Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: «Дослідження впливу методів попередньої обробки текстових даних на якість класифікаційних моделей машинного навчання»

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Цимбалюк Г.О.Б.
(підпис) (прізвище та ініціали)

Керівник _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_28__» _____травня__ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Цимбалюк Гліб Олександр Богданович
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження впливу методів попередньої обробки текстових даних на якість класифікаційних моделей машинного навчання

Керівник роботи Никитюк Вячеслав Вячеславович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_29_» _листопада_ 2024 року № _4/7-1139_

2. Термін подання студентом завершеної роботи 26 травня 2025 р.

3. Вихідні дані до роботи Наукові публікації про методи попередньої обробки даних для обробки природньої мови

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області. 2 Методи попередньої обробки NLP та моделі класифікації текстових даних
3 Експериментальне дослідження впливу попередньої обробки текстових даних на якість класифікації
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Мета. Завдання дослідження. 3 Об'єкт, Предмет дослідження. Наукова Новизна 4 Актуальність дослідження. 5 NLP. 6 Основні завдання визначення авторства. 7 Основні етапи класифікації. 8 Методи попередньої обробки текстових даних. 9 Приклад нормалізації та токенізації. 10 Приклад комбінації традиційних підходів нормалізації тексту зі стемінгом. 11 Приклад комбінації традиційних підходів нормалізації тексту з лематизацією. 12 Класифікаційні моделі, що використовувались в роботі 13 Метрики класифікації 15 Вплив розміру простору ознак на точність класифікаційної моделі наївного Байєса 16 Результати класифікації для різних моделей 17 Вплив вилучення стоп-слів на точність моделі 18 Вплив стемінгу та лематизації на точність моделі класифікації ... 19 Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 29 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.11.2024	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір	02.12.2024-10.01.2025	*Виконано*
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	13.01.2025-14.03.2025	*Виконано*
	роботи		
4.	Оформлення розділу «Аналіз предметної області»	17.03.2025-28.03.2025	*Виконано*
5.	Оформлення розділу «Методи попередньої обробки	31.03.2025-11.04.2025	*Виконано*
	NLP та моделі класифікації текстових даних»		
6.	Оформлення розділу «Експериментальне дослідження	14.04.2025-25.04.2025	*Виконано*
	впливу попередньої обробки текстових даних на якість		
	класифікації»		
7.	Виконання завдання до підрозділу «Охорона праці»	28.04.2025-02.05.2025	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	28.04.2025-02.05.2025	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	05.05.2025-09.05.2025	*Виконано*
10.	Нормоконтроль	12.05.2025-15.05.2025	*Виконано*
11.	Перевірка на плагіат	16.05.2024	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.05.2025	*Виконано*
13.	Захист кваліфікаційної роботи	29.05.2025	

Студент _____ Цимбалюк Г.О.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження впливу методів попередньої обробки текстових даних на якість класифікаційних моделей машинного навчання // Кваліфікаційна робота освітнього рівня «Магістр» // Цимбалюк Гліб Олександр Богданович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2025 // С. 73, рис. – 13, табл. – 5, кресл. – ____, додат. – 2, бібліогр. – 24.

Ключові слова: обробка природної мови, стемінг, токенізація, нормалізація, лематизація, класифікація.

Кваліфікаційна робота присвячена дослідженню впливу методів попередньої обробки текстових даних на точність кінцевої класифікаційної моделі. В першому розділі кваліфікаційної роботи описані задачі класифікації та інтелектуального аналізу ттекстових даних. Проведено огляд досліджень в галузів методів попередньої обробки текстових даних на якість класифікаційних моделей. В другому розділі кваліфікаційної роботи розглянуто теоретичні відомості про основні етапи класифікації даних, зокрема детально описано методики попередньої обробки тестових даних та традиційні і більш сучасні класифікаційні моделі, що використовуються в задачах обробки природної мови.

В третьому розділі кваліфікаційної роботи описано вибір середовища для програмування, наведено основні переваги та недоліки. Обрано та описано відкритий набір даних для дослідження. Наведено експериментальних досліджень впливу різних методик попередньої обробки текстових даних на якість класифікації.

Об'єкт дослідження: обробка природньої мови. Предмет дослідження: Методи попередньої обробки та інтелектуального аналізу в задачах обробки природної мови.

ANNOTATION

Study of the impact of text data preprocessing methods on the quality of machine learning classification models // The educational level "Master" qualification paper // Tsymbaliuk Glib Oleksandr // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2025 // P. 73, fig. – 13, tables – 5, posters – __, annexes – 2, ref. – 24.

Key words: natural language processing, stemming, tokenization, normalization, lemmatization, classification.

Thesis is devoted to the study of the influence of text data pre-processing methods on the accuracy of the final classification model. The first chapter of the qualification paper describes the tasks of classification and intellectual analysis of text data. A review of research in the field of text data preprocessing methods on the quality of classification models is conducted. The second section of the qualification paper deals with theoretical information about the main stages of data classification, in particular, the methods of pre-processing test data and traditional and more modern classification models used in natural language processing tasks are described in detail.

The third chapter of the qualification paper describes the choice of programming environment, the main advantages and disadvantages. The open data set for the study is selected and described. Experimental studies of the impact of various text data preprocessing techniques on the quality of classification are presented.

Object of research: natural language processing. Subject of research: Methods of pre-processing and intelligent analysis in natural language processing tasks.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

BERT (англ. Bidirectional Encoder Representations from Transformers) – двонаправлені кодування кодера з трансформерів.

HTML (англ. HyperText Markup Language) – мова розмітки гіпертексту.

IDF (англ. Inverse Document Frequency) – обернена частота документа.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даними.

LLM (англ. Large Language Model) – велика мовна модель

NER (англ. Named Entity Recognition) – розпізнавання іменованих сутностей.

NLP (англ. Natural Language Processing) – обробка природної мови.

TF (англ. Term Frequency) – частота терма (слова).

ШІ – Штучний інтелект.

ЗМІСТ

ВСТУП

Актуальність теми. Обробка природної мови є однією з найактуальніших галузей штучного інтелекту, що динамічно розвивається. Її актуальність зумовлена постійно зростаючим обсягом текстових даних, які генеруються щодня – від публікацій у соціальних мережах та новинних статей до електронних листів та медичних записів. Здатність машин розуміти, інтерпретувати та генерувати людську мову відкриває безмежні можливості для автоматизації рутинних завдань, покращення взаємодії людини з комп'ютером та видобутку цінної інформації з неструктурованих даних. Завдяки NLP ми можемо створювати інтелектуальних чат-ботів та віртуальних асистентів, які розуміють наші запити, системи автоматичного перекладу, що руйнують мовні бар'єри, а також інструменти для аналізу настроїв та виявлення прихованих закономірностей у тексті.

Актуальність NLP поширюється на безліч галузей. У бізнесі це аналіз відгуків клієнтів, автоматизація обробки звернень, персоналізований маркетинг та підвищення ефективності роботи кол-центрів. У медицині NLP допомагає в аналізі медичних карт для діагностики захворювань, видобутку інформації з наукових статей та автоматичному перекладі медичної термінології. У фінансах – це аналіз ринкових новин для прогнозування цін акцій, виявлення шахрайства та автоматична обробка документів. Юриспруденція використовує NLP для пошуку прецедентів, аналізу контрактів та автоматизації судових процесів. Навіть у сфері освіти NLP сприяє розробці адаптивних навчальних систем та інструментів для перевірки робіт.

Актуальність дослідження методів попередньої обробки NLP на якість класифікаційних моделей є надзвичайно високою, оскільки якість вхідних даних безпосередньо впливає на точність та надійність будь-якої моделі машинного навчання. Неочищені та непідготовлені текстові дані містять шум, надмірність та неоднозначності (наприклад, різні форми одного слова, стоп-слова, пунктуацію), які можуть заплутати модель та знизити її здатність до узагальнення. Методи попередньої обробки, такі як токенізація, нормалізація,

видалення стоп-слів, стемінг та лемматизація, перетворюють сирий текст на більш придатний для аналізу формат. Вони зменшують розмірність даних, усувають зайві елементи та приводять слова до їх базових форм, що дозволяє моделям краще виявляти значущі закономірності та підвищує їхню здатність до правильної класифікації. Дослідження цих методів допомагає визначити оптимальний підхід для конкретного завдання та мови, оскільки ефективність кожного методу може варіюватися залежно від характеристик набору даних та цілей класифікації.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є дослідити вплив методів попередньої обробки текстових даних на основі вибраного набору даних для задачі визначення авторства тексту. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- Проаналізувати стан досліджень в області NLP.
- Дослідити існуючі на даний час методи попередньої обробки текстових даних.
- Проаналізувати відомі методи класифікації в задачах обробки природної мови.
- Вибрати набір даних для дослідження.
- Розробити програмне забезпечення для проведення дослідження.
- Виконати порівняння впливу відомих методів попередньої обробки на точність класифікації.
- Зробити висновки щодо доцільності застосування методів попередньої обробки текстових даних для обраного конкретного випадку.

Об’єкт дослідження – обробка природньої мови.

Предмет дослідження. Методи попередньої обробки та інтелектуального аналізу в задачах обробки природної мови.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що було досліджено вплив методів попередньої обробки текстових даних та їх комбінацій, таких як видалення стоп-слів, стемінг та лематизація на якість класифікаційних моделей NLP для задачі визначення авторства.

Практичне значення одержаних результатів. Розроблено шаблон програного забезпечення, який можна використати як складову інформаційної системи, де стоять задачі обробки природньої мови.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на VIII міжнародній студентській науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2025 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 25 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 73 сторінки, з них 50 сторінки основного тексту, який містить 13 рисунків та 5 таблиць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття текстових даних у контексті штучного інтелекту

У сучасному світі інформаційних технологій текстові дані відіграють ключову роль у формуванні знань, передачі інформації та взаємодії між людьми й комп’ютерними системами. Штучний інтелект (ШІ), як галузь, що займається створенням інтелектуальних агентів і систем, здатних до навчання, аналізу й прийняття рішень, активно використовує текст як основне джерело семантичної інформації. На відміну від числових або структурованих даних, текстова інформація є неструктурованою, що створює як нові можливості, так і низку складних викликів для її обробки.

Текстові дані – це будь-який тип інформації, представлений у вигляді природної мови: речень, абзаців, документів або діалогів. Це можуть бути новинні статті, пости у соціальних мережах, електронна пошта, рецензії на продукти, технічна документація, юридичні акти або медичні записи. Усі ці приклади є джерелами знань, які містять як поверхневу, так і глибоку інформацію про події, наміри, емоції та взаємодії. Саме тому обробка природної мови (Natural Language Processing, NLP) є однією з найважливіших підгалузей штучного інтелекту.

Одна з основних труднощів, з якою стикається штучний інтелект при роботі з текстом, полягає в неоднозначності природної мови. Слова можуть мати кілька значень, граматика є контекстно залежною, а сенс фраз часто визначається не лише їхнім буквальним змістом, а й соціокультурним контекстом. Наприклад, фраза "це просто бомба" у різних контекстах може мати протилежні значення – позитивне як метафора ("це дуже добре") або негативне як пряма загроза.

Штучний інтелект не сприймає текст так, як це робить людина. Машини не мають інтуїтивного розуміння мови чи досвіду спілкування. Тому для того, щоб зробити текст доступним для алгоритмів машинного навчання, його потрібно перетворити на форму, яку система може інтерпретувати – зазвичай це числові вектори або матриці. Це перетворення вимагає глибокої попередньої

обробки, зокрема очищення тексту, нормалізації, розбиття на токени, а також побудови векторних подань (наприклад, через TF-IDF, word2vec, GloVe або сучасні трансформери на кшталт BERT).

Протягом останніх десятиліть спостерігається стрімкий розвиток методів машинного навчання, які працюють з текстом. Серед найбільш значущих досягнень – поява великих мовних моделей (LLM), здатних до контекстного розуміння й генерації тексту. Проте навіть найсучасніші моделі залишаються глибоко залежними від якісного попереднього аналізу та представлення тексту. Від того, як оброблений текст буде подано моделі, залежить її здатність до навчання й точності прогнозування.

Текстові дані в рамках штучного інтелекту можуть виконувати різні функції. У задачах класифікації текст може бути джерелом ознак, які використовуються для визначення тональності висловлювання (позитивне, негативне, нейтральне), тематики документа або авторства. У задачах генерації мова йде про створення осмисленого тексту на основі вхідних параметрів – як у випадку чат-ботів, систем автозаповнення чи автоматизованих редакторів. У прикладних задачах, таких як медичні системи підтримки прийняття рішень, текстова інформація використовується для виявлення симптомів, постановки діагнозу чи пошуку релевантних досліджень.

Слід також зазначити, що текстові дані можуть бути джерелом соціальної, психологічної та поведінкової інформації. За допомогою ШІ можна аналізувати емоційний стан користувачів, виявляти випадки мови ворожнечі або неправдивої інформації, прогнозувати політичні чи споживчі уподобання. Ці можливості відкривають нові горизонти для бізнесу, державного управління, безпеки й медицини, але водночас ставлять серйозні етичні питання щодо конфіденційності, упередженості алгоритмів і права на доступ до персональних даних.

Однією з визначальних особливостей текстових даних є їхня висока варіативність і залежність від домену. Текст, пов’язаний із медициною, значно відрізняється від юридичного тексту або дописів у соціальних мережах як за стилем, так і за структурою. Це означає, що підходи до обробки та аналізу тексту

не можуть бути універсальними: необхідно адаптувати методи до конкретного контексту задачі. Наприклад, в обробці медичних записів важливо зберігати точність термінів, тоді як у випадку з твітом або коментарем до відео важливо правильно інтерпретувати сленг, іронію або емоції.

У контексті машинного навчання текстові дані часто стають основою для побудови класифікаційних моделей. Алгоритми, навчені на великій кількості текстів, здатні визначати категорію документа, передбачати наступне слово або аналізувати зв’язки між поняттями. Проте ці моделі дуже чутливі до якості вхідних даних. Наявність граматичних помилок, скорочень, синонімів чи неочищених символів може суттєво знизити ефективність моделі. Тому етапи підготовки тексту, як-от очистка, нормалізація, перетворення в числове представлення та вибір ознак, є не менш важливими, ніж сама модель.

Штучний інтелект відкриває перед текстовими даними нові можливості, зокрема у галузі багатомовної обробки. Завдяки багатомовним моделям, які навчаються одночасно на десятках мов, з’являється змога створювати універсальні рішення для перекладу, резюмування та класифікації тексту незалежно від мови оригіналу. Це, в свою чергу, сприяє глобалізації знань і доступності інформації для ширшої аудиторії.

Отже, текстові дані в межах штучного інтелекту є не лише джерелом даних, а й об’єктом глибокого дослідження, моделювання та взаємодії. Їх складність, багатозначність та невизначеність вимагають багаторівневого підходу до аналізу. Водночас, потенціал, який приховується в тексті, робить його чи не найціннішим ресурсом для інтелектуальних систем майбутнього.

1.2 Інтелектуальний аналіз тексту та його основні етапи

Інтелектуальний аналіз тексту – це процес автоматизованого аналізу великих обсягів текстової інформації з метою виявлення корисних знань, шаблонів, закономірностей, емоцій чи змістових структур. Завдяки використанню методів обробки природної мови, машинного навчання та статистики, цей аналіз дозволяє структурувати тексти, визначати ключові

поняття, знаходити тематичні напрями, а також класифікувати або кластеризувати документи. Він широко застосовується в бізнесі, науці, журналістиці, праві та охороні здоров'я для прийняття рішень на основі текстових джерел.

У сучасному аналізі природної мови інтелектуальний аналіз тексту є багатоетапним процесом, що охоплює повний цикл – від збирання сирих даних до візуалізації результатів [1-4]. Кожен з етапів має важливе значення для забезпечення якості та ефективності класифікаційної моделі. В [5] наведено наступні етапи інтелектуального аналізу текстових даних:

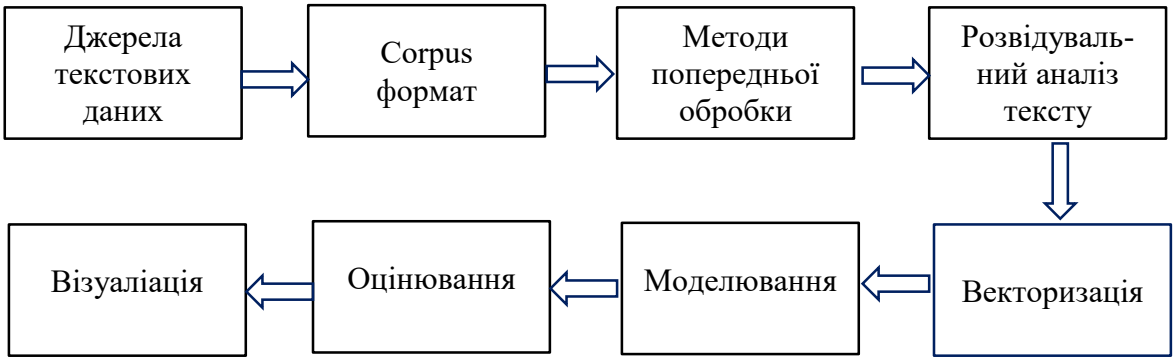


Рисунок 1.1 – Основні етапи інтелектуального аналізу тексту

Розглянемо детальніше етапи, наведені на рисунку 1.1:

- Джерела текстових даних (Text sources). Це перший етап, на якому текстові дані збираються з різних джерел, таких як вебсайти, документи, соціальні мережі або API. Вибір джерела залежить від задачі дослідження та цільової аудиторії. Наприклад, для виявлення настроїв користувачів доречно використовувати дані з Twitter або форумів, тоді як для юридичної класифікації краще підходять структуровані документи або офіційні публікації. Уже на цьому етапі дослідник має враховувати можливу присутність шуму, спотворень, дублювань, а також різноманітність мовних реєстрів і форматів.

- Corpus формат (Corpus format). В контексті класифікації текстів визначає структуру та організацію текстових даних, що використовуються для навчання моделей. Це передбачає приведення зібраних даних до уніфікованої структури, придатної для подальшої обробки. Корпус може бути організований

у вигляді текстових файлів, таблиць чи структурованих JSON-об'єктів. У цьому контексті важливо зберігати не лише сам текст, але й супутню інформацію – метадані, мітки класів або часові позначки. Формат корпусу значною мірою визначає обсяг і глибину обробки на наступних етапах.

- Попередня обробка текстових даних (Text-preprocessing). Включає в себе кроки попередньої обробки тексту, такі як токенізація, видалення стоп-слів, стемінг або лематизація для підготовки даних до аналізу. На цьому етапі здійснюється нормалізація тексту: зниження регістру, видалення пунктуації, зайвих пробілів, а також очищення від HTML-елементів або нестандартних символів. Ключовими складовими є токенізація, тобто розбиття тексту на елементарні одиниці – токени, та фільтрація стоп-слів, що не несуть семантичного навантаження. Часто застосовуються стемінг або лематизація – методи приведення слів до базової форми, що дозволяє зменшити кількість унікальних термінів у словнику. Попередня обробка не тільки знижує розмірність даних, а й усуває випадкові або другорядні особливості, які могли б завадити моделі правильно інтерпретувати вміст.

- Розвідувальний аналіз тексту(Exploratory text analysis) на цьому етапі застосовуються різні методи статистичного або лінгвістичного аналізу, щоб виявити патерни, ключові слова або теми. Такий аналіз передбачає оцінку частотності слів, довжини документів, пошук ключових слів і патернів, а також виявлення аномалій або перекосів у розподілі класів. Це може бути як чисто статистичне дослідження, так і елементи семантичного аналізу. У разі потреби дослідник може здійснити кластеризацію або тематичне групування текстів, щоб отримати уявлення про потенційні категорії або латентні структури у даних. Результати цього етапу часто впливають на вибір технік векторизації й моделювання.

- Векторизація (Vectorizing). Текст перетворюється на числові вектори (наприклад, через TF-IDF або Word2Vec), що дозволяє моделі працювати з ним у числовому вигляді. Найпростішим способом є модель "мішка слів", яка враховує лише частоти появи термінів, ігноруючи порядок слів. Більш просунутою є векторизація з використанням зважених показників, як-от TF-IDF,

де враховується унікальність слова в межах документа. У складніших випадках використовуються векторні представлення слів, відомі як ембедінги (наприклад, Word2Vec, GloVe), що дозволяють зберегти інформацію про контекст. Новітні підходи, такі як трансформерні моделі BERT, дозволяють представляти текст з урахуванням контексту, семантики та синтаксису на рівні всього речення.

- Моделювання(Modeling). На цьому етапі застосовуються алгоритми машинного навчання для тренування моделі, яка буде класифікувати або прогнозувати на основі текстових даних. У залежності від задачі обираються відповідні алгоритми: від класичних моделей, як-от логістична регресія чи наївний баєсівський класифікатор, до складніших варіантів на основі глибокого навчання. Вибір моделі залежить від обсягу даних, складності предметної області та вимог до точності. Важливо враховувати не лише здатність моделі до навчання, але й її стабільність, здатність до генералізації та обчислювальну ефективність.

- Оцінювання (Assessment). Перевіряється ефективність моделі на тестових даних через метрики, такі як точність (accuracy), повнота (recall), достовірність (precision) та F1-міра. Ці показники дають змогу оцінити, наскільки добре модель виконує свої завдання, особливо в умовах дисбалансу між класами. Матриця помилок допомагає виявити типові помилки, а крос-валідація – перевірити стійкість моделі до змін у даних.

- Візуалізація (Vizualization). На останньому етапі результати аналізу візуалізуються через графіки, діаграми або хмарки слів, щоб зручніше зрозуміти висновки з даних. . Завдяки графікам, діаграмам і навіть простим хмарам слів дослідник може зробити дані більш зрозумілими та наочними. Візуальні засоби допомагають не лише краще інтерпретувати результати, але й комунікувати їх для не технічної аудиторії. Вони також дозволяють виявити нові закономірності або сформулювати гіпотези для подальших досліджень.

Таким чином, інтелектуальний аналіз текстових даних є багатоетапним і гнучким процесом, у якому кожен крок має важливе значення для кінцевого результату. Від якості джерела та обробки тексту до вибору моделі й методів

оцінювання – усі елементи мають бути узгодженими, щоб забезпечити ефективні, інтерпретовані та надійні рішення.

Основними завданнями інтелектуального аналізу текстів є: класифікація (віднесення до певної категорії), кластеризація (групування подібних текстів), аналіз тональності (визначення емоційної оцінки), тематичне моделювання (виявлення прихованих тем), витяг іменованих сутностей (наприклад, імен, організацій, дат) та побудова знання на основі змісту. Ці завдання можуть бути як самостійними, так і поєднаними у комплексних аналітичних системах для глибшого розуміння змісту великих текстових масивів.

Отже, інтелектуальний аналіз тексту охоплює широкий спектр завдань, таких як вилучення сутностей, аналіз тональності, виявлення тем, визначення емоційного забарвлення, побудова онтологій тощо. Інтелектуальний аналіз спрямований на виявлення прихованих знань, закономірностей і смислових зв’язків у текстах різного обсягу та походження. Це міждисциплінарна область, яка поєднує методи штучного інтелекту, обробки природної мови, статистики та семантичного аналізу.

Класифікація тексту виступає одним із базових, але водночас важливих компонентів інтелектуального аналізу. Її суть полягає в автоматичному віднесенні текстів до певних наперед заданих категорій. На відміну від загального інтелектуального аналізу, який спрямований на глибше розуміння змісту, класифікація виконує більш вузьке завдання – формальну і часто поверхневу оцінку тексту за певними ознаками. Проте саме класифікація часто лежить в основі складніших аналітичних процесів, слугуючи початковим кроком для фільтрації даних, сегментації аудиторій або моделювання поведінки користувачів. Розглянемо детальніше задачі класифікації текстових даних.

1.3 Задачі класифікації текстових даних

Класифікація тексту – це процес віднесення документів до заздалегідь визначених категорій на основі їхнього змісту [6]. Її мета полягає у присвоєнні кожному тексту певної мітки (класу) на основі його змісту. Така мітка може бути

заздалегідь відомою (наприклад, "позитивний" або "негативний" відгук) або мати більш складну структуру, як-от тематика, тип документа, джерело походження тощо. Класифікація широко застосовується в різних сферах – від автоматичної модерації контенту до фільтрації спаму, аналізу настроїв у соціальних мережах і сортування запитів користувачів у службах підтримки.

Відповідно до [7] існує три основні підходи до класифікації текстових даних, а саме:

- методи машинного навчання;
- лексичні методи;
- гібридні методи.

Методи машинного навчання включають методи керованого та некерованого навчання.

Кероване навчання на основі текстових даних – це підхід у машинному навчанні, за якого модель навчається на попередньо розміченому корпусі текстів. Кожному текстовому прикладу у навчальному наборі відповідає певна мітка або категорія, і завдання алгоритму полягає у тому, щоб навчитися розпізнавати шаблони, які пов’язують вміст тексту з відповідною міткою. У результаті модель зможе автоматично класифікувати нові, невідомі тексти, спираючись на знання, здобуті під час навчання.

Такий підхід застосовується у багатьох практичних задачах, зокрема у фільтрації спаму, визначенні тональності відгуків, тематичній класифікації документів чи автоматичному сортуванні звернень клієнтів. Ефективність керованого навчання значною мірою залежить від якості та обсягу навчального набору, а також від вибору ознак (наприклад, ключових слів, частоти термінів або векторних подань слів) і типу моделі – від простих алгоритмів на кшталт наївного баєсівського класифікатора до складних глибоких нейронних мереж. В даній роботі ми сконцентруємось саме на керованому машинному навчанні.

Некероване навчання на основі текстових даних – це підхід, за якого модель обробляє тексти без попередньо заданих міток або категорій. Метою такого аналізу є виявлення прихованих структур, закономірностей або схожостей між текстами без людського втручання в процес навчання. Алгоритми

некерованого навчання самостійно групують тексти за змістовною подібністю або виокремлюють теми, які найчастіше зустрічаються в корпусі документів.

До основних прикладів застосування некерованого навчання належать кластеризація текстів (групування схожих документів), тематичне моделювання (виявлення прихованих тем), а також візуалізація семантичної структури текстових колекцій. Некероване навчання особливо корисне тоді, коли немає попередньо розмічених даних або коли потрібно дослідити великі обсяги тексту для подальшого аналізу, класифікації або побудови систем рекомендацій.

Лексичні методи базуються на використанні словників, ключових слів, частотного аналізу та простих правил, що визначають належність тексту до певної категорії без навчання моделей. Натомість, гібридні методи поєднують обидва підходи, використовуючи як статистичні алгоритми, так і лексичні ресурси, щоб підвищити точність класифікації або зменшити потребу в великій кількості навчальних даних.

На рисунку 1.2 наведено основні етапи класифікації текстів.

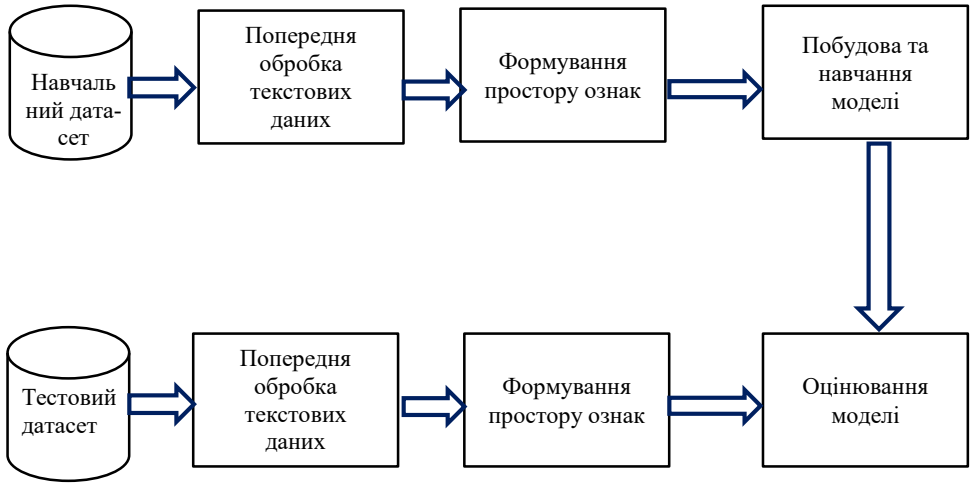


Рисунок 1.2 - Основні етапи класифікації

Класифікація текстових даних складається з кількох ключових етапів, першим з яких є попередня обробка тексту [8]. На цьому етапі тексти очищуються від шуму: видаляються розділові знаки, стоп-слова, зайві пробіли; проводиться токенізація (розбиття тексту на слова), лематизація або стемінг (зведення слів до початкової форми). Метою цього процесу є зменшення

розмірності даних і підвищення їхньої однорідності, що дозволяє алгоритму точніше працювати з текстовими вхідними даними.

Наступний етап – формування простору ознак, тобто перетворення тексту у числове представлення, яке може бути оброблене машинним алгоритмом. Для цього використовують такі методи, як "мішок слів" (bag of words), TF-IDF або векторизацію на основі word embeddings (наприклад, Word2Vec чи BERT) [9]. Після цього відбувається навчання моделі на розмічених даних, тобто алгоритм вивчає, як певні ознаки відповідають певним класам. Останній етап – тестування та оцінка якості моделі, що передбачає застосування моделі до нових текстів і вимірювання її точності, повноти, F1-міри тощо [10]. Це дозволяє оцінити, наскільки ефективно модель справляється із завданням класифікації.

1.4 NLP та зв’язок з класифікацією

NLP (Natural Language Processing) – це обробка природної мови, галузь штучного інтелекту, яка вивчає методи автоматизованого аналізу, розуміння та генерації текстів, написаних людською мовою. Вона охоплює як базові завдання (наприклад, розділення тексту на слова, розпізнавання частин мови), так і складніші – машинний переклад, створення чат-ботів, автоматичне резюмування, аналіз емоцій тощо.

Зв’язок NLP із класифікацією тексту є дуже тісним, адже саме NLP забезпечує технічну базу для цього процесу. Щоб комп’ютер міг класифікувати текст, його спочатку потрібно перетворити на форму, зрозумілу для машинного алгоритму – і це завдання виконує NLP. Наприклад, лематизація, токенізація, видалення стоп-слів та побудова векторних представлень – усе це етапи обробки природної мови, які дозволяють з тексту виділити ключові ознаки для навчання класифікатора. Тобто, NLP – це фундамент, без якого класифікація текстів була б неможливою або неефективною.

NLP охоплюють широкий спектр задач обробки, розуміння й генерації природної мови. Вони поділяються на базові, прикладні та складні задачі, які часто поєднуються в комплексних системах.

- Токенізація – поділ тексту на слова, речення або інші мовні одиниці. Це перший крок у більшості NLP-процесів.
- Лематизація та стемінг – зведення слів до їхньої початкової (словникової) форми для уніфікації тексту.
- POS-тегування (розпізнавання частин мови) – визначення граматичної ролі кожного слова (іменник, дієслово тощо).
- Named Entity Recognition (NER) – виявлення іменованих сутностей у тексті: імен, назв організацій, дат, географічних об'єктів.
- Аналіз тональності (Sentiment Analysis) – визначення емоційного забарвлення тексту (позитивне, негативне, нейтральне).
- Тематичне моделювання – автоматичне виявлення тем у великій кількості документів.
- Класифікація тексту – присвоєння тексту певної категорії (наприклад, тип новини або жанр).
- Кластеризація текстів – об'єднання схожих текстів без попередніх міток.
- Машинний переклад – автоматичне перекладання текстів з однієї мови на іншу.
- Аналіз залежностей (syntactic parsing) – побудова граматичної структури речення.
- Питально-відповідальні системи (Question Answering) – автоматичне знаходження відповіді на поставлене запитання.
- Генерація тексту – створення зв’язного тексту на основі заданого вмісту або шаблону.

Ці задачі можуть бути як незалежними, так і частинами більших систем, таких як чат-боти, голосові помічники, аналітичні інструменти для обробки текстів або пошукові системи.

Методи NLP дозволяють машині розуміти, інтерпретувати та генерувати людську мову, що є критично важливим для ефективного функціонування сучасних інтелектуальних систем.

Класифікація в межах NLP є основою для багатьох прикладних задач – від автоматичного сортування електронної пошти до розпізнавання емоцій у текстах соціальних мереж. Ці задачі можуть бути як незалежними, так і частинами більших систем, таких як чат-боти, голосові помічники, аналітичні інструменти для обробки текстів або пошукові системи.

1.5 Прикладні дослідження ефективності методів попередньої обробки текстових даних

Методи попередньої обробки текстових даних часто недооцінюються, проте вони відіграють ключову роль у забезпеченні якості та ефективності моделей обробки природної мови (NLP). Сирий текст зазвичай містить шум, неоднорідності та зайву інформацію, які можуть спотворювати результати аналізу. Застосування таких технік, як нормалізація, токенізація, видалення стоп-слів, стемінг та лематизація, дозволяє стандартизувати дані, зменшити розмірність та виділити суттєві ознаки, що суттєво покращує продуктивність моделей машинного навчання. Дослідження показують, що належна попередня обробка може підвищити точність класифікації тексту та зменшити обчислювальні витрати, роблячи цей етап незамінним у будь-якому NLP-проєкті. Розгляднемо детальніше дослідження науковців в цій галузі.

Автори [11] показали, що ефект від певного методу може значною мірою залежати від початкового набору даних і задачі класифікації. Крім того, зазначено, що іноді вплив може бути незначним, а в деяких випадках навіть призвести до погіршення результату. Також була обґрунтована необхідність попередньої перевірки набору даних на відсоток елементів, які підпадають під вплив конкретного методу.

Сергієнко, Шан та Мінкер [12] провели порівняльне дослідження існуючих та нових методів попередньої обробки тексту для виявлення тем у висловлюваннях користувачів. Вони застосували сім різних методів зважування термів, як контрольованих, так і неконтрольованих, до двох корпусів: дзвінків клієнтів до кол-центру та відповідей операторів. Дослідження показало, що

колективне використання методів зважування термів та нові методи трансформації ознак можуть суттєво покращити результати класифікації навіть при використанні невеликої кількості ознак.

Автори [13] дослідили, як рішення щодо попередньої обробки тексту можуть впливати на результати неконтрольованого навчання, такого як кластеризація та тематичне моделювання. Вони продемонстрували, що навіть незначні зміни в попередній обробці можуть призвести до суттєвих змін у висновках, зроблених на основі моделей. Автори запропонували статистичну процедуру та програмне забезпечення для оцінки чутливості результатів до різних режимів попередньої обробки, що допомагає дослідникам краще розуміти та контролювати вплив цих рішень на аналіз даних.

В [14] розглядаються методи обробки тексту в контексті інформаційного пошуку, включаючи індексацію, зважування термів та класифікацію документів.

Літературне джерело [15] пропонує глибоке занурення в теоретичні аспекти обробки мови, включаючи морфологічний аналіз, синтаксичний розбір та семантичну обробку.

У [16] аналізують різні алгоритми класифікації тексту, включаючи методи на основі правил, статистичні підходи та глибоке навчання. Вони також розглядають різні методи векторизації тексту та зменшення розмірності.

Автори [17] досліджують перехід від векторів слів до векторів значень, підкреслюючи важливість урахування багатозначності слів у векторних представленнях.

Ламба та Маргам [18] у своєму розділі книги розглядають різні рівні представлення тексту (лексичний, синтаксичний, семантичний) та методи, такі як мішок слів, векторизація слів, частотність термів та зважування, виділення іменованих сутностей та синтаксичний аналіз.

Автор [19] досліджує вплив 26 широко використовуваних технік попередньої обробки на ефективність алгоритмів виявлення мови ненависті та образ у Twitter. Вона виявила, що деякі комбінації технік можуть значно покращити продуктивність моделей

В [14] проведено аналіз, як різні етапи попередньої обробки тексту впливають на синтаксичне узгодження онтологій. Вони виявили, що токенізація та нормалізація є більш ефективними, ніж видалення стоп-слів та стемінг/лематизація, і запропонували новий підхід до виправлення помилкових відповідностей, спричинених попередньою обробкою.

В [15] запропонували моделі Word2Vec для ефективного обчислення векторних представлень слів з великих наборів даних. Ці моделі значно покращили точність у завданнях подібності слів при нижчих обчислювальних витратах.

Автори [16] розробили підхід, який представляє кожне слово як мішок символьних n-грам, що дозволяє краще враховувати морфологію слів, особливо в мовах з великими словниками та багатьма рідкісними словами. Натомість автори [17] представили Sentence-BERT (SBERT), модифікацію попередньо навченого BERT, яка використовує сіамські та триплетні мережі для отримання семантично значущих векторних представлень речень. Цей підхід значно зменшує час обчислень при збереженні високої точності

У [18] проведено аналіз чутливості одношарових згорткових нейронних мереж (CNN) до змін у конфігурації архітектури для завдань класифікації речень. Вони виявили, які компоненти архітектури мають суттєвий вплив на продуктивність моделі, а які – незначний.

Автори [21] досліджують вплив різних методів попередньої обробки тексту на синтаксичне узгодження онтологій. Вони виявили, що базові методи, такі як токенізація та нормалізація, покращують точність та повноту відповідностей, тоді як видалення стоп-слів та стемінг/лематизація можуть призводити до помилкових відповідностей. Для вирішення цієї проблеми автори запропонували контекстно-орієнтований підхід до виправлення помилок у процесі обробки тексту, що дозволяє зменшити кількість хибних відповідностей та покращити загальну ефективність узгодження онтологій.

1.6 Висновок до першого розділу

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» розглянуто поняття текстових даних у контексті штучного інтелекту, зокрема їхню роль як джерела семантичної інформації для аналізу, навчання та прийняття рішень. Описано особливості природної мови, які ускладнюють її обробку – неоднозначність, контекстозалежність та соціокультурні відтінки значення. Розкрито ключові етапи інтелектуального аналізу та класифікації текстових даних. Наведено приклади застосування NLP у різних галузях, таких як медицина, право, соціальні мережі, а також висвітлено значення багатомовних моделей для глобальної обробки інформації.

Також у розділі розглянуто виклики, пов’язані з високою варіативністю текстів залежно від домену, що вимагає адаптації методів до контексту завдання. Підкреслено важливість якісної підготовки тексту для ефективності моделей ШІ, оскільки саме від представлення даних залежить точність та успішність класифікації, прогнозування чи генерації.

2 МЕТОДИ ПОПЕРЕДНЬОЇ ОБРОБКИ NLP ТА МОДЕЛІ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ДАНИХ

2.1 Методи попередньої обробки NLP

У сучасних дослідженнях у сфері обробки природної мов попередня обробка тексту вважається ключовим етапом, що суттєво впливає на результати машинного навчання. Різноманітні прикладні дослідження підтверджують, що ефективність моделей залежить не лише від архітектури, а й від якості попередньої трансформації текстових даних.

Попередня обробка текстових даних є критично важливою на етапі класифікації, оскільки вона дозволяє зменшити шум в даних і покращити якість навчання моделі. Попередня обробка дозволяє позбутися таких елементів, а також знижує розмірність вхідних даних, що допомагає зменшити складність моделі та покращити її ефективність. Без належної обробки навіть найкращий алгоритм класифікації може дати не точні результати.

Основні методи попередньої обробки текстових даних включають [22]:

- нормалізацію;
- токенізацію;
- видалення стоп-слів;
- стемінг;
- лематизацію.

2.1.1 Нормалізація тексту

Нормалізація тексту є базовим, але надзвичайно важливим етапом попередньої обробки текстових даних. Вона передбачає приведення тексту до уніфікованого вигляду для зменшення варіативності лексичних форм, що не несуть додаткового значення. Одним із перших кроків є перетворення всіх символів у нижній регістр. Це дозволяє уникнути подвійного урахування одного й того ж слова у різних написаннях, наприклад, «Data» та «data». Видалення

знаків пунктуації, спеціальних символів, чисел або HTML-розмітки також є типовими кроками на цьому етапі, оскільки такі елементи зазвичай не мають суттєвого впливу на семантику повідомлення в контексті класифікаційних завдань. [23]

Крім того, нормалізація може включати в себе видалення надлишкових пробілів, заміну скорочень на повні форми або уніфікацію специфічних графічних елементів (наприклад, лапок або тире). У деяких випадках можуть застосовуватись правила транслітерації або приведення тексту до єдиної мови, якщо корпус багатомовний. Такі процедури значно спрощують подальше кодування тексту у вектори, знижують кількість унікальних токенів та зменшують розмірність вхідного простору, що сприяє кращому узагальненню моделей машинного навчання.

На рисунку 2.1 наведено приклад нормалізації текстових даних

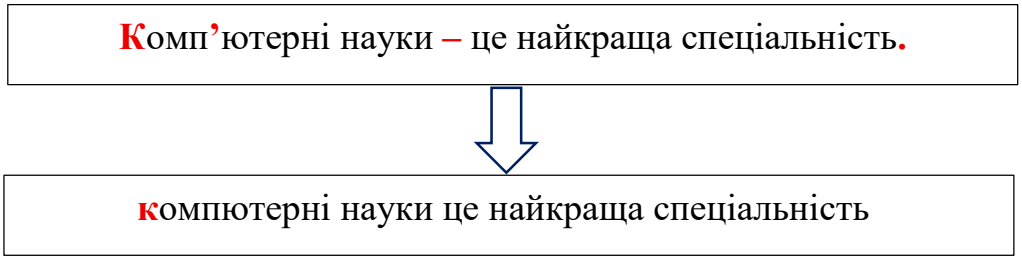


Рисунок 2.1 – Приклад нормалізації тексту

На перший погляд видається, що процес нормалізації є інтуїтивно зрозумілим і легким в імплементації. Проте, не все так просто. Видалення цифр може призвести до втрати важливої інформації, а перетворення чисел у текстовий формат може бути складнішим завданням ніж очікується. Скажімо3-ій, можна перетворити, як три-ій, що буде некоректно. Або ж цифра 7 може бути складовою числа 798, де її потрібно прочитати як сімсот, а може бути числом 7, а може бути порядковим числом сьомий.

Аналогічно існують труднощі обробки часових даних та дат. Адже форматів дат існує чимало (наприклад 01/05/2020 або ж 01.05.2020 або 1-05-2020. У випадку, якщо дати є важливими і несуть цінну для класифікації інформацію,

можна прописувати правила з регулярними виразами для коректного перетворення дат. Після нормалізації тексту потрібно аналізувати список утворених нових слів і перевірити їх на відповідність контексту.

Можуть також існувати труднощі з присутністю в тексті url-адрес, різноманітних скорочень, особливо тих, в яких використовуються певні розділові знаки. Якщо корпус тексту багатомовний, і ми прагнемо привести його до однієї мови, то необхідно використовувати правила транслітерації для власних імен, проте потрібно робити це обережно, адже US – це не ЮС. В такому випадку доцільно спершу виконати заміну скорочень на їх повні відповідники, що вже є нелегкою задачею.

Як бачимо, лише перша техніка нормалізації текстових даних має багато підводних каменів і вимагає неабияких вмінь від експерта, що працює над задачею.

2.1.2 Токенізація тексту

Токенізація тексту – це процес розділення тексту на окремі одиниці, звані токенами, що можуть бути словами, фразами або іншими значущими елементами. Це перший і важливий етап попередньої обробки текстових даних в обробці природної мови. Токенізація дозволяє перетворити текстовий документ на набір елементів, з якими можна працювати далі, наприклад, підраховувати частоти слів або використовувати їх для моделювання [24]. На рисунку 2.1 наведено приклад токенізацію тексту за словами.

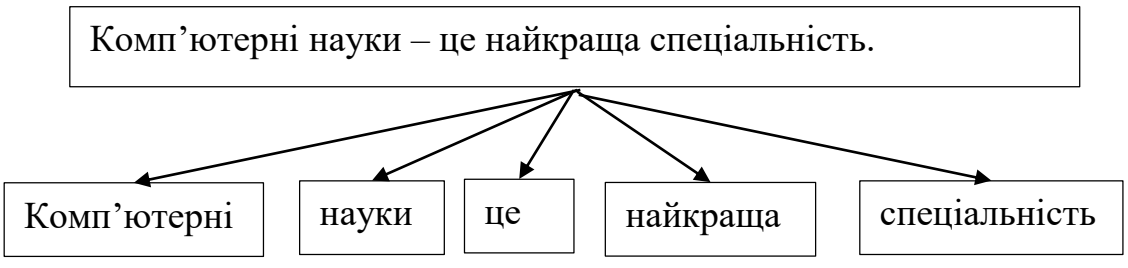


Рисунок 2.2 – Приклад токенізації по словах-токенах

З токенізацією теж не все так просто. Токенами можуть бути:

- речення;
- окремі слова;
- числа;
- знаки пунктуації;
- символи;
- скорочення;
- морфеми (рідше);
- емодзі, хештеги, згадки (у випадку соцмереж).

У випадку, якщо ділити текст на токени речення, здоровою ідеєю виглядає взяти за основу поділу крапки. Але можуть бути речення списками, де в кінці списку крапка, або ж можливі формати дат з крапками, або ж в кінці тексту може бути символ знаку оклику або знак питання.

У випадку виділення окремих слів можемо зустрітись з випадком, коли різними словами будуть вважатись одне і те ж слово, проте вкінці якого є розділовий знак. Тому доцільно виконати нормалізацію тексту перед токенізацією і очистити текст від розділових знаків. У випадку визначення емоційного забарвлення тексту важливим є виділення окремих емоджів, які можуть бути абсолютно зайвою інформацією при розв'язанні інших задач NLP. Перелік таких прикладів можна доповнювати, якщо набір даних – досить великий. Тому раціональним підходом є створення власних методів нормалізації та токенізації за допомогою пробілів з доповненням їх набором регулярних виразів. Таким чином можна створювати власний токенізатор під конкретне конкретну задачу.

2.1.3 Стоп-слова

Стоп-слова – це службові частини мови або загальновживані слова, які хоча й часто трапляються в текстах, зазвичай не мають суттєвого впливу на зміст аналізованого повідомлення. Слова на кшталт they, there, this, where, in, at, on тощо є прикладами таких термінів. Їх видалення дозволяє зменшити розмір

словника та покращити продуктивність моделі, не втрачаючи при цьому важливої інформації. В українській мові прикладами стоп-слів можуть бути:

- Займенники: я, він, вона, це, їх, його.
- Прийменники: в, на, з, до, по, під.
- Зайві слова: взагалі, якби, нібито, також, просто.
- Вигуки: ну, так, що, а.
- Шумові слова: Ну, як.
- Слова, які часто зустрічаються на вебсайтах: сайт, питання, інтернет, відповіді, чат, прайс, замовлення.

Загальний список шумових слів запропонованих Купрієнко можна знайти за посиланням [29]

Проте, як і в попередніх випадках існують певні виклики при видаленні стоп-слів, зокрема, може бути втрата смислової семантики (наприклад у фразі «Я *НЕ* хочу вчитися». Для класифікації теми – стоп-слова можуть бути зайвими. Але для семантичного аналізу, систем питання-відповідь, перекладу, аналізу тону – вони можуть бути критично важливими. Стоп-слова часто змінюють форму: *"це", "цього", "цієї"* тощо. Без попередньої лематизації вони можуть не бути розпізнані як стоп-слова.

2.1.4 Стемінг

Стемінг і лематизація – це споріднені методи в обробці природної мови, обидва належать до етапу попередньої обробки (preprocessing) та нормалізації тексту, а отже є одним з етапів напрямку розуміння природної мови. Обидва методи намагаються привести слова до "основної форми", проте роблять це по різному. Стемінг урізає слова до короткої основи без розуміння значення, натомість лематизація приводить слово до леми – початкової форми [30].

З одного боку, техніка стемінг реалізується шляхом відсікання будь-якого кінця або початку слова, відстежуючи загальновживані префікси і суфікси, що дозволяє нормалізувати слова та зменшити їх кількість для аналізу. Проте з іншого боку ця «груба» техніка може давати наступний результат:

- Слово=studies, суфікс = -es, стема = studi.
- Слово =studying, суфікс =-ing, стема=study.

Як бачимо, стемінг утворив з двох однакових за змістом слів два різних об’єкти в зв’язку з тим, що ця техніка просто обрізає закінчення слів, незалежно від того, чи має отриманий результат якесь змістовне значення. Крім того, як показують дослідження результат стемінгу важко прогнозувати, оскільки він буде різнитися в залежності від обраного алгоритму. Популярна бібліотека nltk в Python містить низку стемерів, які можна використовувати. Експериментальні дослідження показали, що RegexpStemmer очікувано трансформує «studying» до «study», проте «studies» стає «studie». Натомість PorterStemmer та SnowballStemmer дають результат для двох слів «studi».

Для нашого попереднього прикладу стемінг може виглядати наступним чином (див. рисунок 2.3)

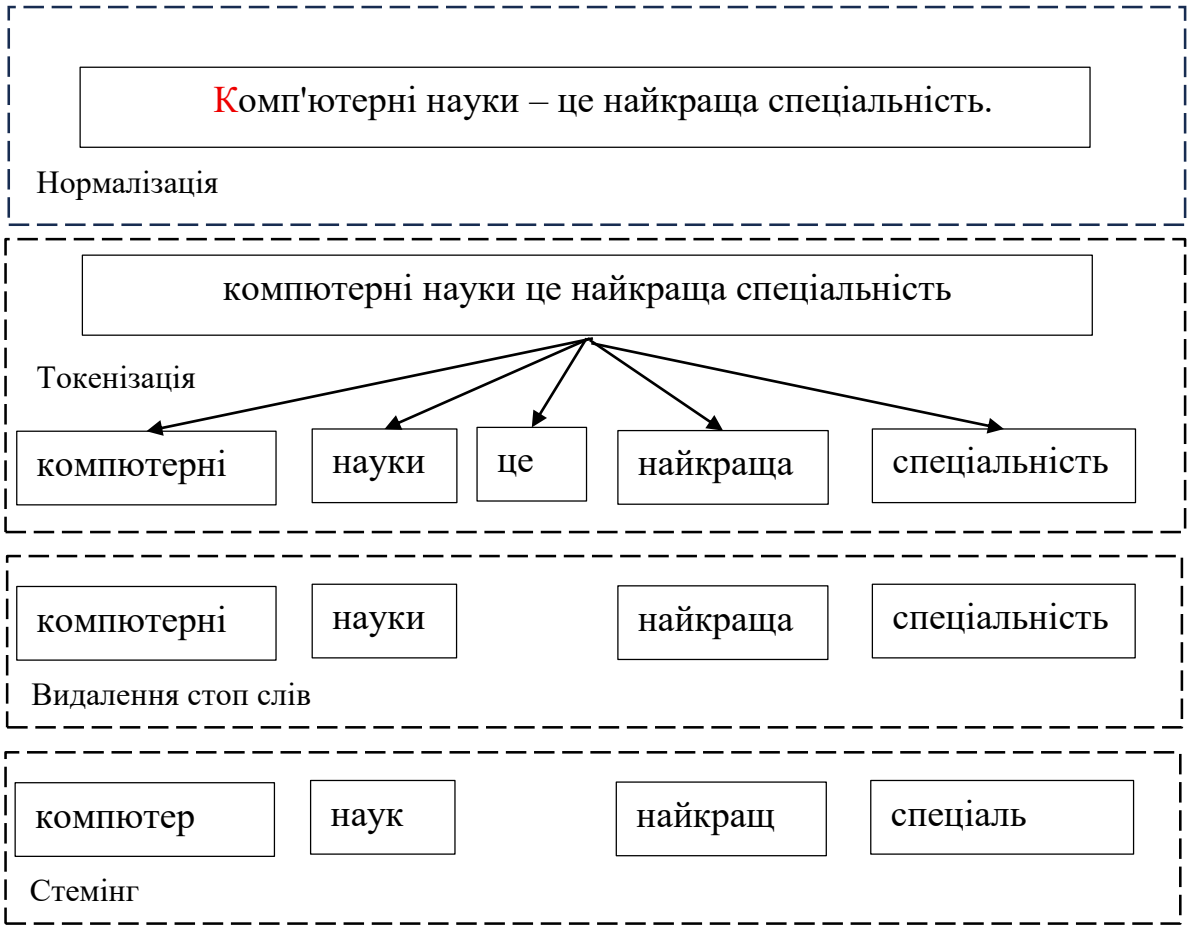


Рисунок 2.3 – Приклад комбінації традиційних підходів нормалізації тексту зі стемінгом

Насправді, для української мови немає такого ж точного та поширеного стемера, як для англійської. Тому необхідно створювати власні стемери у випадку аналізу українських текстів, що є далеко непростою задачею.

2.1.5 Лематизація

Натомість лематизація, яка шукає основну форму слова, завжди приведе ці одного початкового слова:

- Слово=studies, лема = study.
- Слово =studying, лема =study.

Хоча виконання стемінгу є простішим, а отже швидшим ніж процес лематизації, остання все ж є кращим вибором для задач, що вимагають кращої точності. Для лематизації потрібне глибоке лінгвістичне розуміння, щоб сформувати словник, який дозволяє алгоритму знаходити значущу частину слова [31]. Лематизація враховує морфологічні ознаки: рід, число, відмінок, час, вид, ступінь тощо. Вона дозволяє звести всі граматичні варіації слова до однієї форми, що суттєво покращує якість обробки тексту – зокрема у таких задачах, як класифікація, тематичний аналіз, пошукова оптимізація та машинний переклад. Завдяки лематизації, алгоритм може розпізнавати, що слова «найкраща», «найкращі» і «найкращий» є одним і тим самим поняттям.

Для української мови лематизація ускладнюється багатою флективністю (відмінюванням), великою кількістю словоформ та контекстною неоднозначністю. Наприклад, слово *«краще»* може бути і прикметником, і прислівником, що потребує точного аналізу контексту. Крім того, у лексиконі часто трапляються діалектизми, неологізми або суржик, які важко обробити без спеціалізованих словників або моделей. Незважаючи на ці виклики, існують ефективні інструменти лематизації для української, як-от **Stanza** від Stanford NLP, які враховують контекст і граматику слова.

На рисунку 2.4 наведено приклад лематизації для речення українською мовою.

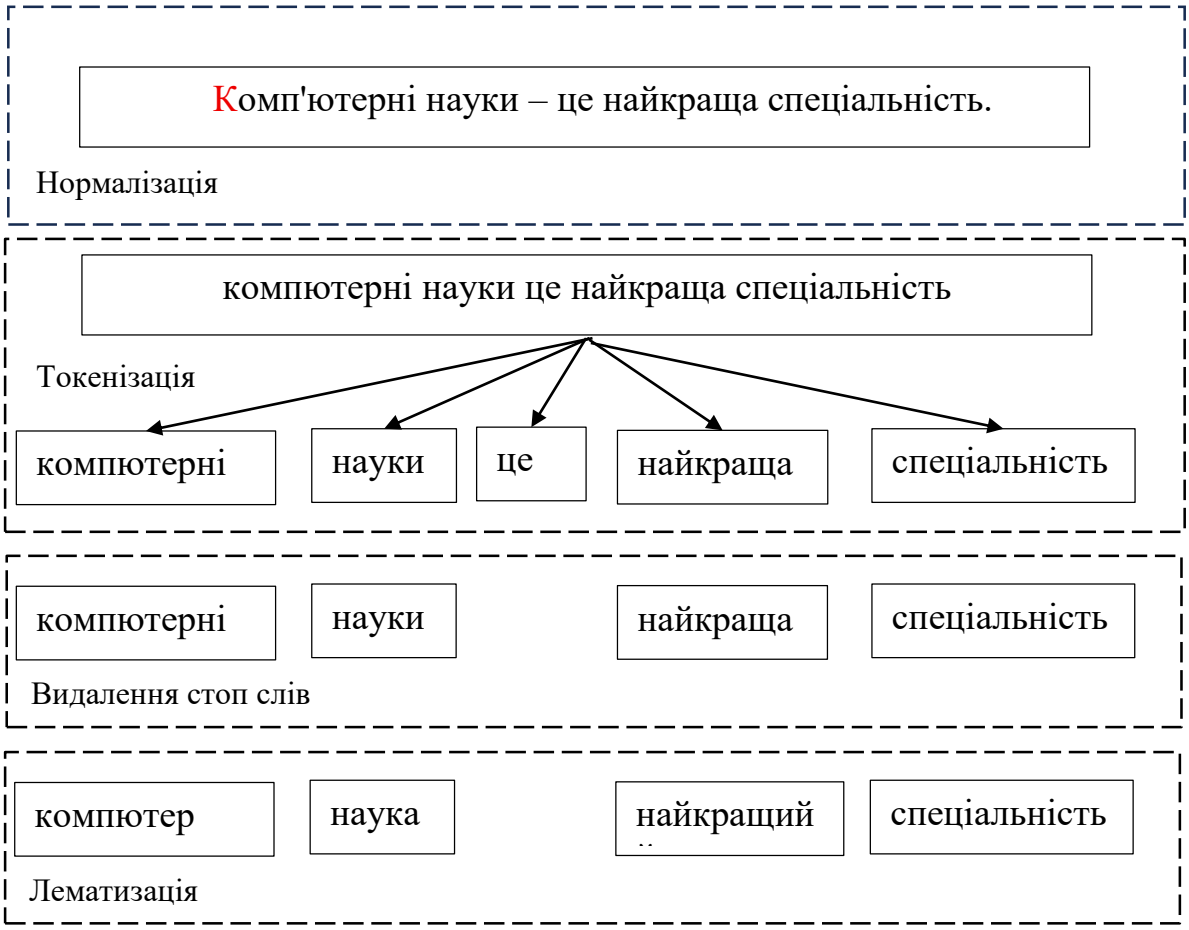


Рисунок 2.4 - Приклад комбінації традиційних підходів нормалізації тексту зі лематизацією

У цьому прикладі слово «найкраща» було приведене до леми «найкращий», оскільки лематизація зводить прикметники до словникової (чоловічої, однини, називного відмінка) форми, незалежно від їхньої граматичної ролі в реченні. Отже, лематизація не зберігає рід або число, а фокусується на загальній словниковій формі, що пояснює появу «найкращий» замість «найкраща». Це корисно для пошуку або класифікації, але може втратити граматичні нюанси речення.

2.2 Основні моделі класифікації текстових даних

Моделі класифікації тексту є ключовими інструментами в NLP і застосовуються для автоматичного визначення категорії або класу, до якого належить певний текстовий фрагмент. Вони використовуються в широкому

спектрі задач – від фільтрації спаму, аналізу настроїв та виявлення тем, до рекомендаційних систем і чат-ботів. Існують різні підходи до побудови таких моделей: традиційні методи машинного навчання, нейронні мережі, а також сучасні трансформерні архітектури. Вибір конкретної моделі залежить від типу даних, обсягу доступної інформації, ресурсоємності задачі та бажаного рівня точності.

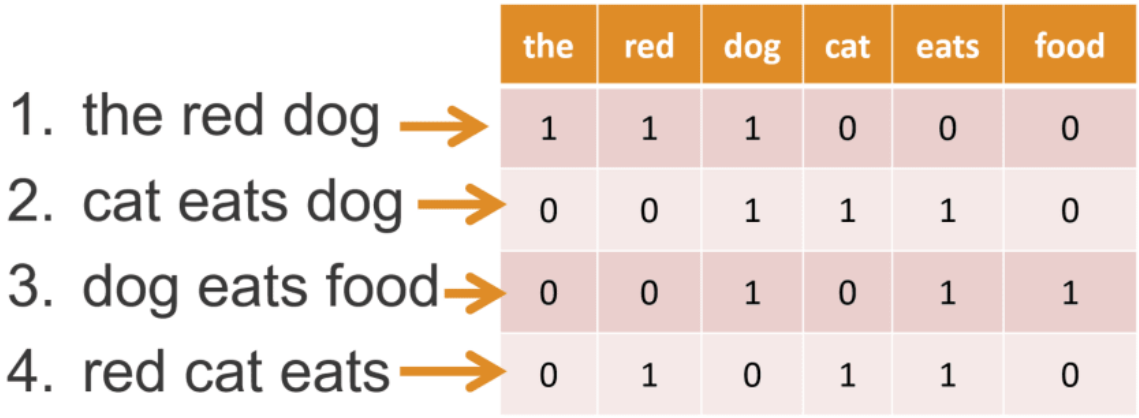
NLP стрімко розвивається завдяки глибокому навчанню та великим мовним моделям, проте традиційні (класичні) моделі машинного навчання залишаються актуальними у багатьох випадках. Вони є швидкими, інтерпретованими, часто менш вимогливими до обчислювальних ресурсів, і добре працюють на невеликих або середніх наборах даних. У цьому підрозділі розглянемо найпопулярніші традиційні моделі, які застосовуються для класифікації тексту: Naive Bayes, Logistic Regression, SVM, дерева рішень, Random Forest і методи бустингу (наприклад, XGBoost, LightGBM).

2.2.1 Формування простору ознак для моделей класичного навчання

Після токенізації та попередньої обробки ми отримуємо послідовності токенів різної довжини, але проблема полягає в тому, що алгоритми машинного навчання потребують векторів чисел фіксованої довжини. Перед подачею тексту на модель, його потрібно перетворити у форму, яку модель здатна обробити. Для цього використовуються два основних підходи:

- Bag of Words (BoW).
- TF-IDF (Term Frequency – Inverse Document Frequency).

Отже, найпростішим способом сформувати простір інформативних ознак є так звана сумка слів (Bag of Words). В такому випадку ми формуємо матрицю рядкам якої відповідають об’єкти тексту, а стовпці визначаються кожним окремим словом, яке зустрічається в текстах. В рамках цього підходу просто рахуємо скільки разів кожне слово зустрічається у відповідному тексті. Ілюстрацію цього можна знайти на рисунку 2.5.



	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

Рисунок 2.5 – Ілюстрація методу Bag of Words

Цей підхід називається Сумка слів, тому що ігнорується порядок та зв'язок між словами, а враховується лише їх кількість в кожному тексті. Існують модифікації методу, коли Сумку слів представляють у бінарному вигляді, тобто просто враховують чи зустрічається слово, чи ні, або ж замість абсолютних значень кількості слів використовують частоту їх використання в тексті.

Власне на частотах побудований і наступний підхід формування простору ознак - TF-IDF. Розглянемо його детальніше.

TF-IDF – це один із найпоширеніших способів числового представлення тексту для задач машинного навчання. Його мета – оцінити важливість кожного слова в документі відносно всього корпусу текстів. На відміну від простого підрахунку частоти слів (Bag of Words), TF-IDF надає меншу вагу словам, які часто зустрічаються в усіх документах (наприклад, "і", "це", "так"), і більшу – словам, які є специфічними для певного документа.

TF-IDF складається з двох частин: TF (Term Frequency) – частота появи слова в конкретному документі, і IDF (Inverse Document Frequency) – обернена частота документів, у яких зустрічається це слово.

TF можна порахувати за формулою:

$$\mathrm{TF}\,(w,d) = \frac{number\ of\ word\ w\ in\ document\ d}{total\ number\ of\ words\ in\ document\ d} \qquad (2.1)$$

Тобто TF – це, по факту, відносна частота слова в кожному окремому документі. Натомість IDF визначається як:

$$IDF\,(w, D) = log\left(\frac{total\ number\ of\ documents\ in\ corpus\ D}{number\ of\ documents\ containing\ word\ w}\right) \quad (2.2)$$

Тобто IDF – це обернена частота слова, яка вимірює важливість слова, встановлюючи наскільки рідко слово зустрічається в документах. Ідея полягає в наступному: якщо слово трапляється в багатьох документах, то воно, швидше за все, є загальним і не варте великої ваги. Якщо ж слово з'являється лише в кількох документах, воно вважається більш інформативним і отримує вищу вагу.

Загальна характеристика обчислюється за формулою:

$$TF - IDF\,(w, d) = TF * IDF \quad (2.3)$$

TF-IDF активно використовується в різноманітних NLP-завданнях, зокрема в класифікації тексту, пошуку інформації, побудові рекомендацій та тематичному аналізі. Його перевага – простота та ефективність, особливо при обмежених обчислювальних ресурсах. Проте TF-IDF не враховує порядок слів або контекст, тому в сучасних системах його поступово замінюють більш складні моделі, такі як word embeddings або трансформери. Утім, TF-IDF і досі залишається популярним базовим інструментом для аналізу тексту.

2.2.2 Модель наївного Байеса

Модель наївного Байєса (Naive Bayes) – це проста, але потужна ймовірнісна модель машинного навчання, яка широко використовується для класифікації тексту, зокрема в задачах спам-фільтрації, тонального аналізу, автоматичного тегування та тематичної класифікації. Її основою є теорема Байєса, яка дозволяє обчислити ймовірність того, що певний документ належить до конкретного класу, з урахуванням присутніх у ньому слів.

В класичному вигляді теорема Байєса виглядає наступним чином

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2.4}$$

де *P(c|x)* – апостеріорна ймовірність даного класу с (тобто даного значення цільової змінної) при даному значенні ознаки х, *P(c)* – апріорна ймовірність даного класу, *P(x|c)* – ймовірність даного значення ознаки при даному класі, *P(x)*– апріорна ймовірність даного значення ознаки.

Модель називається "наївною", тому що припускає незалежність ознак між собою, тобто що кожне слово в тексті не залежить від інших. Це спрощення рідко відповідає реальності, але на практиці модель часто працює дуже ефективно.

Варіант формули Байєса для обчислення апостеріорної ймовірності класу *c* за умови наявності слів-ознак $x_1, x_2, \dots x_n$:

$$P(c|x) = P(x_1|c)P(x_2|c) \dots P(x_n|c)P(c) \tag{2.5}$$

В кінцевому випадку обирається клас, для якого ця ймовірність найбільша.

Метод наївного Байєса має кілька переваг: він дуже швидкий у навчанні та класифікації, ефективний на великих обсягах даних і добре працює навіть при малих обсягах навчальних даних, а також є простим у реалізації та інтерпретації. Однак його основний недолік – це припущення про незалежність ознак, що часто не відповідає реальності, особливо коли між словами існують сильні залежності, як у складних або контекстуальних задачах. Це може знижувати точність моделі в таких випадках, порівняно з більш складними алгоритмами, які враховують ці залежності [32].

2.2.3 Логістична регресія

Логістична регресія – це популярний статистичний метод, який використовується для класифікації даних. Вона є особливим випадком лінійної

регресії, де результат моделі обмежується значеннями в діапазоні від 0 до 1, що робить її ідеальною для задач, де потрібно передбачити ймовірність належності об'єкта до певного класу. Зазвичай логістична регресія використовується в задачах бінарної класифікації, наприклад, для визначення, чи є лист спамом чи ні, чи є пацієнт здоровим або хворим, і т.д.

Основна ідея логістичної регресії полягає в тому, щоб моделювати ймовірність належності об'єкта до певного класу. Вона застосовує логістичну функцію (також відому як сигмоїду), щоб перетворити лінійну комбінацію вхідних ознак у ймовірність. Формула для реалізації логістичної регресії виглядає наступним чином

$$P(y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}}, \quad (2.6)$$

де $(\beta_0, \beta_1, \beta_2, \dots \beta_n)$ – коєфіцієнти моделі, що визначаються під час навчання моделі за допомогою методу максимального правдоподібності $X = (x_1, x_2, \dots, x_n)$ – вектор вхідних ознак, у – клас.

Оцінка коефіцієнтів у логістичній регресії здійснюється за допомогою методу максимального правдоподібності. Завдяки цьому методу, параметри моделі $(\beta_0, \beta_1, \beta_2, \dots \beta_n)$ визначаються таким чином, щоб максимізувати ймовірність спостережуваних даних. Це означає, що ми шукаємо такі значення коефіцієнтів, при яких ймовірність того, що модель правильно класифікує спостереження, є найвищою. Для цього використовуються спеціальні алгоритми оптимізації, такі як градієнтний спуск, які дозволяють поступово знаходити оптимальні значення коефіцієнтів, мінімізуючи функцію втрат (функцію правдоподібності). Після цього коефіцієнти інтерпретуються як логарифмічні зміни відношення шансів, що дозволяє оцінити вплив кожної змінної на ймовірність події.

Замість того, щоб передбачати безпосередньо клас, логістична регресія намагається передбачити ймовірність того, що об'єкт належить до класу 1. Якщо ця ймовірність більше 0.5, об'єкт класифікується як клас 1, інакше – як клас 0.

На рисунку 2.6 наведено ілюстрацію роботи логістичної регресії

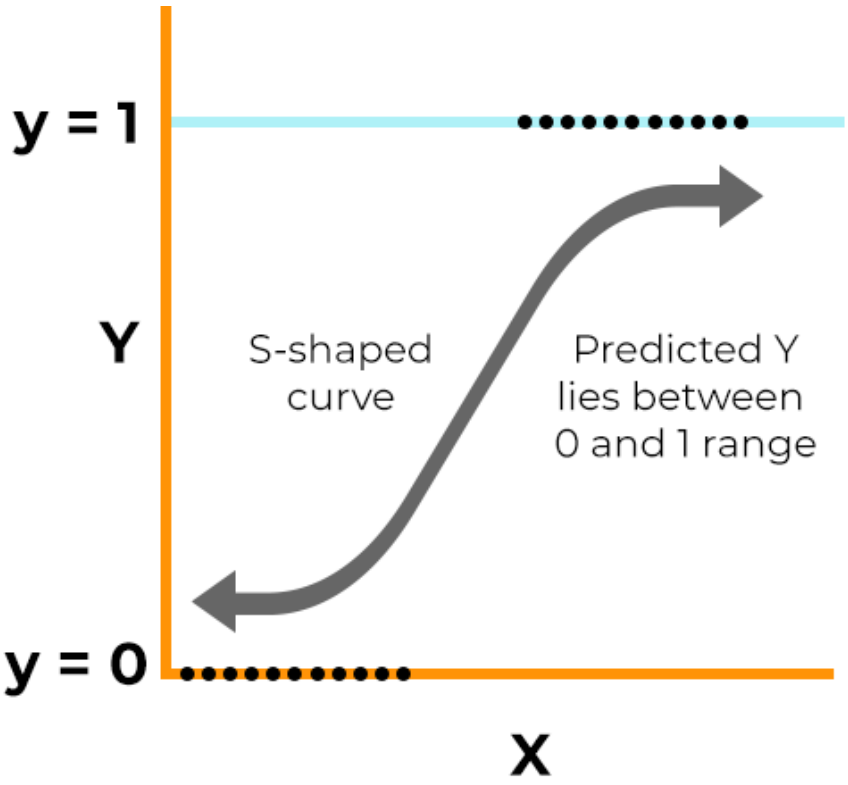


Рисунок 2.6 – Логістична регресія

Логістична регресія є простою, швидкою в навчанні та інтерпретованою моделлю, що дає можливість оцінювати ймовірності належності об'єкта до класу, що робить її корисною для задач бінарної класифікації. Вона добре працює на лінійних даних і швидко застосовується навіть до великих наборів даних. Однак, її головний недолік – припущення лінійності між ознаками та класами, що обмежує її ефективність на нелінійних даних. Крім того, вона чутлива до мультиколінеарності між ознаками і може давати погані результати, коли ознаки сильно корельовані [33].

2.2.4 Метод опорних векторів

Метод опорних векторів (Support Vector Machine, SVM) – це потужний алгоритм машинного навчання, який використовується для задач класифікації та регресії. У контексті класифікації текстів, SVM демонструє високу ефективність завдяки здатності працювати з високовимірними просторами, що характерно для текстових даних після векторизації.

Нехай маємо навчальний набір даних $\{(x_i, y_i)\}_{i=1}^{n}, x_i \in R^d, y_i \in \{-1,1\}$. Тут $X = (x_1, x_2, \dots, x_n)$ – вектор вхідних ознак (наприклад, TF-IDF представлення тексту), y_i– мітка класу.

Задача SVM полягає у знаходженні вектора ваг W та зсуву b, які визначають гіперплощину:

$$W^T X - b = 0 \tag{2.7}$$

На рисунку 2.7 проілюстовано двохвимірний випадок методу SVM

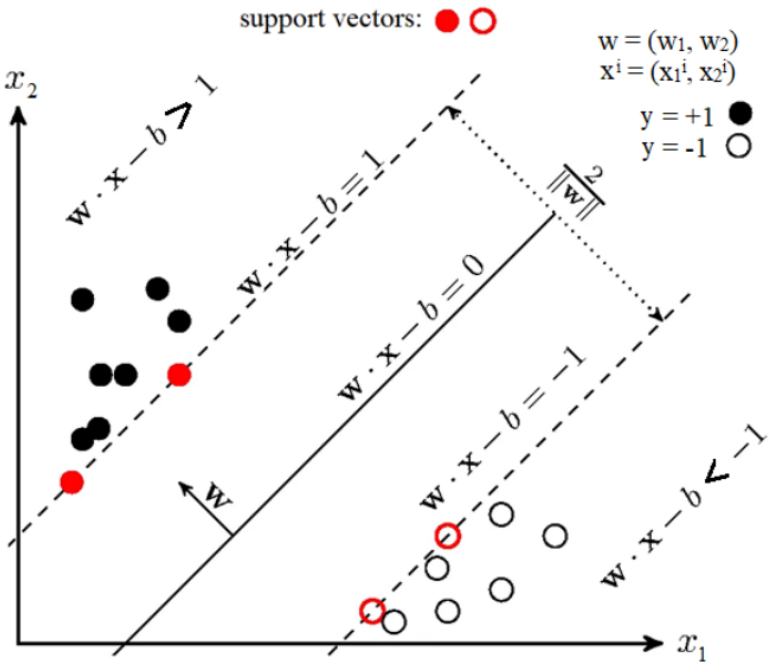


Рисунок 2.7 – Ілюстрація методу опорних векторів

Опорними векторами вважаються дві гіперплощини, які розділяють два класи. В двохвимірному випадку гіперплощини, по факту, є лініями, відстань між якими $\frac{2}{\|W\|}$. Тож очевидно, що для того, щоб максимізувати цю відстань, потрібно мінімізувати $\|W\|$.

Оптимізаційна задача формулюється як:

$$\min_{W,b} \frac{\|W\|^2}{2} \text{ за умови } \; y_i(W^T X - b) > 1 \tag{2.8}$$

Це забезпечує максимальну відстань між класами. Це задача оптимізації з обмеженнями. Її можна вирішити методом множників Лагранжа. Оскільки поверхня квадратична, вона є параболоїдом лише з одним глобальним мінімумом.У випадку, коли дані не є лінійно роздільними вводяться змінні для допущення помилок.

У задачах класифікації текстів, таких як фільтрація спаму, аналіз тональності або категоризація новин, SVM широко використовується завдяки своїм перевагам:

- Висока ефективність у високовимірних просторах: Текстові дані після векторизації мають велику кількість ознак, з якими SVM добре справляється.
- Стійкість до перенавчання: Завдяки максимізації відступу, SVM має хорошу здатність до узагальнення.
- Гнучкість через використання ядер: SVM може використовувати різні ядра (наприклад, лінійне, поліноміальне, RBF) для обробки нелінійно роздільних даних.

Метод опорних векторів є потужним інструментом для класифікації текстів, особливо у випадках з великою кількістю ознак. Його здатність знаходити оптимальну гіперплощину для розділення класів забезпечує високу точність та стійкість до перенавчання, що робить його популярним вибором у задачах обробки природної мови [34].

2.2.5 Інші мовні моделі

Bag-of-Words та TF-IDF підходи дають базове розуміння до підходу обробки мови, проте вже є дещо застарілими. Більш сучасним підходом є числові представлення слів, які захоплюють їхнє семантичне значення та контекст. Слова, які часто зустрічаються в схожих контекстах, мають схожі векторні представлення. До таких моделей належать:

- Word2Vec: Одна з перших і найвпливовіших моделей для створення ембеддингів слів. Вона використовує два основні підходи: Continuous Bag-of-

Words (CBOW), що передбачає центральне слово на основі навколишніх слів та Skip-gram, що передбачає навколишні слова на основі центрального слова. Отже даний підхід побудований на семантичних зв’язках між словами.

- GloVe (Global Vectors for Word Representation): Об'єднує глобальну статистику частотності слів з локальним контекстом для створення векторних представлень, що враховують як локальний, так і глобальний контекст.
- FastText: Розширює Word2Vec, включаючи n-грами символів, що дозволяє краще обробляти рідкісні слова та слова, яких немає в словнику.

Рекурентні нейронні мережі (RNN) були першими нейронними мережами, які могли обробляти послідовності даних, такі як текст. RNN є потужним інструментом для обробки послідовних даних, таких як текст, завдяки здатності зберігати інформацію про попередні елементи послідовності. Це дозволяє моделі враховувати контекст при обробці кожного слова, що є критично важливим для завдань, де порядок слів має значення. У текстовій класифікації RNN ефективно використовуються для аналізу тональності, виявлення спаму, класифікації тематики документів та інших завдань, де важливо розуміти взаємозв'язки між словами. Наприклад, при аналізі відгуків користувачів RNN можуть враховувати контекст, що дозволяє точніше визначити емоційне забарвлення тексту.

Проте класичні RNN мають обмеження у збереженні довготривалих залежностей через проблему зникнення градієнта під час навчання. Для подолання цього були розроблені вдосконалені архітектури, такі як LSTM та модулі з керованими рекурентними одиницями GRU, які дозволяють ефективніше зберігати інформацію на довгих відстанях у тексті. Ці моделі широко застосовуються в задачах машинного перекладу, генерації тексту, розпізнавання мови та інших областях, де необхідно враховувати довготривалі залежності в послідовностях.

Трансформери - це архітектура, яка революціонізувала NLP і стала основою для більшості сучасних передових моделей. На відміну від RNN, трансформери обробляють послідовності паралельно, використовуючи механізми уваги (attention mechanisms), що дозволяє їм краще захоплювати довгострокові залежності та розуміти контекст.

Популярні трансформерні моделі:

- BERT (Bidirectional Encoder Representations from Transformers): Модель, яка навчається на завданнях маскування слів та передбачення наступного речення, що дозволяє їй захоплювати глибокі бінаправлені контекстуальні представлення [26].
- GPT (Generative Pre-trained Transformer): Авторегресивна модель, яка генерує текст, передбачаючи наступне слово на основі попередніх, що робить її ефективною для завдань генерації тексту.
- T5 (Text-To-Text Transfer Transformer): Уніфікує всі NLP-завдання в форматі "текст у текст", що дозволяє моделі вирішувати різноманітні завдання, такі як переклад, узагальнення та відповідь на запитання, за допомогою єдиної архітектури.

Трансформери застосовуються в широкому спектрі завдань, включаючи машинний переклад, узагальнення тексту, відповіді на запитання, аналіз тональності та багато інших.

В таблиці 2.1 наведено порівняльний аналіз між сучасними моделями обробки текстових даних:

Таблиця 2.1 – Порівняльний аналіз сучасних моделей NLP

Характеристика	**Векторні моделі**	**Трансформери**
Представлення контексту	Обмежене або відсутнє	Глибоке контекстуальне
Порядок слів	Ігнорується або частково	Повністю враховується
Навчання	Часто не потребує	Потребує великих обсягів даних
Обчислювальні ресурси	Низькі	Високі
Гнучкість у завданнях	Обмежена	Висока

Векторні моделі є простими у реалізації та ефективними для базових завдань, тоді як трансформери забезпечують високу точність і гнучкість у складних NLP-завданнях, хоча й вимагають значних обчислювальних ресурсів. Таким чином, вибір між векторними моделями та трансформерами залежить від

конкретного завдання, доступних ресурсів та вимог до точності. У багатьох випадках комбінування обох підходів може дати найкращі результати.

2.3 Оцінка точності класифікаційних моделей

Розділення даних на навчальний і тестовий набори є фундаментальним етапом у процесі побудови класифікаційних моделей машинного навчання. Навчальний набір використовується для навчання моделі, дозволяючи їй виявляти закономірності в даних. Тестовий набірслужить для оцінки здатності моделі узагальнювати знання на нові, невідомі дані. Це розділення дозволяє виявити проблему перенавчання, коли модель добре працює на навчальних даних, але погано – на нових. Таким чином, тестовий набір забезпечує об'єктивну оцінку ефективності моделі в реальних умовах.

Матриця помилок (confusion matrix) є базовим інструментом для оцінки продуктивності класифікаційної моделі. Вона відображає кількість правильних і неправильних передбачень, розділених на чотири категорії:

- TP (True Positive): кількість випадків, коли модель правильно передбачила позитивний клас.
- TN (True Negative): кількість випадків, коли модель правильно передбачила негативний клас.
- FP (False Positive): кількість випадків, коли модель помилково передбачила позитивний клас.
- FN (False Negative): кількість випадків, коли модель помилково передбачила негативний клас.

Ідея матриці помилок зображена на рисунку 2.8.

Actually Negative	**True Negative (TN)**	**False Positive (FP)**
Actual Positive	False Negative (FN)	True Positive (TP)
	Predicted Negative	**Predicted Positive**

Рисунок 2.8 - Confusion Matrix

Ця матриця є основою для розрахунку багатьох інших метрик, таких як точність, повнота та F1-міра.

Точність (Accuracy) визначає загальну частку правильних передбачень серед усіх передбачень:

$$\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP} \tag{2.9}$$

Ця метрика є інтуїтивно зрозумілою та часто використовується як первинний показник ефективності моделі. Проте вона може бути оманливою в умовах незбалансованих даних, де один клас значно переважає інший.

Влучність (Precision) визначає точність передбачень позитивного класу:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.10}$$

Повнота (Recall) визначає здатність моделі виявляти всі позитивні приклади:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.11}$$

Ці метрики особливо важливі в задачах, де помилкові передбачення мають різну вартість, наприклад, у медичній діагностиці або виявленні шахрайства.

F1-міра є гармонічним середнім між влучністю та повнотою:

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.12}$$

Ця метрика забезпечує баланс між влучністю та повнотою, особливо корисна в умовах незбалансованих класів.

Коефіцієнт кореляції Метьюза (MCC) є метрикою, яка враховує всі чотири категорії матриці плутанини та є особливо корисною в умовах незбалансованих даних:

$$\text{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.13)$$

Візуальним інструментом оцінки якості класифікаційної моделі є ROC-крива, що відображає співвідношення між повнотою (Recall) та специфічністю (1 - False Positive Rate) при різних порогах класифікації. Площа під кривою (AUC) використовується як узагальнений показник ефективності моделі. На рисунку 2.9 наведено приклад ROC-кривої.

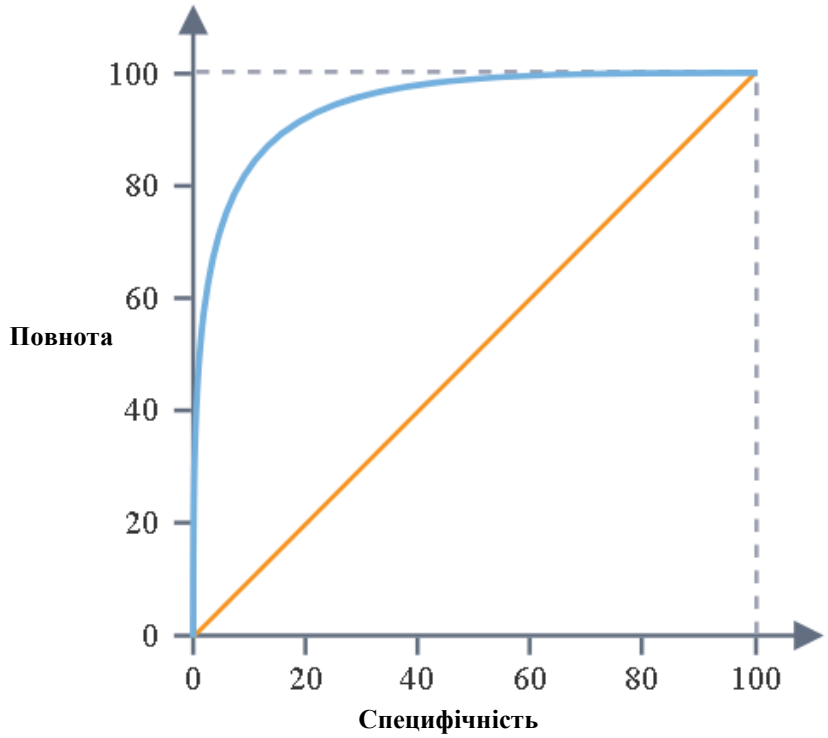


Рисунок 2.9 - Приклад ROC-кривої

У випадку ідеального класифікатора ROC-крива проходить через верхній лівий кут графіка. Це означає, що модель досягає 100% чутливості (усі позитивні випадки правильно класифіковані) при нульовій частці хибнопозитивних передбачень. Чим ближче ROC-крива до цього кута, тим вища прогностична здатність моделі.

Навпаки, якщо ROC-крива наближається до діагоналі, це свідчить про те, що модель працює як випадкове вгадування. Така діагональ відповідає класифікатору, який є неінформативним.

Оцінка точності класифікаційних моделей є багатогранним процесом, що вимагає врахування різних метрик та їхнього взаємозв'язку. Вибір відповідних метрик та інструментів оцінки залежить від специфіки задачі, структури даних та вимог до моделі. Ретельний аналіз результатів оцінки дозволяє не лише визначити ефективність моделі, але й виявити напрямки для її покращення.

2.4 Висновок до другого розділу

В другому розділі кваліфікаційної роботи У другому розділі роботи розглянуто ключові етапи класифікації текстових даних, починаючи з попередньої обробки тексту. Цей процес включає нормалізацію, токенізацію, видалення стоп-слів, стемінг та лематизацію. Ці кроки спрямовані на зменшення розмірності даних та покращення якості вхідних даних для моделей машинного навчання. Наприклад, токенізація допомагає розбити складні тексти на керовані одиниці, а видалення стоп-слів зменшує розмірність даних для більш ефективної обробки.

Далі розглянуто основні традиційні моделі класифікації тексту, зокрема наївний байєсівський класифікатор, логістичну регресію та метод опорних векторів. Метод наївного Байєсу базується на ймовірнісному підході та припущенні незалежності ознак, логістична регресія моделює ймовірність належності до певного класу, а метод опорних векторів шукає оптимальну гіперплощину для розділення класів. Кожна з цих моделей має свої переваги та обмеження, що робить їх придатними для різних типів задач класифікації тексту. Також зроблено огляд сучасних моделей NL. Розглянуто метрики оцінки точності класифікаційних моделей.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ВПЛИВУ ПОПЕРЕДНЬОЇ ОБРОБКИ ТЕКСТОВИХ ДАНИХ НА ЯКІСТЬ КЛАСИФІКАЦІЇ

3.1 Вибір середовища

Python є однією з найпопулярніших мов програмування для аналізу текстових даних завдяки своїй простоті, гнучкості та потужному стандартному функціоналу. У цьому підрозділі розглянемо ключові причини, чому саме Python є оптимальним вибором для обробки та аналізу тексту без використання сторонніх бібліотек.

Python має інтуїтивно зрозумілий синтаксис, що нагадує англійську мову, що робить його легким для вивчення навіть для початківців. Це дозволяє швидко розпочати роботу з аналізом текстових даних без глибоких знань у програмуванні. Крім того, Python є відкритим програмним забезпеченням, що означає вільний доступ до великої кількості ресурсів, документації та спільноти користувачів, які готові допомогти новачкам.

Python підтримує різні парадигми програмування, включаючи об'єктно-орієнтоване, функціональне та процедурне програмування, що надає розробникам гнучкість у виборі стилю кодування. Крім того, Python легко інтегрується з іншими мовами та платформами, що дозволяє масштабувати рішення та використовувати їх у різних середовищах, включаючи веб-додатки, хмарні сервіси та мобільні платформи.

Python має велику та активну спільноту розробників, які постійно вдосконалюють існуючі інструменти та створюють нові рішення для обробки текстових даних. Це забезпечує швидкий доступ до оновлень, виправлень помилок та нових функцій, а також до великої кількості навчальних матеріалів, форумів та обговорень, що полегшує вирішення проблем та обмін досвідом.

Однією з головних переваг Python є наявність широкого спектру вбудованих функцій для обробки тексту та аналізу даних без необхідності використання сторонніх бібліотек:

- Робота з рядками. Python надає потужні можливості для обробки тексту, включаючи методи для пошуку, заміни, розділення та об'єднання рядків.
- Регулярні вирази. Модуль re дозволяє виконувати складні операції з пошуку та заміни в тексті за допомогою шаблонів.
- Обробка файлів. Python дозволяє легко читати та записувати текстові файли, що є корисним при аналізі великих обсягів текстових даних.
- Колекції даних. Вбудовані структури даних, такі як списки, словники та множини, дозволяють ефективно зберігати та обробляти текстову інформацію.

На рисунку 3.1 наведено переваги і недоліки застосування Python:

Рисунок 3.1 - Переваги і недоліки застосування Python

Представлена схема наочно демонструє сильні та слабкі сторони мови програмування Python. Серед ключових переваг виділяються її цінність для наук про дані завдяки багатій екосистемі бібліотек, можливість вбудовування інтерпретатора в інші програми, значні можливості для IoT, підтримка об'єктно-орієнтованого програмування, а також розширюваність за допомогою модулів, написаних іншими мовами. Додатково, Python вирізняється підвищеною продуктивністю розробки завдяки своєму читабельному та лаконічному синтаксису, а також є портативним, безкоштовним та з відкритим вихідним кодом, що робить його доступним та універсальним інструментом.

Водночас, схема вказує на певні недоліки Python. До них належать обмеження швидкості, що є типовим для інтерпретованих мов, а також слабка

підтримка у сфері мобільних обчислень порівняно з іншими спеціалізованими мовами.

Попри деякі недоліки, завдяки своїй простоті, потужному стандартному функціоналу, гнучкості та активній спільноті Python є ідеальним вибором для аналізу текстових даних. Він дозволяє ефективно вирішувати широкий спектр завдань – від базової обробки тексту до побудови складних моделей машинного навчання та глибокого навчання. Це робить Python незамінним інструментом для дослідників, аналітиків та розробників, які працюють з текстовими даними.

3.2 Огляд бібліотек та деяких основних функцій реалізації

3.2.1 Основні бібліотеки для NLP

Однією з головних переваг Python є наявність широкого спектру бібліотек для обробки тексту та аналізу даних [28]. На рисунку 3.2 приведено перелік основних бібліотек, що використовуються в NLP.

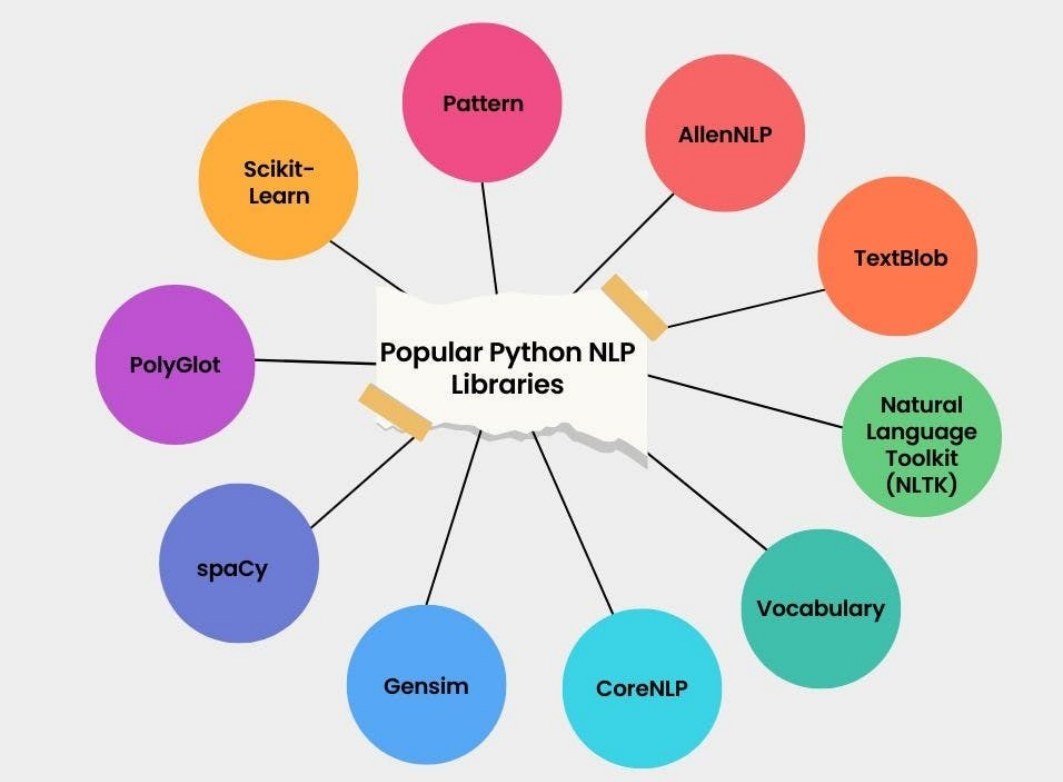


Рисунок 3.2 – Перелік основних бібліотек NLP

Коротко опишемо кожну з цих бібліотек:

- Natural Language Toolkit (NLTK): потужна бібліотека, що підтримує такі завдання та операції, як класифікація, парсинг, тегування, семантичний аналіз, токенізація та стеммінг у Python.
- TextBlob: важлива бібліотека для розробників, які починають свій шлях в обробці природної мови на Python. Вона надає простий API для виконання поширених завдань NLP, таких як тегування частин мови, вилучення іменних фраз, аналіз тональності та переклад.
- SpaCy: відносно молода, але дуже популярна та продуктивна бібліотека для NLP. Вона орієнтована на ефективність та дозволяє обробляти великі обсяги тексту, надаючи готові моделі для багатьох мов.
- Gensim: потужна бібліотека, яка займається виявленням семантичної подібності між двома документами за допомогою інструментарію моделювання тем та моделювання векторних просторів.
- CoreNLP: пропонує широкий спектр лінгвістичних інструментів, таких як токенізація, тегування частин мови, виявлення іменованих сутностей, парсинг та аналіз тональності. Часто використовується через обгортки (wrappers) для Python.
- Vocabulary: відноситься до бібліотек, які надають функціонал для роботи зі словниками, лексикою та синонімами для покращення розуміння тексту.
- Scikit-Learn: надає розробникам широкий спектр алгоритмів для побудови моделей машинного навчання (ML) та інших процесів. Хоча вона не є виключно NLP-бібліотекою, її алгоритми часто застосовуються для класифікації тексту, кластеризації та інших завдань у сфері обробки природної мови.
- Pattern: може бути використана для широкого спектру проектів природної мови, включаючи добування даних з вебу, обробку природної мови, машинне навчання та візуалізацію мереж.
- AllenNLP: один з найдосконаліших інструментів обробки природної мови, що ідеально підходить для бізнесу та дослідницьких застосувань.

Побудований на PyTorch, він надає гнучку архітектуру для розробки нових моделей NLP.

- PolyGlot: Менш відома бібліотека Python, але вона забезпечує широке мовне покриття та глибокий аналіз, підтримуючи багато мов для таких завдань, як визначення мови, токенізація, вилучення іменованих сутностей, тегування частин мови та переклад.

В [35] наведено порівняльний аналіз бібліотек за сильними та слабкими сторонами.

3.2.2 Опис окремих функцій реалізації

Розглянемо окремі функції, що використовувались для експериментальних досліджень. Насамперед для роботи необхідно провести імпорт необхідних бібліотек для обробки та аналізу текстових даних, а також для реалізації моделей машинного навчання(лістинг 3.1)

Лістинг 3.1 – Імпорт необхідних бібліотек

```
import pandas as pd
import nltk
import os
import glob
import math
import re
import pandas as pd
from pathlib import Path
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer, SnowballStemmer,
LancasterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectPercentile,
f_classif
from sklearn.naive_bayes import GaussianNB
from sklearn import linear_model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, recall_score,
precision_score, f1_score
from matplotlib import pyplot as pltimport glob
import pandas as pd
from pathlib import Path
```

Код включає бібліотеки для роботи з датафреймами (pandas), обробки природної мови (nltk), файловою системою (os, glob, pathlib), математичних операцій (math), регулярних виразів (re). Крім того, для векторизації тексту використовується TfidfVectorizer, для відбору ознак - SelectPercentile та f_classif, а для класифікації - GaussianNB, linear_model та KNeighborsClassifier. Також імпортуються функції для оцінки ефективності моделей (accuracy_score, recall_score, precision_score, f1_score) і візуалізації даних (matplotlib.pyplot)

В лістингу 3.2 наведено основні функції для зчитування даних

Лістинг 3.2 – Основні функції для зчитування даних

```
    files = [(Path(f).parent.name, f) for f in
glob.glob("train/**/*.txt", recursive=True)]
    files.sort(key=lambda tup: tup[1])
    for author, file in files:
        print(author, file)
    def load_data(data_dir):
        authors = []
        texts = []
        files = [(Path(f).parent.name, f) for f in
glob.glob(data_dir + '/**/*.txt', recursive=True)]
        files.sort(key=lambda tup: tup[1])
        for author, file in files:
            with open(file, 'r', encoding='utf-8') as f:
                text = f.read()
            authors.append(author)
            texts.append(text)
        return pd.DataFrame({'Author': authors, 'Text': texts})
    def fix_data_types(df):
        df['Author'] = df['Author'].astype(str)
        df['Text'] = df['Text'].astype(str)
        return df
    def load_train_data():
        return fix_data_types(load_data('train'))
    def load_test_data():
        return fix_data_types(load_data('test'))
```

Код лістингу 3.2 призначений для завантаження текстових даних з файлів, організованих за авторами. Спочатку він знаходить усі текстові файли у вказаних директоріях (train або test), групуючи їх за назвою директорії, що відповідає імені автора. Потім функція load_data читає вміст кожного файлу, створюючи DataFrame pandas, де кожен рядок містить ім'я автора та відповідний текст. Функції fix_data_types, load_train_data та load_test_data забезпечують правильне

завантаження та підготовку даних для подальшої обробки, перетворюючи стовпці "Author" та "Text" на рядковий тип.

Для формування простору інформативних ознак використано наступний код, наведений в лістингу 3.3.

Лістинг 3.3 – Код для формування простору ознак

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(sublinear_tf=True, max_df=0.5,
stop_words='english')
features_train_transformed =
vectorizer.fit_transform(train_df['Text'])
features_test_transformed =
vectorizer.transform(test_df['Text'])
```

В лістингу 3.4 приведено код функцій для нормалізації тексту для токенізації текстів

Лістинг 3.4 – Функції для нормалізації та токенізації

```
def normalize_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    return text

def tokenize_text(text):
    stop_words = set(stopwords.words('english'))
    tokens = word_tokenize(text)
    filtered_tokens = [word for word in tokens if word not in
stop_words]
    return filtered_tokens
```

Для перевірки впливу стемінгу та лематизації на якість класифікації було створено нові стовпці датафрейму зі списками стематизованих та лематизованих слів. Код для стемінгу показано в лістингу 3.5. Цей код спочатку нормалізує та токенізує текстові дані в train_df та test_df, створюючи нову колонку 'Processed_Text'. Потім, для кожного токенізованого тексту, застосовується функція stem_words, яка використовує PorterStemmer для стемінгу (зведення слів до їхньої основи), зберігаючи результат у колонці 'Text_stem'.

Лістинг 3.5 – Код для стемінгу

```
    train_df['Processed_Text'] = train_df['Text'].apply(lambda x:
tokenize_text(normalize_text(x)))
    test_df['Processed_Text'] = test_df['Text'].apply(lambda x:
tokenize_text(normalize_text(x)))
    def stem_words(words):
        stemmer = PorterStemmer()
        return [stemmer.stem(word) for word in words]
    train_df['Text_stem'] =
train_df['Processed_Text'].apply(stem_words)
    test_df['Text_stem'] =
test_df['Processed_Text'].apply(stem_words)
    train_df['Text_stem'] =
train_df['Processed_Text'].apply(lambda x: stem_words(x))
    test_df['Text_stem'] = test_df['Processed_Text'].apply(lambda
x: stem_words(x))
```

Приклад побудови моделі та оцінки її точності наведено в лістингу 3.6

Лістинг 3.5 – Побудова моделі та оцінка точності

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
features_train_dense = features_train_transformed.toarray()
clf.fit(features_train_dense, train_df['Author'])
features_test_dense = features_test_transformed.toarray()
predicted = clf.predict(features_test_dense)
accuracy = accuracy_score(test_df['Author'], predicted)
```

Цей фрагмент коду демонструє використання наївного байєсівського класифікатора Гауса (GaussianNB) для прогнозування автора тексту. Після перетворення розріджених матриць ознак на щільні масиви, модель навчається на тренувальних даних, а потім використовується для передбачення авторів для тестових даних, оцінюючи точність за допомогою accuracy_score.

3.3 Опис набору даних

Для дослідження використовувався набір [37]. Цей корпус широко використовується для задач визначення авторства текстів і є підмножиною Reuters Corpus Volume 1 (RCV1). У цьому датасеті кожен автор представлений

рівною кількістю текстів, що забезпечує збалансованість класів для задач класифікації.

У датасеті Reuters-50-50 документи організовані у файловій структурі, де кожен автор має окрему папку, яка містить його тексти. Це стосується як навчального, так і тестового наборів даних. Після розпакування архіву ми отримали дві основні директорії:

- C50train/ – навчальний набір, що містить 2500 документів;
- C50test/ – тестовий набір, що містить 2500 документів.

У кожній з папок знаходяться 50 підкаталогів, названих іменами авторів, наприклад:

- C50train/AlanCrosby/.
- C50train/WilliamKazer/.
- C50test/AlanCrosby/.
- C50test/WilliamKazer/.

Організація датасету Reuters-50-50 у вигляді окремих папок для кожного автора забезпечує зручність обробки даних, оскільки дозволяє легко автоматизувати завантаження та обробку текстів для кожного автора окремо. Завдяки однаковій кількості документів для кожного автора у навчальному та тестовому наборах досягається збалансованість класів, що є важливим для коректного навчання та оцінки моделей класифікації. Крім того, чітке розділення даних на навчальні та тестові папки допомагає уникнути змішування даних та спрощує процес валідації моделей.

Цей датасет є цінним ресурсом для досліджень у галузі обробки природної мови та визначення авторства текстів.

3.4 Результати тестування

Після застосування лістингу 3.3 отримали розріджені матриці для навчального та тестового датасетів розміром 2500x28839. Такий великий обсяг ознак створює певні виклики: він ускладнює обробку і суттєво сповільнює процес класифікації. Тому було прийнято рішення дослідити, як зменшення

кількості ознак вплине на точність класифікації, щоб знайти оптимальний баланс між продуктивністю та якістю моделі. В таблиці 3.1 наведено результати експериментального дослідження залежності точності класифікаційної моделі в залежності від взятої кількості ознак. Потрібно зазначити, що до уваги брали 10%, 20%, 30% , 50% та 100 % усіх ознак. Тестування робили взявши за основу модель наївного Байеса

Таблиця 3.1 – Вплив розміру простору ознак на точність класифікаційної моделі наївного Байєса (GaussianNB)

GaussianNB	**10%**	**20%**	**30%**	**50%**	**100%**
Accuracy	0.577	0.582	0.585	0.594	0.595
Precision	0.610	0.622	0.637	0.643	0.644
Recall	0.577	0.582	0.585	0.594	0.595

Таблиця демонструє ефективність моделі GaussianNB при використанні різної частки ознак (від 10% до 100%). Зі збільшенням кількості ознак спостерігається поступове зростання всіх метрик: Accuracy, Precision та Recall. Найкращі показники досягаються при використанні 100% ознак, де Accuracy становить 0.595, Precision – 0.644, а Recall – 0.595. Наведене дослідження показує, що хоча повний набір ознак забезпечує найвищу точність, значне зменшення їх кількості (на 80%) призводить до мінімальної втрати точності (лише 2%). Цей компроміс є вкрай вигідним, оскільки дозволяє істотно прискорити навчання та прогнозування, зменшити потребу в обчислювальних ресурсах, знизити ризик перенавчання, спростити модель для кращої інтерпретації та підвищити її масштабованість, роблячи розроблене рішення більш практичним та ефективним у реальних умовах.

Незважаючи на те, що модель наївного Баєйеса часто використовується в задачах обробки природньої мови і особливо в спам-фільтрації, бачимо точність визначення авторства з допомогою моделі наївного Байєса є не надто високою. Візьмемо за основу 20% датасету і спробуємо використати інші моделі для

даного датасету. В таблиці 3.2 приведено результати класифікації для різних моделей

Таблиця 3.2 – Результати класифікації для різних моделей

Метрики точності	**GaussianNB**	**KNN**	**SVM**	**Logistic regression**
Accuracy	0.582	0.485	0.673	0.685
Precision	0.622	0.684	0.71	0.697
Recall	0.582	0.485	0.673	0.685

Отримані результати демонструють, що для даного набору даних модель "Логістична регресія" та "SVM" перевершують "GaussianNB" та "KNN" за метриками Accuracy, Precision та Recall. "GaussianNB" показує середні результати, тоді як "KNN" значно відстає від інших моделей, маючи найнижчі показники за оціненими метриками Accuracy та Recall.

Візьмемо за основу модель SVM і дослідимо вплив стоп-слів на точність моделі. При формуванні простору ознак ми використали стандарту функцію TfidfVectorizer. Один раз в налаштуваннях функції, ми обрали використання стоп-слів, а інший раз використали без цієї опції. Експериментальним шляхом було встановлено, що стандартна опція *stopwords='english'* передбачає 247 слів англійської мови, адже простір ознак без вилучення стоп-слів містив 29086 ознак. В таблиці 3.3 наведено отримані експериментальні результати.

Таблиця 3.3 – Вплив вилучення стоп-слів на точність моделі

SVM	**20%**	**20% з вилученням стоп-слів**
Accuracy	0.673	0.665
Precision	0.71	0.702
Recall	0.673	0.665

Попри очікування, що вилучення стоп-слів покращить якість класифікації, бачимо, що для даної задачі, точність моделі, навпаки, навіть трохи зменшилась. Очевидно, що в даному випадку наявність чи відсутність стоп-слів мінімально впливає на точність моделі. Проте, є зміст все-таки вилучати стоп-слова, бо це дає можливість зменшити простір ознак на 247 слів.

І на завершення дослідимо вплив стемінгу та токенізації на точність моделі. Отримані результати експериментальних досліджень наведено в таблиці 3.4

Таблиця 3.4 – Вплив стемінгу та лематизації на точність моделі класифікації

SVM, 20 % ознак	**З вилученням стоп-слів**	**Після стемінгу**	**Після лематизації**	**Стемінг після лематизації**
Accuracy	0.665	0.661	0.66	0.662
Precision	0.702	0.692	0.692	0.695
Recall	0.665	0.661	0.66	0.662

Для моделі SVM, яка показала відносно інших моделей вищу ефективність, вплив таких методів попередньої обробки тексту, як лематизація та стемінг є практично невідчутним. Показник Accuracy після стемінгу та лематизації неочікувано став навіть дещо нижчим, ніж без цих методів обробки. Проте в результаті застосуваввння стемінгу вдалося суттєво знизити простір ознак з 29 тисяч до 21 тисячі. Зважаючи на те, що точність після застосування стемінгу не змінилася, а обсяг даних зменшився майже на 30%, стемінг можна використовувати для скорочення необхідних для класифікації ресурсів. Натомість проведення лематизації вимагає невиправдано великих часових ресурсів, що, як бачимо, в даному конкретному випадку застосування лематизації не призводить до значного покращення, а в навіть трохи знижує ефективність моделі SVM. Таким чином, для цієї моделі та набору даних, лематизація, ймовірно, не є критично необхідним кроком, а інші методи, можуть бути більш ефективними або достатніми. Для даного набору даних доцільніше

йти шляхом налаштування гіперпараметрів моделей або ж застосування нових сучасніших моделей NLP з метою покращення параметрів точності.

3.5 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи обґрунтовано вибір середовища для програмування. Наведено опис основних бібліотек, які використовуються Python для обробки природньої мови. Подано опис окремих функцій, що використовувались для попередньої обробки даних та класифікації текстів. Проведено аналіз впливу розміру простору ознак, а також впливу методів попередньої обробки даних на якість класифікації.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Тема кваліфікаційної роботи освітнього рівня «Магістр» присвячена дослідженню впливу методів попередньої обробки текстових даних на якість класифікаційних моделей. Оскільки, проведення робіт з розробки програмного забезпечення для дослідження передбачає використання комп'ютерної техніки, зокрема ПК та периферійних пристроїв, то обов'язковим є дотримання вимог з охорони праці і техніки безпеки.

Для ефективної і безпечної роботи колективу працівників, в тому числі і фахівців з підвищення ефективності контролю доступу в приміщення, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці. Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Згідно вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом мінрегіону України від 13.11.2014 N 312 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками.

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014. Приміщення, в яких

розміщуються робочі місця операторів сервера загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп'ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97.

У приміщенні, де одночасно експлуатуються понад п'ять комп'ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп'ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання

при відключенні має бути зворотним. Не допускається підключати комп’ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення комп’ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

При підвищенні ефективності контролю доступу в приміщення, де для забезпечення безпеки мешканців, співробітників і збереження майна використовуються ДС, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18. Організація робочого місця фахівця із дослідження методів та програмно-апаратних засобів оптимізаційних процесів на основі ГА повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп’ютера, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля згідно з вимогами ДСанПіН 3.3.2.007-98..

4.2 Безпека в надзвичайних ситуаціях

Тема кваліфікаційної роботи освітнього рівня «Магістр» присвячена дослідженню методів обробки природної мови. Оскільки ми живемо у

військових час, то доцільно розглянути питання про організацію оповіщення і зв’язку у надзвичайних ситуаціях техногенного та природного характеру.

Одним із головних заходів захисту населення від надзвичайних ситуацій (НС) є його своєчасне оповіщення про небезпеку, обстановку, яка склалася внаслідок її реалізації, а також інформування про порядок і правила поведінки в умовах НС. Під час організації оповіщення і доведення інформації до населення України необхідно керуватися вимогами Положення про організацію оповіщення і зв'язку у надзвичайних ситуаціях, затвердженого постановою Кабінету Міністрів України від 15 лютого 1999 року № 192. Кожний громадянин України повинен знати порядок подавання сигналу “Увага всім!”, діяти за ним та іншими сигналами цивільного захисту (ЦЗ) в умовах НС та особливого періоду.

Встановлено, що система оповіщення та інформування у сфері ЦЗ України включає:

- оперативне доведення до відома населення інформації про виникнення або можливу загрозу виникнення НС, у тому числі через загаль¬нодержавну, територіальні і локальні автоматизовані системи централізованого оповіщення;
- завчасне створення та організаційно-технічне поєднання постійно діючих локальних систем оповіщення та інформування населення із спеціальними системами спостереження і контролю (включаючи державну мережу спостереження і лабораторного контролю) в зонах можливого ураження;
- централізоване використання мереж зв'язку, радіомовлення, телебачення та інших технічних засобів передачі інформації незалежно від форми власності та підпорядкування в разі виникнення НС.

Системи оповіщення населення України мають державний, регіональний, місцевий і об'єктовий рівні. Управління системою оповіщення кожного рівня організовується безпосередньо відповідними органами повсякденного управління системи ЦЗ. Рішення на застосування системи оповіщення приймає відповідний голова державної адміністрації (начальник територіальної підсистеми Єдиної системи цивільного захисту). Відповідальність за організацію і практичне здійснення оповіщення несуть керівники органів виконавчої влади,

місцевого самоврядування, підприємств, установ і організацій. Тому керівник об'єкта господарської діяльності і кожний громадянин повинні знати сигнали ЦЗ і уміти правильно за ними діяти.

В результаті наукової розвідки встановлено, що в Єдиній системі ЦЗ України оповіщення населення передбачає спочатку, за будь-якого характеру небезпеки, включення електричних сирен, переривчастий звук яких означає єдиний сигнал небезпеки "Увага всім!". Для вирішення завдань оповіщення на всіх рівнях Єдиної системи ЦЗ створюються спеціальні системи централізованого оповіщення (СЦО). Системою оповіщення будь-якого рівня є організаційно-технічне об'єднання оперативно чергових служб органів управління ЦЗ, спеціальної апаратури управління і засобів оповіщення, а також каналів (ліній зв'язку), які забезпечують передачу команд управління і мовної інформації у НС [39].

СЦО регіонального рівня є основною ланкою системи оповіщення в цілому. Саме з цього рівня планується організація централізованого оповіщення. Завданням СЦО регіонального рівня є оповіщення посадових осіб і сил даного рівня, органів управління, сил місцевого і об'єктового рівнів та їх посадових осіб, а також населення, яке проживає на території, на яку поширюється дія СЦО цього рівня. Інформація, яка доводиться до органів управління і посадових осіб, має оперативний характер, а до населення доводиться інформація про характер і масштаби загрози та про дії в умовах НС, які склалися.

Дослідженням встановлено, що основним способом оповіщення населення про НС в умовах мирного та воєнного часу є передача інформації з використанням державних мереж проводового, радіо і телевізійного мовлення. Для зосередження уваги населення перед передачею інформації вмикаються сирени, виробничі гудки та інші сигнальні засоби, що буде означати подання попереджувального сигналу "Увага всім!", після якого негайно приводяться в готовність радіотрансляційні вузли, радіомовні і телевізійні станції, вмикаються мережі зовнішньої звукофікації. За сигналом населення зобов'язане увімкнути радіотрансляційні та телевізійні приймачі для прослуховування нагального повідомлення. У всіх випадках використання систем оповіщення, з увімкненням

сирен, негайно доводиться до населення відповідне повідомлення засобами проводового, радіо та телевізійного мовлення. Тексти повідомлень передаються протягом 5 хвилин державною мовою і мовою, якою користується більшість населення в регіоні з припиненням іншої передачі. Тексти звернень записуються на магнітних стрічках на весь обсяг касети з обох сторін. Фонограми і друковані тексти звернень зберігаються в запечатаних конвертах в оперативних чергових з питань НС, які в необхідних випадках доводяться до населення. Дублікати фонограм і друкованих текстів звернень зберігаються в запечатаних конвертах на радіотрансляційних вузлах, в апаратних радіомовлення, студіях телебачення і використовуються в разі виходу з ладу апаратури оповіщення або аварії на з'єднувальній лінії зв’язку.

У разі повітряної тривоги: “Увага! Говорить Головне управління (управління, відділ) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації). Громадяни! Повітряна тривога! Відключіть світло, газ, погасіть вогонь у печах. Візьміть засоби індивідуального захисту, документи, запас харчів та води. Попередьте сусідів і допоможіть хворим та людям похилого віку вийти на вулицю. Якнайшвидше дістаньтеся захисної споруди або заховайтеся на місцевості. Дотримуйтеся спокою та порядку. Уважно слухайте повідомлення Головного управління (управління, відділу) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації)”.

Після повітряної тривоги: “Увага! Говорить Головне управління (управління, відділ) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації). Відбій повітряної тривоги! Усім повернутися до місць роботи або проживання. Допоможіть у цьому хворим та людям похилого віку. Будьте готові до можливого повторного нападу противника. Завжди майте з собою засоби індивідуального захисту. Уважно слухайте повідомлення Головного управління (управління, відділу) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації)”.

У разі загрози хімічного зараження: "Увага! Говорить Головне управління (управління, відділ) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації). Громадяни! Виникла безпосередня загроза хімічного

зараження. Одягніть протигази, сховайте дітей у дитячих захисних камерах. Для захисту поверхні тіла використовуйте захисний одяг, комбінезони та чоботи. При собі майте плівкові (полімерні) накидки, куртки або плащі. Перевірте герметизацію житлових приміщень, стан вікон та дверей. Загерметизуйте продукти харчування і запасіться водою. Укрийте сільськогосподарських тварин і корми. Допоможіть хворим та людям похилого віку. Сповістіть сусідів про одержану інформацію. Відключіть електронагрівальні прилади. Надалі дійте відповідно до вказівок Головного управління (управління, відділу) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації)".

У разі загрози радіоактивного зараження: "Увага! Говорить Головне управління (управління, відділ) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації). Громадяни! Виникла безпосередня загроза радіоактивного зараження. Приведіть у готовність засоби індивідуального захисту та постійно майте їх із собою. Після команди управління (відділу) з питань НС та у справах захисту населення від наслідків Чорнобильської катастрофи надягніть їх. Для захисту поверхні тіла від забруднення радіоактивними речовинами використовуйте захисний одяг, комбінезони та чоботи. При собі майте плівкові (полімерні) накидки, куртки або плащі. Перевірте герметизацію житлових приміщень, стан вікон та дверей. Загерметизуйте продукти харчування і запасіться водою. Укрийте сільськогосподарських тварин і корми. Сповістіть сусідів про одержану інформацію. Допоможіть хворим та людям похилого віку. Надалі дійте відповідно до вказівок Головного управління (управління, відділу) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації)".

Уразі загрози біологічного зараження: "Увага! Говорить Головне управління (управління, відділ) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації). Громадяни! Виникла безпосередня загроза біологічного зараження. Для захисту поверхні тіла використовуйте захисний одяг, комбінезони та чоботи. Із собою майте плівкові (полімерні) накидки, куртки або плащі. Перевірте герметизацію житлових приміщень, стан вікон та дверей. Загерметизуйте продукти харчування і запасіться водою. Укрийте

сільськогосподарських тварин і корми. Допоможіть хворим та людям похилого віку. Сповістіть сусідів про одержану інформацію. Відключіть електронагрівальні прилади. Надалі дійте відповідно до вказівок Головного управління (управління, відділу) з питань НС облдержадміністрації (міськвиконкому, райдержадміністрації)".

4.3 Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи описано вимоги щодо безпеки та захисту здоров'я працівників під час роботи з обчислювальними та екранними пристроями. Коротко приведено інформацію по питанню «Організація оповіщення і зв'язку у надзвичайних ситуаціях техногенного та природного характеру»

ВИСНОВКИ

В кваліфікаційній роботі досліджено вплив методів попередньої обробки даних на якість класифікаційних моделей. Для вибраного набору лематизація, не є критично необхідним кроком, а стемінг не дав приросту точності, проте дозволив скоротити простір ознак майже на 30%. Для обраного набору даних доцільніше йти шляхом налаштування гіперпараметрів моделей або ж застосування нових сучасніших моделей NLP з метою покращення параметрів точності. Показано, що стемінг після лематизації дає таке ж зменшення простору ознак, як і без проведення лематизації. Проте такі дослідження доцільно щоразу проводити під час розв'язання задач NLP.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»:

- Розглянуто поняття текстових даних у контексті штучного інтелекту.
- Висвітлено поняття NLP.
- Наведено основні етапи інтелектуального аналізу для задач NLP.
- Здійснено огляд прикладних досліджень, що досліджують ефективність попередньої обробки текстових даних.
- Наведено приклади застосування NLP у різних галузях.

В другому розділі кваліфікаційної роботи:

- Описано основні методи попередньої обробки даних
- Досліджено широкий спектр традиційних моделей машинного навчання та сучасних моделей для задач NLP.
- Наведено основні метрики, що використовуються для оцінки якості класифікаційних моделей.

В третьому розділі кваліфікаційної роботи:

- Обгрунтоване середовище розробки.
- Здійснено огляд основних бібліотек Python для NLP.
- Описано окремі функції розробленого програмно забезпечення.
- Описано вибраний для класифікації набір даних.
- Проаналізовано вплив методів обробки та їх комбінацій на якість кінцевих класифікаційних моделей.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» В четвертому розділі кваліфікаційної роботи описано вимоги щодо безпеки та захисту здоров’я працівників під час роботи з обчислювальними та екранними пристроями. Коротко приведено інформацію по питанню «Організація оповіщення і зв’язку у надзвичайних ситуаціях техногенного та природного характеру»

ПЕРЕЛІК ДЖЕРЕЛ

1 Oleh Pastukh, Oleksandr Bryk Extraction of important data for cognitive software systems based on data science (2025) Scientific Journal of TNTU 117(1) P. 62-66

2 O.Pastukh, I.Stefanyshyn, V.Stefanyshyn, O.Bryk. Robustness evaluation of machine learning algorithms for neurocomputer interface software using distributed and parallel computing.- International Scientific Journal «Computer systems and information technologies».- 2024, 2, 82-88.

3 Zagorodna N., Stadnyk M., Lypa B., Gavrylov M., Kozak R. (2022). Network Attack Detection Using Machine Learning Methods. Challenges to national defence in contemporary geopolitical situation, 2022(1), 55-61. doi:10.47459/cndcgs.2022.7

4 Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., & Lechachenko, T. (2025). Comparison of feature extraction tools for network traffic data. arXiv preprint arXiv:2501.13004

5 Lamba, Manika & Margam, Madhusudhan. (2022). Text Pre-Processing. In book: Text Mining for Information Professionals: An Uncharted Territory P.79-103 10.1007/978-3-030-85085-2_3.

6 Kamruzzaman, Sikder & Haider, Farhana & Hasan, Ahmed. (2010). Text Classification using Data Mining. https://www.researchgate.net/publication/360116833_Text_Pre-Processing

7 https://www.researchgate.net/publication/351544464_Classification_of_Arabic_Tweets_A_Review

8 Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text Classification Algorithms: A Survey. Information, 10(4), 150. https://doi.org/10.3390/info10040150

9 Vidhya, S & Danasingh, Asir Antony & EPIPHANY, JEBAMALAR LEAVLINE. (2015). Feature Extraction for Document Classification.

10 Eang, C., & Lee, S. (2024). Improving the Accuracy and Effectiveness of Text Classification Based on the Integration of the Bert Model and a Recurrent

Neural Network (RNN_Bert_Based). Applied Sciences, 14(18), 8388. https://doi.org/10.3390/app14188388

11 Orlovskyi, O., & Ostapov, S. (2020). Analysis of the text preprocessing methods influence on the destructive messages classifier. Advanced Information Systems, 4(3), 104–108. https://doi.org/10.20998/2522-9052.2020.3.14 [http://ais.khpi.edu.ua/article/view/2522-9052.2020.3.14]

12 Roman B. Sergienko, Muhammad Shan, Wolfgang Minker:A Comparative Study of Text Preprocessing Approaches for Topic Detection of User Utterances. LREC 2016.

13 Denny, Matthew and Spirling, Arthur, Text Preprocessing for Unsupervised Learning: Why It Matters, When It Misleads, and What to Do about It (September 27, 2017). Available at SSRN: https://ssrn.com/abstract=2849145 or http://dx.doi.org/10.2139/ssrn.2849145.

14 Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

15 Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Draft.

16 Kowsari, K., et al. (2019). Text Classification Algorithms: A Survey. ACM Computing Surveys, 52(3).

17 Camacho-Collados, J., & Pilehvar, M. T. (2018). From Word to Sense Embeddings. Journal of Artificial Intelligence Research.

18 Lamba, M., & Margam, M. (2022). Text Pre-Processing. In Text Mining for Information Professionals (pp. 79–103). https://doi.org/10.1007/978-3-030-85085-2_3

19 Glazkova, A. (2023). A comparison of text preprocessing techniques. Social Network Analysis and Mining, 13. https://doi.org/10.1007/s13278-023-01156-y

20 Denny, M., & Spirling, A. (2017). Text Preprocessing for Unsupervised Learning. https://doi.org/10.2139/ssrn.2849145

21 Qiang, Z., Taylor, K., & Wang, W. (2024). How Does A Text Preprocessing Pipeline Affect Ontology Syntactic Matching? https://doi.org/10.48550/arXiv.2411.03962

22 Mikolov, T., et al. (2013). Efficient Estimation of Word Representations. arXiv.

23 Aliero, Abubakar & Bashir, Sulaimon & Aliyu, Hamzat & Tafida, Amina & Kangiwa, Bashar & Dankolo, Nasiru. (2023). Systematic Review on Text Normalization Techniques and its Approach to Non-Standard Words. International Journal of Computer Applications. 185. 975-8887.

24 Webster, Jonathan & Kit, Chunyu. (1992). Tokenization as the initial phase in NLP. 1106-1110. 10.3115/992424.992434.

25 Bojanowski, P., et al. (2017). Enriching Word Vectors. TACL.

26 Reimers, N., & Gurevych, I. (2019). Sentence-BERT. EMNLP.

27 Zhang, Y., & Wallace, B. (2017). A Sensitivity Analysis of CNNs. EMNLP.

28 Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly.

29 https://kuprienko.info/ukrainian-stopwords/

30 Jabbar, Abdul & Iqbal, Sajid & Tamimy, Manzoor & Rehman, Amjad & Bahaj, Saeed & Saba, Tanzila. (2023). An Analytical Analysis of Text Stemming Methodologies in Information Retrieval and Natural Language Processing Systems. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3332710.

31 Khyani, Divya & B S, Siddhartha & Niveditha, N. & M., Divya & Y M, Dr. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology. 22. 350-357.

32 Munawaroh, Khafifah & Alamsyah, Alamsyah. (2023). Performance Comparison of SVM, Naïve Bayes, and KNN Algorithms for Analysis of Public Opinion Sentiment Against COVID-19 Vaccination on Twitter. Journal of Advances in Information Systems and Technology. 4. 113-125. 10.15294/jaist.v4i2.59493.

33 Pranckevicius, Tomas & Marcinkevičius, Virginijus. (2016). Application of Logistic Regression with part-of-the-speech tagging for multi-class text classification. 1-5. 10.1109/AIEEE.2016.7821805.

34 Jannah, Nurul & Kusnawi, Kusnawi. (2024). Comparison of Naïve Bayes and SVM in Sentiment Analysis of Product Reviews on Marketplaces. Sinkron. 8. 727-733. 10.33395/sinkron.v8i2.13559.

35 Darji, Dhara & Goswami, Sachinkumar. (2024). The Comparative study of Python Libraries for Natural Language Processing (NLP). International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 10. 499-512. 10.32628/CSEIT2410242.

36 Python vs R for Data Science [Електронний ресурс]. URL: https://www.stratascratch.com/blog/python-vs-r-for-data-science/

37 UCI Reuters 50 50 [Електронний ресурс]. URL: https://www.kaggle.com/datasets/realabyszero/uci-reuters-50-50

38 ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень. [Електронний ресурс]. URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text

39 Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / В.С. Стручок –Тернопіль: ФОП Паляниця В.А., – 156 с. Отримано з https://elartu.tntu.edu.ua/handle/lib/39196.

ДОДАТКИ

Додаток А

Тези конференції

VIII Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.89
Цимбалюк Г. – ст. гр. СНнм-61
Тернопільський національний технічний університет імені Івана Пулюя

ПОПЕРЕДНЯ ОБРОБКА ТЕКСТОВИХ ДАНИХ ДЛЯ ЗАДАЧ КЛАСИФІКАЦІЇ

Науковий керівник: к.т.н., доцент Никитюк В.В.

Tsymbaliuk G.
Ternopil Ivan Puluj National Technical University

TEXT PREPROCESSING FOR CLASSIFICATION PROBLEMS

Supervisor: PhD, associated professor Nykytiuk V.

Ключові слова: ТОКЕНІЗАЦІЯ, ЛЕМАТИЗАЦІЯ, СТЕММІНГ, СТОП-СЛОВА, НОРМАЛІЗАЦІЯ
Key words: TOKENIZATION, LEMATIZATION, STEMING, STOP-WORDS, NORMALIZATION

Класифікація текстових даних є однією з основних задач в обробці природної мови (NLP). Вона включає в себе розподіл текстів по різних категоріях або класах на основі їх змісту. Задачі класифікації текстів широко застосовуються в таких сферах, як автоматичне сортування електронних листів (спам/не спам), категоризація статей, аналіз тональності (емоційного забарвлення) тексту, а також в машинному перекладі та виявленні тем в документах. Всі ці задачі вимагають точного розуміння тексту і правильного віднесення до відповідної категорії.

В [1] наведено наступні етапи класифікації текстових даних:

- Text sources: це перший етап, на якому текстові дані збираються з різних джерел, таких як вебсайти, документи, соціальні мережі або API.
- Corpus format: в контексті класифікації текстів визначає структуру та організацію текстових даних, що використовуються для навчання моделей.
- Text-preprocessing: включає в себе кроки попередньої обробки тексту, такі як токенізація, видалення стоп-слів, стемінг або лематизація для підготовки даних до аналізу.
- Exploratory text analysis: на цьому етапі застосовуються різні методи статистичного або лінгвістичного аналізу, щоб виявити патерни, ключові слова або теми.
- Vectorizing: текст перетворюється на числові вектори (наприклад, через TF-IDF або Word2Vec), що дозволяє моделі працювати з ним у числовому вигляді.
- Modeling: на цьому етапі застосовуються алгоритми машинного навчання для тренування моделі, яка буде класифікувати або прогнозувати на основі текстових даних.
- Assessment: перевіряється ефективність моделі на тестових даних через метрики, такі як точність, recall, F1-оцінка, щоб оцінити її здатність до класифікації.
- Vizualization - на останньому етапі результати аналізу візуалізуються через графіки, діаграми або хмарки слів, щоб зручніше зрозуміти висновки з даних.

Попередня обробка текстових даних є критично важливою на етапі класифікації, оскільки вона дозволяє зменшити шум в даних і покращити якість

навчання моделі. Попередня обробка дозволяє позбутися таких елементів, а також знижує розмірність вхідних даних, що допомагає зменшити складність моделі та покращити її ефективність. Без належної обробки навіть найкращий алгоритм класифікації може дати не точні результати.

Основні методи попередньої обробки текстових даних включають в себе токенізацію, видалення стоп-слів, стемінг і лематизацію, а також нормалізацію тексту.

Токенізація тексту — це процес розділення тексту на окремі одиниці, звані токенами, що можуть бути словами, фразами або іншими значущими елементами. Це перший і важливий етап попередньої обробки текстових даних в обробці природної мови. Токенізація дозволяє перетворити текстовий документ на набір елементів, з якими можна працювати далі, наприклад, підраховувати частоти слів або використовувати їх для моделювання.

Нормалізація тексту передбачає перетворення всіх слів в нижній регістр, видалення знаків пунктуації тощо.

Стоп-слова – це слова, які найчастіше зустрічаються в тексті і не містять жодної цінної інформації, такі як: they, there, this, where тощо. Бібліотека nltk - це звичайна бібліотека Python, яка використовується для видалення стоп-слів і включає приблизно 180 стоп-слів англійської мови.

Стемінг — це процес зведення слів до їх базових коренів (наприклад, "running" → "run"), що дозволяє зменшити варіативність словоформ.

Лематизація — це процес перетворення слів до їх початкової або лексичної форми (наприклад, "running" → "run", "better" → "good"), з врахуванням контексту та граматичних правил.

Автори [2] показали, що ефект від певного методу може значною мірою залежати від початкового набору даних і задачі класифікації. Крім того, зазначено, що іноді вплив може бути незначним, а в деяких випадках навіть призвести до погіршення результату. Також була обгрунтована необхідність попередньої перевірки набору даних на відсоток елементів, які підпадають під вплив конкретного методу.

Автори [3] показали, що правильно підібрані методи попередньої обробки текстових даних можуть суттєво покращити якість класифікаційних моделей, проте потребують ретельного дослідження в кожній конкретній ситуації.

Проведене дослідження для задачі визначення автора тексту показало несподіваний ефект: стемінг та лематизація не сприяли підвищенню точності класифікаційних моделей, натомість зменшення простору ознак на 90% лиш незначно вплинуло на якість і точність моделей класифікації тексу за автором.

Література.

1. Lamba Manika & Margam, Madhusudhan. (2022). Text Pre-Processing. Text Mining for Information Professionals: An Uncharted Territory (pp.79-103) Edition: 1. Chapter: 310.1007/978-3-030-85085-2_3.
2. Orlovskyi, O., & Ostapov, S. (2020). Analysis of the text preprocessing methods influence on the destructive messages classifier. Advanced Information Systems, 4(3), 104–108. https://doi.org/10.20998/2522-9052.2020.3.14
3. Haddi, Emma & Liu, Xiaohui & Shi, Yong. (2013). The Role of Text Pre-processing in Sentiment Analysis. Procedia Computer Science. 17. 26–32. 10.1016/j.procs.2013.05.005.

VIII Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.89
Цимбалюк Г. – ст. гр. СНнм-61, Цвігун В. – ст.гр. СБ-11
Тернопільський національний технічний університет імені Івана Пулюя

РОЛЬ ТА ОСНОВНІ ВИКЛИКИ ЛЕМАТИЗАЦІЇ ТА СТЕММІНГУ В ОБРОБЦІ ПРИРОДНОЇ МОВИ

Науковий керівник: к.т.н., доцент Загородна Н.В.

Tsymbaliuk G.O., Tsvihun V.
Ternopil Ivan Puluj National Technical University

THE ROLE AND MAIN CHALLENGES OF LEMATIZATION AND STEMING IN NATURAL LANGUAGE PROCESSING

Supervisor: PhD, associated professor Zagorodna N.

Ключові слова: ЛЕМАТИЗАЦІЯ, СТЕММІНГ, ОБРОБКА ПРИРОДНОЇ МОВИ
Key words: LEMATIZATION, STEMING, NATURAL LANGUAGE PROCESSING

Обробка природної мови (NLP, англ. Natural Language Processing) — це галузь штучного інтелекту, що займається взаємодією між комп'ютерами та людськими мовами. Автори [1] поділяють NLP на 2 основних напрямки: генерація природної мови (NLG, англ. Natural Language Generation) і розуміння природної мови (NLU, англ. Natural Language Understanding). Дана стаття хоч і виділяє ці два основні напрямки та наводить сфери застосування NLP, проте не співвідносить їх. В даному дослідженні здійснено розподіл задач NLP між двома базовими напрямками.

До задач NLU можна віднести:

- Аналіз тональності текстів (Sentiment Analysis): займається автоматичним визначенням емоційного забарвлення тексту з метою розуміння ставлення автора до тієї чи іншої ситуації чи особи.
- Обслуговування клієнтів (Customer Service) допомагає здійснювати аналіз дзвінків, настроїв, очікувань клієнтів з метою розуміння намірів та емоцій.
- Керування рекламою (Managing the advertisement) відповідає за виявлення ключових слів для таргетингу.
- Виявлення спаму (Spam Detection) належить до типових задач класифікації текстів.

Хоча існують «чисті» задачі напрямку NLG, на практиці значно частіше використовується комбінація двох методів обробки природної мови:

- Машинний переклад (Machine Translation) - система має зрозуміти зміст вхідного тексту, розпізнати граматичну структуру, контекст тощо, а потім згенерувати відповідний текст іншою мовою, з дотриманням граматики та стилю цільової мови.
- Автоматичне резюмування тексту (Automatic Summary) - система повинна проаналізувати текст, виявити основні ідеї, структуру, важливі речення та ключові слова. Потім на основі аналізу система генерує короткий текст, який передає суть початкового повідомлення в узагальненій формі.
- Чат-боти (Chatbots) – виконують дві задачі – розуміння тексту запиту та формулювання відповіді.

Як бачимо розуміння природної мови має широкий спектр застосування і застосовується як для вирішення окремих задач, так і в комбінованих підходах. NLU охоплює розпізнавання, аналіз та інтерпретацію природної мови, яку люди використовують у повсякденному спілкуванні.

Стемінг і лематизація — це споріднені методи в обробці природної мови, обидва належать до етапу попередньої обробки (preprocessing) та нормалізації тексту, а отже є одним з етапів напрямку розуміння природної мови. Обидва методи намагаються привести слова до "основної форми", проте роблять це по різному. Стемінг урізає слова до короткої основи без розуміння значення, натомість лематизація приводить слово до леми – початкової форми.

З одного боку, техніка стемінг реалізується шляхом відсікання будь-якого кінця або початку слова, відстежуючи загальновживані префікси і суфікси, що дозволяє нормалізувати слова та зменшити їх кількість для аналізу. Проте з іншого боку ця «груба» техніка може давати наступний результат:

Слово=studies, суфікс = -es, стема = studi

Слово =studying, суфікс =-ing, стема=study.

Як бачимо, стемінг утворив з двох однакових за змістом слів два різних об'єкти в зв'язку з тим, що ця техніка просто обрізає закінчення слів, незалежно від того, чи має отриманий результат якесь змістовне значення. Крім того, як показують дослідження результат стемінгу важко прогнозувати, оскільки він буде різнитися в залежності від обраного алгоритму. Популярна бібліотека nltk в Python містить низку стемерів, які можна використовувати. Експериментальні дослідження показали, що RegexpStemmer очікувано трансформує «studying» до «study», проте «studies» стає «studie». Натомість PorterStemmer та SnowballStemmer дають результат для двох слів «studi».

Натомість лематизація, яка шукає основну форму слова, завжди приведе ці одного початкового слова:

Слово=studies, лема = study

Слово =studying, лема =study.

Хоча виконання стемінгу є простішим, а отже швидшим ніж процес лематизації, остання все ж є кращим вибором для задач, що вимагають кращої точності. Для лематизації потрібне глибоке лінгвістичне розуміння, щоб сформувати словник, який дозволяє алгоритму знаходити значущу частину слова. Лематизація використовує словниковий запас і морфологічний аналіз слів, щоб отримати результат.

В [2] зазначено, що є обмежений набір інструментів для реалізації стемінгу та лематизації українською мовою і запропоновано програмний модуль на PHP для стемінгу текстів українською мовою. Проте для задач NLP зручно використовувати бібліотеки Python, а отже необхідно дослідити можливості бібліотеки nltk для україномовних текстів.

Підсумовуючи, можна сказати, що стемінг, зазвичай використовують у задачах, де потрібна швидка обробка тексту, таких як пошукові системи або кластеризація текстів. Лематизація застосовується у задачах, що вимагають більш високої точності, таких як машинний переклад, аналіз тональності чи лінгвістичний аналіз.

Література.

1. Khyani Divya. An Interpretation of Lemmatization and Stemming in Natural Language Processing/ Khyani Divya , Siddhartha B.S., Niveditha N.M., Divya Y M, Manu Y.M. // Journal of University of Shanghai for Science and Technology. -2021. – 22(10).P. 350-357.

2. Глибовець А. М. Алгоритм токенізації та стемінгу для текстів українською мовою / Глибовець А. М., Точицький В. В. // Наукові записки НаУКМА. Комп'ютерні науки. - 2017. - Т. 198. - С. 4-8.