охорони праці проводять у тому разі, якщо на підприємстві: введено в дію нові або переглянуті НПАОП, внесено зміни та

доповнення до них; змінено технологічний процес, замінено або модернізовано устаткування, прилади, інструменти, вихідну сировину, матеріали тощо; працівниками порушено вимоги НПАОП, й це призвело до травм, аварій, пожеж тощо; у разі перерви понад 30 календарних днів у роботі виконавця робіт з підвищеною небезпекою та понад 60 днів – у роботі виконавця інших робіт.

Цільовий інструктаж з охорони праці.

Проводять у разі ліквідації аварії або стихійного лиха та проведення робіт, на які потрібен наряд-допуск, наказ або розпорядження.

У разі виконання робіт, на які потрібно оформлювати наряд-допуск, цільовий інструктаж реєструють у цьому наряді-допуску, а в журналі реєстрації інструктажів – не обов’язково.

Ризик, як кількісна оцінка небезпек.

У Законі Україні «Про об'єкти підвищеної небезпеки», приводиться таке визначення: *ризик* – це ступінь імовірності визначеної негативної події, що може відбутися у визначений час у випадку визначених обставин на території об'єкта підвищеної небезпеки і/чи за його границями. *Ризик* – це усвідомлена кількісна оцінка ймовірності виникнення події з певними небажаними наслідками. *Аналіз ризику* – це систематичне використання інформації про ризик, порівняння його з прийнятним ризиком, обґрунтування раціональних заходів захисту. Нульового ризику (абсолютної безпеки) немає. *Існує поняття допустимого ризику* – це ризик, який у конкретній ситуації вважається допустимим до рівня, прийнятого у суспільстві, виходячи з економічних та соціальних чинників. *Прийнятний ризик* – це ризик, який не перевищує на території об'єкта підвищеної небезпеки або за її межами гранично допустимого рівня. *Максимально прийнятний ризик* – при якому може постраждати не більше 5%% видів біогеоценозу. *Оцінка ступеня ризику* – це кількісна оцінка впливу будь-якої небезпеки.

Оцінка ступеня ризику (R) визначається математичною формулою як відношення кількості небезпек, що проявляються з негативними наслідками (n) до можливої їх кількості (N) за конкретний період часу:

$$R = \frac{n}{N}.$$

Гранично допустимий ризик – це максимальний ризик, який не повинен перевищуватись, незважаючи на очікуваний результат. Надмірний ризик характеризується виключно високим рівнем, який у переважній більшості випадків призводить до негативних наслідків. *Інтегральний ризик* – сумарний ризик для населення, соціальних, техногенних і природних об`єктів від всіх можливих негативних подій природного і техногенного походження. *Аналіз ризику* – систематичне використання наявної інформації для ідентифікації небезпек і визначення ризику для однієї людини, населення, майна, соціальних і техногенних об`єктів та навколишнього природного середовища. *Оцінка ризику* – процес визначення імовірності виникнення аварій або надзвичайних ситуацій та відповідних їм збитків.

Індивідуальний ризик характеризує реалізацію небезпеки певного виду для конкретної особи. Індивідуальний ризик характеризує розподіл ризику в часі та просторі. Його можна визначити за кількістю що реалізувалися факторів ризику:

$$R_i = \frac{P}{L},$$

де R_i - індивідуальний ризик; P - кількість потерпілих (загиблих) в одиницю часу t від певного фактору ризику f; L - число людей, схильних до відповідного фактору ризику в одиницю часу t.

При визначенні індивідуального ризику необхідно враховувати частку часу перебування в "зоні ризику" та постійне місце проживання особи.

ВИСНОВКИ

За результатам. виконан. робот. одержано наступ. висновки:

1. Зпроектован. програм.. систему для класифікац. текстов. інформації по категоріях на основі програмн. технологі.Python та різних програм. компонент бібліот.: Scikit-lear., NLTK, тощо, що дало змогу перейти до етапу її практичног. конструювання.

2. Практич. зконструйован. програм. систему для класифікації текстов. інформації по категор., оцінено точність її роботи (точніст. класифікації корпус. текстов. документів склала 96.4%), що дає змогу її прикладног. застосуванн. у різних галузях людської діяльності, наприклад, класифікац. потоку новин в реальном. часі для медіа агенстю, тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. pandas documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 07.03.2024)
2. NumPy documentation. URL: https://numpy.org/devdocs/ (дата звернення: 07.03.2024)
3. Пастух О.А., Яцишин В.В., Жаровський Р.О., Шаблій Н.Р. Software tool for productivity metrics measure of relational database management system // Математичне моделювання. 2023. №1 (48). С.7-17.
4. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. Technology of relational database management systems performance evaluation during computer systems design // Scientific Journal of TNTU. 2023. №1. С. 54–65.
5. Pastukh O., Yatsyshyn V. Brain-computer interaction neurointerface based on artificial intelligence and its parallel programming using high-performance calculation on cluster mobile devices // Scientific Journal of TNTU. 2023. №4. С. 26-31.
6. Pastukh O., Stefanyshyn V., Baran I., Yakymenko I., Vasylkiv V. Mathematics and software for controlling mobile software devices based on brain activity signals // The International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2023). 2023. №3628. С. 684–689.
7. Pastukh O., Petryk M., Bachynskiy M., Mudryk I., Stefanyshyn V. Processing of Cerebral Cortex Neurosignals from EEG Sensors and RecognizinSpecific Types of Mechanical Movements Elements of Pacient Limbs under the Cognitive Feedback Influenses // International Workshop on Computer Information Technologies in Industry 4.0. 2023. №3468. С. 61–70.
8. Yasniy O., Pastukh O., Didych I., Yatsyshyn V., Chykhira I. Application of machine learning for modeling of 6061-T651 aluminum alloy stress− strain diagram // Procedia Structural Integrity. 2023. №48. С. 183-189.

9. Matplotlib 3.8.3 documentation. URL: https://matplotlib.org/stable/index.html (дата звернення: 07.03.2024)

10. Keras 3 API documentation. URL: https://keras.io/api/ (дата звернення: 07.03.2024)

11. statsmodels. URL: https://www.statsmodels.org/stable/api.html (дата звернення: 07.03.2024)

12. Мелех Л.В. Безпека життєдіяльності та охорона праці : Навчальний посібник. Львів: Львівський державний університет внутрішніх справ. 2022. 219 с.

13. Сокуренко В.В. Безпека життєдіяльності та охорона праці : Підручник. Харків: Харків. нац. ун-т внутр. справ. 2021. 308 с.

ДОДАТКИ

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп’ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПІ
_____ Петрик М.Р.
”___” _____ 2024 р

РОЗРОБКА СИСТЕМИ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ ПО КАТЕГОРІЯХ

ТЕХНІЧНЕ ЗАВДАННЯ

Вид робіт: Кваліфікаційна робота
на здобуття освітнього ступеня ”Бакалавр”
спеціальність 121 ”Інженерія програмного забезпесення”

”УЗГОДЖЕНО”
керівник кваліфікаційної роботи
_____ д.т.н., проф. Пастух О.А.
”____” _____ 2024 р.

”ВИКОНАВЕЦЬ”
студент групи СП-41
_____ Бабинець К.М.
”____” _____ 2024 р.

Тернопіль 2024

ЗМІСТ

1. Підстави до розробки

2. Призначення до розробки

3. Вимоги до програмної системи

3.1. Функціональні характеристики

3.2. Склад та параметри технічних засобів

3.3. Інформаційна та програмна сполучність

4. Стадії розробки

5. Програмна документація

6. Порядок контролю та приймання

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка програмної системи виконується у відповідності до графіку навчального плану на 2023 – 2024 н.р. та згідно наказу ректора Тернопільського національного технічного університету імені Івана Пулюя на виконання кваліфікаційної роботи на здобуття освітнього ступеня бакалавр.

Тема кваліфікаційної роботи на здобуття освітнього ступеня бакалавр: ”Розробка системи для класифікації текстової інформації по категоріях” для оптимізації роботи щодо класифікації потоків текстових новин у сфері медіа агенств, що працюють в середовищі Інтернету.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмна система для класифікації текстової інформації по категоріях надає можливість віднести текстову інформацію до того чи іншого класу – категорії. Це, наприклад, потрібно для категорізації інформаційного текстового потоку для медіа компаній. Цим самим полекшує, прискорює та здишевлює попередню обробку текстової інформації – потоку новин для медійних компаній. Окремим питанням є оцінювання точності роботи програмної системи на основі різних трьох метрик якості, які мають різну складність в інтерпритації, стійкості для незбалансованих даних та надійності. Програмна система повинна для наочності сприйняття точності її роботи візуалізовувати точності характеристики в межах окремих класів. Програмна система повинна давати можливість зберігати алгоритми свого функціонування, щоб їх могли використовувати сторонні програми, тобто, зберігати алгоритми машинного навчання у файли даних. Програмна система повинна давати можливість модифікувати алгоритми структуризації текстової слабоструктурованої

інформації на основі різних моделей, наприклад, моделі word embedding, тощо. Також система повинна онлайн у часі класифіковувати потік текстової інформації ”на льоту”.

3. ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ

3.1. Функціональні характеристики

- формувати та візуалізувати корпус текстових документів

- формувати та візуалізувати метрики якості завдання класифікації, накшталт accuracy, f1_score, roc_auc_score

- прогнозувати категорію для інших текстових документів текстової інформації

- зберігати алгоритм машинного навчання для класифікації текстової інформації у вигляді текстового документа

3.2. Склад та параметри технічних засобів

1) персональний комп’ютер з 1024 Мб оперативної пам‘яті, встановленою системою Windows 8, 8.1, 10. Більше 300 Мб вільного місця на жорсткому диску. Двоядерний процесор із двома потоками та тактовою частотою не менше 1.2 GHz.

2) наявність встановленого дистрибутиву Анаконди та ядра мови програмування Python із біліотеками Keras, Pandas, Numpy, Statsmodels.

3.3. Інформаційна та програмна сполучність

Програмна система повинна коректно функціонувати в операційній системі Windows різних поколінь. Програмні компоненти програмної системи повинні працювати з високою степінню точності та стійкості. Розробку виконувати з використанням технологій Python в середовищі програмування – інтерфейсі взаємодії Spyder.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи програмна система повинна пройти крізь стадії життєвого циклу розробки програмного забезпечення:

- аналіз предметної області
- проектування архітектури програмної системи
- практичне конструювання програмної системи
- реалізація певних елементів візуалізації даних
- тестування якості роботи програмної системи щодо точності класифікації текстової інформації по категоріях
- оформлення супровідної документації
- здача роботи

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмної системи для класифікації текстової інформації по категоріях повинні бути розроблені наступні документи:

- пояснювальна записка
- технічне завдання
- презентаційний матеріал
- додатки

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблена програмна система має виконувати всі вимоги, що складаються з перерахованих у даному технічному завданні.

Приймання проводиться спеціально створеною екзаменаційною комісією у відведені терміни, що вказані у календарному плані завдання кваліфікаційної роботи на здобуття освітнього ступеня бакалавр.

Додаток Б

Тези доповіді на конференції

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2024 р.

(збірник тез конференції)

Тернопіль 2024

Марків К. **РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КРУЇЗНОЇ КОМПАНІЇ З ВИКОРИСТАННЯМ SPRING FRAMEWORK**	**331**
Музика В. **АСИНХРОННІСТЬ ПРОГРАМУВАННЯ В РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**	**332**
Петришин Я., Марцинюк Я. **РОЛЬ CI/CD У ПІДВИЩЕННІ ЕФЕКТИВНОСТІ ТА НАДІЙНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**	**333**
Бурса В., Мудрик І. **АВТОМАТИЗОВАНА СИСТЕМА ОЦІНКИ РІВНЯ ЗНАННЯ ІНОЗЕМНОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ**	**335**
Антонюк Д., Пастух О. **АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АНСАМБЛЮВАННЯ АЛГОРИТМІВ**	**336**
Чорна Х., Пастух О. **ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУ ПАСАЖИРОПЕРЕВЕЗЕНЬ**	**337**
Бабинець К., Пастух О. **ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ**	**338**
Бойко Д., Пастух О. **ПРОГРАМА ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ ЩОДО КРЕДИТУВАННЯ**	**339**
Яковів Б., Пастух О. **ПРОГРАМНА СИСТЕМА ДЛЯ НЕВРОЛОГІЧНОЇ ДІАГНОСТИКИ**	**340**
Гашинський Р., Мельник Н. **АВТОМАТИЗОВАНИЙ АНАЛІЗ СХЕМИ ДОКУМЕНТІВ З ЧАСТКОВО-СТРУКТУРОВАНИМИ ДАНИМИ**	**341**

УДК 629.735.072.8.08:004(043.2)
Бабинець К. – ст. гр. СП-41, Пастух О.
Тернопільський національний технічних університет імені Івана Пулюя

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВОЇ ІНФОРМАЦІЇ

Науковий керівник: д.т.н., професор Пастух О.А.

Babynets Kh., Pastukh O.
Ternopil Ivan Puluj National Technical University

SOFTWARE FOR CLASSIFICATION OF TEXT INFORMATION

Supervisor: Dr.Sc., Professor Pastukh O.A.

У сьогоденні очевидним є факт потоків великих даних різної природи інформації, серед яких важливе місце займає текстова інформація. Говорять навіть про те, що ми живемо не лише в еру інформаційних технологій, але і в еру технологій даних. З плином часу кількість даних збільшується по експоненційному закону, тому виникає гостра потреба автоматичної обробки текстової інформації на основі комп'ютерів, які у свою чергу потребують використання сучасного програмного забезпечення для класифікації текстової інформації. На рис.1 наведено програмний конвеєр для класифікації текстової інформації.

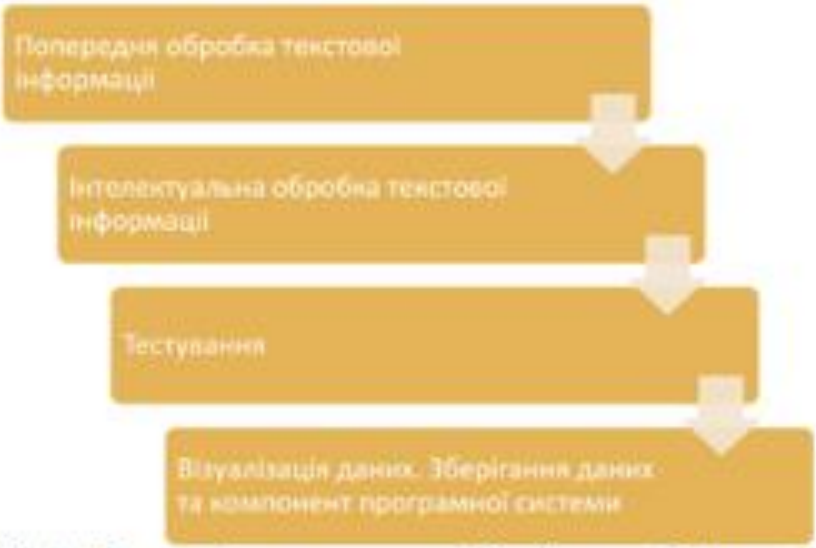

Рис.1 – Програмний конвеєр для класифікації текстової інформації

Програмний конвеєр для класифікації текстової інформації містить базові кроки, що зображені на рис.1. Перший крок – це попередня обробка текстової інформації: узгодження регістерних форм даних, виправлення граматичних помилок, тощо. Другий крок – це класифікація на основі алгоритму машинного навчання. Третій крок – це процес тестування. Четвертий крок – це візуалізація даних і зберігання окремих програмних компонент для їх використання сторонніми програмами.

Додаток В

Диск із кваліфікаційною роботою бакалавра