ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Сегментація тексту документів на основі нейронної мережі»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та
технології спеціальності
126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТмд_22
Очеретяний І.І.
Керівник: Калініченко А.В.
Рецензент: Біловод О.І.

Полтава – 2023 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю ефективно аналізувати та розуміти масиви текстової інформації є фундаментом для розвитку новітніх технологій, що використовуються в умовах цифрової трансформації, в тому числі, і для електронного документу обігу. Одним з ключових етапів у цьому процесі є сегментація тексту. Вона допомагає розбивати текст на частини і проводити аналіз мови, розподіляючи її на важливі сегменти. Однак питання реалізації сегментації на основі нейронних мереж потребують додаткових досліджень. Все це свідчить про актуальність теми роботи.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в рамках науково-дослідної роботи «Управління стратегією інноваційного розвитку підприємств в контексті підвищення їх конкурентоспроможності на аграрному ринку, сталого розвитку та забезпечення продовольчої безпеки держави» (2021 р.), що фінансувалась господарськими договорами із замовниками, Концепції розвитку штучного інтелекту в Україні (розпорядження Кабінету Міністрів України № 1787-р від 29.12.2021), тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є підвищення ефективності системи електронного документообігу.

Завданнями кваліфікаційної роботи є:

- аналіз основних складових процедури сегментації тексту;
- розроблення сегментація тексту документів на основі нейронної мережі;
- оцінки продуктивності синтезованих нейронних мереж;

– обґрунтування рекомендацій щодо вирішення завдання сегментації проїзної частини дороги.

Об'єктом дослідження є процес сегментації тексту за допомогою штучних нейронних мереж.

Предметом дослідження є методи обробки даних у нейронній мережі сегментації тексту.

Методами дослідження в рамках визначення інструментарію для сегментації тексту, і техніко-економічного обґрунтування прийнятих рішень використовувався аналітичний метод досліджень, а для розробки моделі глибокого навчання нейронної мережі сегментації тексту – моделювання.

Інформаційна база кваліфікаційної роботи сформована з ресурсів, що містять інформацію про інструментарій для розробки та дослідження сегментації тексту.

Елементи наукової новизни роботи полягають у створенні моделі глибокого навчання нейронної мережі сегментації зображень документів.

Практична значущість роботи полягає у розробці рекомендацій щодо використання штучного інтелекту в системах електронного документообігу, які можуть бути використані для подальших досліджень за даною тематикою та при систем електронного документообігу.

Апробація результатів відбувалася в рамках V-ої Міжнародної студентської конференції «Теоретичне та практичне застосування результатів сучасної науки» (жовтень 2023 р., м. Рівне), III-ої Міжнародної студентської конференції «Міждисциплінарні наукові дослідження та перспективи їх розвитку» (листопад 2023 р., м. Дніпро).

За результатами досліджень здійснено 2 публікації тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 68 сторінок формату А4. Вона містить 21 рисунок і 1 таблицю.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ СЕГМЕНТАЦІЇ ТЕКСТУ

1.1 Оцінка напрямів використання сегментації тексту

В епоху швидкої цифрової трансформації здатність ефективно обробляти та розуміти великі обсяги текстових даних стала базисом сучасних технологій [1]. Критичним кроком у цьому процесі є техніка – сегментація тексту. Вона дозволяє розкладати на фрагменти та аналізувати людську мову, розділяючи її на значущі одиниці. При цьому виконується завдання розпізнавання згадок сутностей, зокрема іменованих сутностей (Namer Entity Recognition, NER), у тому, щоб виділити і класифікувати певні фрагменти тексту на заздалегідь відомі типи (рис. 1.1). Ще один приклад сегментації тексту (Daniel Defoe – Robinson Crusoe, 1719 р.) наведений на рис. 1.2.

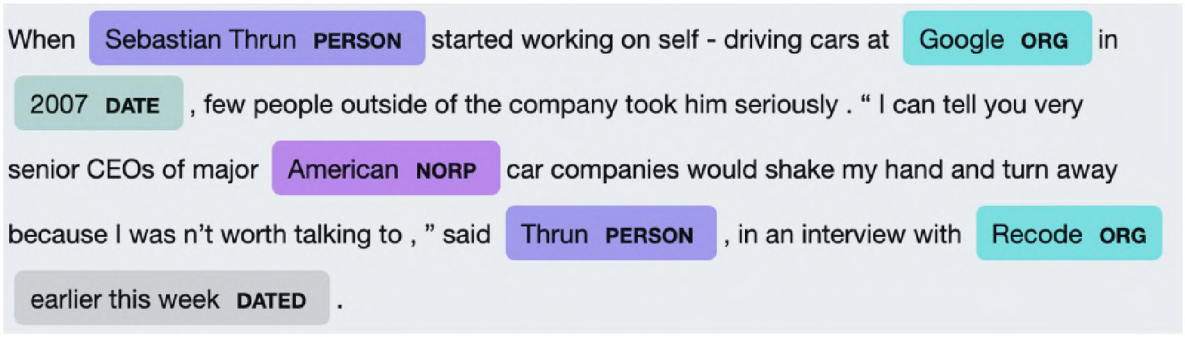

Рисунок 1.1 – Сегментація тексту

Так залишив я острів 19 грудня 1686 року, судячи з корабельного календаря, пробувши на ньому двадцять вісім років, два місяці та дев'ятнадцять днів; з цього вторинного полону я був звільнений того дня місяця, в який я колись врятувався втечею на баркасі від маврів міста Сале.

S1 – дати
S2 – міста

Так залишив я острів **<S1> 19 грудня 1686 року </S1>**, судячи з корабельного календаря, пробувши на ньому двадцять вісім років, два місяці та дев'ятнадцять днів; з цього вторинного полону я був звільнений того дня місяця, в який я колись врятувався втечею на баркасі від маврів міста **<S2> Сале </S2>**.

Рисунок 1.2 – Сутність реалізації NER

Будучи фундаментальним компонентом обробки природної мови (Natural Langauge Processing, NLP) [2], сегментація тексту є невід'ємною частиною численних додатків і галузей, що покращує нашу здатність обробляти й розуміти величезні обсяги текстових даних, створених у нашу цифрову епоху (рис. 1.3). Оскільки сфера продовжує розвиватися, сегментація тексту відіграватиме все більш важливу роль у формуванні майбутнього NLP і штучного інтелекту (Artificial Intelligence, AI), відкриваючи нові можливості для розуміння та взаємодії з людською мовою.

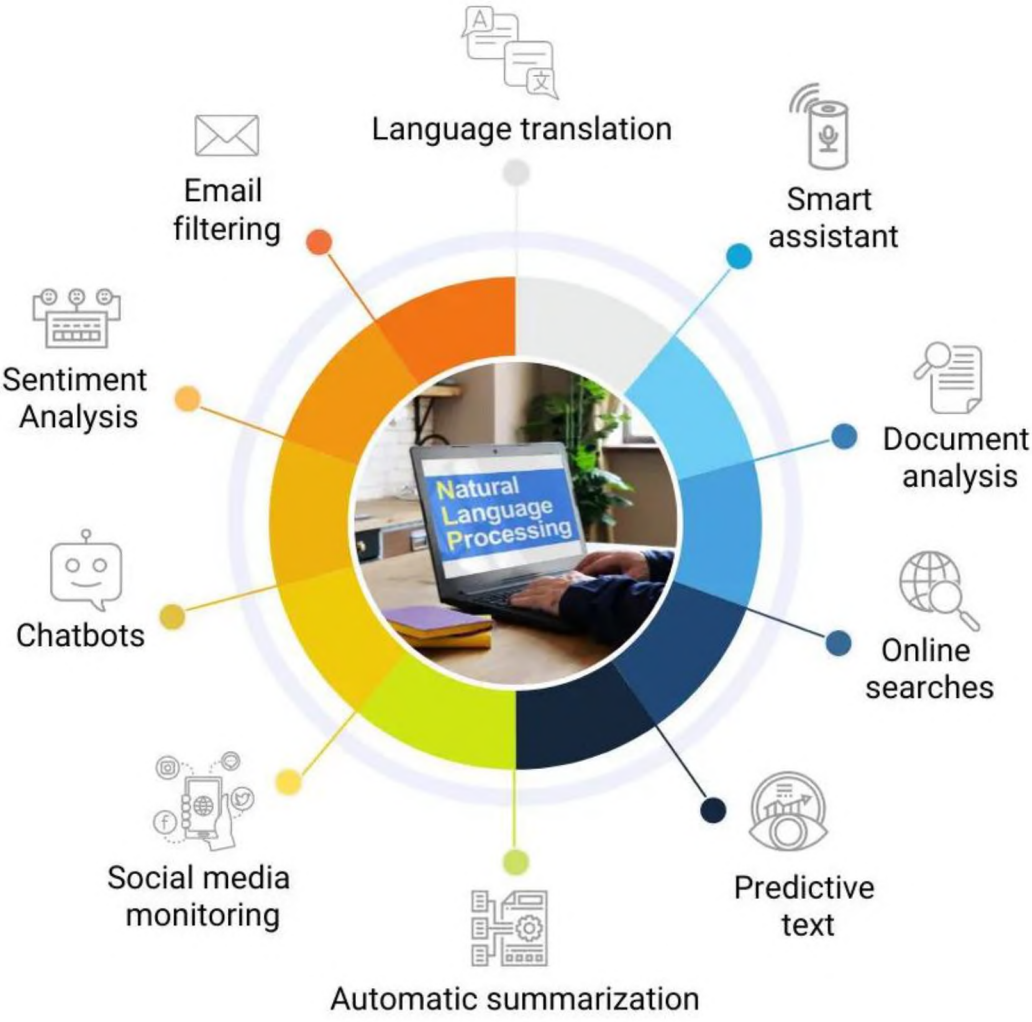

Рисунок 1.3 – Застосування NLP

Надалі доцільно розглянуто пріоритетні напрями використання сегментації тексту, демонструючи її цінність у формуванні цифрового суспільства [3].

Пошук інформації. Сегментація тексту є ключовим компонентом систем пошуку інформації, таких як пошукові системи та індексація документів. Розбиваючи текст на значущі блоки, ці системи можуть ефективно зіставляти запити користувачів із релевантним вмістом, покращуючи загальний досвід користувача та якість результатів пошуку.

Аналіз настроїв. В аналізі настроїв сегментація тексту має вирішальне значення для визначення настрою чи емоційного тону окремих речень або абзаців у документі. Ця інформація може бути безцінною для компаній, щоб відстежувати відгуки клієнтів, аналізувати громадську думку та інформувати про маркетингові стратегії.

Машинний переклад [4]. Сегментація тексту відіграє важливу роль у машинному перекладі, оскільки дозволяє системам перекладу обробляти вихідний текст із різною деталізацією, як-от слова, фрази чи речення. Точна сегментація може призвести до кращої якості перекладу, зберігаючи зміст і контекст оригінального тексту.

Резюмування тексту. Сегментація тексту на менші одиниці, такі як теми чи дискурси, є важливою для генерації стислого та зв'язного резюме великих документів або колекцій текстів [5]. Резюмування тексту має багато застосувань, зокрема агрегацію новин, транскрипцію зустрічей і наукові дослідження.

Сегментація тексту є фундаментальною для чат-ботів і розмовних систем AI, щоб розуміти та реагувати на введення користувача. Завдяки точному сегменту повідомлень користувачів на слова, речення та теми ці системи можуть генерувати більш релевантні та відповідні контексту відповіді, що призводить до більш природної та привабливої взаємодії.

Модерація вмісту. Сегментація тексту є важливим кроком у системах модерації вмісту, які спрямовані на виявлення та фільтрування неприйнятного чи шкідливого вмісту. Розбиваючи текст на менші блоки, ці системи можуть ефективніше виявляти порушення принципів спільноти, захищаючи користувачів і сприяючи розвитку здорових онлайн-спільнот [6].

Оскільки сегментація тексту та технології NLP стають все більш поширеними, попит на кваліфікованих фахівців у цих галузях зростає. У цьому розділі досліджуються наслідки цих подій для освіти та навчання робочої сили, а також потенційний вплив на різні галузі.

Щоб задовольнити зростаючий попит на спеціалістів з NLP та штучного інтелекту, навчальні заклади та навчальні програми повинні адаптувати свої навчальні програми, щоб включити відповідні навички та знання. Це включає базові теми з інформатики, лінгвістики та статистики, а також спеціалізовані курси з методів NLP, таких як сегментація тексту, машинне та глибоке навчання. Крім того, міждисциплінарні програми, які поєднують NLP з іншими галузями, такими як психологія, соціологія чи предметна експертиза, можуть допомогти створити всебічно розвинених професіоналів, здатних вирішувати складні виклики реального світу. Все більша поширеність сегментації тексту та технологій NLP створює нові можливості працевлаштування в різноманітних галузях. Кваліфіковані професіонали потрібні для таких ролей, як дослідники даних, інженери з NLP, дослідники AI та консультанти з мовних технологій. Крім того, оскільки компанії та організації все більше покладаються на ідеї, керовані NLP, зростатиме попит на експертів у галузі, які зможуть ефективно спілкуватися та співпрацювати з технічними командами.

Широке впровадження технологій сегментації тексту та NLP має потенціал для трансформації різних галузей, забезпечуючи більш ефективну та точну обробку текстових даних. Наприклад, у галузі охорони здоров'я NLP можна використовувати для отримання та аналізу інформації з медичних записів (рис. 1.4). Наприклад, у реченні є три симптоми (лихоманка, кашель і блювання), а поняття блювання заперечується. Концепції, що заперечуються, часто зустрічаються в клінічних записах. В даному контексті варто також розглядати наукових статей і відгуків пацієнтів, що призводить до кращих результатів для пацієнтів і більш обґрунтованого прийняття рішень. Подібним чином у юридичній сфері NLP може допомогти з переглядом

документів, аналізом справ і моніторингом відповідності, допомагаючи фахівцям з правових питань заощадити час і ресурси.

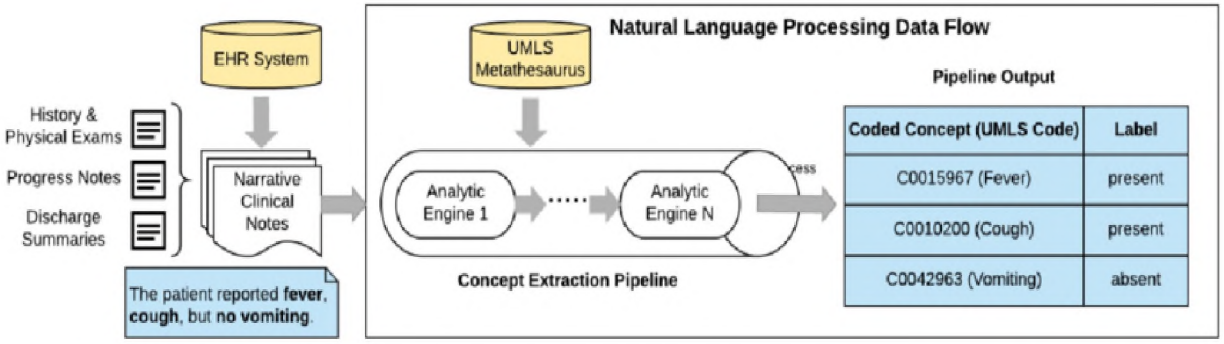

Рисунок 1.4 – Послідовність операцій конвеєра медичної NLP, який перетворює оповідальне речення в клінічній записці структуровані результати

Етичні та соціальні міркування. Як і будь-який технологічний прогрес, зростання сегментації тексту та технологій NLP викликає етичні та соціальні міркування. Забезпечення того, щоб ці технології розроблялися та відповідально розгорталися, вимагає постійного діалогу між політиками, зацікавленими сторонами галузі та громадськістю. Основні проблеми, які необхідно вирішити, включають конфіденційність даних, справедливість алгоритмів і потенційне переміщення робочих місць через автоматизацію. Беручи участь у цих дискусіях і враховуючи етичні міркування в розробці та впровадженні систем NLP, можна працювати над створенням більш справедливого та інклюзивного цифрового середовища.

Таким чином, прогрес у сегментації тексту та технологіях NLP має далекосяжні наслідки для освіти, робочої сили та промисловості. Стимулюючи кваліфіковану робочу силу та вирішуючи етичні та соціальні проблеми, пов’язані з цими інноваціями, ми можемо використовувати потенціал NLP та AI для створення більш інформованого, ефективного та пов’язаного світу.

1.2 Проблемні питання реалізації сегментації тексту

В ході проведених досліджень встановлено, що існує кілька проблем під час сегментації тексту [7]. До них слід віднести наступні.

1. Неоднозначність – межа між словами та реченнями часто є неоднозначною в тексті природною мовою, що ускладнює визначення того, де закінчується одне слово чи речення та починається інше.

2. Варіативність мови – різні мови мають різні граматичні та синтаксичні структури, через що може бути складно розробити єдиний алгоритм сегментації тексту, який ефективно працює для всіх мов.

3. Нестандартний текст. Тексти, які відрізняються від стандартних правил написання, як-от публікації в соціальних мережах або текстові повідомлення, можуть створити додаткові проблеми для алгоритмів сегментації тексту.

4. Помилки OCR (рис. 1.5) – системи оптичного розпізнавання символів (OCR), які використовуються для оцифровки текстових документів, часто створюють помилки, які можуть вплинути на точність алгоритмів сегментації тексту.

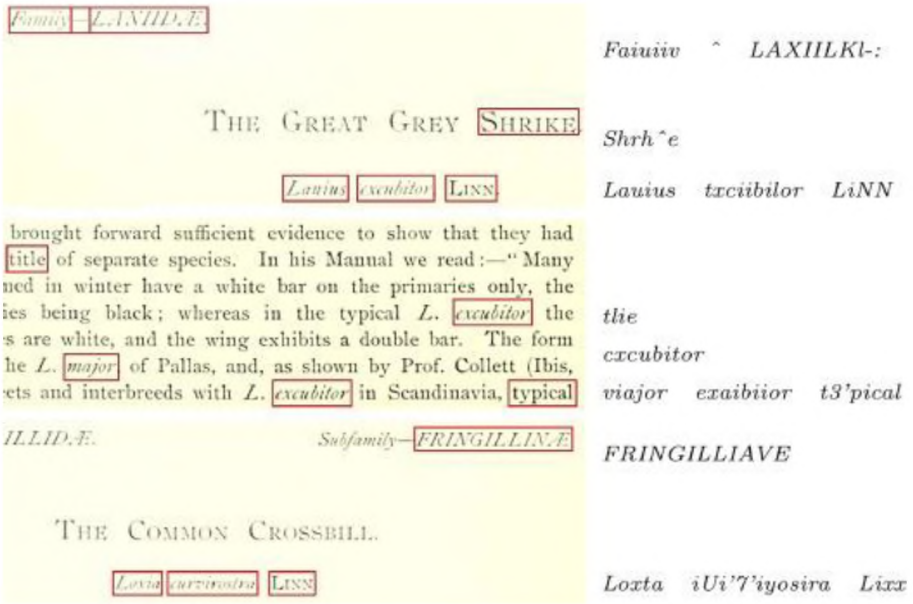

Рисунок 1.5 – Помилки OCR

5. Обробка багатослівних виразів: у багатьох мовах є багатослівні вирази, які слід розглядати як окремі одиниці під час сегментації тексту, але алгоритми можуть не відрізнити їх від окремих слів.

6. Обчислювальна ефективність – алгоритми сегментації тексту повинні бути обчислювально ефективними, оскільки вони часто використовуються як етапи попередньої обробки для великих завдань NLP, і повинні мати можливість обробляти великі обсяги текстових даних у реальному часі.

1.3 Основи сегментації тексту

Мова – це неструктуровані дані, які використовуються людьми для спілкування між собою. Структуровані або напівструктуровані дані, у свою чергу, включають поля або розмітку, що дозволяють комп'ютеру аналізувати їх. Але, попри відсутність машиночитабільної структури, неструктуровані дані є випадковими [8]. Навпаки, вони підкоряються лінгвістичним правилам, які роблять ці дані зрозумілими людям.

Методи машинного навчання, і особливо навчання з учителем, в даний час є найбільш вивченими і перспективними інструментами обробки природної мови. Машинне навчання дозволяє навчати (і перенавчати) статистичні моделі зі зміною мови [9]. Створюючи моделі на контекстно-залежних корпусах, програми можуть використовувати вузькі смислові вікна збільшення точності без необхідності вдаватися до додаткової інтерпретації. Наприклад, для створення програми, яка читає медичні карти і генерує рекомендації з лікування, потрібна зовсім інша модель, ніж для програми, що відбирає новини з урахуванням особистих переваг користувача.

Формально, модель мови повинна приймати на вході неповну фразу і доповнювати її словами, які не вистачають, найбільш ймовірними для завершення висловлювання [10]. Цей тип моделей мови сильно впливає на аналітичну обробку тексту, тому що демонструє основний механізм

застосування обробки мови – використання контексту для вгадування сенсу. Моделі мови також розкривають базову гіпотезу машинного навчання на текстах – передбачуваний текст. Моделі мови здатні виводити або визначати відносини між лексемами, які моделі спостерігають як рядки даних у кодуванні UTF8, а люди розпізнають як слова з певним змістом. Формально модель використовує контекст для визначення вузького простору рішень, в якому є невелика кількість варіантів. Далі визначимося з термінологією NLP.

Головним завданням будь-якої програми машинного навчання є визначення те, що вважати корисним сигналом як і виділити його з інформаційного шуму [12]. З цією метою виконується аналіз ознак, в ході якого встановлюється, які ознаки, властивості чи розмірності в тексті найкраще передають його зміст та базову структуру. У галузі NLP розрізняють кілька типів ознак для аналізу та розуміння мови [13].

Лінгвістичні ознаки. Ці ознаки пов'язані з лінгвістичними аспектами тексту, такими як граматика, семантика та синтаксис. Вони включають елементи, такі як частини мови (іменники, дієслова, прикметники та ін.), морфологічні особливості (наприклад, час, число, рід), та синтаксичні структури (наприклад, порядок слів, структура речення).

Контекстні ознаки. Ці ознаки відносяться до значення слів та фраз у контексті конкретної речення чи абзацу. Вони допомагають визначити, як контекст впливає сенс слова чи фрази. Наприклад, те саме слово може мати різні значення в різних контекстах, і контекстні ознаки допомагають вловлювати ці нюанси.

Структурні ознаки. Ці ознаки відносяться до організації та структури тексту. Вони можуть містити інформацію про розділення тексту на абзаци, заголовки, списки та інші структурні елементи. Структурні ознаки також можуть враховувати ієрархічні та логічні зв'язки між різними частинами тексту [14].

Складність людської мови та властива мінливість текстових даних вимагають широкого спектру підходів до вирішення сегментації тексту. До

найбільш відомих методів, які використовуються в цій галузі, зазвичай, відносяться [15].

Методи на основі правил покладаються на попередньо визначені правила та шаблони для сегментування тексту. Ці методи часто включають регулярні вирази, евристику та ручну розробку функцій [16]. Хоча підходи на основі правил можуть бути ефективними для конкретних завдань і мов, їм може бути важко адаптуватися до нових доменів або мовних варіацій.

Статистичні методи використовують розподіл слів і фраз у тексті для визначення точок сегментації [17]. Такі методи, як підхід ковзного вікна, лексична зв'язність і n-грами, можуть бути використані для аналізу частоти та моделей спільного використання слів. Хоча статистичні методи є більш гнучкими, ніж підходи, засновані на правилах, вони можуть бути чутливими до шуму та вимагати значних обчислювальних ресурсів.

Методи машинного та глибокого навчання зробили революцію в сегментації тексту, дозволивши моделям вивчати шаблони та функції із самих даних. Методи керованого навчання, такі як опорні векторні машини (SVM) [18] і приховані марковські моделі [19], широко використовуються в таких завданнях, як токенізація та сегментація речень. Зовсім недавно архітектури глибокого навчання, такі як рекурентні нейронні мережі (RNN), мережі довготривалої короткочасної пам'яті (LSTM) і моделі на основі трансформаторів (наприклад, BERT і GPT), показали чудову продуктивність у різних завданнях сегментації тексту, включно з тематичними та сегментація дискурсу [20].

Гібридні підходи. Щоб використати сильні сторони різних методів і пом'якшити їхні обмеження, було розроблено гібридні підходи, які поєднують методи на основі правил, статистики та машинного навчання [21]. Ці методи можуть використовувати експертні знання, лінгвістичні ресурси та функції, керовані даними, для досягнення більш точної та надійної сегментації тексту. Оскільки сфера сегментації тексту продовжує розвиватися, виникають нові проблеми та можливості.

Синтаксис – це зведення правил формування речень. Зазвичай, він визначається граматикою. Речення – це одиниці мови, які використовуємо для створення змісту; вони кодують набагато більше інформації, ніж окремі слова. Мета синтаксичного аналізу – показати значущі зв'язки між словами, зазвичай, шляхом поділу речення на частини, або зв'язки між лексемами в деревоподібній структурі (подібно до синтаксичного розбору речень, який ви могли робити в школі) [22-24]. Синтаксис є обов'язковою основою для міркувань про систему понять чи семантику, тому що це важливий інструмент для розуміння того, як слова впливають один на одного при формуванні фраз.

Морфологія визначає форму речей, а аналізі тексту – форму окремих слів чи лексем [25]. Структура слів може допомогти нам виявити множину, рід, час, відмінювання по особах, і т. д. Аналіз морфології – складне завдання, тому що в більшості мов є винятків з правил та спеціальних випадків.

У загальному випадку, корпус – це колекція взаємозалежних документів (текстів) природною мовою [26]. Корпус може бути великим або маленьким, але зазвичай складається з десятків і десятків і з десятків і навіть сотень гігабайт даних у тисячах документів. Корпус можна розбити на категорії документів або окремі документи. Документи, своєю чергою, можна розбити на абзаци – смислові одиниці промови (units of discourse), які зазвичай висловлюють одну ідею. Абзаци також можна розбити на речення – синтаксичні одиниці; закінчена пропозиція структурно звучить як конкретний вираз. Пропозиції складаються зі слів і розділових знаків – лексичних одиниць, які визначають загальний зміст але набагато корисніших у поєднаннях. Нарешті, самі слова складаються зі складів, фонем, афіксів і символів, тобто одиниць, які мають сенс, тільки коли вони об'єднані в слова.

Корпус можна розбити на категорії документів або окремі документи (рис. 1.6) [27]. Документи, своєю чергою, можна розбити на абзаци – смислові одиниці промови (units of discourse), які зазвичай висловлюють одну

ідею. Абзаци також можна розбити на речення – синтаксичні одиниці; закінчена пропозиція структурно звучить як конкретний вираз. Пропозиції складаються зі слів і розділових знаків – лексичних одиниць, які визначають загальний зміст але набагато корисніших у поєднаннях. Нарешті, самі слова складаються зі складів, фонем, афіксів і символів, тобто одиниць, які мають сенс, тільки коли вони об'єднані в слова.

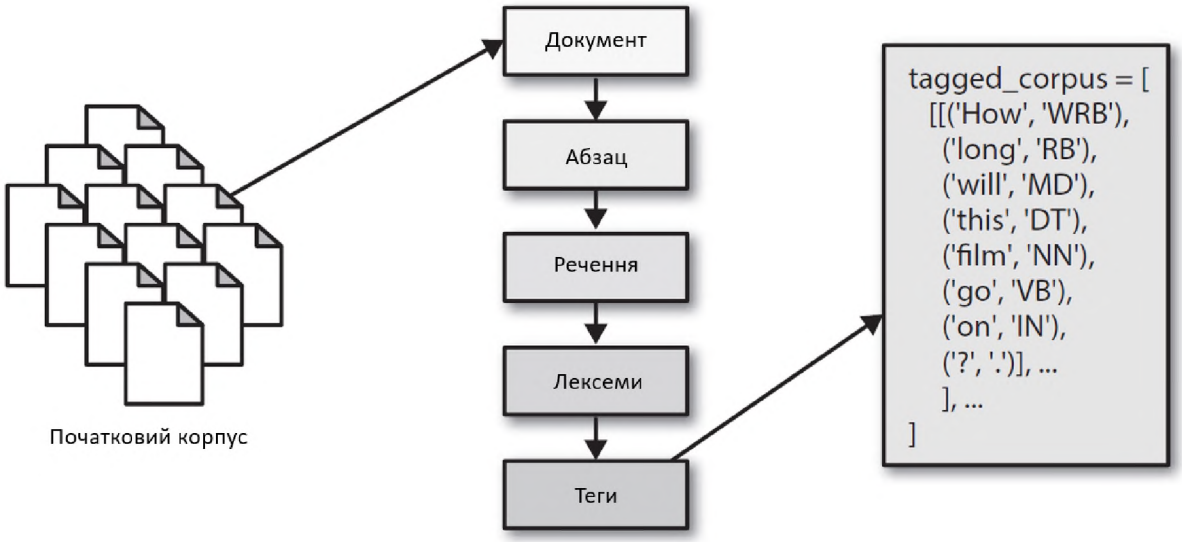

Рисунок 1.6 – Розбивка документа, сегментація, виділення лексем та маркування

Щоб працювати з текстовими документами, потрібно, щоб слова (в даному випадку, слово має назву Token) були перетворені в унікальні ідентифікатори. Для цього необхідно створити словник, який зіставить кожне слово з унікальним ідентифікатором. Щоб створити Словник, перетворимо текст або пропозиції в список. Розглянемо, навіщо потрібен об'єкт словника та як його використовувати. Об'єкт словника, зазвичай, використовується створення так званого мішка слів «bag of words» (рис. 1.7). Саме цей словник і «bag of words» (корпус, зазвичай, у коді Python позначається – corpus) використовуються як вхідні дані для тематичного моделювання та інших моделей [27, 28]. «Документ», зазвичай, може відноситись як до «речення», так і до «абзацу», а «корпус», зазвичай, є «збором документів» у вигляді

пакета слів або як його ще називають мішка слів. Тобто, для кожного документа корпус містить ідентифікатор кожного слова та його частоту у цьому документі. На жаль, будь-який корпус у його вихідному вигляді зовсім непридатний для аналізу. Його необхідно попередньо обробити та стиснути.

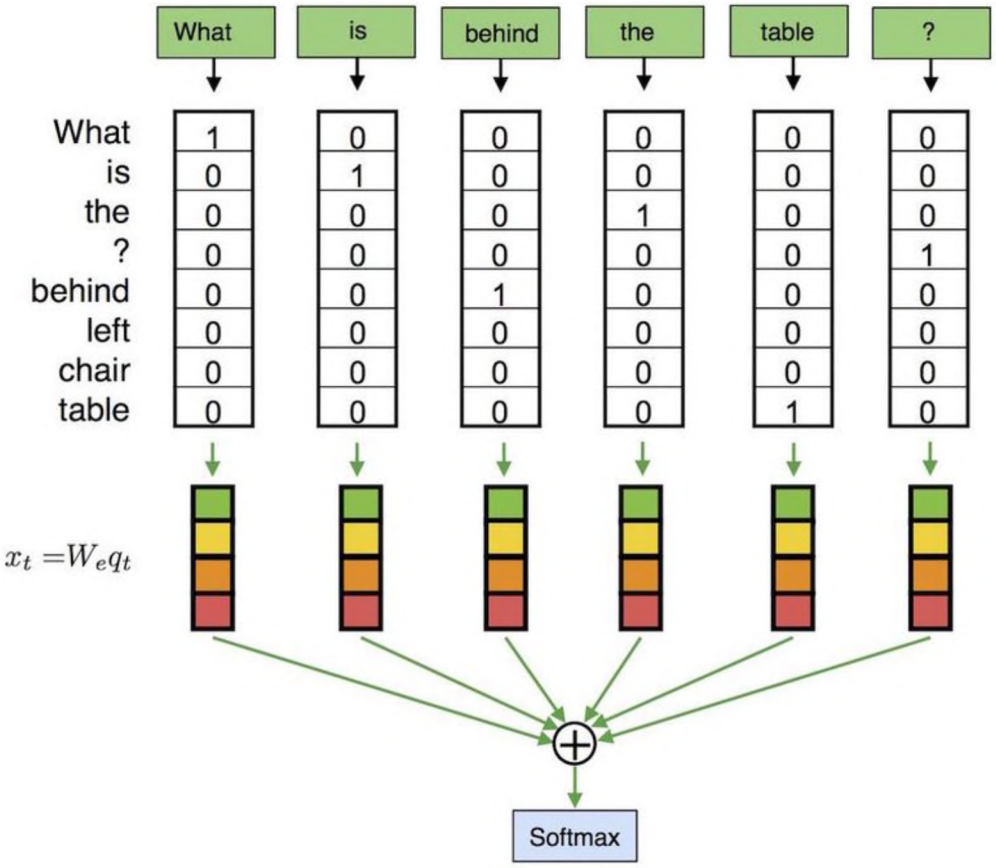

Рисунок 1.7 – Приклад використання «bag of words» для навчання нейронної мережі

Якщо абзаци можна як структурні одиниці документа, тоді речення можна вважати смисловими одиницями. Подібно до абзацу, що виражає єдину ідею, пропозиція містить закінчену думку, яку ми хотіли б сформулювати та висловити. Сегментацію – поділ тексту на речення, які потім оброблятимуться методами маркування слів тегами частин мови. Синтаксична сегментація є необхідною умовою для маркування тексту частинами промови. У деяких випадках може знадобитися спочатку виконати маркування, щоб правильно розбити текст на речення, наприклад, при аналізі

мовної інформації або транскрибованого тексту, де межі між реченнями не такі очевидні. У письмовому тексті попередня сегментація полегшує маркування лексем частинами мови. З мовними даними часто краще справляються інструменти SpaCy, тоді як обробки письмової мови краще підходить NLTK. Сегментація тексту охоплює низку завдань, від токенізації та сегментації речень до більш складних завдань сегментації теми та дискурсу. Кожне з цих завдань має окрему мету, але всі вони сприяють досягненню кінцевої мети подолання розриву між людською мовою та машинним розумінням. Для сегментації тексту, зазвичай, виконується лематизація – процес приведення словоформи до леми (її нормальній формі). Одним з інструментів, яким користуються для такої сегментації є Word2vec – метод ефективного створення вкладень [29]. В цілому, глибоке навчання нейронних мереж багато в чому спростило виконання сегментації, водночас показавши вражаючу якість.

Сегментація тексту відіграє ключову роль у NLP, оскільки вона служить основою для завдань більш високого рівня, таких як пошук інформації, аналіз настроїв, машинний переклад і резюмування тексту. Він передбачає розбиття тексту на менші керовані блоки, які легше обробляти та розуміти машинам. Надалі розглянемо ключові компоненти сегментації тексту та їх відповідну важливість.

Токенізація – це процес поділу тексту на окремі слова або лексеми [30]. Ці лексеми представляють собою основні будівельні блоки людської мови та мають вирішальне значення для наступних завдань NLP. Токенізація передбачає обробку різних мовних нюансів, таких як пунктуація, скорочення та спеціальні символи. Різні мови також можуть вимагати спеціальних стратегій токенізації для врахування їх унікальних структур і морфології.

Сегментація речення. Після того, як текст токенізовано, наступним кроком є визначення та розділення повних речень [1]. Сегментація речень дає змогу машинам аналізувати текст на рівні речень, що важливо для таких завдань, як аналіз настроїв і машинний переклад. Цей процес може бути

складним через непослідовність у використанні пунктуації, скорочень та інших мовних варіацій.

Тематична сегментація передбачає групування текстових сегментів за їх тематичним змістом [1-3]. Ця техніка особливо корисна для організації та узагальнення великих документів або аналізу розмов. Сегментація теми базується на виявленні змін у темі й може бути досягнута за допомогою різних підходів, таких як лексична зв'язність, вбудовування слів або алгоритми машинного навчання.

Сегментація дискурсу робить аналіз тексту ще одним кроком вперед, розділяючи текст на послідовні сегменти, які зберігають логічні зв'язки та контекст (рис. 1.8) [31].

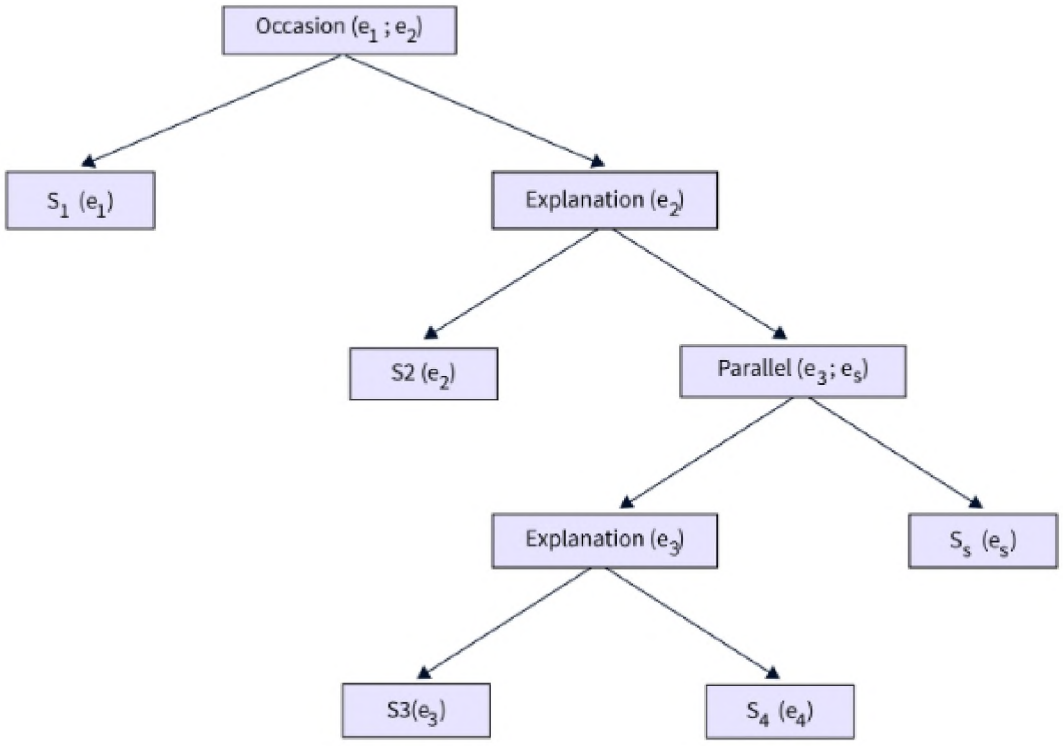

Рисунок 1.8 – Сегментація дискурсу

Вона відноситься до процесу поділу тексту на складові частини, що становлять одиниці дискурсу. Ці одиниці дискурсу, або сегменти, є більшими частинами тексту, ніж речення і зазвичай містять одну або кілька ідей або тим, які пов'язані між собою смислово і контекстуально. Мета сегментації

дискурсу – поняття структури та організації тексту на вищому рівні, ніж просто аналіз окремих речень чи слів. Ось кілька ключових аспектів сегментації дискурсу. Визначення меж частин. Одне з основних завдань – визначити, де починається і закінчується кожен сектор дискурсу. Це може бути засноване на мовних ознаках, як-от спілки, маркери переходу чи зміни у темі. Розуміння того, наскільки різні сегменти пов'язані між собою (наприклад, причинно-наслідкові зв'язки, порівняння, контраст), є важливим елементом аналізу дискурсу. У сегменті дискурсу NLP використовується для покращення розуміння та генерації природної мови, включаючи завдання, пов'язані з автоматичним реферуванням, відповідями на запитання та визначенням тональності тексту. Існують різні методи та алгоритми сегментації дискурсу, включаючи правильні підходи та машинне навчання, особливо з використанням нейронних мереж. Сегментація дискурсу є складним завданням, оскільки вимагає глибокого розуміння як мови, а й контексту, у якому він використовується. Цей процес вимагає глибшого розуміння структури тексту та зв’язків між його складовими. Сегментація дискурсу є складним завданням, яке часто використовує передові методи NLP, такі як аналіз дискурсу та теорія риторичної структури.

1.4 Аналіз номенклатури архітектур нейронних мереж для сегментації текстів

Сегментація тексту, яка передбачає поділ тексту на значущі одиниці, такі як речення, слова чи інші семантичні одиниці, використовує різні архітектури нейронних мереж, кожна з яких підходить для різних аспектів завдання. В цілому, для сегментації тексту можна використовувати наступні нейронні мережі. RNN досить ефективні для послідовних даних, таких як текст (рис. 1.9). Вони можуть обробляти вхідні послідовності змінної довжини, що робить їх придатними для таких завдань, як сегментація

речень, де важливий контекст. Мережі довгострокової короткочасної пам'яті (LSTM) – особливий вид RNN. LSTM особливо добре фіксує довгострокові залежності в тексті, що є вирішальним для розуміння структури та меж у текстовій послідовності [3].

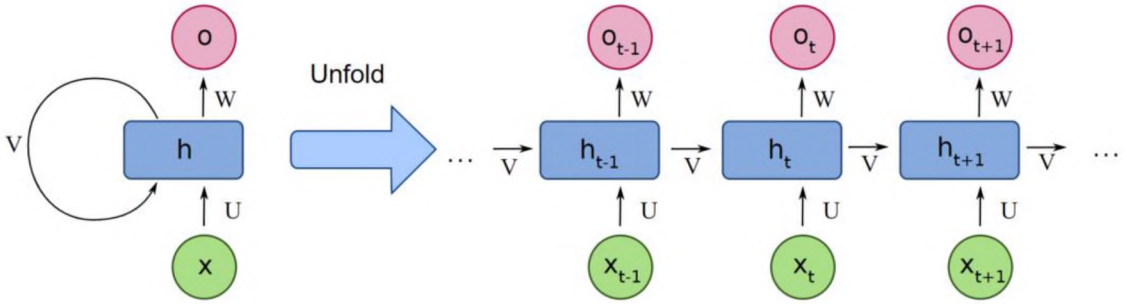

Рисунок 1.9 – RNN для NLP

Gated Recurrent Units (GRU), подібно до LSTM, є ще одним варіантом RNN. Вони розроблені, щоб допомогти фіксувати залежності в послідовностях даних, але їх простіше та часто швидше навчати, ніж LSTM.

Згортокові нейронні мережі (CNN). Хоча CNN в основному відомі для обробки зображень, вони також можуть використовуватися для сегментації тексту. CNN можуть фіксувати ієрархічні шаблони в даних і ефективно використовуються для таких завдань, як виявлення меж слів.

Трансформери. Цей тип мережі, найкращим прикладом якого є такі моделі, як BERT [32, 33] або GPT, спирається на механізми самоконтролю . Вони дуже ефективні для захоплення контексту та розуміння нюансів у тексті, що робить їх придатними для складних завдань сегментації. Представлення двонаправленого кодера від Transformers (BERT). моделі BERT, спеціально розроблені для розуміння контексту слова в реченні, дуже ефективні в завданнях, які вимагають розуміння семантичних меж у тексті.

Гібридні нейронні мережі також використовуються для сегментації тексту. Вони поєднують різні типи архітектур нейронних мереж, щоб використовувати сильні сторони кожної з них, усуваючи обмеження, які

можуть бути присутніми при використанні одного типу мережі (рис. 1.10). Використання гібридних мереж у сегментації тексту відображає постійну еволюцію та інновації у сфері NLP, де поєднання різних методологій часто призводить до більш ефективних та адаптованих рішень.

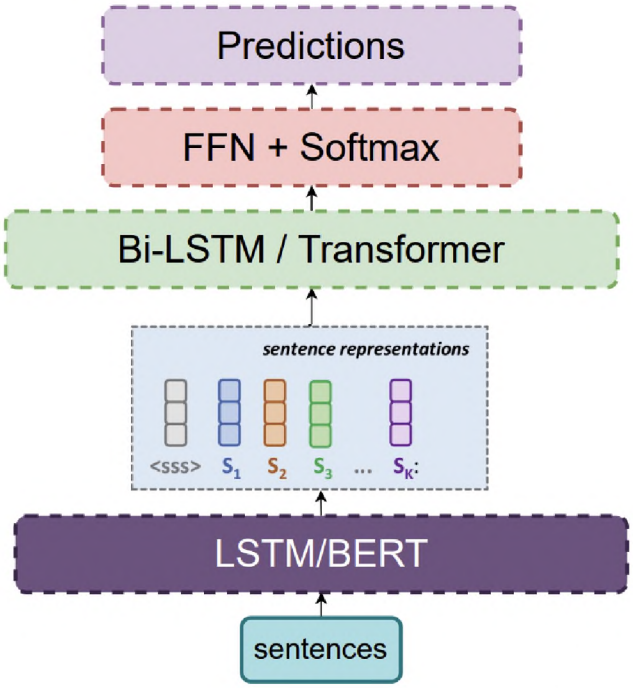

Рисунок 1.10 – Варіант гібридної моделі

Цей підхід може призвести до більш надійних і ефективних моделей, особливо в таких складних завданнях, як сегментація тексту, де різні аспекти даних можуть вимагати різних аналітичних підходів. До прикладів такого підходу слід віднести наступні.

Поєднання RNN/LSTM з CNN – цей гібридний підхід може охопити як послідовний характер тексту (з використанням RNN або LSTM), так і ієрархічні моделі (з використанням CNN). Наприклад, LSTM можна використовувати для розуміння послідовності слів у реченні, тоді як CNN може витягувати ознаки з цієї послідовності для подальшої обробки.

Інтеграція механізмів привернення уваги з RNN/LSTM – механізми привернення уваги, популяризовані моделями Transformer, можна комбінувати з RNN або LSTM, щоб покращити здатність моделі

зосереджуватися на відповідних частинах текстової послідовності для кращої сегментації.

Використання попередньо навчених моделей із користувальницькими шарами. Такі моделі, як BERT, які попередньо навчені на великих текстових корпусах, можна поєднувати з додатковими рівнями нейронної мережі, адаптованими до конкретних завдань сегментації. Цей підхід дозволяє використовувати складне розуміння мовного контексту, властивого цим моделям, а також налаштовувати мережу відповідно до конкретних вимог завдання сегментації.

Поєднання методів на основі правил із нейронними мережами – у деяких випадках гібридні підходи можуть також включати компоненти ненейронної мережі, такі як системи на основі правил, особливо в мовах або контекстах, де певні правила сегментації є добре встановленими та надійними.

Важливо відзначити, що вибір архітектури нейронної мережі залежить від конкретного завдання та набору даних, що використовується. Деякі архітектури можуть працювати краще, ніж інші, тому важливо експериментувати з різними моделями, щоб знайти найкращу для конкретного завдання.

Висновки до розділу 1

Щоб визначити структуру тексту, треба проводити його сегментацію Без сегментування довгого документа на тематично зв’язні фрагменти важко зрозуміти текст, не кажучи вже про те, щоб знайти важливу інформацію. Проблема може посилитись відсутністю сегментації в стенограмах аудіо/відеозаписів та ін. Будучи фундаментальним компонентом NLP, сегментацію тексту можна використовувати для пошуку інформації, аналізу

настроїв, машинного перекладу, резюмування тексту, чат-ботів та розмовного AI, модерації вмісту.

Для реалізації сегментацією тексту потрібно вирішити низку завдань: варіативність мови, нестандартний текст, помилки OCR, обробка багатослівних виразів, обчислювальна ефективність.

Складність текстових даних вимагає застосування великого інструментарію: методів на основі правил, статистичних методів, методів машинного та глибокого навчання, а також їх комбінацій.

На даний час, для сегментації тексту можна використовувати кілька архітектур нейронних мереж, наприклад, RNN, LSTM, GRU, CNN, трансформери та гібридні моделі.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ ДЛЯ СЕГМЕНТАЦІЇ ТЕКСТУ ДОКУМЕНТІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

2.1 Використання морфологічного аналізу та Word2vec

Сегментація текстів – сортування фрагментів вхідних текстів відповідно до їхнього змісту. В загальному випадку, програмна реалізація використання морфологічного аналізу та техніки Word2vec embeddings [34].

Морфологічний аналіз – це процес вивчення та обробки слів у тексті з метою розуміння їх форми та функцій у реченнях. Тобто, це визначення характеристик слова з урахуванням того, як це слово пишеться. При морфологічному аналізі не має інформації про сусідні слова. Тобто, морфологічний аналізатор використовується насамперед для приведення слів текстів до так званої нормальної форми, що дозволяє скоротити різноманіття слів, заощаджує пам'ять, а в ряді випадків (коли інформація про взаємозв'язок слів у документі не така важлива в порівнянні зі словниковим складом) покращує точність під час роботи нейронних мереж з текстом. Морфологічний аналіз текстів включає кілька основних аспектів.

Морфологічний аналіз важливий для розуміння структури та змісту тексту, що робить його важливим інструментом у багатьох додатках NLP, таких як машинний переклад, автоматичне реферування, отримання інформації та аналіз сентиментів.

Лематизація – перетворення слова на його базову, або словникову, форму (лему). Наприклад, перетворення дієслів на інфінітив («робив» на «робити») або іменників на називний відмінок однини («будинків» на «будинок»).

Визначення частини мови (POS-тегінг). Кожне слово в тексті класифікується з його частини мови (іменник, дієслово, прикметник та ін.). Це допомагає зрозуміти роль слова у реченні.

Визначення морфологічних ознак – аналізуються такі характеристики слів, як рід, число, відмінок, час, спосіб та інші граматичні категорії.

Зміна слова відповідно до його граматичних характеристик, наприклад, зміна закінчень у іменників для різних відмінків або відмінювання дієслів у різних часах.

Для української мови існує кілька бібліотек та інструментів, наприклад:

1. Lang-uk – набір інструментів для обробки української мови, який включає морфологічний аналізатор, інструменти для сегментації тексту, токенізації та лематизації. Lang-uk також надає інструменти для синтаксичного та семантичного аналізу.

2. Ukrainian model for spaCy – це популярна бібліотека для NLP, яка підтримує множину мов, включаючи українську. Для української мови існує модель, яка включає можливості токенізації, лематизації, морфологічного аналізу та розпізнавання іменованих сутностей.

3. Apertium, хоча більше відома як система машинного перекладу, вона також включає інструменти для морфологічного аналізу та може бути використана для опрацювання українського тексту.

4. Stanza – розроблена Стенфордським університетом, ця бібліотека підтримує безліч мов, включаючи українську. Вона надає інструменти для багатьох завдань NLP, таких як лематизація, морфологічний аналіз, синтаксичний парсинг та розпізнавання іменованих сутностей.

Ці інструменти можуть бути корисними у різних завданнях обробки української мови, від базової обробки тексту до складніших додатків, таких як машинний переклад та аналіз сентиментів.

Як відомо, Word2vec – це група зв'язаних моделей, які використовуються для векторних уявлень слів. Ці векторні представлення, чи вбудовування слів (embeddings), розробляються з допомогою нейронних мереж [35]. Важливою особливістю Word2vec є те, що вони охоплюють велику кількість лінгвістичних контекстів і відносин між словами [36]. Наприклад, вектори для слів, які часто зустрічаються у схожих контекстах, будуть близькими у векторному просторі.

Є дві основні архітектури Word2vec: (Continuous Bag of Words, CBOW) [37] і Skip-gram.

1. CBOW – модель передбачає поточне слово на основі контексту. Наприклад, у реченні «Сонце зійшло на сході» модель може використовувати «Сонце зійшло на» як контекст для передбачення слова «сході».

2. Skip-gram – модель працює навпаки, використовуючи поточне слово для передбачення його контексту. Наприклад, зі слова «зійшло» вона може намагатися передбачити «Сонце», «на» та «сході».

Word2vec моделі навчаються на великих текстових корпусах і можуть бути використані для різних завдань в галузі NLP, таких як класифікація текстів, машинний переклад, автоматичне резюмування та ін. Однією з ключових переваг Word2vec є те, що моделі здатні вловлювати тонкі семантичні відносини між словами, наприклад, розуміючи, що слова «король» і «королева» мають схожу природу, але відрізняються за ознакою статі. Word2vec embeddings (Word2vec, W2V) – попередньо навчені моделі нейронних мереж, які вже мають у своєму складі великий запас переведених у векторне представлення слів. Зазвичай, такі моделі працюють краще, ніж власні рукописні ембеддинги, оскільки крім величезної бази слів, де вони навчені, у яких закладено попередні логічні чи математичні ідеї, поліпшують аналіз і обробку тексту. Word2vec можна реалізувати, наприклад за допомогою бібліотеки Gensim. Тематичне моделювання – це метод отримання основних тем, яким присвячений оброблюваний текст. У пакеті Gensim реалізовано основні алгоритми тематичного моделювання LDA та LSI. Ви можете помітити, що ці алгоритми доступні і в інших пакетах, таких як Scikit, R та ін. Але швидкість обробки та якість результату та оцінки тематичних моделей не мають аналогів у Gensim, плюс у пакеті багато зручних засобів для обробки тексту. Ще одна істотна перевага Gensim: вона дозволяє обробляти великі текстові файли, не завантажуючи весь файл на згадку. Модель від GENSIM, звичайно, не єдина, є моделі, які вважаються найкращими по відношенню до неї. Але це хороша класична модель, що має низку переваг, що дозволяє

зрозуміти переваги цього класу моделей. Алгоритми машинного навчання діють у просторі числових ознак, очікуючи отримати на вході двовимірний масив, рядки якого представляють екземпляри, а стовпці – ознаки. Щоб виконати машинне навчання на тексті, потрібно перетворити документи на векторні представлення, до яких можна застосувати чисельне машинне навчання. Цей процес називається вилученням ознак, або просто векторизацією, і по суті є першим кроком на шляху аналізу природної мови.

Перетворення документів у чисельний вид дає можливість здійснювати їх аналіз та створювати екземпляри, з якими зможуть працювати алгоритми машинного навчання. В аналізі тексту екземплярами є цілі документи або висловлювання, які можуть мати різні розміри, від коротких цитат або твітів до цілих книг. Але самі вектори мають однакову довжину. Кожна властивість у векторному поданні це ознака. У випадку з текстом, ознаки представляють атрибути та властивості документів – включаючи їх вміст та метаатрибути, такі як довжина документа, ім'я автора, джерело та дата публікації. Разом, всі ознаки документа описують багатовимірне простір ознак, якого можуть застосовуватися методи машинного навчання. З цієї причини тепер повинні докорінно змінити свій погляд на природний мову – перейти від речень і слів до точок у багатовимірному семантичному просторі. Крапки в просторі можуть розташовуватися близько або далеко один від одного, утворювати тісні групи або рівномірно розподілятися. Отже, семантичний простір формується з документів, близьких за змістом, розташовуються поруч, а різні – далеко друг від друга. Кодуючи подібність як відстань, ми можемо почати породжувати первинні компоненти документів та визначати межі рішення у нашому семантичному просторі. Найпростішим представленням семантичного простору є модель мішок слів, основна ідея якої полягає в тому, що сенс та подібність кодуються у вигляді словника. Незважаючи на простоту, ця модель дуже ефективна і може бути відправною точкою для складніших моделей. Для векторизації корпусу в мішок слів (Bag-Of-Words, BOW) представимо кожен документ у вигляді вектору, довжина якого дорівнює розміру словника

корпусу. Можна спростити обчислення, відсортувавши в алфавітному порядку лексеми у векторі, як показано на рис. 2.1. Також можна сформувати словник, що відображає позиції лексем у векторі. У будь-якому випадку ми отримаємо векторне відображення корпусу, що дозволяє однозначно подати кожен документ.

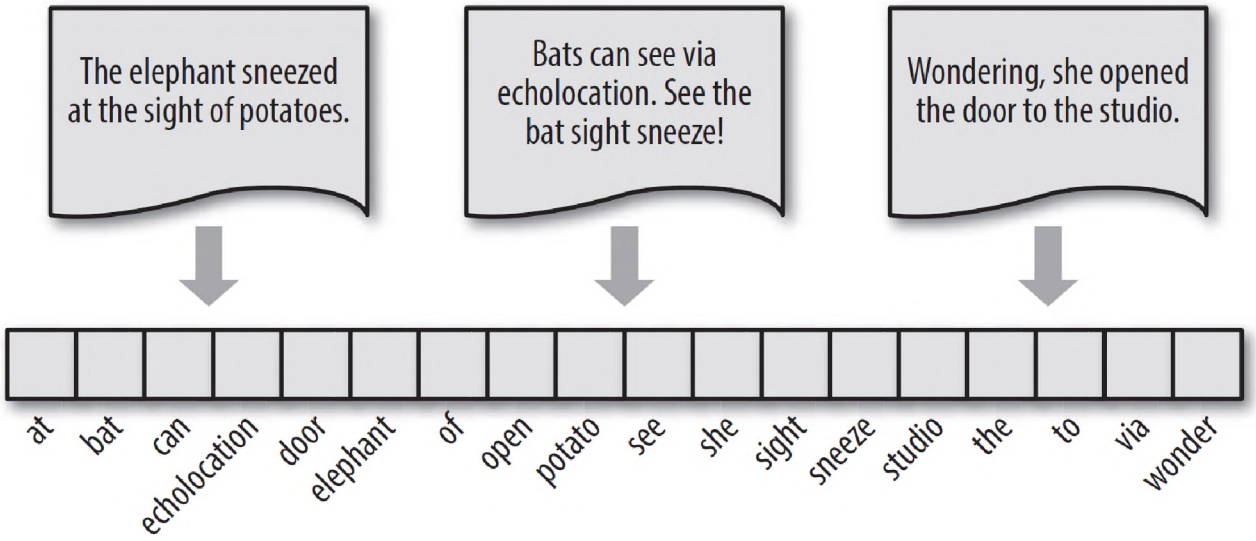

Рисунок 2.1 – Подання документів у вигляді векторів

Інформація повинна зберігатися в кожному елементі вектору, що представляє документ. Надалі розглянемо чотири види кодування вектору:

– частотне;
– пряме;
– TF-IDF;
– розподілене представлення.

В загальному випадку, будемо спиратись на їх реалізацію з застосуванням ресурсів Gensim. Як зразок буде використовуватися маленький корпус із трьох речень.

Для початку створимо список документів і виконаємо їхню лексемізацію, щоб потім приступити до векторизації. Наступний метод, tokenize, виконує спрощену лексемізацію, відкидає розділові знаки, використовуючи набір символів string.punctuation, і перетворює символи, що залишилися, в нижній

регістр. Додатково ця функція згортає властивості за допомогою SnowballStemmer, видаляючи суфікси, наприклад, що позначають множину («bats» і «bat» інтерпретуються як та сама лексема). Приклади в наступному розділі використовуватимуть цей демонстраційний корпус і деякі з них – це метод лексемізації [27]:

```
import nltk
import string
def tokenize(text):
stem = nltk.stem.SnowballStemmer('english')
text = text.lower()
for token in nltk.word_tokenize(text):
if token in string.punctuation: continue yield stem.stem(token)
corpus = [
"The elephant sneezed at the sight of potatoes.",
"Bats can see via echolocation. See the bat sight sneeze!",
"Wondering, she Opened the Door to the Studio.",
]
```

Вибір конкретного способу векторизації значною мірою визначається предметним простором. Аналогічно вибір реалізації – Scikit-Learn або Gensim – диктується вимогами програми. Наприклад, NLTK [38] пропонує множину методів, які чудово підходять для обробки текстових даних, але спричиняють занадто багато залежностей.

Бібліотека Scikit-Learn створювалася не для обробки тексту, але пропонує надійний API та множину різних зручностей, особливо цінних у прикладному контексті. Gensim може серіалізувати словники та посилання у формат Matrix Market, що дозволяє використовувати дані на різних платформах. Однак, на відміну від Scikit-Learn, Gensim не виконує жодних операцій з документами для їхньої лексемізації або стеммінгу. З цих причин, представляючи кожен із чотирьох методів кодування, доцільно розглядати кілька варіантів реалізації, з застосуванням Scikit-Learn або Gensim.

2.2 Частотні вектори

Найпростіший спосіб векторизації полягає у заповненні вектору частотами появи слів у документі. У цій схемі кодування кожен документ представляється як мультимножина складових його лексем, а значенням для кожної позиції слова у векторі служить лічильник відповідного слова. Значення можуть бути простими цілими лічильниками, як на рис. 2.2, або зважуватись загальною кількістю слів у документі.

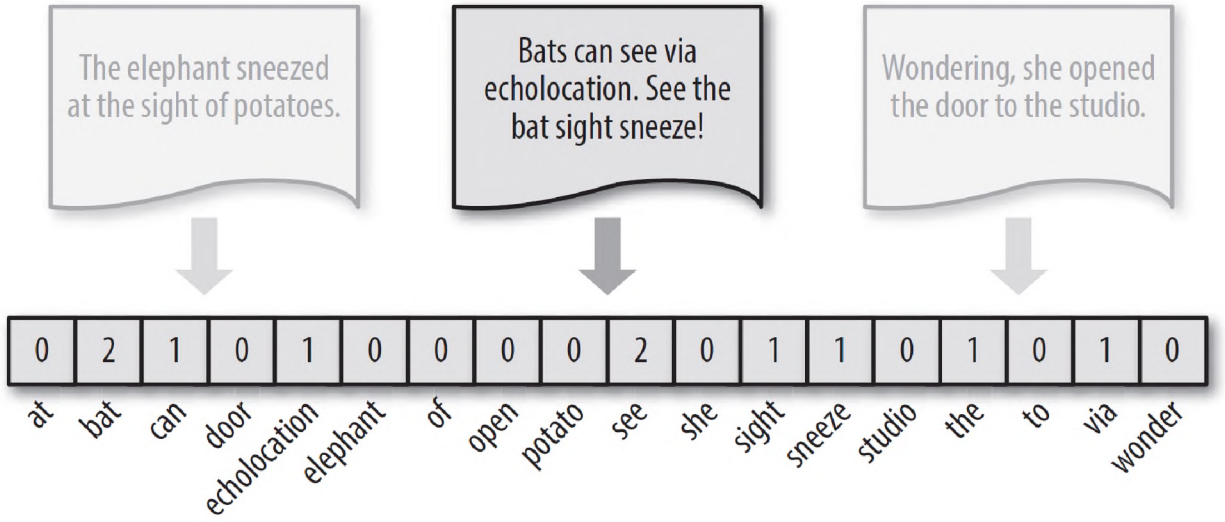

Рисунок 2.2 – Подання частот лексем у вигляді вектору

Використовуємо об'єкт defaultdict, який дозволяє вказати, яке значення має повернути словник при зверненні до неіснуючого ключа. Викликаючи конструктор як defaultdict(int), ми вказуємо, що за умовчанням має повертатися значення 0, і цим створюємо простий словник-лічильник. Можна застосувати цю функцію до всіх документів у корпусі за допомогою map, як показано в останньому рядку наступного лістингу, та створити векторне подання документів:

```
from collections import defaultdict
    def vectorize(doc):
    features = defaultdict(int)
    for token in tokenize(doc): features[token] + = 1 return
features
    vectors = map(vectorize, corpus)
```

У моделі sklearn.feature_extraction є перетворювач CountVectorizer із власними методами лексемізації та нормалізації. Метод fit цього перетворювача приймає послідовність, що ітерується, або список рядків або об'єктів файлів і створює словник корпусу. Метод transform перетворює кожен окремий документ на розріджений масив, роль індексів у якому грають кортежі з ідентифікаторами документів та лексем зі словника, а роль значень – лічильники лексем [27]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
vectors = vectorizer.fit_transform(corpus)
```

Вектори можуть виходити дуже розрідженими, особливо зі збільшенням розміру словника, що може істотно впливати на продуктивність моделей машинного навчання. Тому для великих корпусів рекомендується використовувати перетворювач HashingVectorizer з Scikit-Learn, який підтримує трюк з хешуванням для пошуку рядків з лексемами і вилучення відповідних їм індексів. Він економно використовує пам'ять і добре підходить для обробки великих наборів даних, тому що не зберігає весь словник цілком і тим самим прискорює збереження та навчання. Однак зворотне перетворення (з вектору в документ) неможливе, через що можливі конфлікти і не підтримується зворотне зважування частоти документів.

Частотний кодувальник Gensim називається doc2bow [39]. Щоб задіяти його, спочатку потрібно створити об'єкт словника Dictionary з Gensim, що відображатиме лексеми в індекси в порядку їхнього прямування в документі (завдяки чому усуваються накладні витрати на сортування). Метод doc2bow приймає лише один екземпляр документа, тому для векторизації всього корпусу ми використовуємо генератор списків, що завантажує лексемізовані документи на згадку, і позбавляємося ризику вичерпати генератор:

```
import gensim
corpus = [tokenize(doc) for doc in corpus]
id2word = gensim.corpora.Dictionary(corpus) vectors = [
id2word.doc2bow(doc) for doc in corpus
]
```

Об'єкт словника можна зберегти на диск або завантажити з диска та реалізувати бібліотеку doc2bow, яка приймає лексемізований документ і повертає розріджену матрицю кортежів (id, count), де id – ідентифікатор лексеми у словнику.

2.3 Пряме кодування

Ігноруючи граматику та відносні позиції слів у документах, частотні методи кодування страждають проблемою витягнутого хвоста, або розподілу Zipfian distribution, характерною для природних мов. В результаті лексеми, що зустрічаються дуже часто, виявляються на порядки «значнішими», ніж інші, що зустрічаються набагато рідше. Це може істотно впливати на деякі моделі (наприклад, узагальнені лінійні моделі), які передбачають нормальний розподіл ознак.

Вирішенням цієї проблеми є пряме кодування, метод логічної векторизації, який поміщає у відповідний елемент вектору значення true (1), якщо лексема є в документі, і false (0) – якщо відсутня [40]. Іншими словами, кожен елемент вектору при прямому кодуванні говорить про наявність або відсутність лексеми в документі, що описується, як показано на рис. 2.3.

Пряме кодування послаблює проблему дисбалансу розподілу лексем, спрощуючи документ до його компонентів. Цей метод найбільш ефективний для коротких документів (речень), що містять не так багато елементів, що повторюються, і зазвичай застосовується в моделях, що мають хороші згладжуючі властивості. Пряме кодування також часто застосовується у штучних нейронних мережах, функції активації яких вимагають подачі на вхід значень дискретного діапазону [0,1] або [-1,1].

У бібліотеці Gensim немає спеціалізованого методу, що реалізує пряме кодування, але метод doc2bow повертає список кортежів, якими ми можемо оперативно управляти. Доповнивши код із прикладу частотної векторизації із

застосуванням Gensim у попередньому розділі, ми можемо отримати вектори прямого кодування зі словника id2word.

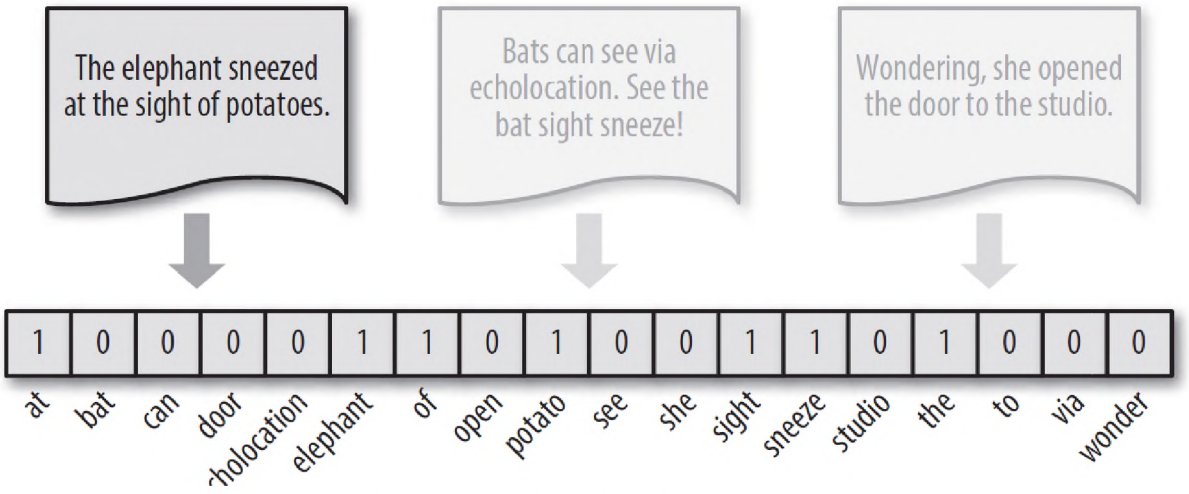

Рисунок 2.3 – Пряме кодування

Для цього додаємо вкладений генератор списків, що перетворює список кортежів, що повертається методом doc2bow, до списку кортежів (token_id, 1) і в зовнішньому генераторі списків застосуємо це перетворення до всіх документів у корпусі:

```
corpus = [tokenize(doc) for doc in corpus]
id2word = gensim.corpora.Dictionary(corpus) vectors = [
[(token[0], 1) for token in id2word.doc2bow(doc)] for doc in
corpus
]
```

Пряме кодування представляє подібність і відмінність лише на рівні Документів, але, оскільки всі слова виявляються рівновіддаленими, цей спосіб неспроможна закодувати подібність лише на рівні слів. Крім того, оскільки всі слова рівновіддалені, форма слів набуває особливої важливості; лексеми «trying» (спроба) і «try» (спробувати) будуть так само далекі один від одного, як незв'язані слова, такі як «red» (червоний) і «bicycle» (велосипед). Нормалізація лексем та їх приведення до одного класу за допомогою стеммінгу або лематизації, про які розповідається далі в цьому розділі, гарантують, що слова, що відрізняються числом, регістром символів, родом, відмінком, часом та ін., інтерпретуватимуться як єдиний компонент вектору, що забезпечить

зменшення розмірності простору ознак та збільшення продуктивності моделей. Подання «мішок слів», про яке розповідалося досі, описує документи без урахування контексту корпусу. Найбільш зручний підхід ґрунтується на зіставленні відносної частоти або рідкості лексем у документі з їх частотою в інших документах. Головна ідея цього підходу полягає в тому, що основний зміст документа закодований у рідкісних словах. Наприклад, у корпусі на спортивну тему такі лексеми, як «суддя», «база» та «землянка» (лава запасних) з'являються частіше в документах на бейсбольну тему, тоді як інші слова, такі як «біг», «окуляри» та « гра», що частіше зустрічаються в інших документах у корпусі, відіграють менш важливу роль.

Метод кодування «частота слова – зворотна частота документа» (Term Frequency-Inverse Document Frequency, TF-IDF) нормалізує частоту лексем у документі з урахуванням вмісту в іншому корпусі (рис. 2.4).

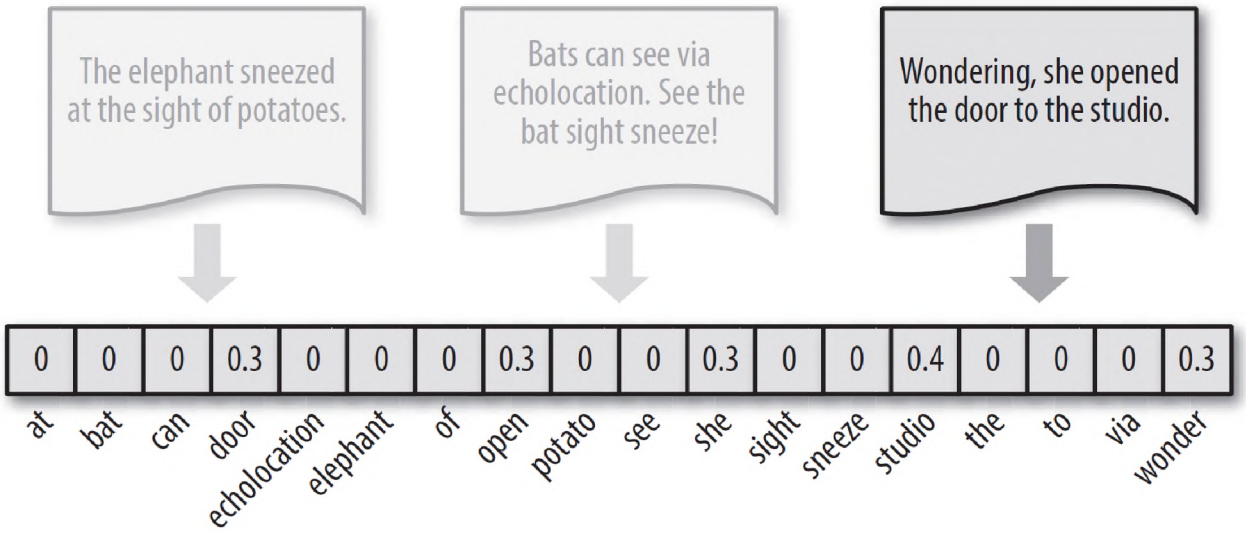

Рисунок 2.4 – Метод кодування TF-IDF

Цей метод надає більше ваги термінам, релевантним конкретного екземпляра, наприклад, лексема studio має найвищу релевантність для даного документа, тому що з'являється тільки в ньому. Оцінка TF-IDF обчислюється лише на рівні лексем, тому релевантність лексеми у документі вимірюється масштабованої частотою появи лексеми в документі, нормалізованої зворотної масштабованої частотою появи лексеми у всьому корпусі. Частота слова для

заданої лексеми в документі, tf(t, d) може бути логічною частотою (як у прямому кодуванні, 1, якщо t присутній в d, і 0 в іншому випадку) або лічильником. Однак зазвичай частота слова та зворотна частота документа масштабуються логарифмічно, щоб уникнути упередженості на користь найбільш довгих документів або слів, що з'являються особливо часто: tf(t, d) = 1 + log ftd. Аналогічно обернену частоту документа для слова в заданому наборі документів можна перевести в логарифмічний масштаб: idf(t, D) = log 1 + N/nt , де N - число документів, а nt – число входжень слова t у всіх документах. Після цього повна оцінка TF-IDF обчислюється як tfidf(t, d, D) = tf(t, d) x idf(t, D).

Оскільки відношення логарифмічної функції idf більше або дорівнює 1, величина TF-IDF завжди більша або дорівнює нулю. Відповідно, згідно з нашими представленнями, чим ближче до 1 оцінки TF-IDF слова, тим більш інформативним є це слово для цього документа. І навпаки, що ближче ця оцінка до нуля, то менш інформативне слово.

У бібліотеці Gensim є структура даних TfidfModel, яка так само, як об'єкт Dictionary зберігає лексеми з їх індексами у векторі в порядку їхнього прямування в документі, але, крім того, зберігає частоту лексем у корпусі для можливої наступної векторизації документів. Як було показано вище, Gensim дозволяє нам застосовувати свій метод лексемізації, очікуючи отримати корпус у вигляді списку списків лексем. Насамперед ми повинні сконструювати словник лексем і на його основі створити екземпляр TfidfModel, який обчислить нормалізовану зворотну частоту документа. Після цього можна отримати подання TF-IDF для кожного вектора за допомогою getitem, використовуючи синтаксис звернення до словників, а потім до кожного документа застосувати метод doc2bow словника:

```
corpus = [tokenize(doc) for doc in corpus]
lexicon = gensim.corpora.Dictionary(corpus)
    tfidf = gensim.models.TfidfModel(dictionary = lexicon,
normalize = True) vectors = [tfidf[lexicon.doc2bow(doc)] for doc
in corpus]
```

У бібліотеці Gensim є допоміжні методи для запису словників та моделей на диск у компактній формі, а це означає, що є зручна можливість зберігати

моделі TF-IDF та словники з метою подальшого їх використання для векторизації нових документів. Той самий результат (нехай і з трохи більшими зусиллями) можна отримати за допомогою комбінації модуля pickle та Scikit-Learn. Ось як можна зберегти модель на диск засобами з бібліотеки Gensim:

```
lexicon.save_as_text('lexicon.txt', sort_by_word = True)
tfidf.save('tfidf.pkl')
```

Перша інструкція збереже словник текстовий файл у відсортованому вигляді, а друга збереже модель TF-IDF у вигляді стиснутої розрідженої матриці. Зверніть увагу, що об'єкт Dictionary можна зберегти в більш компактному двійковому форматі викликом його методу save, але save_as_text дозволяє легко перевірити та скоригувати збережений словник вручну. Ось як можна завантажити моделі з диска:

```
lexicon = gensim.corpora.Dictionary.load_from_text('lexicon.txt')
tfidf = gensim.models.TfidfModel.load('tfidf.pkl')
```

Одна з переваг методики TF-IDF – вона природно вирішує проблему стоп-слів, які майже напевно присутні у всіх документах у корпусі (наприклад, «a», «the», «of» та ін.) і тому отримують дуже невелику вагу в цій схемі кодування. Завдяки цьому найбільшої значущості в моделі TF-IDF набувають відносно рідкісні слова. Як результат, TF-IDF широко застосовується до моделей «мішок слів» і є чудовою відправною точкою в багатьох видах аналізу природної мови.

2.4 Розподілене представлення

Частотне і пряме кодування, а також кодування методом TF-IDF дозволяють нам перетворювати документи у векторний простір, але часто також корисно закодувати подібності між документами в контексті того ж векторного простору. На жаль, ці методи векторизації виробляють вектори з невід'ємними елементами, що не дозволяє порівнювати документи, що не

мають загальних лексем (бо два вектори, косинус кута між якими дорівнює 1, будуть вважатися далекими один від одного навіть за очевидної семантичної подібності).

Якщо контексті програми подібність між документами грає значної ролі, текст можна закодувати як числової послідовності методом розподіленого представлення, як показано на рис. 2.5. При такому підході вектор документа є не простим відображенням позицій лексем у їх числові значення, а набором ознак, що визначають схожість слів. Складність простору ознак (і довжина вектору) визначається особливостями навчання цього представлення і безпосередньо пов'язані з самим документом.

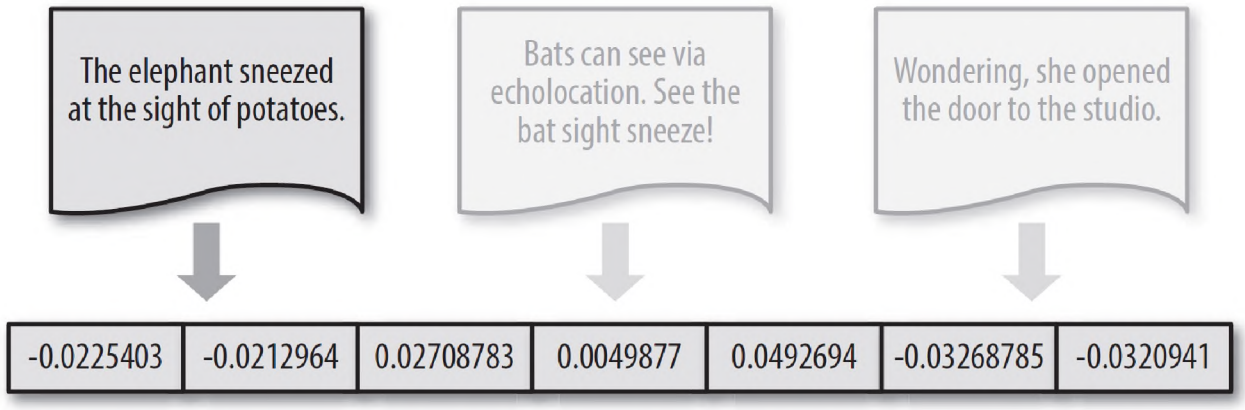

Рисунок 2.5 – Розподілене представлення

Інструмент word2vec реалізує модель векторного представлення слів (word embedding model) і дозволяє створювати такого роду розподілені представлення. Алгоритм word2vec навчає представлення слів, спираючись на модель безперервного мішка слів (Continuous Bag-of-Words, CBOW) чи skip-gram, у результаті слова у просторі ознак виявляються поруч із подібними словами, залежно від контексту. Наприклад, реалізація Gensim використовує мережу прямого поширення.

Алгоритм doc2veci є розширенням алгоритму word2vec. Він пропонує paragraph vector (вектор абзацу) – алгоритм навчання без вчителя, який створює простір ознак фіксованої довжини документів різної довжини. Це

представлення намагається успадкувати семантичні властивості слів, щоб, наприклад, слова «red» (червоний) і «colorful» (різнокольоровий) вважалися більш схожими один на одне, ніж на «river» (річка) або «governance» (влада).

Крім того, алгоритм paragraph vector враховує порядок слів у вузькому контексті, нагадуючи модель n-грам. Комбінований результат виходить набагато ефективнішим, ніж моделі «мішок слів» або «мішок n-грам», тому що краще узагальнює та має меншу розмірність, а завдяки фіксованій довжині може використовуватись у поширених алгоритмах машинного навчання.

Ні NLTK, ні Scikit-Learn не реалізують отримання такого векторного представлення слів. Реалізація Gensim дає користувачам можливість навчити моделі word2vec і doc2vec на своїх корпусах, а також включає готову модель, попередньо навчену на корпусі новин Google.

Щоб скористатися заздалегідь навченими моделями з бібліотеки Gensim, ви повинні завантажити двійковий файл моделі, розмір якого становить близько 1,5 ГБ. Для додатків, де масивні залежності небажані (наприклад, якщо вони виконуються на екземплярі AWS Lambda), цей підхід може бути неможливим.

Для навчання моделі можна використовувати наступний алгоритм. Спочатку, використовуючи генератор списків, завантажити корпус на згадку. (Gensim підтримує потокову обробку, але таке рішення допоможе уникнути вичерпання генератора.) Потім створити список об'єктів класу TaggedDocument, що успадковує клас LabeledSentence, який, у свою чергу, реалізує розподілене представлення word2vec. Об'єкти TaggedDocument складаються зі слів та тегів. Створити екземпляр TaggedDocument можна зі списку лексем з єдиним тегом, що однозначно ідентифікує екземпляр. У наступному прикладі ми помітили кожен документ як «d{}».format(idx), наприклад, d0, d1, d2 та ін. Після отримання списку об'єктів TaggedDocument створюється модель Doc2Vec і задається розмір вектора, а також мінімальний поріг – лексеми, що зустрічаються рідше за цей поріг, ігноруються. Розмір вектора (параметр size) зазвичай вибирається щонайменше 5; у цьому прикладі ми вибрали невелике значення виключно для простоти демонстрації. Мінімальний поріг (параметр

min_count) у прикладі обраний рівним нулю, щоб гарантувати облік всіх лексем, але на практиці частіше мінімальний поріг встановлюється рівним значенню від 3 до 5, залежно від того, як багато інформації має захопити модель. Після створення моделі проводиться навчання нейронної мережі для отримання векторних уявлень, які будуть доступні через властивість docvecs:

```
from gensim.models.doc2vec import TaggedDocument, Doc2Vec
corpus = [list(tokenize(doc)) for doc in corpus] corpus = [
TaggedDocument(words, ['d{}'.format(idx)]) for idx, words in
enumerate(corpus)
]
model = Doc2Vec(corpus, size = 5, min_count = 0)
print(model.docvecs[0])
# [ 0.01797447 -0.01509272 0.0731937 0.06814702 -0.0846546 ]
```

Вибір способу векторизації та бібліотеки для його реалізації залежить від конкретного випадку застосування, як показано в табл. 2.1.

Таблиця 2.1 – Огляд методів векторизації тексту

Метод	Принцип дії	Застосування	Недоліки
Частотний	Підрахунок частоти входження лексем	Байєсівські моделі	Найпоширеніші слова не завжди є інформативними
Пряме кодування	Визначення логічного ознаки присутності лексеми (0, 1)	Нейронні мережі	Всі слова виявляються рівновіддаленими, тому дуже важлива нормалізація
TF-IDF	Нормалізація частоти лексем за документами	Програми загального призначення	Слова, що помірно часто зустрічаються, можуть бути не репрезентативними для теми документа
Розподілене представлення	Кодування подібності лексем на основі контексту	Моделювання складних відносин	Великий обсяг обчислень, складність масштабування без застосування додаткових інструментів (наприклад, Tensorflow)

При правильному використанні розподілені представлення дозволяють отримати набагато якісніші результати, ніж моделі TF-IDF. Саму модель можна зберегти на диск і повторно навчити її, що робить її надзвичайно гнучкою для різних випадків використання. Однак для навчання на великих корпусах може знадобитися дуже багато часу та пам'яті, а результат може виявитися гіршим,

ніж при використанні моделі TF-IDF у сукупності з методом головних компонентів (Principal Component Analysis, PCA) або сингулярного розкладання (Singular Value Decomposition, SVD) скорочення простору ознак. Але, як би там не було, цей вид уявлень виявився справжнім проривом і допоміг досягти значних покращень у обробці тексту.

2.5 Створення на основі Gensim перетворювача для векторизації та нормалізації

Прийоми векторизації в Gensim становлять особливий інтерес, тому що ця бібліотека дозволяє зберігати та завантажувати корпуси з диска, відокремлюючи їх від конвеєра обробки. Однак є можливість написати свій перетворювач, який використовує механізм векторизації із Gensim.

Перетворювач GensimVectorizer буде обгортати об'єкт Dictionary з Gensim, згенерований методом fit(), чий метод doc2bow використовується методом transform(). Об'єкт Dictionary (наприклад, TfidfModel) можна зберегти на диск і завантажити з диска, тому наш перетворювач також користуватиметься такою можливістю, приймаючи шлях при створенні екземпляра. Якщо цей файл існує, він завантажується негайно.

Додатково за допомогою save() ми зможемо записувати свій Dictionary на диск після виклику fit().

Метод fit() конструює об'єкт Dictionary, передаючи лексемізовані та нормалізовані документи конструктору Dictionary. Потім екземпляр Dictionary негайно зберігається на диск, щоб перетворювач міг завантажити його без повторного навчання. Метод transform() викликає метод Dictionary.doc2bow, який повертає розріджене подання документа у вигляді списку кортежів (token_id, frequency). Однак таке представлення може викликати проблеми в Scikit-Learn, тому ми використовуємо допоміжну функцію sparse2full із Gensim для перетворення розрідженого представлення в масив NumPy:

```
import os
from gensim.corpora import Dictionary
from gensim.matutils import sparse2full
class GensimVectorizer(BaseEstimator, TransformerMixin):
def init (self, path = None):
self.path = path self.id2word = None self.load()
def load(self):
if os.path.exists(self.path):
self.id2word = Dictionary.load(self.path)
def save(self):
self.id2word.save(self.path)
def fit(self, documents, labels = None): self.id2word =
Dictionary(documents) self.save() return self
def transform(self, documents):
for document in documents:
docvec = self.id2word.doc2bow(document)
yield sparse2full(docvec, len(self.id2word))
```

Неважко помітити, як проводиться обгортання методів векторизації, що обговорювалися вище в цьому розділі, перетворювачами Scikit-Learn. Це дає нам велику гнучкість у виборі підходів і водночас дозволяє використовувати утиліти машинного навчання кожної бібліотеки. Подальше розширення цього прикладу та реалізацію перетворювачів для отримання оцінок TF-IDF та розподілених уявлень ми залишаємо читачам для самостійної вправи.

Багато сімейства моделей страждають від «прокляття розмірності»: зі збільшенням розмірності простору ознак дані стають дедалі більше розрідженими і менш інформативними для базового простору рішень. Нормалізація тексту зменшує кількість вимірів і зменшує розрідженість. Крім простої фільтрації лексем (видалення розділових знаків і стоп-слів) існує два основних методи нормалізації: стемінг і лематизація.

Стемінг полягає у використанні серії правил (або моделі) для розділення рядка на менші підрядки. Його мета полягає у видаленні приставок і суфіксів зі слів, що змінюють їх значення. Наприклад, видаляються суфікси 's' і 'es', які у латинських мовах зазвичай вказують на множину. Лематизація, з іншого боку, виконує пошук кожної лексеми у словнику та повертає канонічну (словникову) форму слова, яка називається лемою. Оскільки цей метод заснований на

пошуку еталона, він допомагає справлятися з незвичайними випадками та обробляє лексеми, що є різними частинами мови. Наприклад, дієслово 'gardening' (садівництво) має перетворюватися на лему 'to garden', тоді як іменники 'garden' (сад) і 'gardener' (садник) – у різні леми. Стеммінг перетворив би всі ці слова на одну лексему 'garden' (сад).

Стемінг та лематизація мають свої переваги та недоліки. Стемінг вимагає лише розбити слово на підрядки, тому виконується швидше. Лематизація, навпаки, вимагає пошуку у словнику чи базі даних і використовує теги частин мови для виявлення кореневої леми слова, що робить її набагато повільнішим за стеммінг, але також набагато ефективнішим. Щоб забезпечити систематичну нормалізацію тексту, напишемо свій перетворювач, який поєднує всі ці етапи. Наш клас TextNormalizer приймає на вході мову, яка використовується для завантаження правильного набору стоп-слів із корпусу NLTK. Також можна передбачити налаштування для TextNormalizer, щоб дати можливість вибирати між стемінгом та лематизацією, і передачу мови у SnowballStemmer. Для фільтрації сторонніх лексем визначимо два методи. Перший, is_punct(), порівнює першу букву в назві категорії Unicode кожного символу з 'P' (Punctuation – розділові знаки); другий, is_stopword(), перевіряє, чи присутня дана лексема в нашій множині стоп-слів:

```
import unicodedata
from sklearn.base import BaseEstimator, TransformerMixin
class TextNormalizer(BaseEstimator, TransformerMixin):
def __init__(self, language = 'english'):
self.stopwords = set(nltk.corpus.stopwords.words(language))
self.lemmatizer = WordNetLemmatizer()
def is_punct(self, token): return all(
unicodedata.category(char).startswith('P') for char in token
)
def is_stopword(self, token):
return token.lower() in self.stopwords
```

Тепер додаємо метод normalize(), який приймає єдиний документ, тобто список абзаців, що містять списки речень, які представлені списками кортежів (token, tag) – цей формат даних ми отримали в результаті попередньої обробки документів HTML:

```
def normalize(self, document): return [
self.lemmatize(token, tag).lower() for paragraph in document for
sentence in paragraph for (token, tag) in sentence if not
self.is_punct(token) and not self.is_stopword(token) ]
```

Цей метод застосовує функції фільтрації видалення небажаних лексем і потім виконує лематизацію. Метод lemmatize() спочатку перетворює теги частин мови з набору Penn Treebank, який використовується функцією nltk.pos_tag, на теги WordNet, вибираючи за іменником іменник.

```
def lemmatize(self, token, pos_tag): tag = {
'N': wn.NOUN,
'V': wn.VERB,
'R': wn.ADV,
'J': wn.ADO
}.get(pos_tag[0], wn.NOUN)
return self.lemmatizer.lemmatize(token, tag)
```

Нормалізація тексту – лише одна з багатьох методологій, яка до того ж активно використовує NLTK, що може збільшити небажані для вашої програми накладні витрати. У числі інших можливих варіантів можна назвати видалення лексем, що з'являються частіше або рідше деякої межі, або видалення стоп-слів з наступним вибором перших 5-10 тисяч слів, що найчастіше зустрічаються. Ще один варіант - просто обчислити накопичену частоту і відібрати слова, що становлять 10-50% накопиченої частоти. Ці методи дозволили б нам ігнорувати слова, що зустрічаються в тексті дуже рідко або занадто часто, і виявити терміни, потенційно найбільш значущі для прогнозування цього корпусу

Операція нормалізації тексту має бути необов'язковою та застосовуватися з обережністю, тому що має руйнівний характер, оскільки видаляє частину інформації. Регістр символів, розділові знаки, стоп-слова та різноманітність конструкцій слів – все це дуже важливо для розуміння тексту природною мовою. Деякі моделі можуть вимагати наявності індикаторів, таких як регістр символів. Прикладом може бути класифікатор іменованих сутностей, тому що в англійській мові власні імена прийнято записувати з великої літери. Альтернативний спосіб зменшення розмірності – використовувати метод

головних компонентів (Principal Component Analysis, PCA) або сингулярне розкладання (Singular Value Decomposition, SVD) для зменшення розмірності простору ознак до певної величини (наприклад, п'ять або десять тисяч вимірювань) на основі частоти слів. Ці перетворювачі повинні застосовуватися після векторизації і можуть мати ефект злиття слів, схожих у даному векторному просторі.

Висновки до розділу 2

Для сегментації тексту документів може використовуватись морфологічний аналіз і Word2vec. Інструменти морфологічного аналізу можуть бути корисними у різних завданнях обробки української мови

Важливою особливістю Word2vec є те, що вони охоплюють велику кількість лінгвістичних контекстів і відносин між словами. На даний час існує дві основні архітектури Word2vec: CBOW і Skip-gram.

Word2vec можна реалізувати, наприклад за допомогою бібліотеки Gensim. Перетворення документів у чисельний вид дає можливість здійснювати їх аналіз та створювати екземпляри, з якими зможуть працювати алгоритми машинного навчання. Швидкість обробки, якість результату та оцінки тематичних моделей не мають аналогів у Gensim. Існує чотири види кодування вектору: частотне, пряме, TF-IDF, розподілене представлення.

Прийоми векторизації в Gensim становлять значний інтерес, тому що ця бібліотека дозволяє зберігати та завантажувати корпуси з диска, відокремлюючи їх від конвеєра обробки. Однак є можливість написати свій перетворювач, який використовує механізм векторизації із Gensim.

Крім простої фільтрації лексем (видалення розділових знаків і стоп-слів) існує два основних методи нормалізації: стемінг і лематизація.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО РЕАЛІЗАЦІЇ МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ ДОКУМЕНТІВ

3.1 Оцінювання продуктивності моделі

Для перевірки працездатності запропонованої в роботі моделі глибокого навчання нейронної мережі сегментації тексту використана база, що містить 432 договори. Варіант договору наведений на рис. 3.1.

ДОГОВІР ОРЕНДИ БУДИНКУ ТА ЗЕМЕЛЬНОЇ ДІЛЯНКИ НА ЯКІЙ ВОНО РОЗМІЩЕНО № _____
м. _____ _____«____» _____ _____ р.
_____ в особі _____, що діє на підставі
_____, іменований надалі «Орендодавець», з одного боку, і
_____ в особі _____, що діє на підставі
_____, іменований надалі «Орендар», з іншого боку, іменовані уклали цей договір, надалі «Договір», про наступне:
1. ПРЕДМЕТ ДОГОВОРУ
1.1. <s1>За цим договором Орендодавець зобов'язується передати Орендареві в тимчасове володіння та користування _____, іменоване надалі «Будівля»,<s3> за плату, з метою: _____.</s3></s1>
1.2. <s4>Термін дії цього договору _____ років.</s4>
1.3. <s3>Розмір орендної плати складає: _____ рублів на місяць.</s3>
1.4. <s1>Орендна плата вноситься в наступні строки та в наступному порядку: _____ і не може переглядатися Сторонами частіше ніж раз на рік.</s1>
1.5. <s1>Одночасно з передачею Орендарю прав володіння та користування будинком за цим договором, йому передаються права на ту частину земельної ділянки, на якій розташована будівля та необхідна для її використання.</s1>
1.6. Також Орендареві надається право оренди на прилеглу до будівлі земельну ділянку, що належить Орендодавцю на праві власності _____.
1.7. <s3>Орендна плата, встановлена у п.1.3. цього договору, включає плату за користування будівлею, плату за користування земельною ділянкою, на якій вона розташована, а також плату за прилеглу до будівлі земельну ділянку, зазначену в п.1.6.</s3>
2. ЗОБОВ'ЯЗАННЯ СТОРІН ЗА СПРАВЖНІМ ДОГОВОРОМ
2.1. <s1>Орендодавець зобов'язується:
• передати Орендарю будівлю не пізніше «___»_____ _____ року Вказане зобов'язання вважатиметься виконаним після надання Орендарю будівлі за володіння та користування та підписання Сторонами акту про його передачу;
• передати Орендарю права на прилеглу до будівлі земельну ділянку, що належить Орендодавцю на праві власності, розмір якої встановлено у п.1.5. справжньої угоди;
• <s3>здійснювати за свій рахунок капітальний ремонт будівлі.</s1></s3>
2.2. <s1>Орендар зобов'язується:
• <s3>своєчасно вносити орендну плату;</s3>
• користуватися будинком відповідно до його призначення;
• підтримувати будівлю у справному стані;
• <s3>здійснювати за свій рахунок поточний ремонт будівлі та нести витрати на її утримання;</s3>
• повернути Орендодавцю при припиненні цього договору будинок у тому стані, в якому його було отримано, з урахуванням нормального зносу.</s1>
3. ПОЛІПШЕННЯ, ВІДДІЛЬНІ І НЕ ВІДДІЛЬНІ ВІД БУДІВЛІ, ВІДПОВІДАЛЬНІСТЬ СТОРІН
3.1. Вироблені Орендарем відокремлені покращення будівлі є його власність.
3.2. <s3>У разі, коли Орендар зробив за рахунок власних коштів та за згодою Орендодавця поліпшення будівлі, не відокремлені від неї без шкоди, Орендар має право після припинення цього договору на відшкодування вартості таких покращень.</s3>
3.3.<s1> За невиконання або неналежне виконання цього договору Сторони несуть відповідальність, передбачену чинним цивільним

Рисунок 3.1 – Варіант тегованого договору

Для прикладу, в роботі виконується сегментація тексту за 6-ма тегами, що наведені на рис. 3.2.

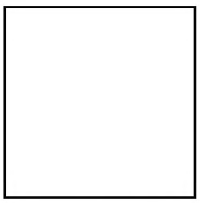

Рисунок 3.2 – Теги для сегментації тексту

Для реалізації моделі використаний хмарний сервіс Google Colab. Для коду на мові Python необхідно підключити бібліотеки (рис. 3.3).

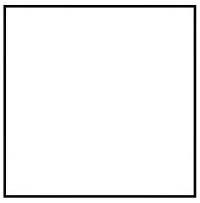

Рисунок 3.3 – Підключення бібліотек

Для реалізації моделі вибрано кілька архітектур, наприклад U-Net та CNN (рис. 3.4 і 3.5).

Лінійна Conv1d-мережа

```
model_conv1d = create_Conv1d(xLen, embeddingSize) # Створюємо просту мережу
model_conv1d.summary() # Дивимося отриману архітектуру
```

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 256, 300)]        0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 256, 16)           14416
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 256, 16)           784
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 256, 16)           784
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 256, 6)            294
=================================================================
Total params: 16,278
Trainable params: 16,278
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] history = model_conv1d.fit(xTrainGENSIM, yTrainGENSIM, epochs=20, batch_size=200, validation_split = 0.2)
    # Згодовуємо моделі отримані вектори
```

Рисунок 3.4 – Архітектура CNN

Рисунок 3.5 – Фрагмент архітектури U-Net

Повний текст програмного коду наведений у Додатку А. В якості основи використана бібліотека Gensim. Для навчання нейронної мережі форматувався датасет, що містить вибірки xTrain та yTrain за допомогою методу Word2wec Gensim. Довжина xTest дорівнює 414, а yTrain – 256. Для цього виконується поділ окремої вибірки на вікна. Довжина вікна дорівнює 256, крок навчання – 30, а кількість вимірювань для векторного простору – 300. Кожному індексу (слову) з договору необхідно поставити у відповідність вектор із 6-ти елементів, що відповідають тегам категорій (рис. 3.6). Для кожного варіанту архітектури виковувався підбір гіперпараметрів. Бібліотека Keras не має Conv1DTranspose. Тому такий шар прописуємо вручну. Він необхідний для розвороту після шару Maxpooling.

Специфіка роботи в Google Colab вимагає збереження масивів з оперативної пам'яті у пам'ять runtime. В загальному випадку, це може бути корисним у кількох ситуаціях. У процесі тривалих обчислень чи аналізу

даних часто виникає потреба зберігати проміжні результати. Це дозволяє уникнути повторення тривалих обчислень у разі втрати даних через переривання з'єднання або обмеження часу сесії. Переміщення даних між оперативною пам'яттю і пам'яттю runtime може допомогти оптимізувати використання доступної оперативної пам'яті, особливо при роботі з великими обсягами даних. Google Colab має обмеження часу сесії. Після того, як сесія закінчується, всі дані в оперативній пам'яті губляться. Збереження важливих даних у пам'ять runtime дозволяє відновити стан вашого проекту під час перезапуску сесії. Іноді зручніше зберегти результати в пам'ять runtime для подальшого використання в тій же сесії замість того, щоб зберігати їх на Google Диск або завантажувати щоразу заново. Доступ до даних, збережених в пам'яті runtime, зазвичай швидше, ніж до даних, що зберігаються на віддалених сховищах, таких як Google Диск. Важливо пам'ятати, що пам'ять runtime в Colab не є постійною і дані будуть втрачені після завершення сесії. Тому для довгострокового зберігання даних слід використовувати інші методи, такі як збереження на Google Диск.

Після проведення навчання, можна спостерігати результат роботи моделі, наприклад, рис. 3.6.

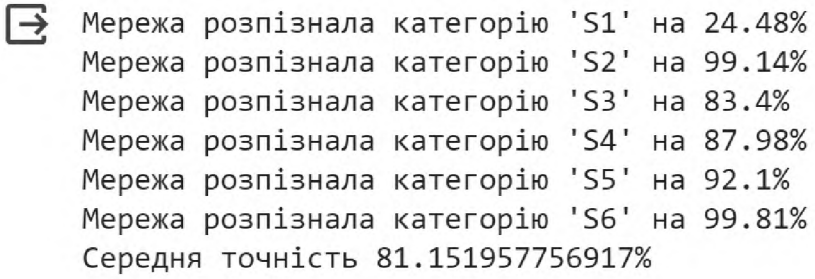

Рисунок 3.6 – Вектор

Таким чином, продуктивність складає 81 %. Для підвищення продуктивності моделі глибокого навчання нейронної мережі сегментації тексту доцільно оптимізувати параметри, розширити та збагатити датасет, а також реалізувати гібридні мережі.

3.2 Перспективні тенденції та технології в сегментації тексту

Оскільки сфера сегментації тексту продовжує розвиватися, кілька нових тенденцій і технологій формують майбутнє NLP і AI. Тому надалі доцільно охарактеризувати їхній потенційний вплив.

Впровадження попередньо підготовлених мовних моделей, таких як BERT, GPT та їх наступників, призвело до зміни парадигми в NLP. Ці моделі, попередньо навчені на масивних текстових корпусах, фіксують багату лінгвістичну інформацію, яку можна точно налаштувати для різних завдань, включаючи сегментацію тексту. Очікується, що безперервний розвиток більш потужних і ефективних попередньо навчених моделей ще більше покращить продуктивність сегментації тексту та інших завдань NLP.

Із зростаючим інтересом до мультимодального AI, інтеграція сегментації тексту з іншими модальностями, такими як зображення, аудіо та відео, відкриває нові можливості для покращення розуміння мови. Враховуючи додаткову контекстну інформацію з цих модальностей, моделі сегментації можуть краще зрозуміти значення та структуру складних текстових даних, що призведе до більш точних і контекстно-орієнтованих систем NLP.

Активне навчання та напівконтрольоване навчання. Позначення даних для контрольованого навчання може бути тривалим і дорогим процесом. Техніки активного навчання та напівконтрольованого навчання дозволяють моделям навчатися на обмежених позначених даних, одночасно використовуючи величезні обсяги непозначених даних. Ці підходи можуть бути особливо корисними в задачах сегментації тексту для мов із низьким ресурсом і спеціалізованих доменів, де отримання даних з мітками може бути складним завданням.

Оскільки моделі NLP стають все складнішими та складнішими, розуміння процесів прийняття рішень, що лежать в основі їхніх прогнозів, має вирішальне значення для надійності та підзвітності. Дослідження

можливостей пояснення та інтерпретації моделей сегментації тексту можуть допомогти практикам краще зрозуміти поведінку моделі, діагностувати помилки та підвищити загальну надійність систем NLP.

Навчання суперництву та стійкість. Змагальний тренінг, який передбачає навчання моделей для захисту від зловмисного введення, є новою тенденцією в NLP та AI. Розробка моделей сегментації тексту, стійких до агресивних атак, життєво важлива для забезпечення безпеки та надійності систем обробки мови, особливо в критичних програмах, таких як кібербезпека та модерація вмісту.

Динамічний ландшафт сегментації тексту продовжує створювати нові виклики та можливості, стимулюючи інновації в NLP та AI. Враховуючи ці нові тенденції та технології, дослідники та практики можуть ще більше вдосконалити наше розуміння людської мови та розробити більш ефективні, надійні та інклюзивні мовні технології для майбутнього [41].

Методологію сегментації зображення [42] можна адаптувати для сегментації тексту кількома способами. Один із підходів полягає в тому, щоб розглядати слова або символи в тексті як області на зображенні та використовувати методи сегментації зображення для сегментації цих областей [43]. Для потрібно виконати кілька процедур.

Попередня обробка тексту. Текст попередньо обробляється, щоб перетворити його на представлення зображення, де кожне слово чи символ розглядається як область. Сегментація зображення – методи сегментації зображення потім використовуються для розділення тексту на його складові області, такі як слова чи символи. Це можна зробити за допомогою таких методів, як аналіз зв’язаних компонентів, зростання області або активні контури. Постобробка: нарешті, результати сегментації зображення піддаються наступній обробці для створення бажаної сегментації тексту, наприклад списку слів або списку речень. Перевагою цього підходу є можливість використання багатьох досягнень у сегментації зображень, таких як моделі на основі глибокого навчання, для підвищення точності та

ефективності сегментації тексту. Крім того, цей підхід також може обробляти нестандартний текст, наприклад рукописний текст, який може бути складним для традиційних алгоритмів сегментації тексту. Однак важливо зазначити, що якість представлення зображення тексту може вплинути на продуктивність алгоритмів сегментації зображення, а отже, необхідно ретельно розглянути етап попередньої обробки.

Варто також зазначити, що цей підхід можна поєднувати з іншими методами сегментації тексту для покращення продуктивності. Наприклад, традиційні алгоритми сегментації тексту можна використовувати для надання початкових результатів сегментації, які потім можна вдосконалити за допомогою методів сегментації зображень.

Загалом, застосування методології сегментації зображення до сегментації тексту забезпечує альтернативний підхід, який може бути корисним у певних сценаріях і може бути інтегрований з іншими методами для покращення результатів. Однак важливо враховувати конкретні вимоги кожного завдання сегментації тексту та оцінювати ефективність обраного підходу за допомогою відповідних показників оцінювання.

3.3 Техніко-економічне обґрунтування прийнятих рішень

Сегментування тексту підвищує якість електронного документу обігу та інших чинників, що впливають на бізнес процеси. Для практичної реалізації розглянутої моделі глибокого навчання нейронної мережі сегментації тексту трудомісткість розробки коду складає 380 год. Крім цього, необхідно до 20 годин на тестування програмного продукту та до 10 год для процедури DevOps. При цьому, вважаємо, що набір даних для формування датасету зібраний. Таким чином, загальна трудомісткість складає:

$$T = 380 + 20 + 10 = 410. \quad (3.1)$$

Відповідно, перерахунок трудомісткості у кількість робочих днів можна виконати з виразом:

$$M = \frac{T}{8} \approx 52 \text{ днів або} \approx 2{,}6 \text{ місяця.} \quad (3.2)$$

Враховуючи обсяг зазначених робіт, доцільно найняти програміста-фрілансера на мові Python з відповідною класифікацією. Згідно [44], в середньому по Україні, його заробітна плата складає $C = 44000$ грн. Як наслідок, витрати на заробітну плату програміста-фрілансера дорівнюють:

$$V = M \cdot C = 2{,}6 \cdot 44000 = 114400 \text{ [грн].} \quad (3.3)$$

Згідно, п. 3.1, для реалізації процесу навчання нейронної мережі використовується хмарний сервіс Google Colab [45]. Щоб оптимізувати пошук оптимальних архітектури та гіперпараметрів синтезованої нейронної мережі доцільно отримати тарифний план Google Colab Pro+. Він забезпечує підключення серверних відеокарт (наприклад, NVIDIA A100-SXM4-40GB [46]) та/або TPU. За 3 місяця витрати на цей тарифний план складають 4560 грн. Таким чином, орієнтовні витрати на реалізацію запропонованої моделі складають ≈ 119000 грн.

Висновки до розділу 3

Для перевірки працездатності запропонованої в роботі моделі глибокого навчання нейронної мережі сегментації тексту використана база, що містить 432 договори. На їх основі створюється датасет, що поділяється на вибірки за класичною схемою.

Модель реалізується на мові Python за допомогою хмарного сервісу Google Colab. В якості базової розглядається бібліотека Gensim. З метою збереження проміжних результатів виконується збереження масивів з оперативної пам'яті у пам'ять runtime. Продуктивність нейронної мережі складає 81 %. Це підтверджує висунуті в роботі теоретичні положення. Для

підвищення продуктивності доцільно оптимізувати архітектуру мережі, гіперпараметри та ін. На основі проведених досліджень обґрунтовано перспективні напрями подальшого розвитку та технології в сегментації тексту. Основна увага приділяється застосуванню методології сегментації зображення. Це дозолить підвищити інтелектуальний рівень платформ електронного документообігу.

ВИСНОВКИ

Для розуміння та ефективної обробки тексту, важливо провести його сегментацію, яка включає розділення документа на тематично суміжні частини. Цей процес є критичним, особливо у випадках довгих документів чи стенограм аудіо та відеозаписів, де без сегментації важко вийти на ключову інформацію. У сфері NLP сегментація тексту займає фундаментальне місце і використовується в широкому спектрі застосувань, від пошуку інформації та аналізу настроїв, до машинного перекладу, автоматичного резюмування текстів, розробки чат-ботів та систем розмовного AI, а також модерації контенту. Щоб ефективно реалізувати сегментацію тексту, потрібно вирішити ряд складних завдань. Це включає в себе адаптацію до мовної варіативності, обробку нестандартних текстів, корекцію помилок OCR, аналіз полісемічних виразів та забезпечення обчислювальної ефективності. Також необхідно застосовувати широкий арсенал методів, починаючи від традиційних правил, до статистичних та сучасних методів машинного та глибокого навчання, часто комбінуючи ці підходи для досягнення найкращих результатів.

На сучасному етапі розвитку обчислювальної лінгвістики, для задачі сегментації текстів, можна застосувати різноманітні нейронні архітектури, зокрема, RNN, LSTM, GRU, CNN, архітектури на основі трансформерів та гібридні системи. Для детального аналізу текстів документів застосовується морфологічний аналіз та моделі Word2vec, які демонструють високу ефективність у обробці української мови. Особливість Word2vec полягає у здатності виявляти різноманітні лінгвістичні контексти та зв'язки між словами, з двома основними архітектурами, які варто відзначити. Word2vec можна реалізувати з використанням бібліотеки Gensim, що дозволяє перетворювати текстові документи у числовий формат для подальшого аналізу та використання в алгоритмах машинного навчання. Gensim вирізняється своєю швидкістю обробки, високоякісними результатами та точними оцінками тематичних моделей, пропонуючи чотири методи

кодування векторів: частотний, прямий, TF-IDF та розподілене представлення. Важливою особливістю Gensim є також можливість збереження та завантаження корпусів з диску, що дозволяє відокремити їх від основного процесу обробки. Крім того, користувачі мають змогу створювати власні перетворювачі, використовуючи механізми векторизації, надані бібліотекою Gensim.

Крім базового видалення непотрібних слів і знаків пунктуації, важливими методами обробки тексту є стемінг і лематизація. У цій дисертації для оцінки ефективності розробленої моделі глибокого навчання для сегментації тексту використовувались дані, що містили 432 договори, на базі яких було створено датасет, розподілений згідно з класичною схемою.

Розробка моделі велася на Python із використанням хмарного сервісу Google Colab, при цьому основою стала бібліотека Gensim. Для збереження проміжних даних використовувалась передача даних з оперативної пам'яті в пам'ять runtime. Ефективність моделі становила 81 %, що підтверджує теоретичні засади, висунуті в роботі. Подальше підвищення продуктивності можливе завдяки оптимізації архітектури мережі, гіперпараметрів та іншого. Зроблені висновки вказують на перспективні напрями розвитку та технології у сфері сегментації тексту, з особливим акцентом на методологію сегментації зображень для підвищення інтелектуального потенціалу систем електронного документообігу.

Таким чином, результатами роботи є створенні моделі глибокого навчання нейронної мережі сегментації зображень документів, рекомендацій щодо використання штучного інтелекту в системах електронного документообігу. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні систем електронного документообігу.