

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Факультет \_\_\_\_\_ Комп'ютерних наук \_\_\_\_\_  
(повна назва)

Кафедра \_\_\_\_\_ Штучного інтелекту \_\_\_\_\_  
(повна назва)

**КВАЛІФІКАЦІЙНА РОБОТА**  
**Пояснювальна записка**

рівень вищої освіти \_\_\_\_\_ другий (магістерський) \_\_\_\_\_

\_\_\_\_\_ Дослідження та застосування графів знань та нейромережових моделей \_\_\_\_\_  
\_\_\_\_\_ для обробки природномовних текстів \_\_\_\_\_  
(тема)

Виконав:  
студент 2 курсу, групи \_\_\_\_\_ СШМ-20-3 \_\_\_\_\_  
\_\_\_\_\_ Громак О. В. \_\_\_\_\_  
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту \_\_\_\_\_  
\_\_\_\_\_ (повна назва спеціалізації)

Керівник \_\_\_\_\_ проф. к.т.н, доц. Рябова Н. В. \_\_\_\_\_  
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри \_\_\_\_\_  
(підпис)

\_\_\_\_\_ В.О. Філатов \_\_\_\_\_  
(прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет \_\_\_\_\_ Комп'ютерних наук \_\_\_\_\_  
(повна назва)  
Кафедра \_\_\_\_\_ Штучного інтелекту \_\_\_\_\_  
(повна назва)  
Рівень вищої освіти \_\_\_\_\_ другий (магістерський) \_\_\_\_\_  
Спеціальність \_\_\_\_\_ 122 Комп'ютерні науки \_\_\_\_\_  
(код і повна назва)  
Тип програми \_\_\_\_\_ освітньо-наукова \_\_\_\_\_  
(освітньо-професійна або освітньо-наукова)  
Освітня програма \_\_\_\_\_ Системи штучного інтелекту (СШІ) \_\_\_\_\_  
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри \_\_\_\_\_  
(підпис)

«\_\_\_\_\_» \_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ**  
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові \_\_\_\_\_ Громак Олександрі Валеріївні \_\_\_\_\_  
(прізвище, ім'я, по батькові)

1. Тема роботи \_\_\_\_\_ Дослідження та застосування графів знань та нейромережових моделей для обробки природномовних текстів \_\_\_\_\_

затверджена наказом університету від 24 березня 2022 р. № 414Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16 травня \_\_\_\_\_ 2022 р.

3. Вихідні дані до роботи Науково-технічні публікації щодо дослідження методів вилучення інформації, розробки глибоких нейронних мереж і графів знань, та методів їх навчання, дані з інтернет-джерел щодо розробки алгоритмів методів видобутку іменованих сутностей, текстові дані для навчання та обробки графу знань та методів нейронних мереж.

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі. Дослідження методів видобутку сутностей, токенизація сутностей, тегування частин мови, розпізнавання та відбукоток іменованих сутностей.

Опис алгоритму видобутку іменованих сутностей з тексту для вирішення задачі побудови графу знань та нейромережових моделей.

Постановка цілей та задачі дослідження.

Розробка та опис програмного додатку.

Проведення аналізу виконаної роботи та формування висновків.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1.1 - Речення у вигляді дерева, Рисунок 1.2 – Приклад роботи методу NER, Рисунок 1.3 – Приклад роботи методу RE, Рисунок 1.4 – Приклад роботи методу реферування, Рисунок 1.5 – Завдання та архітектури нейронних мереж, Рисунок 1.6 - ШНМ із зворотними (рекурентними) зв'язками, Рисунок 1.7 – Архітектура одновимірної згорткової нейронної мережі, Рисунок 1.8 – Архітектура мережі з увагою, Рисунок 1.9 – Архітектура перенесення навчання нейронних мереж, Рисунок 1.10 – Відображення графу знань, Рисунок 1.11 – Приклад графу знань, Рисунок 1.12 – Представлення даних у вигляді графу знань, Рисунок 1.13 - Представлення даних у вигляді графу знань, Рисунок 2.1 – Приклад роботи вилучення інформації, Рисунок 2.2 – Приклад роботи NER, Рисунок 2.3 – Схема матриці помилок, Рисунок 2.4 – Підходи NER, Рисунок 2.5 – Приклад BIOES-схеми

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

### КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	28.03.22	
2	Аналіз предметної області та об'єкту дослідження	28.03.22-03.04.22	
3	Пошук та аналіз теоретичного матеріалу	03.04.22-10.04.22	
4	Дослідження існуючих методів рішення задачі	10.04.22-15.04.22	
5	Вибір програмних та технічних засобів	15.04.22-17.04.22	
6	Проектування та розробка програмного додатку	17.04.22-23.04.22	
7	Аналіз результатів роботи програмного додатку	23.04.22-24.04.22	
8	Оформлення пояснювальної записки	25.04.22-03.04.22	
9	Попередній захист	14.05.22	
10	Захист перед ЕК	16.05.22	

Дата видачі завдання 28 березня 2022 р.

Студент \_\_\_\_\_  
(підпис)

Керівник роботи \_\_\_\_\_  
(підпис) \_\_\_\_\_  
(посада, прізвище, ініціали)

## РЕФЕРАТ

Пояснювальна записка: 93 с., 52 рис., 2 табл, 2 дод., 22 джерела.

ВИДОБУВАННЯ ІНФОРМАЦІЇ, ГРАФ ЗНАНЬ, ОБРОБКА ПРИРОДНОЇ МОВИ, ШТУЧНИЙ ІНТЕЛЕКТ, ШТУЧНА НЕЙРОННА МЕРЕЖА, GOOGLE COLAB, PYTHON

Об'єкт дослідження – методи обробки природномовних текстів для задачі розпізнавання іменованих сутностей.

Предмет дослідження – використання графів знань та нейронних мереж для задач обробки природномовних текстів. Для реалізації проекту була обрана мова програмування Python, так як вона є найбільш зручна та пристосована для роботи із штучним інтелектом.

Мета роботи – дослідження графів знань та штучних глибинних нейронних мереж для вирішення задачі обробки природної мови.

Методи дослідження – аналіз теоретичного матеріалу, аналіз технічної літератури та практична реалізація.

В роботі проведено теоретичний аналіз існуючих методів для вирішення задач, архітектур нейронних мереж, графів знань та демонстрація їх практичного застосування для обробки природномовних текстів.

## **ABSTRACT**

Explanatory note: 93 p., 52 figures, 2 tables, 2 apps, 22 sources,

ARTIFICIAL INTELLIGENCE, ARTIFICIAL NEURAL NETWORK,  
GOOGLE COLAB, INFORMATION EXTRACTION, KNOWLEDGE GRAPH,  
NATURAL LANGUAGE PROCESSING, PYTHON

The object of research – methods of natural language process for the task of named entities recognition.

The subject of research – using of knowledge graphs and neural networks for the tasks of natural language process. For the project was chosen programming language Python, as it is the most convenient and suitable for working with artificial intelligence, and the environment for developing Google Colab.

Methods of work – methods for knowledge graphs and artificial deep neural networks training to solve the problem of natural language processing.

Research methods – analysis of theoretical material, analysis of technical literature and practical implementation.

The paper presents a theoretical analysis of existing methods for solving problems, neural network architectures, knowledge graphs and demonstration of their practical application for processing natural language texts.

## ЗМІСТ

Перелік скорочень, умовних позначень, символів, одиниць і термінів .....	8
Вступ.....	9
1 Аналіз предметної області та постановка задачі .....	11
1.1 Загальні відомості .....	11
1.2 Методи обробки природномовних текстів.....	13
1.3 Проблеми використання NLP .....	18
1.4 Використання нейронних мереж для задач NLP .....	19
1.5 Використання графів знань для задач NLP .....	23
1.6 Постановка задачі .....	27
2 Методи видобування інформації .....	29
2.1 Загальні відомості .....	29
2.2 Основні задачі та підходи видобування інформації.....	33
2.3 Named entity recognition.....	34
2.3.1 Задачі вирішення NER.....	36
2.3.2 Проблеми використання NER.....	37
2.3.3 Оцінка Named Entity Recognition.....	38
2.3.4 Підходи по вирішення NER .....	41
2.3.5 Вирішення задачі NER .....	44
2.4 Word Embeddings.....	49
2.4.1 Word2Vec .....	52
3 Графи знань та нейромережеві моделі.....	58
3.1 Графи знань .....	58
3.1.1 Загальні відомості та області застосувань графів знань .....	58
3.1.2 Побудова графу знань.....	62
3.2 Нейронні мережі.....	66
3.2.1 Нейронні мережі для задачі обробки природної мови.....	66
3.2.2 RNN .....	67
3.2.3 Архітектура LSTM.....	69

4 Практичне застосування отриманих результатів досліджень .....	75
4.1 Вибір програмних засобів .....	75
4.1.1 Google Colab .....	75
4.2 Розробка програми .....	76
Висновки .....	85
Перелік джерел посилання .....	87
Додаток А Вихідний код програми .....	90
Додаток Б Відомість кваліфікаційної роботи магістра .....	93

## **ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ**

ШІ – штучний інтелект;

ШНМ – штучна нейронна мережа;

ІЕ – information extraction – видобування інформації;

KG – knowledge graph – граф знань;

NEL – named entity linking – зв'язування іменованих сутностей;

NER – named entity recognition – видобування іменованих сутностей;

NLP – natural language processing – обробка природної мови;

POS – part of speech tagging – тегування частин мови;

RE – relation extraction – видобування інформації.

## ВСТУП

Мова – це спосіб спілкування, за допомогою якого ми можемо говорити, читати і писати. Наприклад, ми думаємо, приймаємо рішення, плануємо, тощо природною мовою; точно, словами. Однак велике питання, яке постає перед нами в цю епоху штучного інтелекту, полягає в тому, чи можемо ми так само спілкуватися з комп'ютерами. Іншими словами, чи можуть люди спілкуватися з комп'ютерами своєю природною мовою? Для нас є складним завданням розробити програми для обробки природовмивних текстів, оскільки комп'ютерам потрібні структуровані дані, але людська мова неструктурована і часто неоднозначна за своєю природою.

У цьому сенсі ми можемо сказати, що обробка природної мови (NLP) – це підгалузь комп'ютерних наук, особливо штучного інтелекту (ШІ), яка стурбована тим, щоб комп'ютери могли розуміти й обробляти людську мову. Технічно основним завданням NLP було б програмувати комп'ютери для аналізу та обробки величезної кількості даних природною мовою.

Можна виділити три високорівневі групи завдань обробки текстів природною мовою. Перша група – це лінгвістичний аналіз. Методи в галузі лінгвістичного аналізу спрямовані на розбір структури тексту на різних рівнях. Друга група – методи отримання ознак з текстів. Вони частково перетинаються з першою групою, але, тоді як завдання лінгвістичного аналізу – це самостійні завдання, завдання отримання ознак завжди передують застосуванню методів машинного навчання. Третя група – прикладні завдання, вони ближчі до бізнесу, до користувача. Як правило, для їх вирішення використовуються методи з перших груп зі спеціальними надбудовами.

Наше людське знання забезпечує формальне розуміння світу. Було здійснено багато спроб вирішити проблему неструктурованих даних, однак

графи знань (Knowledge Graphs, KG) та нейромережеві методи – це найсучасніші та найкращі способи узгодження даних.

Ми можемо використовувати обробку природної мови для обробки текстових джерел і створення графів знань, які можуть підтримувати різноманітну аналітику. Усі дані, джерела даних та бази даних будь-якого типу можуть бути представлені та реалізовані у вигляді графу – тобто представляють структурні відносини між сутностями.

Графи знань стають все більш популярним напрямком досліджень у сфері пізнання та інтелекту на рівні людини. Вони надають найбільш відповідний для інтерпретації і ефективний спосіб зберігання різноманітних знань, пропонують простоту обслуговування і самоаналізу, що перевершує багато альтернативи та дозволяють здійснювати організацію та представлення неструктурованих текстових даних у вигляді знань.

Як нейронний, так і knowledge-based підходи для обробки природної мови мають сильні та слабкі сторони. Нейронні методи надзвичайно потужні і постійно займають лідируючі позиції поточних таблиць лідерів NLP. Однак вони також чутливі до таких проблем, як кількість і якість навчальних даних або пов'язування моделей із тим, як люди використовують мову та своє розуміння світу. З іншого боку, хоча і не повністю вільні від таких проблем, системи NLP, засновані на структурованих уявленнях знань, як правило, краще підходять для вирішення деяких з них. Однак вони можуть вимагати значної інженерної роботи, щоб постійно контролювати такі структуровані уявлення.

Враховуючи вищесказане, в даній роботі буде проведено дослідження методів обробки природномовних текстів за допомогою графів знань та нейромережевих моделей.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Загальні відомості

У якості предметної області було обрано задачу обробки проридомовних текстів за допомогою графів знань та нейромережевих моделей.

Обробка природної мови – це здатність комп'ютерної програми розуміти людську мову, як вона звучить та пишеться – називається природною мовою. Це компонент штучного інтелекту (ШІ).

Хоча NLP існує більше 50 років і має коріння в галузі лінгвістики, історія машинного перекладу сягає сімнадцятого століття, коли такі філософи, як Лейбніц і Декарт, висунули пропозиції щодо кодів, які б зв'язували слова між мовами. Усі ці пропозиції залишалися теоретичними, і жодна не привела до розробки справжньої машини. А сфера обробки природної мови почалася в 1940-х роках, після Другої світової війни. У той час люди усвідомлювали важливість перекладу з однієї мови на іншу і сподівалися створити машину, яка зможе виконувати такий переклад автоматично. Проте, очевидно, завдання виявилось не таким легким, як люди спочатку уявляли [1].

До 1958 року деякі дослідники виявляли важливі проблеми в розвитку NLP. Одним із цих дослідників був Ноам Хомскі, який вважав занепокоєним те, що моделі мови розпізнають безглузді, але граматично правильні речення так само нерелевантні, як і безглузді та граматично неправильні. Хомскі вважав проблематичним те, що речення «Безбарвні зелені ідеї сплять люто» (Colorless green ideas sleep furiously) було класифіковано як малоймовірне в тій же мірі, що і «Люто сплять ідеї безбарвної зелені» (Furiously sleep ideas green colorless) (рисунок 1.1); будь-який носій англійської може розпізнати

перший граматично правильним, а другий – неправильним, і Хомскі вважав, що того ж слід очікувати від моделей машин [1].

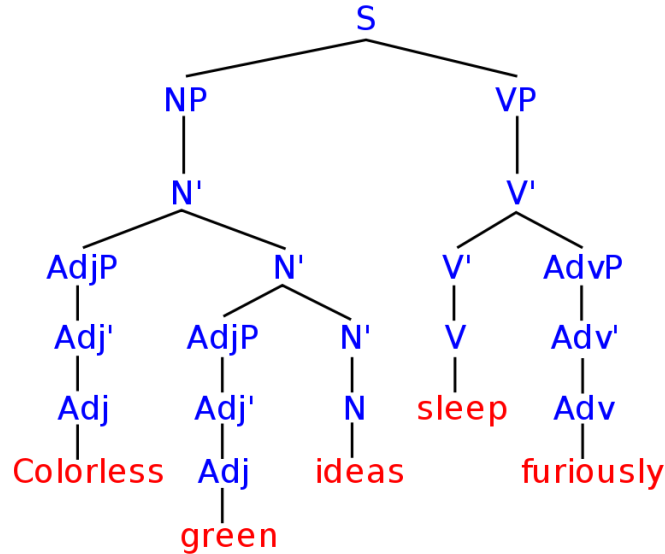


Рисунок 1.1 – Речення у вигляді дерева

Щоб машини розуміли мову, вони повинні інтуїтивно осягати знання, які лежать в основі ситуацій, з якими вони стикаються в тексті. Таке просте твердження, як іде дощ, одразу ж має на увазі сукупність спільного контексту для будь-якого читача: вони повинні взяти з собою парасольку, дороги будуть слизькими, посилений рух може спричинити запізнення, дощові чоботи краще, ніж сандалі, і багато іншого. Системи розуміння мови повинні мати можливість надійно використовувати ці знання для прийняття рішень або дій. Спостереження за світом завжди більш насичені та детальніші, ніж інформація, яка явно передається мовою, і машини повинні бути в змозі заповнювати деталі, що залишилися, висновками здорового глузду.

Нещодавні досягнення в обробці природної мови досягли значного прогресу у визначенні розумних наслідків ситуацій, описаних у тексті. Ці

методи, як правило, передбачають навчання моделей мови з високими параметрами і продемонстрували помітне покращення у виконанні різноманітних еталонних кінцевих завдань у розумінні природної мови. Однак ці системи є крихкими – часто виходять з ладу, коли надаються вхідні дані поза розповсюдженням – і не піддаються інтерпретації – не здатні надати уявлення про те, чому ці різні вхідні дані викликають зміщену поведінку. Тим часом традиційні підходи до розуміння природної мови, які зосереджуються на зв'язуванні мови з базовими знаннями з великих онтологій, залишаються обмеженими через їх нездатність масштабуватися до ситуативного різноманіття, вираженого мовою.

## 1.2 Методи обробки природномовних текстів

Оскільки NLP має широке коло завдань з обробки текстів природною мовою, розглянемо набір класичних завдань, вирішення яких несе практичну користь.

Перше і найважливіше завдання – це машинний переклад [4]. Це з перших завдань обробки мови, разом із яким виникла і розвивалася сама ця область. Перші програми перекладу були побудовані в середині минулого століття та були засновані на найпростішій стратегії послів. В даний час існує цілий спектр комп'ютерних систем машинного перекладу (різної якості), від великих міжнародних дослідницьких проєктів до комерційних автоматичних перекладачів. Істотний інтерес представляють проєкти багатомовного перекладу з використанням проміжної мови, якою кодується сенс фраз, що перекладаються. Але завдання отримання автоматичного перекладу високої якості так і залишається невирішеним. Це в якомусь сенсі двигун NLP, одне з найбільших завдань, яким можна займатися.

Друге завдання – класифікація текстів [4]. Суть завдання полягає в тому, що нам дано набір текстів, і є завдання класифікувати ці тексти за категоріями. Класифікація означає віднесення кожного документа до певного класу із заздалегідь відомими параметрами, а кластеризація – розбиття безлічі документів на кластери, тобто підмножини тематично близьких документів. Для вирішення цих завдань застосовуються методи машинного навчання, у зв'язку з чим ці прикладні завдання часто відносять до напряму Text Mining, що розглядається як частина наукової галузі Data Mining (інтелектуальний аналіз даних). Завдання класифікації набуває все більшого поширення і має безліч способів застосування.

Перший і один із найважливіших з практичної точки зору способів застосування – класифікація листів на спам і не спам. Інший класичний варіант – рубрикація.

Рубрикація – багатокласова класифікація новин за категоріями, тобто віднесення тексту до однієї із заздалегідь відомих тематичних рубрик (зазвичай рубрики утворюють ієрархічне дерево тематик) – зовнішня політика, спорт, кіно тощо. Або, припустимо, коли приходять на пошту листи, і ми хочемо відокремити замовлення з інтернет-магазину від авіаквитків та броні готелів. Третій класичний варіант застосування завдання текстової класифікації – сентиментний аналіз. Наприклад, класифікація відгуків на позитивні, негативні та нейтральні.

Оскільки можливих категорій, на які можна ділити тексти, можна вигадати дуже багато, текстова класифікація є однією з найпопулярніших практичних завдань NLP.

Третє завдання – вилучення іменованих сутностей (Named Entity Recognition, NER). При вирішенні цього завдання здійснюється виділення в тексті природною мовою певних об'єктів – іменованих сутностей (імен персоналій, географічних назв, назв фірм тощо), їх відносин і пов'язаних з

ними подій. Як правило, це реалізується на основі часткового синтаксичного аналізу тексту, що дозволяє обробляти великі масиви текстів, зокрема, потоки новин від інформаційних агентств. Виділені дані тим чи іншим чином структуруються чи візуалізуються (рисунок 1.2).

Тобто ми виділяємо в тексті ділянки, які відповідають заздалегідь вибраному набору сутностей. Найпростіший метод – словниковий. Він дозволяє досить надійно виділити відомі і при цьому унікальні імена, такі як «Samsung» або «Apple». Ці слова не мають інших сенсів чи функції, крім позначення даних конкретних організацій. Для деяких типів сутностей (наприклад, адреса електронної пошти або номер телефону) добре підходять регулярні вирази. Однак на практиці для підвищення точності, тобто зменшення кількості хибнопозитивних пророцтв, необхідно застосовувати імовірнісні моделі. У тексті «Тім Кук – генеральний директор компанії Apple» ви повинні зрозуміти, що Тім Кук – це персона, а Apple – це організація.

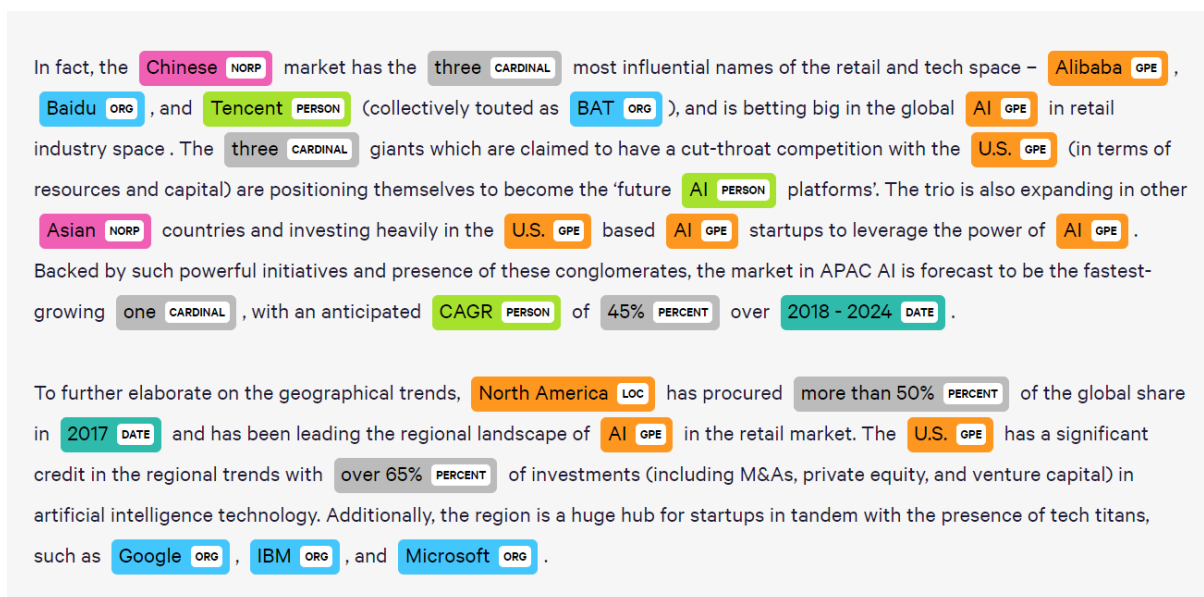


Рисунок 1.2 – Приклад роботи методу NER

З третім завданням пов'язано четверте – завдання вилучення фактів і відносин (relation extraction). Наприклад, є відносини до роботи. З тексту «Тім Кук – генеральний директор компанії Apple» ясно, що наш герой пов'язаний професійними відносинами з Apple. Те саме можна сказати безліччю інших способів: «Компанію Apple очолює Тім Кук» або «Тім Кук пройшов шлях від менеджера корпорації ІВМ до генерального директора компанії Apple». Ці пропозиції відрізняються як предикатом, а й структурою.

Прикладами інших відносин, що часто виділяються, є відносини купівлі/продажу, володіння, факт народження з атрибутами – датою, місцем тощо.

Таке завдання використовується під час структуризації неструктурованої інформації. Крім того, це важливо в питаннях і відповідях і діалогових системах, в пошукових системах – завжди, коли нам потрібно аналізувати питання і розуміти, до якого типу він відноситься, а також, які обмеження є на відповідь (рисунок 1.3).

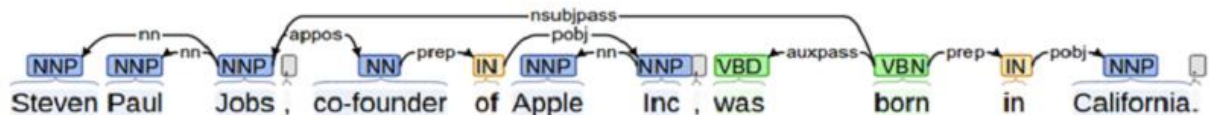


Рисунок 1.3 – Приклад роботи методу RE

Четвертою і однією з найпоширеніших завдань є, згадані раніше, питання-відповіді та діалогові системи (чат-боти) [4]. Приклад можливого питання: Хто придумав вилку? Завдання вирішується шляхом визначення типу питання, пошуком текстів, що потенційно містять відповідь на це питання (при цьому зазвичай застосовуються пошукові машини), а потім отримання відповіді з виданих текстів. У системах питання-відповідь ми

ставимо машині питання, машина шукає відповідь на нього в базі даних або корпусі текстів. Прикладами таких систем можуть бути IBM Watson або Wolfram Alpha.

Amazon, Alexa – це традиційні приклади діалогових систем. Щоб вони нормально працювали, має вирішити багато завдань NLP. Наприклад, текстова класифікація допомагає визначити, чи ми потрапляємо в один зі сценаріїв goal-oriented чат-бота. Допустимо, «питання про курси валют». Вилучення відношень потрібно для визначення заповнювачів шаблону сценарію, а завдання ведення діалогу на загальні теми допоможе нам у ситуації, коли ми не потрапили до жодного сценарію.

П'ятий приклад класичного завдання NLP – саммаризація (реферування) [4]. Реферування тексту – скорочення його обсягу та генерація короткого викладу його змісту, назви або анотації – реферату, що робить швидшим пошук у колекціях документів. Реферат може складатися також для кількох близьких за темою документів (наприклад, кластеру новинних документів). Основним методом автоматичного реферування досі є відбір найбільш значущих пропозицій тексту, що реферується, на основі статистики слів і словосполучень, а також структурних і лінгвістичних особливостей текстів (рисунок 1.4).

---

**Article**

(cnn) to allay possible concerns, boston prosecutors released video friday of the shooting of a police officer last month that resulted in the killing of the gunman. the officer wounded, john moynihan, is white. angelo west, the gunman shot to death by officers, was black. after the shooting, community leaders in the predominantly african-american neighborhood of (...)

---

**Human-written summary**

boston police officer john moynihan is released from the hospital. video shows that the man later shot dead by police in boston opened fire first. moynihan was shot in the face during a traffic stop.

---

**Generated summary (See et al., 2017)**

boston prosecutors released video friday of the shooting of a police officer last month. the gunman shot to death by officers , was black . one said the officers were forced to return fire. he was placed in a medically induced coma at a boston hospital.

---

Рисунок 1.4 – Приклад роботи методу реферування

Це, звісно, не повний список задач NLP, але перераховані вище є найбільш популярними на поширених у застосуванні.

### 1.3 Проблеми використання NLP

Формулювання завдань не дуже складні, однак самі завдання зовсім не є простими, тому що ми працюємо з природною мовою. Явлення полісемії (багатозначні слова мають загальний вихідний зміст) та омонімії (різні за змістом слова вимовляються і пишуться однаково) характерні для будь-якої природної мови. І якщо носій української добре розуміє, що в теплому прийомі мало спільного з бойовим прийомом, автоматичній системі доводиться довго вчитися. Чому «Press space bar to continue» краще перекласти «Для продовження натисніть пробіл», ніж «Бар космічної преси продовжить роботу». Роздивимось детальніше такі явлення:

- полісемія – це наявність в одного й того ж слова кількох пов'язаних між собою значень зупинка (процес або будівля), стіл (організація чи об'єкт);

- омонімія – подібність слів у звуковому відношенні за відмінності значень: ключ, коса, замок;

- анафора – наприклад, нехай, нам дано текст «Двірник дві години мів сніг, він був незадоволений». Займенник «він» може відноситися як до двірника, так і до снігу. По контексту ми легко розуміємо, що він двірник, а не сніг. Але домогтися, щоб комп'ютер це теж легко розумів, непросто. Завдання займенникової анафори і зараз вирішено не дуже добре, активні спроби покращити якість рішень продовжуються;

- еліпсис – це ще одна додаткова складність. Наприклад, «Микола з'їв зелене яблуко, а Катерина – червоне». Ми розуміємо, що Катерина з'їла червоне яблуко. Тим не менш, щоб машина теж зрозуміла зробити непросто.

## 1.4 Використання нейронних мереж для задач NLP

За останні кілька років нейронні мережі знову з'явилися як потужні моделі машинного навчання, показали кращі результати в таких галузях, як розпізнавання образів та обробки мови [23]. Ще зовсім недавно нейромережеві моделі почали застосовуватися також до різних завдань обробки природної мови з дуже добрими результатами. Традиційна модель «bag of words» разом із класифікаторами, які використовують цю модель, такими, як баєсовський метод, були успішно використані для того, щоб отримувати дуже точні прогнози щодо аналізу настроїв. З появою технологій глибокого навчання та їх застосуванням в обробці природної мови було покращено точності цих методів у двох основних напрямках: використання ШНМ з учителем для навчання класифікатора та без вчителя для оптимізації попередньої обробки даних та вибору характеристик [2].

Прикладом найбільш успішного застосування нейронних мереж поки є аналіз зображень, проте нейромережеві технології докорінно змінили роботу з текстовими даними [24]. Якщо раніше кожен елемент тексту (літера, слово чи речення) потрібно було описувати за допомогою безлічі ознак різної природи (морфологічних, синтаксичних, семантичних тощо), то тепер у багатьох завданнях необхідність складних описів пропадає.

Теоретики та практики нейромережевих технологій часто говорять про «навчання уявленню» (representation learning) – у сирому тексті, розбитому тільки на слова та пропозиції, нейронна мережа здатна знайти залежності та закономірності та самостійно скласти ознаковий простір. На жаль, у такому просторі людина нічого не зрозуміє – під час навчання нейронна мережа ставить кожному елементу тексту у відповідність один щільний вектор, що складаються з деяких чисел, що представляють виявлені «глибинні» взаємозв'язки [3].

Акцент при роботі з текстом зміщується від конструювання підмножини ознак та пошуку зовнішніх баз знань до вибору джерел даних та розмітки текстів для подальшого навчання нейронної мережі, для якого потрібно значно більше даних у порівнянні зі стандартними методами. Саме через необхідність використовувати великі обсяги даних і через слабку інтерпретованість і непередбачуваність нейронні мережі не затребувані в реальних додатках промислового масштабу, на відміну від інших алгоритмів навчання, що добре зарекомендували себе, таких як випадковий ліс і машини опорних векторів. Проте нейронні мережі використовують у цілій низці завдань автоматичної обробки текстів (рисунок 1.5).

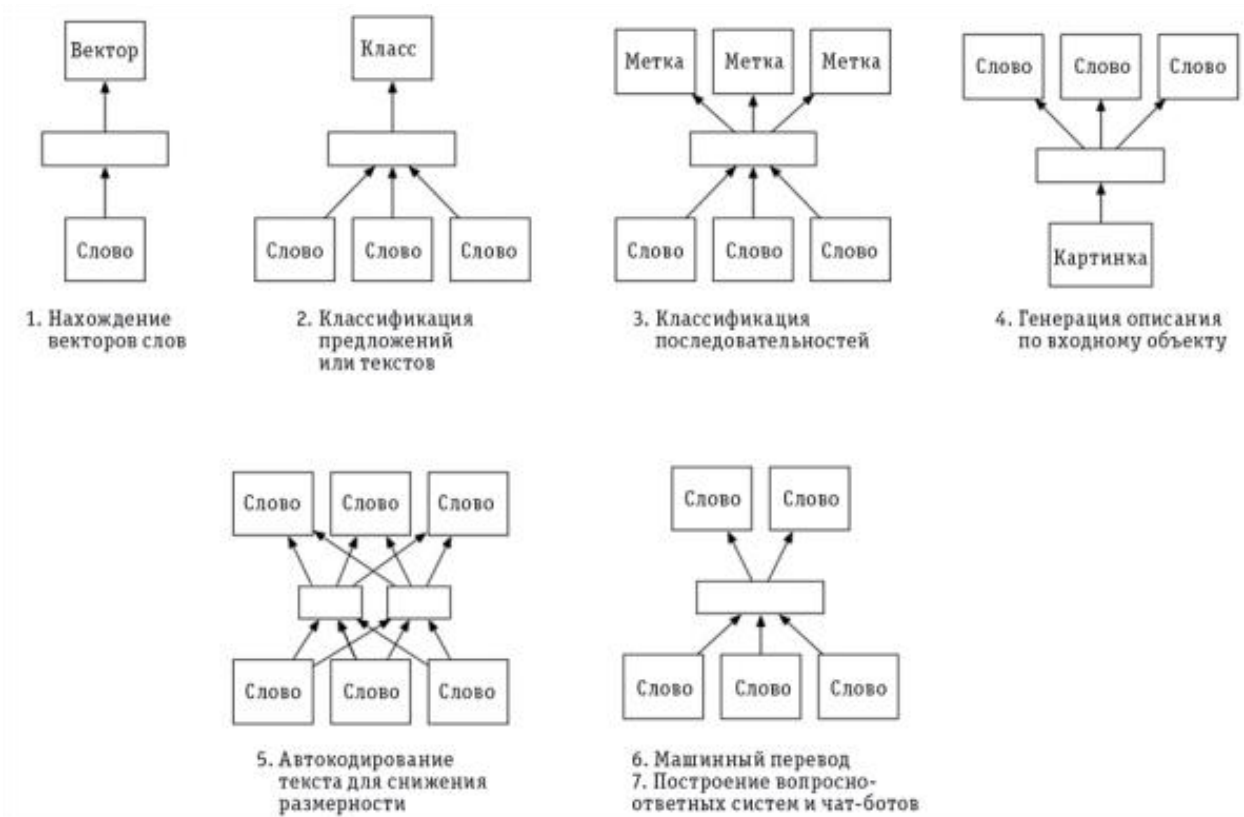


Рисунок 1.5 – Завдання та архітектури нейронних мереж

Архітектури нейронних мереж, що використовуються для обробки текстів природною мовою [5]:

- рекурентні нейронні мережі (рисунок 1.6) – мережі, у яких можливі цикли, тобто вихід одного нейрона підключений до його входу або входу інших нейронів (прості, GRU, LSTM);

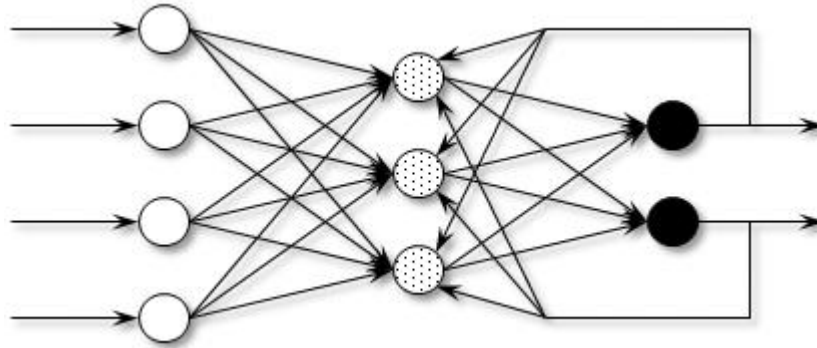


Рисунок 1.6 – ШНМ із зворотними (рекурентними) зв'язками

- одновимірні згорткові нейронні мережі (рисунок 2.7) також застосовуються для аналізу текстів. Однак для цієї мети використовуються лише одновимірні;

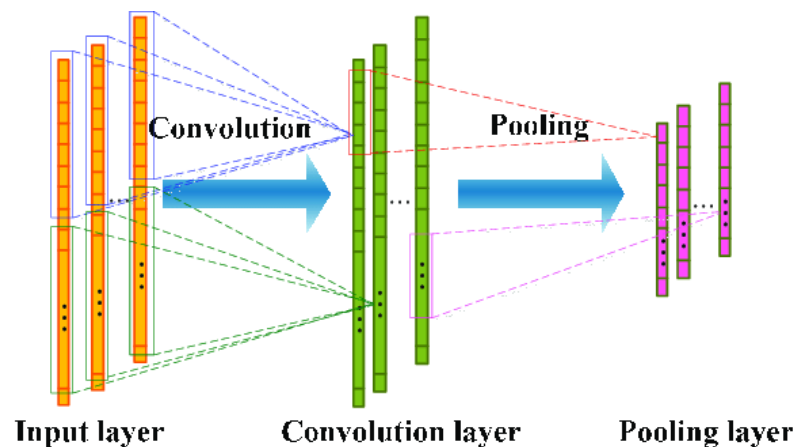


Рисунок 1.7 – Архітектура одновимірної згорткової нейронної мережі

- нейронні мережі з увагою (attention) – механізм уваги нейронних мережах дуже корисний саме під час аналізу текстів, оскільки інформація, яка важлива для розуміння поточного фрагмента тексту могла бути у фрагменті, який був досить далеко.

Саме механізм уваги дозволяє знаходити такі зв'язки. Приклад архітектури та роботи такої мережі наведений нижче (рисунок 1.8).

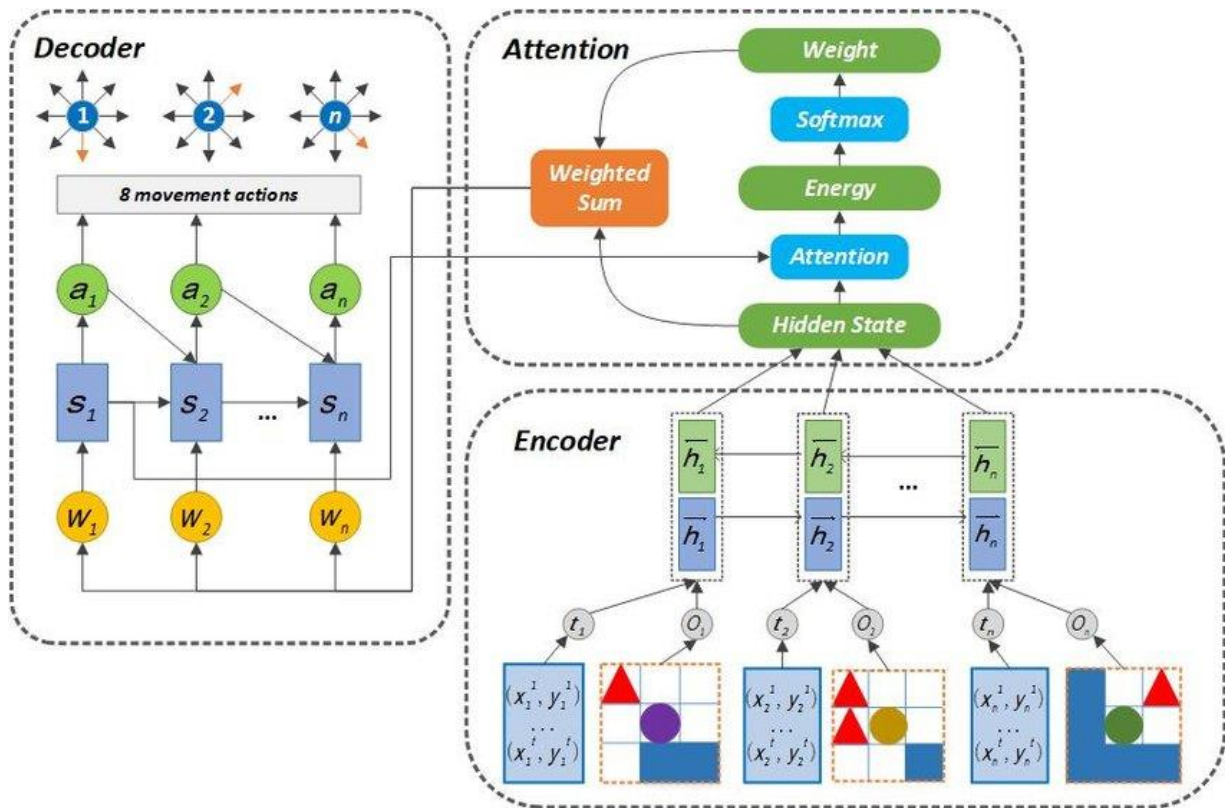


Рисунок 1.8 – Архітектура мережі з увагою

- перенесення навчання (рисунок 1.9) для аналізів тексту (transfer learning) – дозволяє отримувати найкращі результати на задачах аналізу тексту.

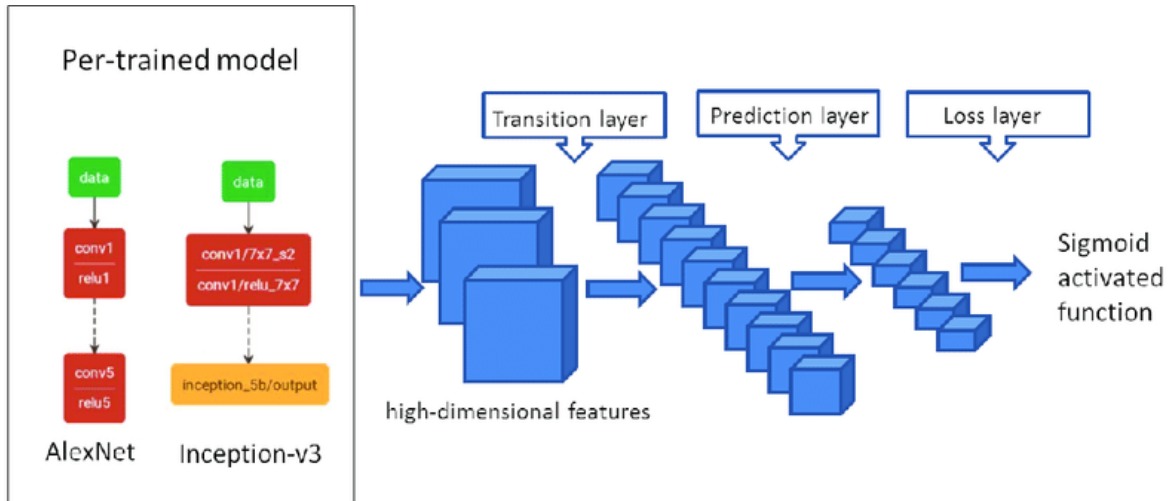


Рисунок 1.9 – Архітектура перенесення навчання нейронних мереж

### 1.5 Використання графів знань для задач NLP

Граф знань (рисунок 1.10) є збором фактів: об'єкти (вузли) з'єднані один з одним типізованими зв'язками. Інформація, що зберігається у графі, може бути різною: знання з багатьох областей (DBpedia та Yago); знання з якоїсь конкретної області (у Bio2RDF та UMLS містяться знання з галузі біології та медицини); знання, що стосуються конкретної галузі чи підприємства. Щоб з великої бази знань витягти граф конкретної області, виконують його обхід починаючи з однієї чи більше початкових об'єктів (концепцій) [6]. Цей спосіб не завжди ефективний, тому що навіть після всього двох-трьох переходів у субграф може увійти більше 50% вихідного вмісту. Щоб обмежити обхід для енциклопедичних джерел, вибираються релевантні зв'язки (ребра) шляхом підрахунку індексу специфічності – приналежності до області, що цікавить. Такі домен-специфічні субграфи стали активно використовуватися в ряді перспективних застосувань.

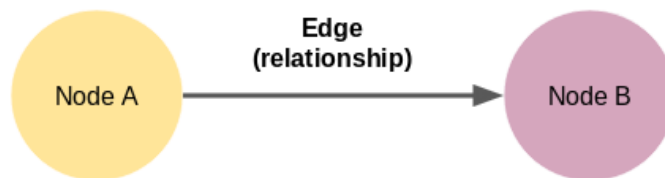


Рисунок 1.10 – Відображення графу знань

Графи знань можна застосовувати для доповнення навчальних вибірок для алгоритмів машинного навчання, що дозволяє покращувати робочі характеристики додатків з обмеженим обсягом навчальних даних – наприклад, систем аналізу тональності (емоційного забарвлення, *sentiment analysis*) голосових висловлювань, розпізнавання іменованих об'єктів, формування рекомендацій, відповідей на питання, і навіть систем комп'ютерного зору. Проте навчальні дані рідко представлені у тій формі, як і граф знань, тому завжди їх об'єднати [7].

Вузли А та В – це дві різні сутності. Ці вузли з'єднані ребром, що представляє відносини між двома вузлами. Тепер це найменший граф знань, який ми можемо побудувати – він також відомий як потрійний (*triple*). Зазвичай, графи знань бувають різних форм і розмірів.

Для представлення даних у вигляді графа знань візьмемо для прикладу таке речення: «Лондон – столиця Англії. Вестмінстер розташований у Лондоні». Після деякої базової обробки, ми отримаємо 2 тріпли, такі як: «Лондон, бути столицею, Англія», «Вестмінстер, визначити місцезнаходження, Лондон». Отже, у цьому прикладі ми маємо три унікальні сутності (Лондон, Англія та Вестмінстер) і два відносини (бути столицею, місцезнаходження). Щоб побудувати граф знань, ми маємо лише два

пов'язані вузли в графі з сутностями та вершинами з відношеннями, і ми отримуємо щось подібне (рис. 1.11):

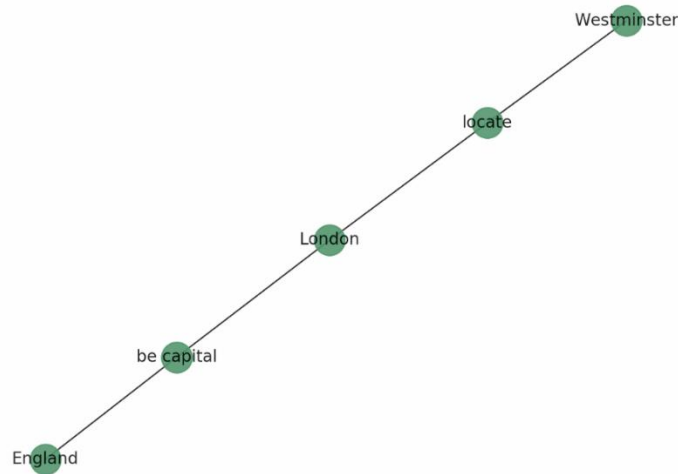


Рисунок 1.11 – Приклад графу знань

Або розглянемо представлення даних на прикладі речення із Тімом Куком: «Тім Кук – генеральний директор компанії Apple». Якщо вузол А = Тім Кук і вузол В = Apple, то цілком ймовірно, що ребро буде «ген. директор». Вузол або сутність також можуть мати кілька відносин. Тім Кук – це не тільки ген. директора Apple, він також працював в IBM. Але як нам включити цю нову інформацію про Тім Кук? Просто додаємо ще один вузол для нової сутності, IBM (рис. 1.12).

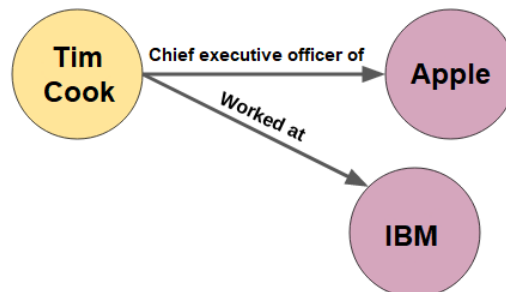


Рисунок 1.12 – Представлення даних у вигляді графу знань

Нові зв'язки можуть виникати не тільки з першого вузла, а й з будь-якого вузла на графіку знань, як показано нижче (рис. 1.13). Штаб-квартира компанії Apple розташована у США.

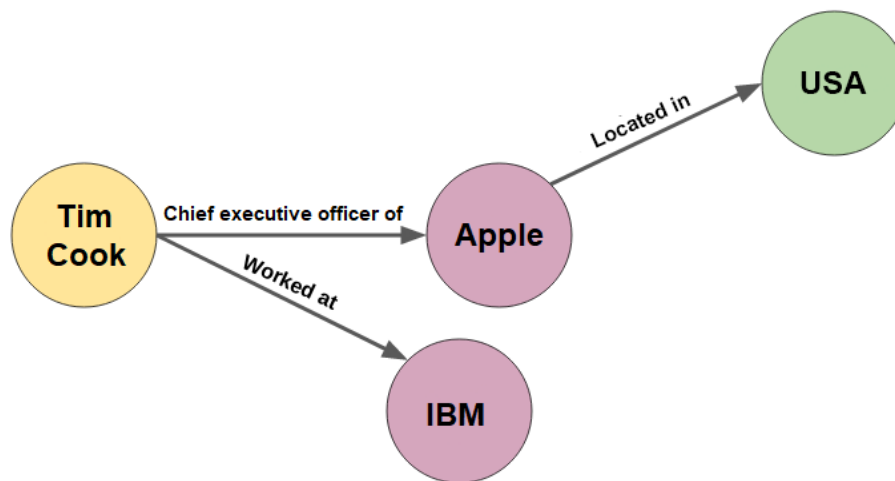


Рисунок 1.13 – Представлення даних у вигляді графу знань

Існують три завдання NLP, які мають безпосереднє відношення до побудови графа знань: виділення сутності, виділення зв'язків та розв'язання сутності.

Вилучення об'єктів – це завдання визначення ключових об'єктів, що представляють інтерес (наприклад, організації, людей, місць тощо) із тексту. Ці об'єкти зазвичай складають вузли в графі знань.

Виділення відношення – це завдання, у якому надані дві цікаві об'єкти та деякий текст ми витягуємо з тексту відносини між ними (наприклад, чисті продажі, інформацію про команду управління тощо). Іноді витяг відношення також використовується для вилучення властивостей даної сутності. Вилучені зв'язки та властивості зазвичай стають зв'язками або властивостями вузлів у графі знань [8].

Розділення сутності – це завдання визначити, чи посилаються кілька згадок у тексті до однієї сутності. Наприклад, в абзаці тексту «Джон Сміт», «він» і «її батько» можуть посилатися на одну сутність.

Граф знань – це свого роду графова модель. Однак така модель фокусується на зберіганні всіх даних як на рівні схеми, так і на рівні окремих сутностей. Така модель на мою думку може бути одним із найкращих варіантів для обробки природномовних текстів, тому що саме графи знань можуть надати такі ключові переваги для роботи зі знаннями:

- описи мають формальну семантику, яка дозволяє як людям, так і комп'ютерам обробляти їх ефективним і однозначним чином;
- описи сутностей доповнюють один одного, утворюючи мережу, де кожна сутність являє собою частину опису пов'язаних з нею сутностей і забезпечує контекст для їх інтерпретації.

Саме ці фактори грають ключову роль в роботі з природними мовами.

## 1.6 Постановка задачі

Метою даної кваліфікаційної роботи є дослідження та застосування на практиці графів знань та нейромережових моделей для обробки природномовних текстів, а саме за допомогою методу розпізнавання іменованих сутностей (NER). У якості архітектури НМ краще всього підійде рекурентна штучна нейронна мережа.

- провести аналіз науково-технічних публікацій та даних Інтернет джерел в області сучасних підходів для обробки природномовних текстів;
- провести порівняльний аналіз основних відомих методів для вирішення задачі;
- здійснити вибір моделі та методів для проведення дослідження;

- провести аналіз основних задач та методів NLP з метою потенційного їх застосування задля побудови графів знань;
- здійснити вибір та формування наборів даних для експериментальних досліджень;
- провести огляд методів видобування сутностей та відносин з тексту та проаналізувати, які методи переважають, виділити їх переваги та недоліки;
- провести обґрунтований вибір методів видобування сутностей, відносин, та визначити, які з цих підходів будуть найкращими для побудови графів знань та нейронних мереж;
- реалізувати та граф знань;
- здійснити дослідження можливостей реалізованого програмного додатку.

## 2 МЕТОДИ ВИДОБУВАННЯ ІНФОРМАЦІЇ

### 2.1 Загальні відомості

Як згадувалося раніше, однією з найпопулярніших і широко застосовуваних завдань є завдання видобування інформації (information extraction).

Видобування інформації – це завдання автоматичного отримання конкретної інформації, пов'язаної з обраною темою, з основної частини або частин тексту (рисунок 2.1).

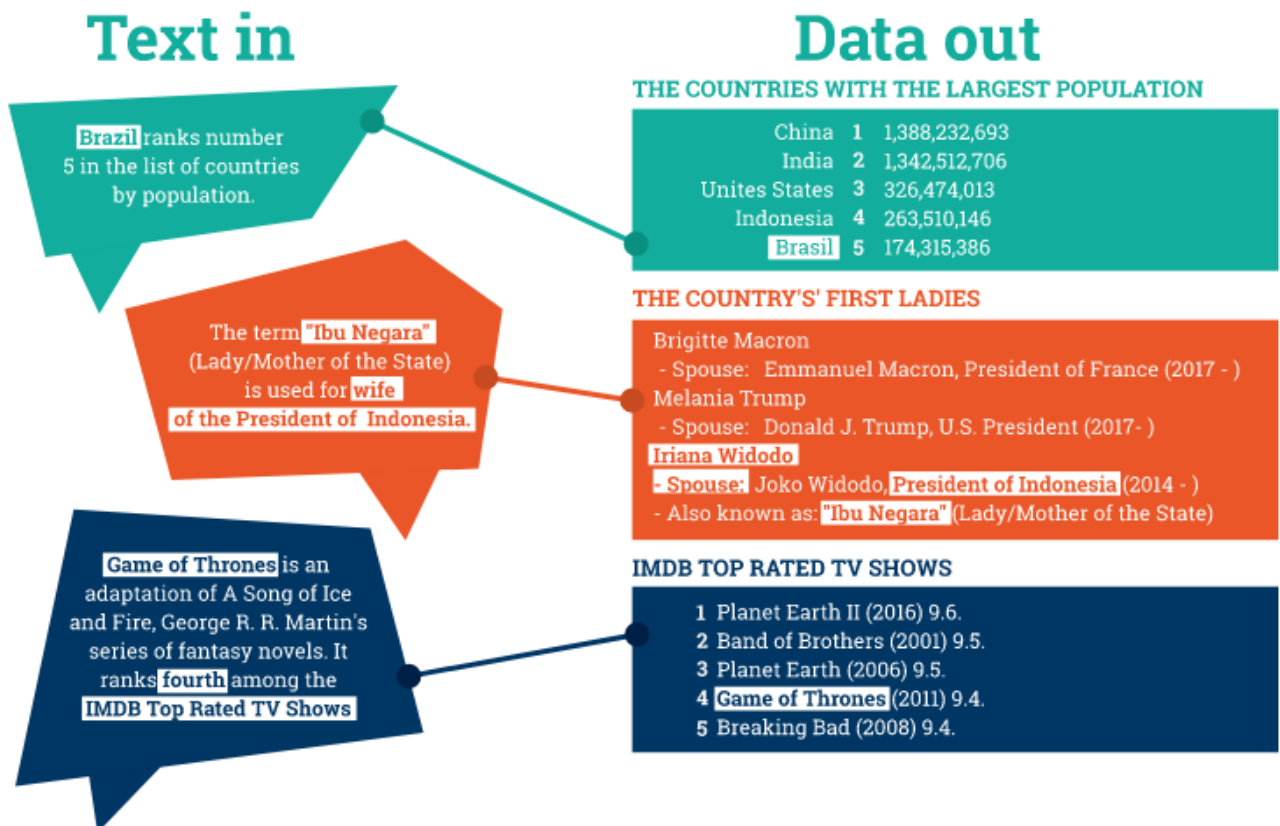


Рисунок 2.1 – Приклад роботи вилучення інформації

Ідея, що лежить в основі вилучення інформації, полягає в тому, щоб дозволити легко ідентифікувати та асимілювати дані, що стосуються конкретної діяльності, без необхідності вручну проходити великі обсяги інформації, щоб знайти точні дані. Цей процес аналогічний ідеям концептуального майнінгу або мережевого аналізу, оскільки всі ці підходи спрямовані на збирання корисної інформації з ширшого набору доступних даних [5].

Інструменти видобування інформації дозволяють отримувати інформацію з текстових документів, баз даних, веб-сайтів або кількох джерел. ВІ може витягувати інформацію з неструктурованого, напівструктурованого або структурованого, машиночитаного тексту. Проте зазвичай ВІ використовують у обробці природної мови (NLP) для вилучення структурованого з неструктурованого тексту. Де структуровані дані – це дані, які мають семантично чітко визначену схему для інформації, що зберігається в них, які інтерпретуються щодо різних категорії та сенсу тексту (контексту).

Як правило, для вилучення структурованої інформації з неструктурованих текстів задіяні такі основні підзадачі:

- попередня обробка тексту – текст готується до обробки за допомогою деяких засобів обчислювальної лінгвістики, таких як лексема, розбиття речень, морфологічний аналіз тощо;
- пошук і класифікація понять – виявлення та класифікація згадок про людей, речі, місця, події та інші заздалегідь визначені типи понять;
- з'єднання понять – завдання виявлення зв'язків між вилученими поняттями;
- уніфікація – подання витягнутих даних у стандартну форму;
- позбавлення від шуму – передбачення усунення повторюваних даних;
- заповнення бази знань – саме тут отримані знання завантажуються у базу даних для подальшого використання.

Вилучення інформації може бути повністю автоматизовано або виконуватися за допомогою людського внеску. Як правило, найкращі рішення для вилучення інформації – це комбінація автоматизованих методів із людьми.

Загальний підхід до отримання інформації вимагає використання програм, здатних сканувати джерела інформації, що вважаються машиночитаними. Це може включати друковані документи, які були відскановані в якісь електронні файли, документи, підготовлені у вигляді електронних таблиць або документів для обробки текстів, або навіть дані, що містяться в полях, що читаються в базі даних. Як правило, встановлюються параметри, які дозволяють програмному забезпеченню отримати доступ до цих джерел даних та швидко сканувати їх, використовуючи конкретні критерії для визначення пріоритетів та отримання певних типів інформації з доступного набору. Цей процес зазвичай відрізняється від простого процесу пошуку тим, що метод вимагає невідповідності конкретним словам або фразам як таким, але натомість використовує NLP, який допомагає не тільки оцінювати фактичні слова, але також контекст і значення, що розуміється цим контекстом.

Видобування інформації залежить від розпізнавання іменованих об'єктів (NER), допоміжного інструменту, який використовується для пошуку цільової інформації для вилучення. NER спочатку розпізнає організації як одну з кількох категорій, таких як місцезнаходження, особи або організації. Після того, як категорія інформації розпізнається, утиліта вилучення інформації витягує пов'язану інформацію з названого об'єкта і створює з неї машиночитаний документ, який алгоритми можуть додатково обробляти для вилучення значення. ВІ знаходить сенс у даних за допомогою інших підзадач, включаючи розділення спільного посилання, виділення зв'язків, аналіз мови

та словарного запасу, а іноді й автоматичне вилучення інформації з аудіо, відео або зображень.

Завдання ВІ є досить складним, тому через його складність сучасні підходи до вилучення інформації зосереджені на конкретних галузях. Наприклад витягування з новинної стрічки інформації про стан бойових дій на сході України.

Історія завдання видобування інформації сягає корінням ще з кінця 1970-х років, коли методи завдання обробки природної мови ще тільки починали розроблятися. Ця галузь розвивалася дуже швидко з кінця 1980-х років, коли оборонна агенція США DARPA, фінансувала конкуруючі дослідницькі групи для проведення дослідження видобування інформації, а також комерційна система JASPER, яка була створена компанією Carnegie Group Inc для агенції Рейтер для надання фінансовим трейдерам потрібних їм новин у той же час[7].

Свій подальший розвиток ВІ отримала після низки конференцій Message Understanding Conference (MUC) з 1987 року. MUC – це конференції-змагання, які були призначені для вирішення наступних задач:

- MUC-1 у 1987 році та MUC-2 у 1989 році – вирішення завдання щодо військово-морських операцій;
- MUC-3 у 1991 році та MUC-4 у 1992 році – вирішення завдання стосовно тероризму у латиноамериканських країнах;
- MUC-5 у 1993 році – венчурні операції у галузі мікроелектроніки;
- MUC-6 у 1995 році – вирішення завдання щодо новинних статей про зміни в управлінні компаніями;
- MUC-7 у 1998 році – інформація щодо звіти про запуски супутників.

На теперішній час значення ВІ зростає із збільшенням обсягу інформації у всесвітньої мережі, яка доступна у неструктурованій формі, у яких відсутні семантичні метадані. Тому, можемо зробити висновок, що ВІ є

значною частиною задачі обробки природномовних текстів. Також, можна зазначити, що ВІ використовується й в завданні інформаційного пошуку для класифікації та індексування великих обсягів інформації або документів. Завдання вилучення інформації передбачає існування набору текстових документів, в якому кожен документ створений за деяким єдиним шаблоном, але із різною інформацією у кожному про сутності та події. Система ВІ вимагає розуміння про сутності для того, щоб вилучити ті дані, які відповідають слотам у заданому шаблоні.

## 2.2 Основні задачі та підходи видобування інформації

Загальною метою видобування інформації є створення коректного машиночитаного тексту для змоги обробляти речення. Тому роздивимось спочатку які існують задачі, що відносяться до ВІ:

- видобування термінологій – пошук термінів, які відповідають заданому корпусу текстів;
- видобування звуків – на основі деякого шаблону шукаються звуки, тобто за відповідною характеристикою в звуковому сигналі;
- видобування відносин – пошук відносин між сутностями або суб'єктами;
- розпізнавання або видобування іменованих сутностей – розпізнавання сутностей, що стосуються людини або організації, використовуючи заздалегідь задані знання про ці сутності (домен або інформація), які були витягнуті з інших речень тексту;
- кореферентний пошук – пошук різних зв'язків між текстовими сутностями, тобто якщо в реченні «Біл Гейст один із найуспішніших людей у світі. Він заснував компанію Microsoft», потрібно виявити, що «він» це є та ж особа, що й «Біл Гейтс».

Перейдемо до списку існуючих підходів у ВІ. Наразі існують наступні стандартні підходи:

- рекурентна нейронна мережа (RNN);
- марковська модель (умовна та прихована);
- регулярні вирази;
- наївний баєсів класифікатор;
- моделі максимальної ентропії;
- умовні випадкові поля.

Також для ВІ виділяють ще й гібридні підходи, що об'єднують у собі класичні зазначені вище підходи.

Звісно список усіх задач та підходів ВІ є набагато більше, ніж зазначено, але саме ці перелічені є найбільш популярними та актуальними у використанні на сьогоднішній день.

### 2.3 Named entity recognition

Для подальшої розробки програмного застосунку кваліфікаційної роботи було обрано саме метод розпізнавання іменованих сутностей (NER).

Визначимо для початку, що саме є іменовані сутності. У будь-яких текстових даних іменовані сутності є об'єктами, які існують у реальному світі. У першій, класичній постановці, яка була сформульована на конференції MUC-6 у 1995 році, це персони, локації та організації [12],[13]. З того часу з'явилося кілька доступних корпусів, у кожному з яких свій набір іменованих сутностей. Зазвичай до персон, локацій та організацій, які можуть бути представлені в будь-яких даних за допомогою їх власного імені, додаються нові типи сутностей. Найпоширеніші їх – числові (дати, фінансові суми), і навіть сутності Misc (від miscellaneous – інші іменовані сутності) [13]. Більш формально можна сказати, що іменована сутність позначає власну

назву будь-якого об'єкта. Приклад роботи NER можна побачити на рисунку 2.2.

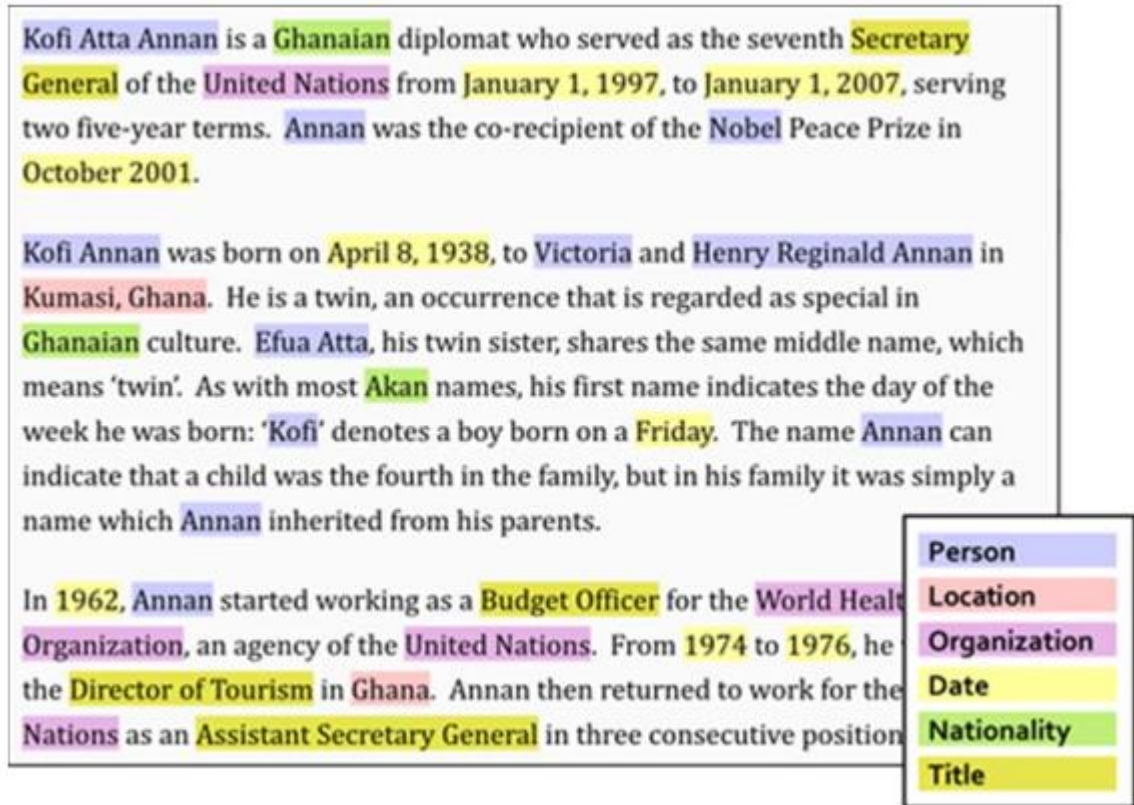


Рисунок 2.2 – Приклад роботи NER

Розпізнавання іменованої сутності – це процес, коли іменованний об'єкт ідентифікується та пов'язується з його класом. Оскільки ми знаємо, що будь-які вихідні текстові дані складаються з різних типів слів, наприклад, деякі з них є стоп-словами, слова частини мови, також можуть існувати різні ласкаві слова, які можуть бути представлені в текстовому файлі, який можна відокремити як іменовані об'єкти [12]. Ці слова можуть представляти зв'язок між двома реченнями або двома словами.

Тому іноді стає дуже важливим визначити та класифікувати їх, щоб модель, яка буде працювати з даними, могла легко розуміти текстові дані та точно отримувати з них результати. Наприклад, із речення «Корпорація Microsoft придбала Activision Blizzard за 68,7 мільярда доларів у 2022 році». А система розпізнавання названих сутностей дасть результати як «microsoft (компанія) придбала activision blizzard (компанія) за 68,7 мільярда доларів (ціна) у 2022 році (час)». Тут, у реченні, ми можемо побачити процес розпізнавання моделі NER, класифікуючи слова на назвах компаній, ціну та час.

Як вже зазначалось раніше, метою систем розпізнавання іменованих об'єктів є ідентифікація імен людей, місць розташування, організацій та інших об'єктів, які представляють інтерес у текстових документах.

### 2.3.1 Задачі вирішення NER

Розберемося навіщо потрібно вирішувати задачу NER. У класичній постановці завдання NER може знадобитися на вирішення, як згадувалося раніше, структуризації неструктурованих даних. Наприклад, маємо текстовий документ, з якого необхідно занести дані до бази даних або таблиці. Іменовані сутності можуть відповідати як рядкам бази даних, так і як зміст осередків. Щоб коректно заповнити цю базу або таблицю, необхідно вміти визначати у тексті необхідні нам дані. Як правило, удосконаленням такої системи було б, наприклад, вміння ідентифікувати сутність у тексті (завдання Named Entity Linking). Тобто, необхідно вміти розпізнавати, що такі слова, як, наприклад, «NATO» та «North Atlantic Treaty Organization» відносяться до однієї і тієї ж сутності.

Також можна зазначити те, що коли ми знаємо, де в тексті знаходиться, іменовані сутності, ми можемо вже надалі вирішувати інші завдання з

фрагментами тексту – виділяти абзаци, в яких зустрічаються сутності певного типу і надалі з ними працювати, виділяти в тексті лише потрібні або важливі частини, наприклад, для зручності роботи аналітикам.

Виділені сутності можуть складатися з колокацій, і необхідно вміти правильно їх виділяти для вирішення багатьох завдань, наприклад, для коректного перекладу або для синтаксичного парсингу [14].

Як згадувалося в попередньому розділі NER є фундаментом для деяких завдань обробки природних мов, наприклад, для побудови запитально-відповідних систем або визначення займенникової анафори. Займенникова анафора допомагає нам зрозуміти як займенник відноситься до певної сутності [15]. Наприклад, у реченні «Білл Гейтс успішна людина. Він заснував корпорацію Microsoft». Виділяючи сутність «Білл Гейтс» як персону, машині необхідно зрозуміти, що «Білл Гейтс» і «він» – та сама сутність.

Що стосується питання-відповіді системи, то тут, в основному лежить завдання для пошуковика [14]. Наприклад, якщо ввести запит «Хто озвучував лева Алекса у мультфільмі “Мадагаскар”», швидше за все отримаємо правильну відповідь на своє запитання – виділяються сутності (мультфільм, лев тощо) та шукається відповідь у базі даних.

Насправді, не варто вважати, що ми маємо видобувати тільки ті сутності, що відносяться до імен, локацій або організацій. Залежно від умов та постановки задач ми можемо виділяти будь-які необхідні нам сутності та фрагменти тексту, що робить NER одною з самих розв’язуваних задач.

### 2.3.2 Проблеми використання NER

Така проблема, як неоднозначність у мові існує в цілому в задачі обробки природних мов. Наприклад, для NER великою проблемою можуть

стати омоніми, коли різні сутності можуть називатися одним і тим самим словом [16]. Візьмемо на розгляд слово «коса». Що конкретно може бути на увазі – коса з волосся чи інструмент, щоб косити траву? Або візьмемо слово «Джорджія» – така сутність може мати значно більше значень: штат, ім'я чи фамілія людини, назва фільму, закладу, газети, прізвисько тощо. У цьому разі необхідно враховувати те, про що був попередній зміст тексту до цільового слова (локальний контекст) і загальний сенс тексту (глобальний).

Ще однією складністю можна назвати технічну частину. Якщо людина все може врахувати щодо тексту, то машину необхідно навчити, що зробити дуже непросто. При видобуванні сутностей необхідно також розуміти, що нам включати в колокацію, коли потрібно та коли не потрібно виділити сутність тощо. Припустимо є текст, в якому необхідно виділити назву закладу «Ласкаво просимо до Школи Чаклунства та Чарівництва Гогвортс». У такому складному прикладі не зовсім зрозуміло чи треба виділяти повністю «Школа Чаклунства та Чарівництва Гогвортс» або коректніше буде виділити лише «Гогвортс». Для цього прикладу обидва варіанти будуть вірними, однак такі випадки необхідно зафіксувати та враховувати надалі, щоб у всіх місцях решти тексту такі приклади були розмічені однаково, та надалі не було суперечностей та помилок. Така робота може бути досить трудомісткою та витратною [15].

Саме такі випадки є основними проблемами NER і це показує нам чому цей метод ще не використовується повсюдно.

### 2.3.3 Оцінка Named Entity Recognition

Основною оцінкою для NER, тобто метрикою, є f-міра. Наприклад, маючи як результат роботи системи готову тестову розмітку тексту, і заздалегідь правильно розмічений текст (еталон), можемо обчислити оцінки

повноти та влучності. Влучність та повнота використовуються для визначення точності системи, в якій прості обчислення точності мало що говорять або просто дають хибні результати [17]. Щоб зрозуміти ці терміни, краще розглянути таблицю 2.1 (рисунок 2.3), відому як матриця помилок. Прикладом точного, але неповного класифікатора можна назвати такий, що виділяє з тексту лише одну коректну сутність, а прикладом повного, але неточного класифікатора можна назвати такий, що виділяє цю сутність по всьому тексту, із цим виділяючи велику кількість непотрібних даних.

Таблиця 2.1 – Матриця помилок

	Predicted: Abnormal	Predicted: Normal
Actual: Abnormal	True positive	False negative
Actual: Normal	False positive	False negative

Розглянемо детальніше про основні метрики оцінки NER:

- влучення (precision) – співвідношення true positive і загальних прогнозованих positive, враховує лише кількість правильно призначених міток, тобто true positive (2.1):

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} = \frac{\text{True positive}}{\text{Actual results}} \quad (2.1)$$

- повнота (recall) – співвідношення true positive і загальних actual positives, окрім true positive враховує також positive-теги, які були пропущені, а тому помилково замінені негативними (тобто false negative) (2.2):

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} = \frac{\text{True positive}}{\text{Predict results}} \quad (2.2)$$

- точність (accuracy) – співвідношення коректних прогнозів, зроблених до розміру тестових даних, вказує на частку true результатів (як true positive, так і true negative) у розмітці, яка також включає хибнопозитивні (false positive) та хибнонегативні (false negative) результати (2.3):

$$\text{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{Total}}, \quad (2.3)$$

де Total – сума true positive, true negative, false positive та false negative.

- f-міра – середнє гармонійне значення точності та повноти (2.4):

$$F - \text{score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.4)$$

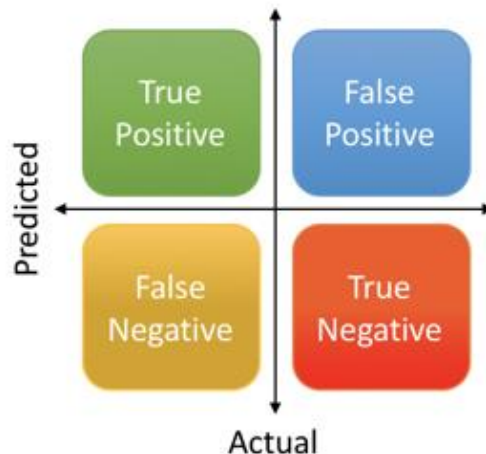


Рисунок 2.3 – Схема матриці помилок

Для того, щоб була можливість оцінити якість роботи системи, нам потрібен корпус. Корпус – набір текстів. Як відомо, створювати розмітку для корпусів – дуже дорого (якісна розмітка одного тексту потребує роботи декількох людей, щоб уникнути помилок). Тому на теперішній час у загальному доступі не дуже багато корпусів із коректною заздалегідь означеною розміткою. Найчастіше розмітки створюються на конференціях по вирішенню задачі NER. Серед таких конференцій, на яких були створені корпуси із іменованими сутностями, можна назвати CoNLL, TAC та MUC, який вже згадувався раніше [18].

Серед перелічених корпусів для оцінки якості найчастіше виділяють саме CoNLL 2003 корпус [11]. Вміст цього корпусу для англійської мови описаний у таблиці 2.2.

Таблиця 2.2 – Вміст CoNLL 2003

	Articles	Sentences	Tokens	LOC	MISC	ORG	PER
Training set	946	14 987	203 621	7140	3438	6321	6600
Develop. set	216	3466	51 362	1837	922	1341	1842
Test set	231	3684	46 435	1668	702	1661	1617

#### 2.3.4 Підходи по вирішення NER

Існує три основні підходи до NER: заснований на правилах (rule-based), на основі машинного навчання та на основі нейронних мереж (рисунок 2.4) [15]. Розглянемо ці підходи детальніше.

Перша та найдавніша категорія – це підходи, засновані на правилах або знаннях. Більшість досліджень у цій категорії засновані на правилах, створених вручну [13]. Перевага таких методів полягає у тому, що вони не вимагають заздалегідь анотованих навчальних даних, оскільки вони спираються на лексичні ресурси. Ще одна перевага полягає в тому, що точність ручних методів може бути високою завдяки знанням у предметній галузі. Недоліком можна назвати те, що це також робить їх залежними від домену, що ресурси лексикону можуть бути недоступні, і що створення та підтримка таких ресурсів для багатьох мов є дорогою [14].

Друга категорія систем NER – це підходи, засновані на машинному навчанні (ML). Ці моделі використовуються для заміни створених людиною правил, необхідних першої категорії. Методи цієї категорії можна розділити на три типи: контрольовані (supervised), напівконтрольовані (semi-supervised) та неконтрольовані (unsupervised) [18]. У методах з учителем (контрольовані) і напівконтрольовані модель машинного навчання навчається на вхідних прикладах разом з їх цільовими вихідними даними. У цій категорії поширені машини опорних векторів (Support Vector Machines, SVM), приховані марковські моделі (Hidden Markov Models, HMM), умовні випадкові поля (Conditional Random Fields, CRF) та дерева рішень [12]. У методі, заснованому на машинному навчанні, NER сприймається як проблема маркування послідовності. У порівнянні з проблемою класифікації поточна мітка прогнозування у проблемі маркування послідовності не тільки пов'язана з поточною вхідною функцією, але також пов'язана з попередньою міткою прогнозування, тобто послідовність міток прогнозування має сильний взаємозв'язок взаємозалежності. У традиційному машинному навчанні умовне випадкове поле (CRF) є основною моделлю NER. Його цільова функція не тільки враховує функцію вхідного стану, але також включає функцію передачі міток. Точність NER іноді обмежується використанням

класифікатором. Наприклад, коли використовуються HMM і SVM, залежності між словами не враховуються. Метод навчання без вчителя (unsupervised) є більш автоматизованим, хоча йому потрібен мінімальний набір навчальних даних (початкові значення) [9]. Хоча ці методи не вимагають стільки зусиль, як перша категорія, вихідні дані все ж таки потрібні для навчання.

Остання й найновіша категорія – це нейромережеві підходи із визначенням характеристик. Вони покладаються на машинне навчання, але вони відрізняються від підходів, заснованих на правилах і машинному навчанні, автоматично виводячи функції за допомогою глибокого навчання [16]. Останні дослідження показують, що вони, таким чином, перевершують попередні методи. На відміну від вищезазначених підходів, вони не вимагають початкових значень, онтологій або словників, специфічних для предметної області, і, таким чином, більш незалежні від предметної області. Вони також виграють від точності передбачуваних ознак. З іншого боку, для побудови надійних моделей потрібні великі набори даних.

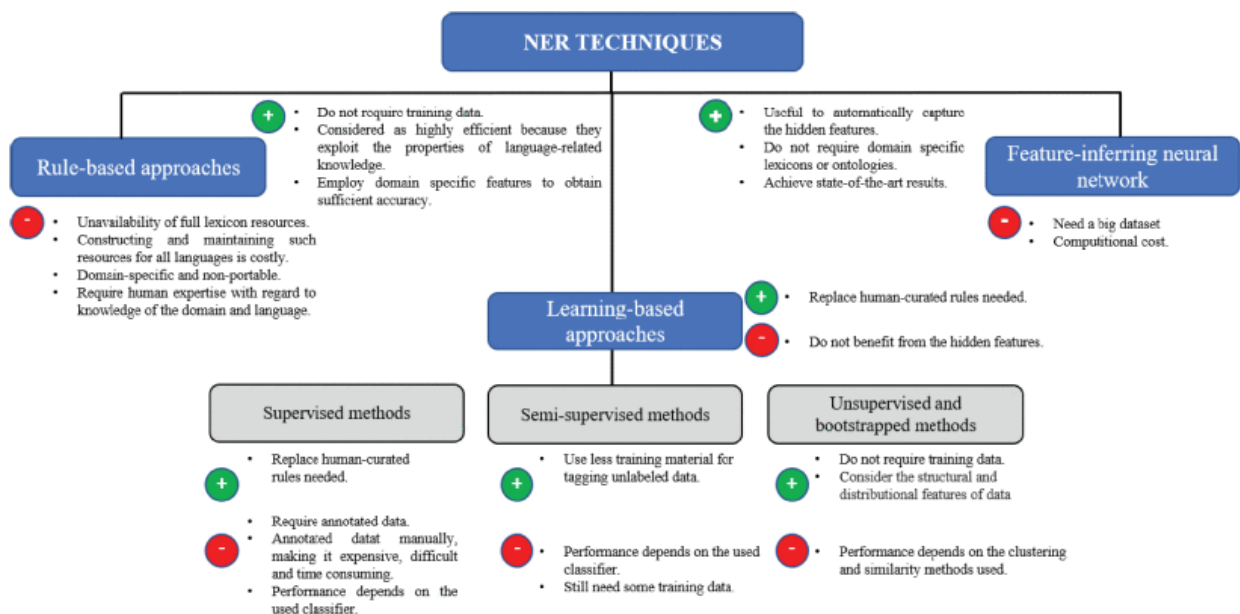


Рисунок 2.4 – Підходи NER

### 2.3.5 Вирішення задачі NER

Зазвичай, задача видобування іменованих сутностей зводиться до завдання класифікації токенів до того чи іншого класу. Один із найвідоміших засобів для вирішення цієї задачі є BIOES-схема[21]. Сенс такої схеми полягає у тому, що ми додаємо префікс, який позначає позицію розміщення токена у реченні, до мітки сутності:

- beginning – позначає перший токен у неперервному фрагменті тексту сутності, який може складатися більше ніж одного токена (слова);
- inside – токен, що знаходиться всередині сутності;
- out або просто o – токен, який не відноситься ні до однієї з сутностей;
- ending – токен, що знаходиться в кінці сутності, тобто останній, який також може складатися більше ніж одне слово;
- single – токен у сутності, який складається із одного слова.

Наприклад, маємо речення “Jane Villanueva of United Airlines Holding discussed the Chicago route.” (рисунок 2.5). У цьому прикладі ми бачимо, що є персона (PER) – Jane Villanueva, є назва організації (ORG) – United Airlines Holding та назва локації (LOC) – Chicago, а також решта токенів, що не відносяться до сутностей (O). Як бачимо, для кожної із сутностей, яка складається більше ніж з одного слова, є початок “B” та кінець “E”, у випадку с організацією маємо токен, що знаходиться всередині “I”. Інші токени, такі як артиклі, розділові знаки тощо, позначаються “O”.

Коли ми говоримо про природну мову, основна сутність із якою ми працюємо – слово, будь-яке слово. Але у тексті можемо зустріти не тільки слова, а й цифри, символи тощо. Тому формально використовується термін – токен. Зазвичай токен може бути розділений з обох боків пробілами або розділовими знаками.

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Рисунок 2.5 – Приклад BIOES-схеми

Таким чином ми можемо встановити межі всіх анотацій сутностей, тобто ми знаємо яка сутність починається та закінчується якимось токеном та чи можемо закінчити анотацію цієї сутності, або потрібно розширювати на наступні токени.

Але у цього методу також є великий недолік – такий метод не дає нам змоги працювати із вкладеними сутностями, або сутностями, що перетинаються. Наприклад, маємо речення або сутність “Харківський національний університет ім. Каразіна”. Ми бачимо, що в цілому це сутність закладу, але “Каразін” – персона, яка може бути вказана окремо. Взагалі такі випадки потрібно якось задавати у розмітці, але для даної схеми неможливо передавати декілька фактів для одної сутності одночасно – один токен має мати тільки одну мітку. Тобто “Каразін” може бути лише тільки розміткою закладу, або розміткою персони.

Слід також зазначити, що окрім стандартного методу зведення задачі до класифікації, існує стандартний формат даних CoNLL-U [22]. У такому форматі дуже зручно зберігати розмітку для NER у якості таблиці. Така

таблиця містить дані, де кожен рядок – токен, а кожен стовпець – конкретний тип ознак токenu. На рисунку 2.6 наведений приклад даних у форматі CoNLL-U. У прикладі наведені само слово, його початкова форма, частина мови (POS-тег), морфологічні характеристики, синтаксична залежність (відношення між лексемами) та розширені залежності токенів.

```
# sent_id = 1
# text = They buy and sell books.
1  They    they    PRON    PRP    Case=Nom|Number=Plur          2  nsubj  2:nsubj|4:nsubj  -
2  buy     buy     VERB    VBP    Number=Plur|Person=3|Tense=Pres 0  root    0:root          -
3  and     and     CONJ    CC      -                               4  cc      4:cc            -
4  sell    sell    VERB    VBP    Number=Plur|Person=3|Tense=Pres 2  conj    0:root|2:conj   -
5  books   book    NOUN    NNS    Number=Plur                    2  obj     2:obj|4:obj     SpaceAfter=No
6  .       .       PUNCT   .      -                               2  punct   2:punct         -
```

Рисунок 2.6 – Приклад відображення даних у CoNLL-U

Тепер розглянемо стадії розробки системи NER, які включають до себе попередню обробку даних, таких як сегментація, токенізація, виділення функцій, застосування моделей ML до даних для тегування, а потім подальшу обробку для усунення деяких невідповідностей тегування (рисунок 2.7). Роздивимося детальніше про стадії.

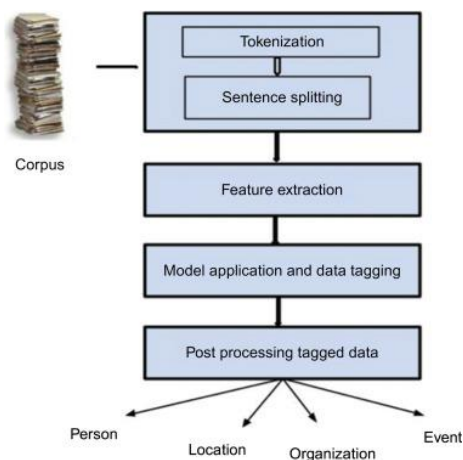


Рисунок 2.7 – Стадії розробки NER

Найбільш часто використовуваними методами попередньої обробки природних мов є токенізація та тегування частини мови (POS). Інші поширені методи включають: видалення стоп-слів, нормалізація, розбиття речень, лемматизацію (приведення слова до початкової форми), розбір залежностей, а також структурний розбір. Хоча попередня обробка використовується за замовчуванням, деякі дослідження не описують детально, як вона виконується. Можливою причиною є те, що більшість досліджень, особливо щодо NER, використовують стандартні набори даних, які вже були попередньо оброблені. Наприклад, CoNLL2003, найпопулярніший набір даних для NER, який згадувався раніше, уже із застосованою токенізацією, POS-тегами (part of speech) і фрагментацією до необроблених даних.

Як вже було зазначено вище, першими кроками у розробці є поділ тексту на речення (сегментація) та поділ речень на токени (токенізація). Далі вже потрібно визначити ознаки кожного токена – визначення контекстно-незалежних ознак токена.

Контекстно-незалежні ознаки – набір ознак, які не залежать від сусідніх від токена слів. Серед таких ознак зазвичай виділяють:

- word embeddings або вкладання слів;
- символічні ознаки;
- додаткові спеціальні ознаки для конкретної мови або задачі.

Разом ці ознаки формують вектор ознак токена.

Вкладання слів – метод, у якому слова або словосполучення із заданого словника відображаються у вигляді вектору дійсних чисел. Таким чином ми отримаємо стисле уявлення про контекст слова. Детальніше про ембедінги роздивимося далі.

Символьні ознаки нам потрібні для того, щоб для кожного токена отримати вектор ознак, що залежить тільки від символів, з яких складається токен.

Серед додаткових ознак виділяють позицію токена у тексті. Це потрібно для того, щоб визначати, що конкретна сутність може зустрічатись у початку або кінці тексту найчастіше.

Також найчастіше виділяють частини мови або POS-теги. Такі ознаки грають дуже важливу роль у вирішенні задачі синтаксичного парсингу. Наприклад, якщо використовується мова із складною морфологією – українська або російська, нам також потрібно визначити рід або відмінок слова. Морфологічний аналіз також потрібен для приведення слів до початкової форми (лематизація) для того, щоб скоротити розмірність простору ознак.

Що стосовно контекстно-залежних ознак токена – це такі ознаки, або набір ознак, які містять інформацію не тільки про сам токен, а ще й про сусідні токени. Раніше це реалізовувалось таким чином – ознаки визначались шляхом вибору кількох сусідніх токенів (частіше три або чотири) по обидві сторони від токена. Але такий підхід не виявився надійним, бо будь-яка важлива інформація може знаходитись на відстані більшої, ніж три або чотири, що приводило до неточного аналізу.

На сьогоднішній день усі контекстно-залежні ознаки на рівні речень визначаються за допомогою двосторонніх рекурентних нейромереж – GRU (Gated Recurrent Unit) та LSTM (Long short-term memory).

На вхід рекурентної мережі (RNN) (рисунок 2.8) подаються контекстно-незалежні ознаки токена, з яких необхідно отримати контекстно-залежні. Вихід RNN в момент часу  $i$  є контекстно-залежною ознакою  $i$ -того токена, в якому міститься інформація про попередні та наступні токени.

Після отримання контекстно-залежних ознак для всіх токенів з контекстно-незалежних ознак, потрібно для кожного токена отримати правильну мітку.

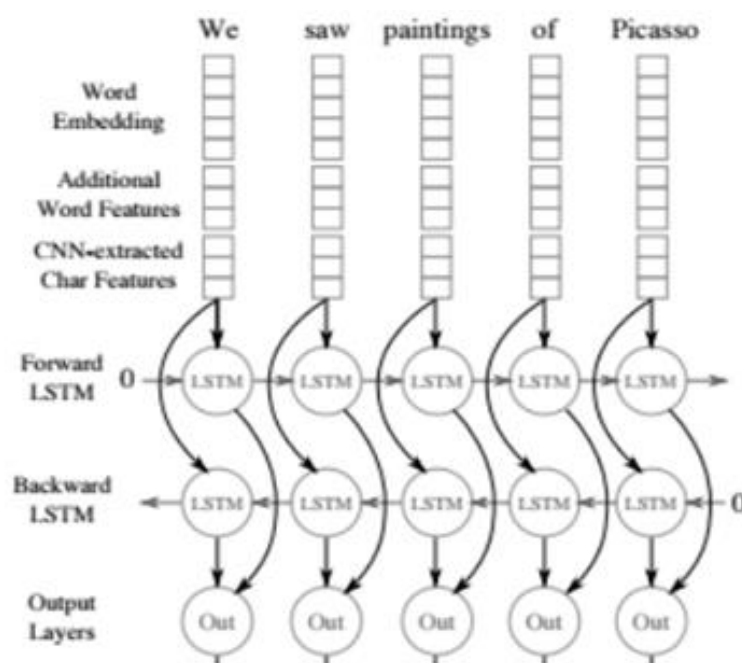


Рисунок 2.8 – Приклад роботи LSTM

## 2.4 Word Embeddings

Розуміючи те, що комп'ютер не розуміє слів, було запропоновано ідею закодувати слова у цифри по порядку прямування у словнику.

Слід зазначити. Що до появи ембедінгів (вкладання слів або embeddings), головною ознакою токена була словоформа, тобто індекс слова у словнику. Сенс у тому, що для кожного токена призначається у відповідність булевий вектор розмірності словнику. У такому словнику замість індексу слова ставлять 1, а на інших місцях ставлять 0. Але такий метод має недоліки – у словнику усі слова мають бути розташовані за абеткою. Тому, якщо додається нове слово, необхідно перенумерувати частину слів.

Роздивимося такий приклад – маємо слова “кінь”, “кобила”, “жереб’я”, ці слова за написанням майже не мають спільного між собою та розташовані у різних частинах словнику, хоча за змістом ми розуміємо, що це позначення

самця, самки та дитинча одного виду тварин. Тобто з цього виходить, що ми можемо виділити два види близькості слів – лексичний та семантичний. Також, можна привести у приклад слова, які за змістом не мають нічого спільного (семантично далекі), а за написанням дуже близькі (лексично) – “золото” та “зола”. Це доказує те, що літерне написання слова ніяк не зв’язано з його змістом – таку гіпотезу у кінці дев’ятнадцятого століття висунув лінгвіст Фердінанд де Соссюр.

Для вирішення цієї проблеми був запропонований метод ембедінгу, основна ідея якого полягає у тому, щоб мати змогу представити семантичну близькість слів, тобто зіставити слову вектор, який відображає його значення у “просторі сенсів”. Останні десятиліття стався значний розвиток цієї концепції для нейронних моделей (останні розробки включають контекстуалізовані вкладки слів у моделей, як BERT, ELMo та GPT2).

На сьогодні вектори слів є основою багатьох систем обробки природної мови, які підкорили сучасний світ – Amazon Alexa, Google translate тощо.

Слова-вектори (word vectors) – це чисельні уявлення слів, які зберігають семантичний зв’язок між ними. Наприклад, маємо вектори “cat”, “dog” та “pencil” (рисунку 2.9). Для вектора “cat” одним із найближчих за сенсом буде слово “dog”. Проте векторне уявлення слова “pencil” досить сильно відрізнятиметься від вектора “cat”. Ця схожість обумовлена частотою народження двох слів (тобто [cat, dog] або [cat, pencil]) в одному контексті.

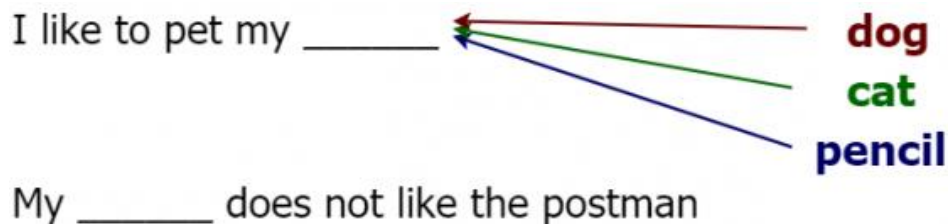


Рисунок 2.9 – Приклад речення із словами-векторами

Тож, спочатку визначимо, що таке вкладання. Вкладання або ембедінг – це зіставлення довільної сутності (наприклад, вузла у графі або шматочки картинки) деякому вектору. Якщо казати спрощено про вкладання слів – це стиснене уявлення про контекст слова. Алгоритми вбудовування слів дають щільні (dense) вектори, отже слова зі схожим контекстом розподілу з'являються у тому ж області простору вбудовування.

Тепер роздивимось як саме можна отримати вектор зі слова. Для цього існують такі підходи – one-hot encoding та bag of words.

One-hot encoding (ONE) – метод, суть якого у том, що потрібно взяти вектор довжини нашого словника і поставити лише одну одиницю у позиції, що відповідає номеру слова у словнику (рисунок 2.10). Однак такий метод не має властивості семантичної близькості.

motel [○ ○ ○ ○ ○ ○ ○ ○ ○ ○ 1 ○ ○ ○ ○ ] AND  
 hotel [○ ○ ○ ○ ○ ○ ○ 1 ○ ○ ○ ○ ○ ○ ○ ] = ○

Рисунок 2.10 – Приклад One-hot encoding

Bag of words (рисунок 2.11). Змістове значення одного окремого слова нам може бути не так важливо, так як наша мова складається з набору таких слів. Ці набори ми називаємо текстами. Отже, якщо нам потрібно якоесь перетворити текст, потрібно для кожного слова в тексті взяти його ONE-вектор та зібрати їх усіх разом. Що ми отримаємо на виході? Отримаємо один вектор, у якому підраховані кількості слів у тексті. Такий підхід називається «мішок слів» (bag of words, BoW), тому що ми отримаємо лише один вектор без інформації про розташування слів у тексті.

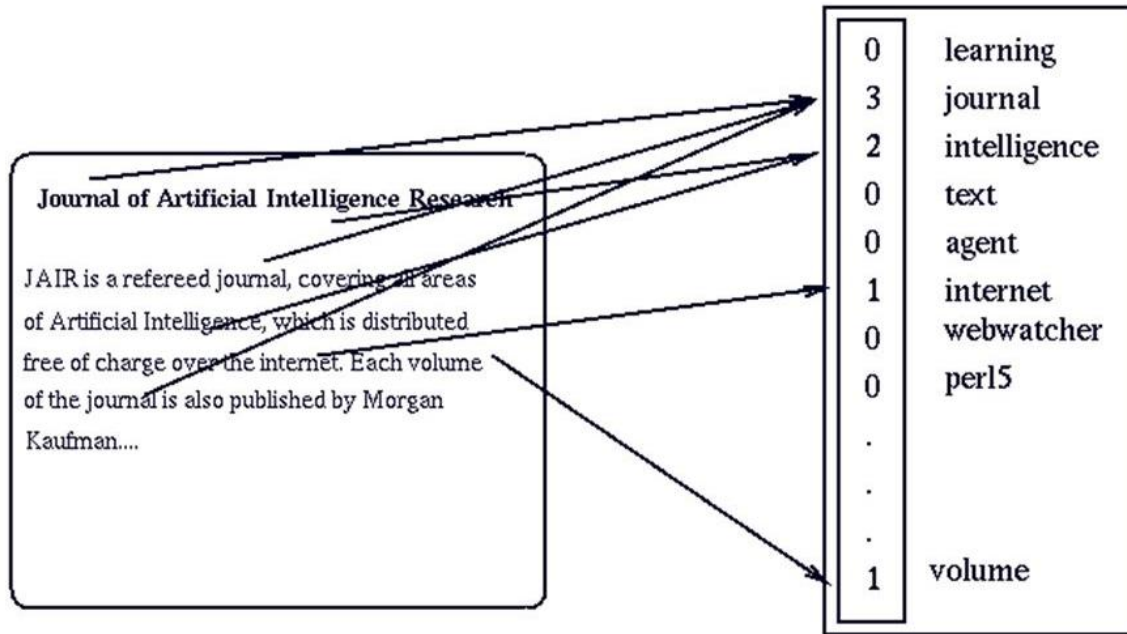


Рисунок 2.11 – Приклад Bag of words

Такі підходи були і залишаються підходящими для областей, де кількість текстів невелика, а словник обмежений. Та коли у наше життя прийшла світова павутина інтернету все стало складніше і простіше: у доступі тепер маємо безліч текстів, а словник постійно збільшується та змінюється. Лише тільки в українській мові кількість становить близько мільйона, тому матриця спільних спостережуваностей тільки пар слів буде розмірністю  $10^6 \times 10^6$ , що для обробки вже є дуже ресурсомісткою навіть для сучасних комп'ютерів.

#### 2.4.1 Word2Vec

Word2vec – метод ефективного створення вкладень, розроблений у 2013 році чеським аспірантом Томашем Миколовим, який запропонував свій підхід до word embedding [17]. Даний метод був заснований на гіпотезі, яку в науці

називають гіпотезою локальності – «слова, що зустрічаються в однакових оточеннях, мають близькі значення». Алгоритми word2vec використовують контекст, щоб сформувати чисельні уявлення слів, тому слова, що використовуються в тому самому контексті, мають схожі вектори. Word2vec у порівнянні з іншими підходами добре справляється з великими обсягами даних.

Модель підходу проста – необхідно передбачати ймовірність слова за його контекстом або оточенням (2.5). То треба вчити такі вектори слів, щоб ймовірність, що присвоюється моделлю слову, була близька до ймовірності зустріти це слово в цьому оточенні в реальному тексті.

$$P(w_0|w_c) = \frac{e^{s(w_0, w_c)}}{\sum_{w_i \in V} e^{s(w_i, w_c)}}, \quad (2.5)$$

де  $w_0$  – вектор цільового слова;

$w_c$  – вектор контексту, обчислений із векторів оточуючих потрібне слово інших слів;

$s(w_i, w_c)$  – функція, яка зіставляє одне число двом векторам.

Представлена формула називається softmax. Ця функція потрібна для того, щоб модель могла навчитися за допомогою методу зворотного поширення помилки (backpropagation).

Навчання моделі відбувається за таким алгоритмом – беремо послідовно  $(2k+1)$  слів. Де слово в центрі – це те саме слово, яке потрібно передбачити, а оточуючі слова – його сенс за змістом довжиною  $k$  з кожної сторони від цього слова. Для кожного слова в моделі представлено унікальний вектор. Цей вектор змінюється у процесі навчання моделі. Такий підхід називається CBOW – continuous bag of words.

Слідом за CBOW був також запропонований протилежний йому підхід skip-gram (словосполучення з пропуском) (рисунок 2.12). Суть підходу у том, що треба з цього слова вгадати його контекст (вектор контексту).

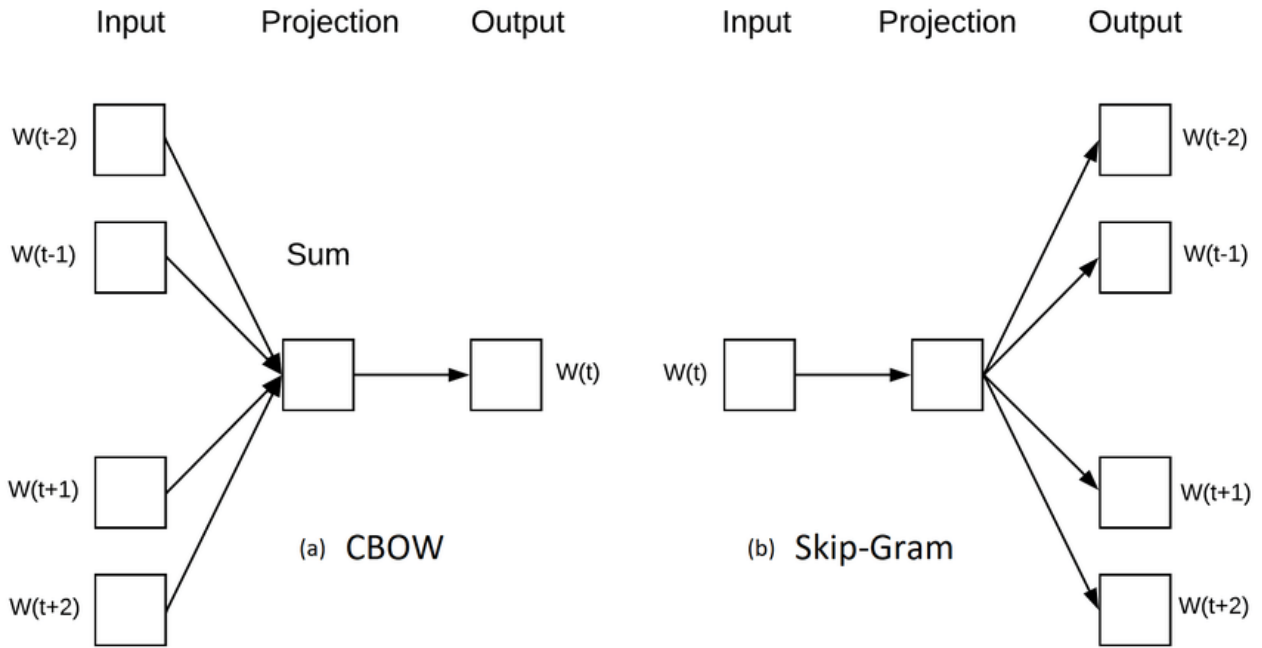


Рисунок 2.12 – Архітектури моделей CBOW та Skip-gram

На сьогоднішній день векторні уявлення слів застосовуються в багатьох областях:

- моделювання мов;
- чат-боти;
- машинний переклад;
- питання-відповіді системи;
- аналіз тональності тексту;
- тегування частин мови (POS);
- виявлення іменованих сутностей тощо.

Розглянемо процес створення векторних слів, наприклад, за допомогою нейромережових методів. Для того щоб навчити вибірку без заздалегідь розмічених даних, потрібно спочатку вирішити такі задачі:

- необхідно вирішити завдання ONE – створити кортежі даних у форматі [вхідне слово, вихідне слово], де кожне слово представляється у вигляді вектору довжиною  $n$ , де  $i$ -значення розташування слова кодується як 1 на  $i$ -ої позиції та 0 на решті позицій;
- далі необхідно створити модель, до якої на вхід та вихід надходять створені ONE вектори;
- для точного передбачення правильного слова, треба визначити функцію втрат;
- переконатися, що усі схожі за змістом слова мають схожі вектори.

Для розуміння як створюються структуровані дані з тексту візьмемо наприклад пропозицію “The cat pushed the glass off the table.” (рисунок 2.13).

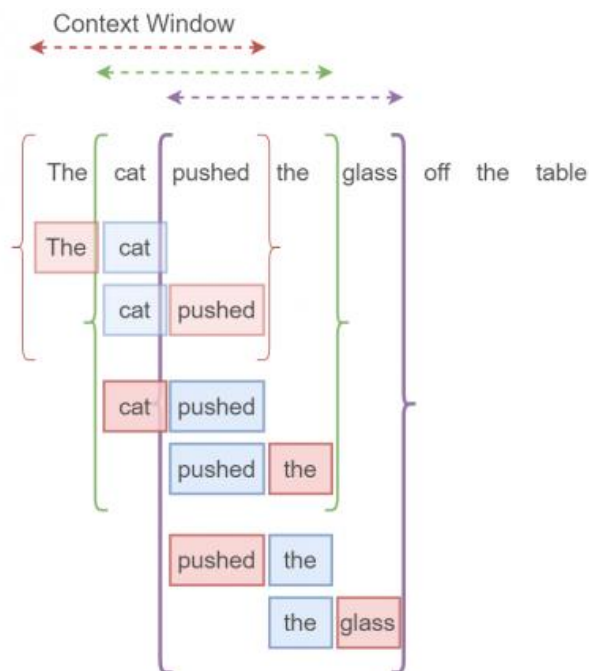


Рисунок 2.13 – Приклад структуризації даних

На зображенні можемо побачити фігурні дужки, що позначають контекстне вікно. Також ми бачимо, що кожний токен виділяється двома кольорами, де червоний – вихідний вектор, тобто цільовий, а синій – відповідно вхідний, тобто контекстний. Як згадувалося раніше про роботу контекстних вікон, ми бачимо, що у кожному такому вікні виділено по два токени (слова). Тобто на кожне цільове слово припадає кілька сусідніх.

Якщо вирішувати цю задачу за допомогою нейромереж, окрім наборів вхідних та вихідних ONE векторів, нам потрібен також шар вкладання (embedding layer). Такий шар містить у собі усі вектори слів у вигляді матриці розміром – кількість слів у словнику на розмір простору векторного уявлення слів (рисунок 2.14). Така матриця заповнюється рандомно, як і нейромережа, та коректно налаштовується в процесі навчання та оптимізації.

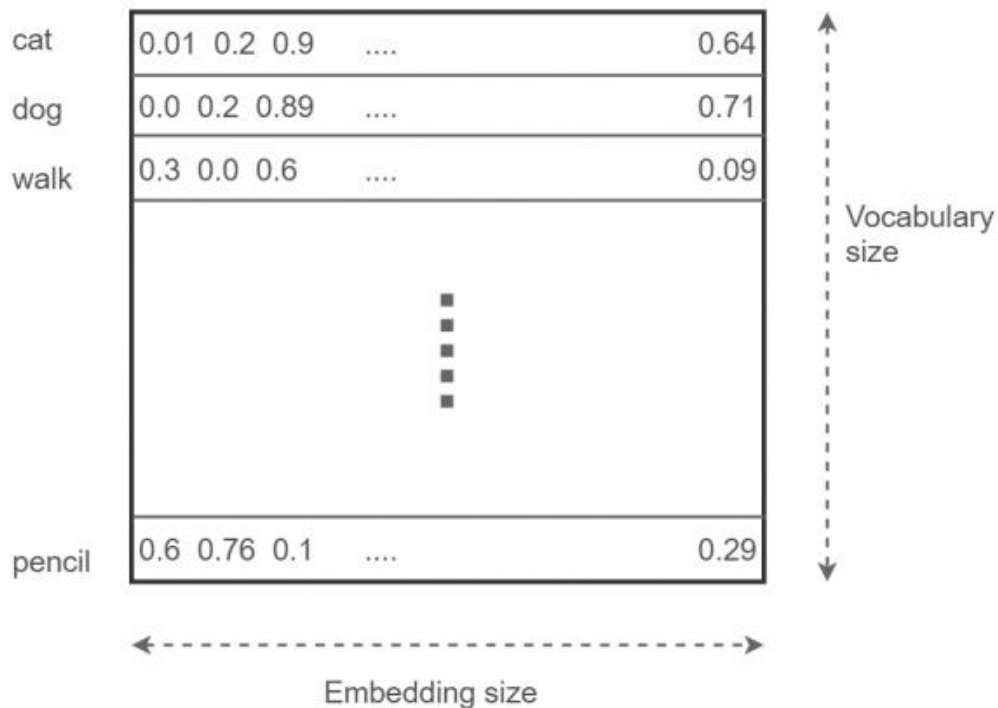


Рисунок 2.14 – Матриця ембедінгу

В загалом, приклад результату роботи методу Word2Vec зображено на рисунку 2.15. Тут ми можемо побачити, що усі слова, щ відносяться до котів – cats, kitten, cat, виділені однією областю синім кольором. Таким же чином у просторі виділені сутності, що відносяться до собак та домашніх тварин.

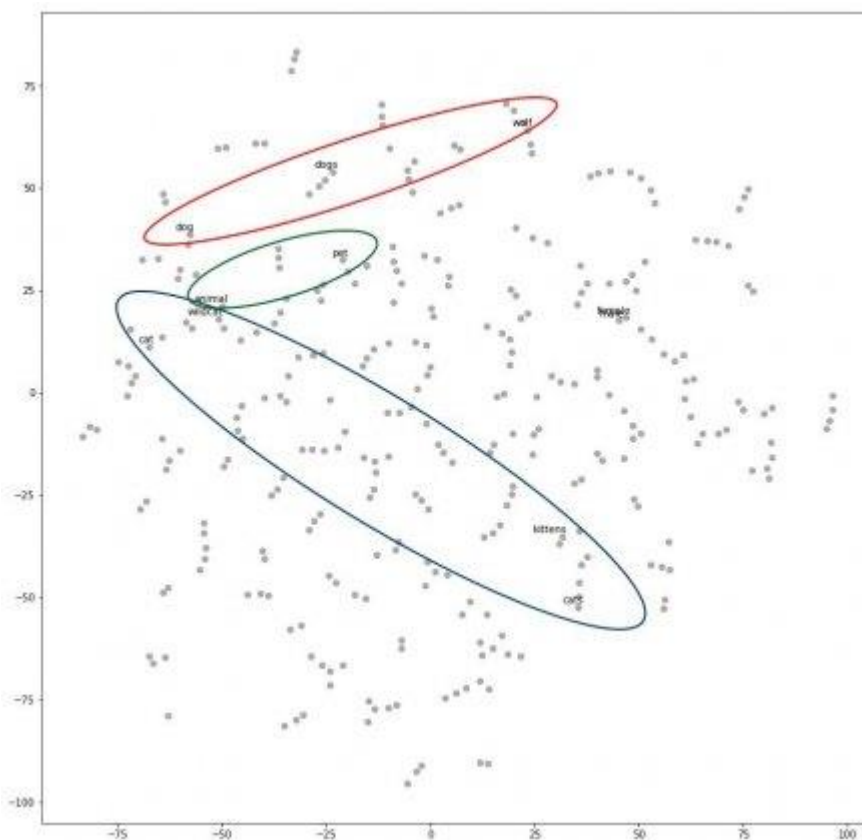


Рисунок 2.15 – Приклад роботи Word2Vec

Недоліком word2vec є те, що з його допомогою не можуть бути представлені слова, які не зустрічаються в навчальній вибірці. Але на сьогодні Word2Vec показав свою користь на практичних завданнях аналізу текстів, і на поточний момент у завданнях обробки природних мов використовується в основному саме він.

## 3 ГРАФИ ЗНАНЬ ТА НЕЙРОМЕРЕЖЕВІ МОДЕЛІ

### 3.1 Графи знань

Людське знання забезпечує формальне розуміння світу. Графи знань, які представляють структурні відносини між сутностями, стають дедалі популярнішим напрямом досліджень у сфері пізнання та інтелекту на рівні людини.

Об'єднання людських знань є одним із напрямків досліджень штучного інтелекту (ШІ). Подання знань та міркування, натхненні людиною розв'язанням проблем, повинні представляти знання для інтелектуальних систем, щоб отримати можливість вирішувати складні завдання. Останнім часом граfi знань як форма структурованих людських знань привернули велику увагу дослідників як у академічних колах, і у промисловості.

#### 3.1.1 Загальні відомості та області застосувань графів знань

Графи знань (KG) організують дані з кількох джерел, збирають інформацію про об'єкти, що нас цікавлять, в даній області або задачі (наприклад, про людей, місця або події) і встановлюють зв'язки між ними. У науці про дані та штучний інтелект граfi знань зазвичай використовуються для:

- полегшення доступу до джерел даних та їх інтеграцію;
- додавання контексту і глибини до інших методів штучного інтелекту, переважно заснованих на даних, таких як машинне навчання;
- служіння зв'язувальною ланкою між людьми та системами, наприклад, генерувати легкочитані пояснення або, у ширшому масштабі, створювати інтелектуальні системи для вчених та інженерів.

Подання знань має довгу історію розвитку в галузі логіки та штучного інтелекту. Ідея графічного представлення знань вперше переформулюється з 1956 року як концепція семантичної мережі, запропонована Річенсом [10], тоді як символічно-логічне знання може сягати загального вирішувача завдань [1] у 1959 року. База знань вперше використовується з засновані на знаннях системи міркувань та вирішення проблем. MYCIN [2] – одна з найвідоміших експертних систем для медичної діагностики, що базується на правилах, з базою знань близько 600 правил.

Пізніше у співтоваристві репрезентації людських знань розвинулися мови на основі фреймів, на основі правил та гібридні репрезентації. Приблизно наприкінці цього періоду розпочався проект Сус, спрямований на збирання людських знань. Фреймворк опису ресурсів (RDF) та мова веб-онтологій (OWL) були випущені по черзі та стали важливими стандартами семантичної мережі. Потім було опубліковано безліч відкритих баз знань чи онтологій, таких як WordNet, YAGO, Freebase та інші.

Як термін, “Knowledge Graph” був придуманий ще у 1970-х роках та за змістом мав дуже схоже значення із семантичними мережами, а у 1988 році Стокман та Фріз [7] запропонували сучасну ідею структурування знань у графі. Пізніше, вже у 2007 році на основі графів у якості репозиторія знань були засновані: DBpedia, яка працювала лише із даними з Вікіпедії, та Freebase, який працював також із набором загальнодоступних даних [3]. Сучасний термін «графів знань» був запроваджений Google у 2012 році [7] для позначення своєї бази знань загального призначення, хоча аналогічні підходи існували від початку сучасного ШІ у таких галузях, як представлення знань, отримання знань, обробка природної мови, онтологія, інженерія та семантична мережа.

Сьогодні KG широко використовуються у всьому, від пошукових систем та чат-ботів до рекомендацій продуктів та автономних систем. У науці

про дані поширені варіанти використання пов'язані з додаванням ідентифікаторів та описів до даних різних модальностей, щоб забезпечити осмислення, інтеграцію та зрозумілий аналіз. У штучному інтелекті графі знань доповнюють методи машинного навчання для того, щоб:

- зменшити потребу у великих помічених наборах даних;
- сприяти передачі навчання та зрозумілості;
- кодувати знання предметної області, завдань та додатків, які було б дорого вивчати лише на основі даних.

Граф знань організовує та інтегрує дані відповідно до онтології, яка називається схемою графа знань, та застосовує алгоритм міркувань для отримання нових знань. Такі моделі знань можуть бути створені з нуля, наприклад, експертами в предметній галузі, вивчені з неструктурованих або напівструктурованих джерел даних, або зібрані з існуючих граф знань, як правило, за допомогою різних напівавтоматичних або автоматичних механізмів перевірки та інтеграції даних.

Хоча існує безліч визначень графів знань, більшість з них сходяться на думці на наступних визначеннях:

- на відміну від баз знань, зміст KG організовано як граф, де однаково важливі вузли (сутності та його типи), відносини між вузлами та його атрибути. Це спрощує інтеграцію нових наборів даних та підтримує дослідження шляхом переходу від однієї частини графа до іншої за їх посиланнями;

- значення даних кодується для програмного використання в онтології, яка описує типи сутностей у графі та їх характеристики та можуть бути представлені як схеми підграфу. Це означає, що граф є одночасно місцем для організації та зберігання даних, а також для обміркування того, про що йдеться, та отримання нової інформації;

- графи знань є гнучкими з погляду типів даних і схем, які можуть підтримувати. Вони, включаючи їхні схеми, розвиваються, щоб відображати зміни в предметній області, і нові дані додаються до графа в міру його появи.

Деякі графи знань використовуються переважно всередині організації, яка їх створила. Найбільш поширеним прикладом є діаграма знань Google, яка використовується у веб-пошуку, або діаграма продуктів Amazon. Інші графи знань знаходяться у відкритому доступі. До них відносяться DBpedia, Wikidata, WordNet, Geonames тощо.

Сучасні графи знань є результатом складного складання ручного та автоматичного моделювання та конвеєрів прийому даних. Щоб залишатися в курсі цих процесів, забезпечуючи при цьому актуальність, несуперечність та достовірність інформації, потрібні спеціалізовані соціально-технічні методи, починаючи від отримання знань і закінчуючи обробкою природної мови, машинним навчанням та взаємодією людини з комп'ютером. Графи знань не використовуються ізольовано від інших методів, вони співіснують та доповнюють рішення на основі даних, включаючи обробку природної мови (розуміння тексту, отримання знань, генерація тексту) та управління даними (семантичне маркування, зв'язування записів, очищення даних тощо). Нещодавні дослідження розглядають, наприклад, компроміси між семантичними уявленнями та глибоким навчанням, а також у більш широкому сенсі проектування архітектури ШІ.

Слід зазначити, що графи знань лежать в основі багатьох технологій, орієнтованих на людину, таких як пошук, відповіді на запитання, діалоги та рекомендації. Вони також можуть допомогти, крім іншого, в управлінні даними, виявленні шахрайства, управлінні знаннями, пошуку, чат-ботах, рекомендаціях, а також в інтелектуальних системах у різних організаційних підрозділах.

### 3.1.2 Побудова графу знань

Граф знань (KG) представляє дані у вигляді семантичного триплету (тобто упорядкованих наборів термінів), що складаються з  $(s, p, o)$ : суб'єкт  $s - s \in U \cup V$ , предикат  $p - p \in U$ , об'єкт  $o - o \in U \cup V \cup L$ , які можуть бути інтернаціоналізовані ідентифікатори ресурсів (Internationalized Resource Identifier, IRI) [15]  $i \in I$ , порожні вершини  $b \in V$  або літерали  $l \in L$ , де  $U, V$  і  $L$  – нескінченні множини і попарно не перетинаються [16]. Такий триплет також часто позначається як (head, relation, tail) або  $(h, r, t)$ .

Кожен триплет визначає одне з'єднання між двома об'єктами у графі. Набір допустимих відносин та типів сутностей визначає онтологію KG, яка також є його загальною структурою. Наприклад, це може бути графік географічних об'єктів, біомедичних споруд або веб-сторінок. Враховуючи деякий набір записів, KG дозволяє зробити висновок (рисунок 3.1).

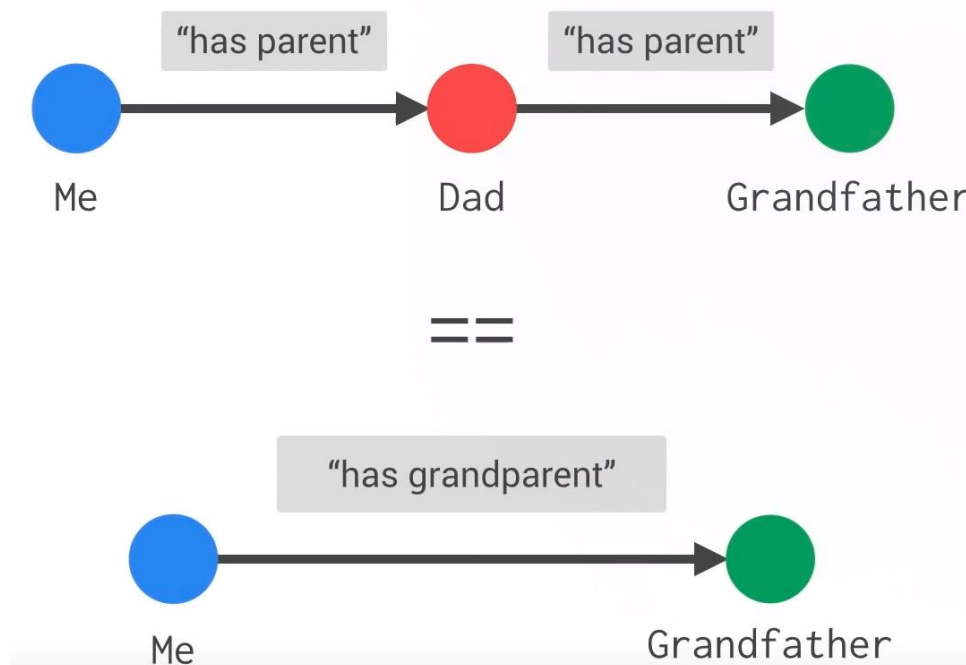


Рисунок 3.1 – Приклад висновку графу

Отже, роздивимося стадії розробки графу знань для задач обробки природномовних текстів. На рисунку 3.2 зображений дуже узагальнений пайплайн для роботи із текстами. Тому, роздивимося його детальніше.

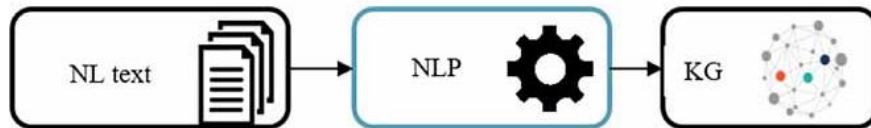


Рисунок 3.2 – Паплайн Knowledge Graph

На рисунку 3.3 вже можна побачити детальніший пайплайн (стадії розробки) графу знань для задач обробки природних мов в цілому.

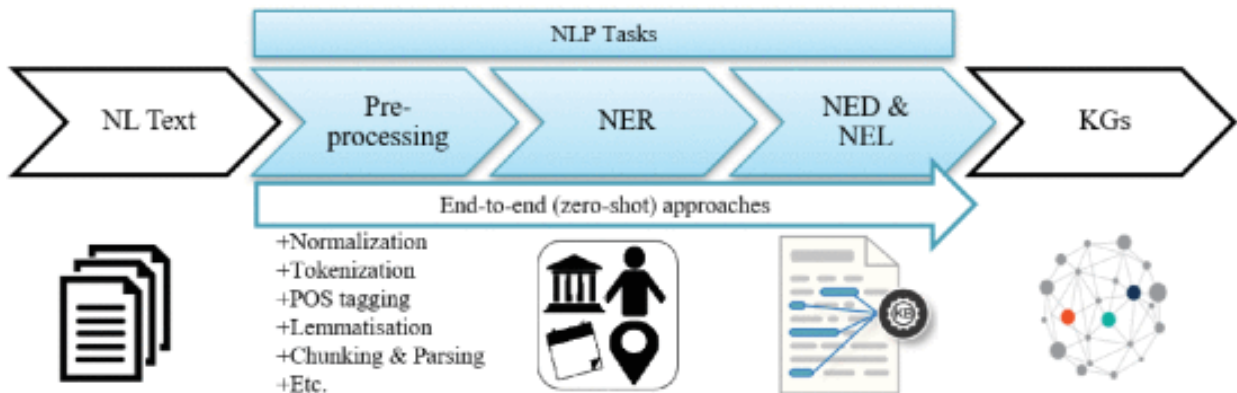


Рисунок 3.3 – Детальний паплайн Knowledge Graph

На вхід до моделі подаються корпуси – набір текстових даних. Дані можуть подаватися такі як:

- структуровані – реляційні бази даних;
- напівструктуровані – HTML, XML, JSON тощо;

- неструктуровані – текст, зображення, документи.

Наступним кроком є попередня обробка даних, що включає такі пункти:

- сегментація – розбиття текстового документу на речення;
- токенизація – розбиття речень на слова;
- POS-тегування – визначення частин мови;
- лемматизація – приведення слів до канонічної форми тощо.

Як ми бачимо, тексти та текстові документи є тільки неструктурованими даними, тому наступним кроком є вилучення інформації (knowledge extraction). Як зазначалося раніше вилучення інформації включає три частини: вилучення сутностей, вилучення відносин і вилучення атрибутів.

Вилучення сутності (entity extraction) – класифікація знань на заздалегідь позначені категорії. Такий метод належить до автоматичного розпізнавання іменованих сутностей з вихідного корпусу. Оскільки сутність є найголовнішим елементом у графі знань, повнота, точність та ступінь запам'ятовування її добування безпосередньо впливатимуть на якість підсумкового графу знань. Методи добування сутностей можна розділити на методи: на основі правил і словників, на основі статистичного машинного навчання та методи добування, орієнтовані на відкриту предметну галузь.

Вилучення відносин (relation extraction) – пошук зв'язків між сутностями та отримує семантичну інформацію для побудови графів знань. До методів вилучення відносин відносять: вилучення відносин відкритих сутностей та вилучення взаємозв'язків сутностей на основі спільних міркувань.

Вилучення атрибутів (attribute extraction) – визначення інтенційної семантики сутностей та більш чітко визначає поняття сутності. Тобто для сутності з допомогою атрибута можна формувати повну схему сутності. Оскільки атрибут сутності можна розглядати як іменованій зв'язок між

сутністю і значенням атрибута, проблема вилучення атрибутів сутності може бути перетворена на завдання вилучення відносини.

Ці два кроки необхідні саме для неструктурованих та напіструктурованих даних. Для структурованих даних попередня обробка потрібна залежно від постановки задачі.

Саме для даної кваліфікаційної роботи була обрана задача видобування іменованих сутностей (NER), тобто крок із задачами NED та NEL можна не роздивлятись. Але якщо ми маємо справу із повноцінною системою із візуалізацією самого графу, яка працює і обробляє природомовні тексти, в залежності від постановки задачі, ці методи вилучення інформації (IE) надають більше можливостей для роботи із текстом. Тож, основні кроки як вирішується задача NER наведені у розділі 2.

Після вирішення завдання NER, за допомогою програмних методів можна створювати візуалізацію рішення у вигляді графа (рисунок 3.4).

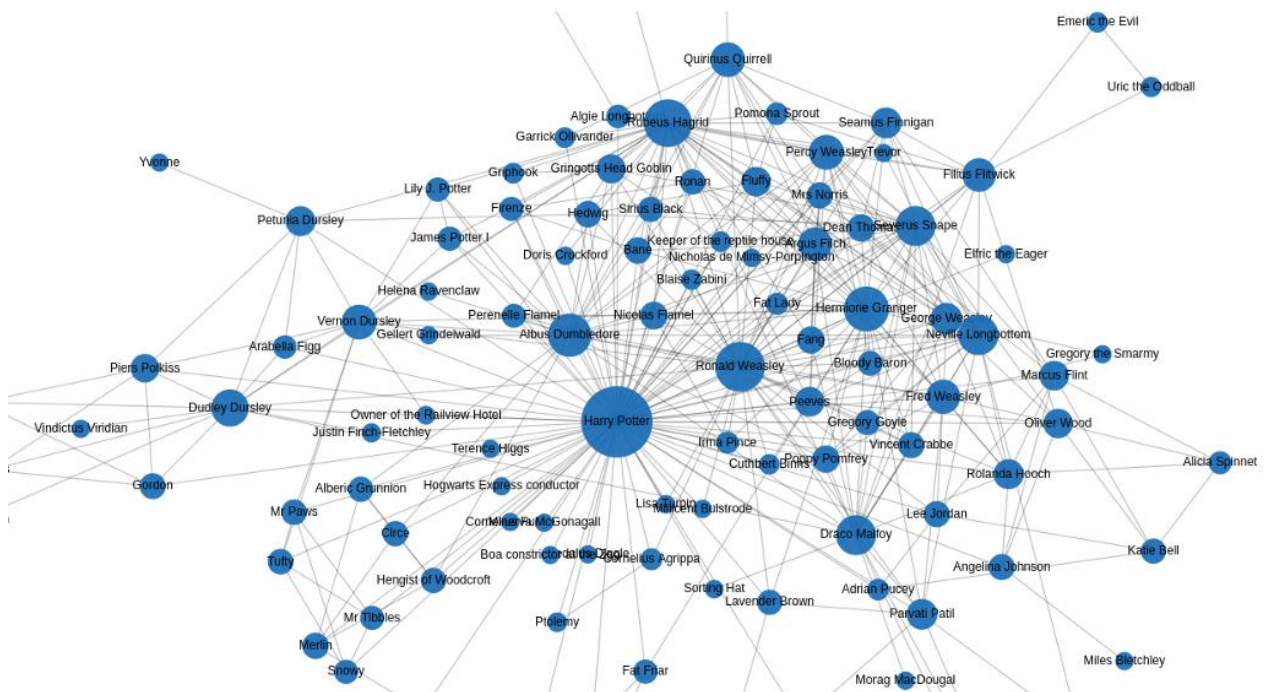


Рисунок 3.4 – Приклад графу знань

## 3.2 Глибинні нейронні мережі

Ще одним із методів вирішення NER виділяють нейромережеві моделі. Такий підхід для вирішення задач обробки природномовних текстів є самим новим та ефективним.

### 3.2.1 Глибинні нейронні мережі для задач обробки природної мови

Однією з причин активного поширення технології обробки природної мови стало те, що для цього завдання стали застосовувати нейронні мережі, а саме нейронні мережі з архітектурою трансформерів, що дозволило досягти великого прогресу в автоматичній обробці тексту.

Також для обробки природної мови відіграло також значну роль наявність готових до використання бібліотек: Hugging Face, TensorFlow, PyTorch, spaCy, nltk, rumorphy тощо.

Ми знаємо, що вхідними даними до нейронної мережу повинні бути числа числа, тому, коли ми використовуємо нейромережі для аналізу даних, наші дані необхідно перевести в набір чисел. Якщо ми працюємо зі структурованими даними, такими як таблиці, в яких там можливо два варіанти, якщо дані в числовому вигляді, то з ними нічого не робимо, якщо дані в категоріальному вигляді, наприклад, чоловік або жінка, то такі дані подаються у вигляді векторів у форматі ONE.

На першому етапі нам необхідно розбити текст на окремі частини, кожна з яких представлятиметься у цифровому вигляді окремо. Цей процес називається векторизацією. Тут ми можемо використовувати різні методи.

Перший спосіб – це числове кодування кожному токеноу призначаємо окремий код. Це може бути частота, з якою токен зустрічаються в тексті або якийсь код, наприклад, популярне кодування utf-8, ASCII.

Другий спосіб – коли кожному токєну ставимо відповідність не одне число, а цілий вектор який містить кілька чисел. Найпростіший варіант – використовувати формат у ONE. У цьому випадку вектор містить стільки чисел, скільки можна використовувати токєнів і всі значення вектору дорівнюють 0, крім одного, яке відповідає потрібному нам токєну.

Ще один варіант – це щільні векторні уявлення (embedding). У цьому випадку кожному токєну ставиться відповідно не одне число, а цілий вектор, але розмірність такого вектору значно нижче, ніж у ONE. У такому векторі можуть використовуватися не тільки 0 та 1, але й будь-які числа.

### 3.2.2 RNN

До того, щоб коректно аналізувати текст і розуміти його сенс, необхідно аналізувати текст набір із ізольованих токєнів, символів, слів або речень, а як послідовність токєнів. Для того, щоб це робити нам необхідно використовувати спеціальну архітектуру нейронних мереж, які вміють аналізувати саме цю послідовність, а не тільки дані, які залежать один від одного. Для завдання аналізу послідовностей добре підходять рекурентні нейронні мережі (RNN) (рисунок 3.5).

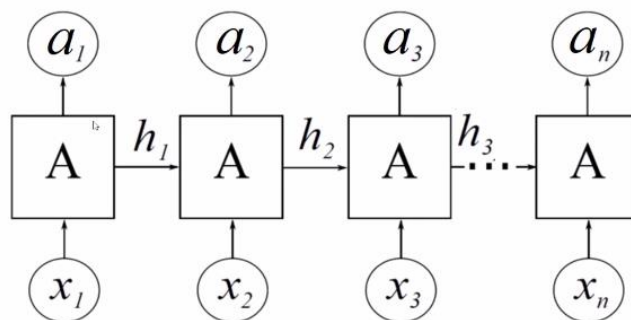


Рисунок 3.5 – RNN

У рекурентних мережах дозволені цикли у разі вихід нейронів. Може бути підключений до входу всіх нейронів у поточному або іншому шарі. Рекурентну нейронну мережу також можна подати у вигляді мережі з прямим поширенням сигналу. Для цього використовується прийом, який називається розгортанням у часі – створюється кілька копій рекурентної нейронної мережі. На вхід копії нейронної мережі надходять елементи послідовності.

Що видає RNN? Вона видає два значення – вихідне  $a_1$  (надходить на вихід з нейронної мережі), вхідний  $h_1$  (надходить на вхід копії мережі в наступний момент часу). Це можна назвати прихованим станом, який враховує те, що було на попередніх етапах аналізу послідовності. Копія нейронної мережі в наступний момент часу на вхід отримує другий елемент послідовності  $x_2$ , а також стан попереднього етапу. Залежно від результатів аналізу так само видає два значення – вихідне та значення із прихованим станом.

На відміну від повнозв'язаної нейронної мережі, рекурентна нейронна мережа може працювати з послідовністю вхідних даних будь-якої довжини. Для цього потрібно створити стільки копій RNN, розгорнутої в часі, скільки у нас елементів вхідних даних у послідовності. Рекурентна нейронна мережа може працювати у двох режимах і описаний був режим послідовності (sequence to sequence). Такий режим роботи корисний для автоматичного перекладу або генерації тексту.

Інший режим називається послідовність вектор (sequence to vector). У цьому випадку на вхід нейронної мережі подається послідовність, а на виході беремо тільки останнє значення та ігноруємо все, що видала нам нейронна мережа на попередніх етапах. Такий режим корисний, наприклад, для задач класифікації.

Коли ми розвертаємо рекурентну нейронну мережу в часі, отримуємо звичайну мережу з прямим поширенням сигналів. Таку мережу можна

навчити за допомогою навчання з учителем та методом зворотного поширення помилки. Складність полягає в тому, що кількість шарів такої розгорнутої нейронної мережі залежить від кількості елементів даних послідовності.

Такі глибокі нейронні мережі складно навчити через проблеми зникаючого градієнта. Вона полягає в тому, що сигнал про помилки, за яким можна розрахувати, як нам потрібно змінювати ваги в шарах, зменшується при передачі цього одного шару до іншого. І якщо нейронна мережа містить 5 або 10 шарів, то цілком можна обійтися з традиційними методами, то для мережі 200 прихованих шарів це стає великою проблемою.

Для того, щоб вирішити цю проблему використовується складніша архітектура нейронних мереж, в яких не просто вихід нейрона з'єднаний з його входом, а сама структура нейрона складніша. Серед таких архітектур популярними зараз є LSTM, GRU. Роздивимося саме LSTM.

### 3.2.3 Архітектура LSTM

Мережі LSTM влаштована складно – в цих мережах елементи мережі є не одним нейроном або шаром з нейронів, а цілим набором шарів, які взаємодіють один з одним за певними правилами. Такі набори називаються осередками у мережах LSTM (LSTM cell). На рисунку 3.6 показана мережа LSTM розгорнута в часі. У середині, поточна копія, ліворуч копія мережі в попередній момент часу та праворуч копія мережі в наступний момент часу. На вхід мережі в різні моменти часу надходять елементи послідовності  $x$  і кожен момент часу мережа видає якесь значення  $h$ .

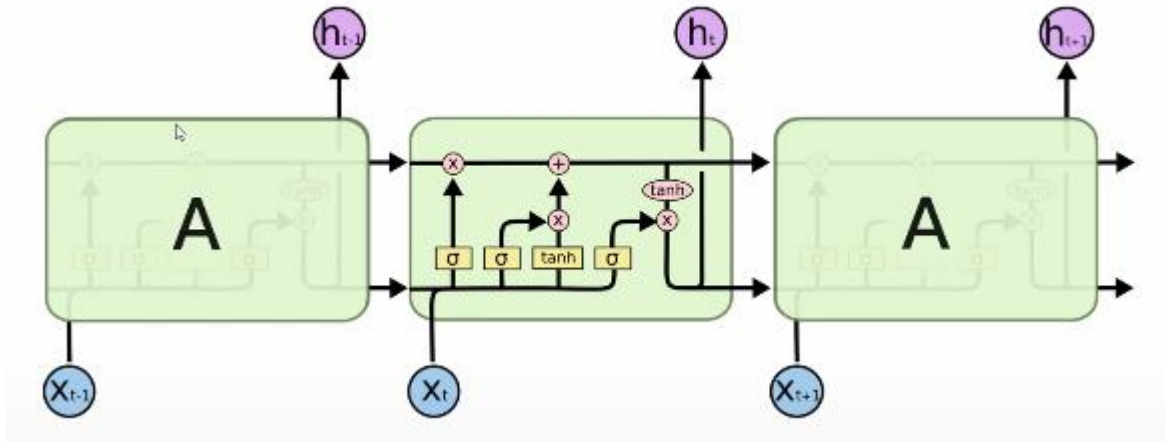


Рисунок 3.6 – Архітектура LSTM

Мережа LSTM передає на вхід своєї копії наступного часу не одне значення, а цілих два. Архітектура осередку мережі LSTM складається з декількох шарів нейронів, які з'єднуються між собою за допомогою операції поелементного множення або складання. Основою в осередку LSTM є стан саме вона дозволяє зберегти дані на тривалий проміжок часу.

Роздивимося детальніше про архітектуру. На рисунку 3.7 зображений сигнал, що надходить з попереднього етапу роботи мережі  $C_{t-1}$ , передається на наступний етап  $C_t$ .

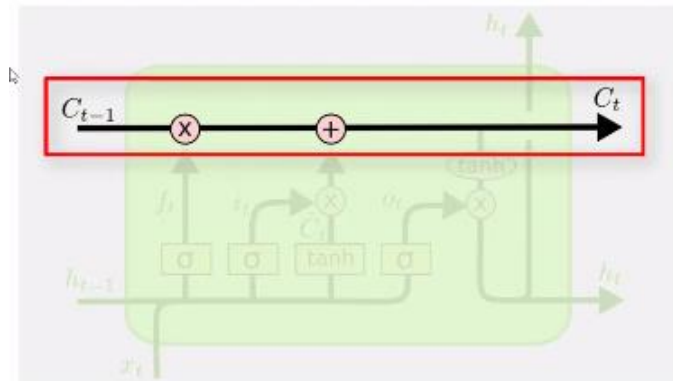


Рисунок 3.7 – Архітектура LSTM

Саме тому, що у мережі LSTM є лінія передачі стану, осередок LSTM може зберігати дані необмежено довго. Для того, щоб керувати станами осередку LSTM використовуються так звані вентиля (gate) (рисунок 3.8).

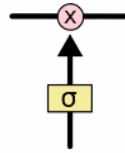


Рисунок 3.8 – Вентиль LSTM

Вони влаштовані дуже просто – це шар нейронів, що мають на виході функції sigmoid, в якому значення змінюється від нуля до одиниці і наступна операція з елементного множення. Якщо наш нейрон видав 1, елемент сигналу проходить без змін, якщо на виході сигмоїда, отримуємо 0, елемент сигналу не передається. Також сигнал може бути ослаблений, якщо на виході із сигмоїду отримуємо значення в діапазоні від нуля до одиниці. У мережах LSTM використовуються три типи таких вентилів:

- вентиль забуття (forget gate) – управляє тим, коли дані стираються з комірки пам'яті (рисунок 3.9);

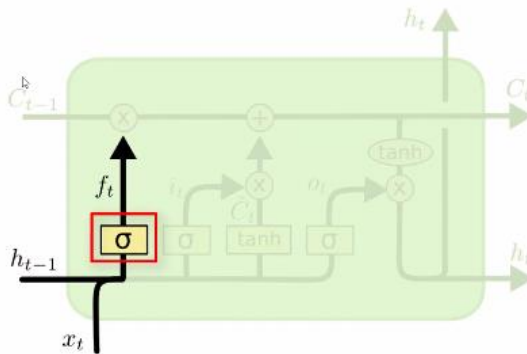


Рисунок 3.9 – Вентиль забуття

- вхідний клапан (input gate) – керує тим, які дані записуються в комірку пам'яті (рисунки 3.10);

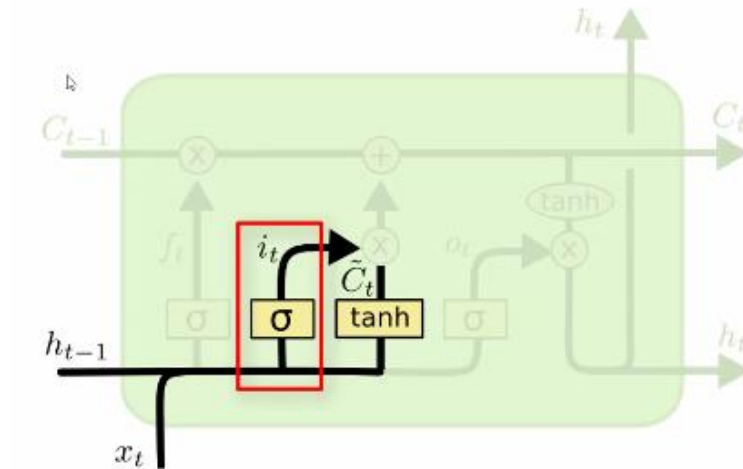


Рисунок 3.10 – Вхідний клапан

- вихідний клапан (output gate) – потрібен для того, щоб визначити який сигнал буде подаватися на вихід нейронної мережі (рисунки 3.11).

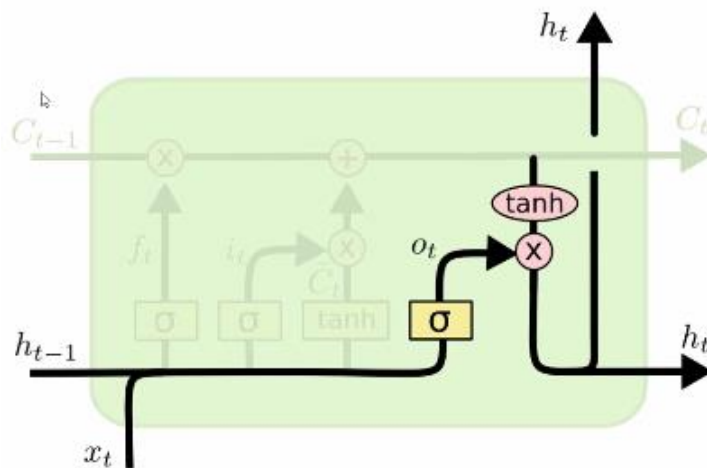


Рисунок 3.11 – Вихідний клапан

Мережа LSTM зараз дуже популярна, проте недолік її у цьому, що у ній досить багато елементів. Тому, щоб навчити таку мережу, потрібні великі обчислювальні ресурси.

Отже, ми виявили, що найкращою можливою моделлю, яка може вирішити проблему звичайної RNN, є моделі Long-Short Time Memory (LSTM). Тепер роздивимось двонаправлений LSTM (Bi-directional LSTM). Двонаправлений LSTM є комбінацією двох LSTM – одна рухається вперед «справа наліво», а інша – назад «зліва направо», таким чином захоплюючи всю суть/контекст даних. Для NER, оскільки контекст охоплює минулі та майбутні мітки у послідовності, ми повинні враховувати як минулу, так і майбутню інформацію. Архітектуру мережі можна побачити на рисунку 3.12.

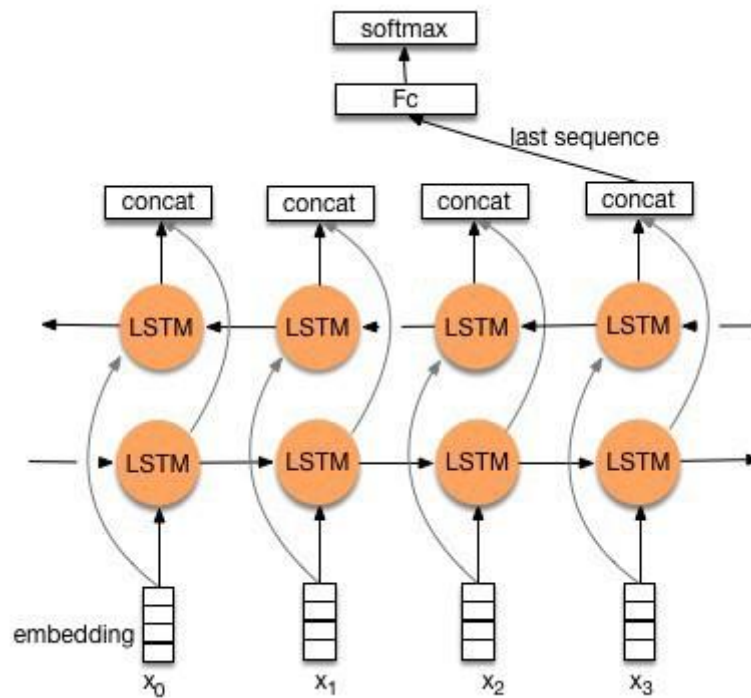


Рисунок 3.12 – Архітектура Bi-directional LSTM

Embedding Layer: ELMo (Embedding from Language Models) – це глибоке контекстуалізоване представлення слів, яке моделює складні характеристики використання слів (наприклад, синтаксис і семантика) і те, як це використання змінюється в різних лінгвістичних контекстах (тобто для моделювання полісемії). Приклад: хоча термін «Apple» є поширеним, але ELMo надає різні вбудовування для обох (фруктів та організації) через контекстну логіку.

Ми вже знаємо початковий алгоритм для навчання нейромережі – робота з текстовими даними, вона не дуже відрізняється від алгоритму для графів знань, однак у випадку з мережею нам потрібен шар ембедінгу, який описується вище.

Після навчання мережі, та виконання задачі NER, необхідно перевірити оцінку NER за допомогою метрики f-міра, про яку згадувалося у розділі 2. Тому що у разі NER ми можемо мати справу з важливими фінансовими, медичними або юридичними документами, і точне визначення названих об'єктів у цих документах визначає успіх моделі.

## 4 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

### 4.1 Вибір програмних засобів

У якості методу вирішення задачі видобування іменованих сутностей (NER) було обрано граф знань. Також для програмної реалізації задачі та графу було обрано мову Python.

Python – високорівнева мова програмування, яка в основному призначена для роботи зі штучним інтелектом.

Оскільки задача обробки текстів є дуже об'ємною та вимагає багато ресурсів від комп'ютеру, було прийнято рішення реалізувати програмний додаток у Google Colab.

Також, для подальшої розробки нам знадобляться такі бібліотеки як:

- nltk – набір бібліотек та програм для символної та статистичної обробки природної мови.;
- spaCy для попередньої обробки тексту та глибокого навчання та matplotlib;
- matplotlib для візуалізації графа.

#### 4.1.1 Google Colab

Google Colab, також відомий як Colaboratory – це середовище Jupyter, яке надається та керується компанією Google із можливістю роботи з ЦП, графічними процесорами і навіть TPU. Він працює так само, як і будь-який інший нотпад Jupyter. Ми можемо програмувати на Python, писати описи як виноски. Він також має інші функції Jupyter та багато іншого.

Чому саме Google Colab? Існує багато причин, за якими хтось віддає перевагу використовувати такі служби, як Google Colab:

- не займає час для налаштування щоб почати працювати з Google Colab;

- незалежність від платформи: оскільки доступ до нотпадів Jupyter можна отримати безпосередньо з браузера, ми можемо зробити це на будь-якому комп'ютері, Mac, Windows, Linux тощо, і він працюватиме точно так само;

- наявність безкоштовних ресурсів: навчання моделі глибокого навчання вимагає багато ресурсів, і тому не всі ноутбуки та настільні комп'ютери витримують це завдання. Google Colab надає безкоштовний доступ до графічних процесорів, які ми можемо використовувати для наших завдань глибокого навчання.

Тепер, коли ми зрозуміли, що таке Google Colab і які переваги він приносить, перейдемо до реалізації NER за допомогою графу знань.

## 4.2 Розробка програми

Отже, необхідно побудувати граф знань із нуля. Тому, для цього необхідно обрати набір текстів або речень. Для цього скористуємося датасетом із набором речень із різних статей Вікіпедії про фільми. Ця вибірка знаходиться у вільному доступі в інтернеті, містить 4318 речень, які були вилучені з більш ніж 500 статей. Кожне з цих речень містить рівно два об'єкти – один суб'єкт і один об'єкт. На рисунку 4.1 наведений приклад таких речень.

Отже, необхідно побудувати граф знань для іменованих сутностей із нуля. У якості набору даних буде невеликий текст, який заздалегідь вже був поділений на окремі речення (рисунок 4.1).

index	sentence
0	Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much.
1	Mr. Dursley was the director of a firm called Grunnings, which made drills.
2	Nothing like this man had ever been seen on Privet Drive.
3	He was tall, thin, and very old, judging by the silver of his hair and beard, which were both long enough to tuck into his belt.
4	His blue eyes were light, bright, and sparkling behind half-moon spectacles and his nose was very long and crooked, as though it had been broken at least twice.
5	This man's name was Albus Dumbledore.
6	Albus Dumbledore didn't seem to realize that he had just arrived in a street where everything from his name to his boots was unwelcome.
7	Potter wasn't such an unusual name.
8	He was sure there were lots of people called Potter who had a son called Harry.
9	True, I haven't introduced myself. Rubeus Hagrid, Keeper of Keys and Grounds at Hogwarts.

Show [35] [v] [n] [p] [o] [s] [e] [t] [a] [r] [t] [i] [n] [g]

Рисунок 4.1 – Приклад речень

Також слід зазначити, що у програмі для обробки англomовних текстів був використаний конвеєр `en_core_web_sm` (рисунок 4.2). Такий конвеєр складається з одного або кількох компонентів, які викликаються по порядку.

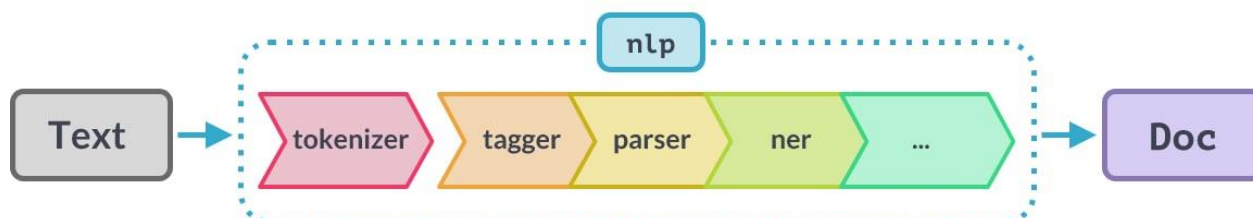


Рисунок 4.2 – Приклад схеми конвеєру

Тож, перед тим як працювати із текстом, спочатку нам потрібно провести токенизацію кожного слова та символу за допомогою вбудованого до `sraCu` методу (рисунок 4.3). Тобто кожне слово, розділовий знак будуть роздивлятися окремо, окрім випадків, коли розділовий знак “.” є частиною слова – скорочення, як у випадку з першим реченням “Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much.”.

```

    Mr.
    and
    Mrs.
    Dursley
    ,
    of
    number
    four
    ,
    Privet
    Drive
    ,
    were
    proud
    to
    say
    that
    they
    were
    perfectly
    normal
    ,
    thank
    you
    very
    much
    .

```

Рисунок 4.3 – Токенізація тексту

Після проведення токенизації необхідно також провести тегування частин мови (POS). У цьому випадку нам допоможе конвеєр із бібліотеки spaCy, який дозволяє нам прогнозувати які мітки та теги будуть застосовуватися у нашому тексті.

Тегування частин мови відбувається за допомогою таких атрибутів:

- text – просто слова із нашого тексту;
- lemma – привід слів до канонічної форми;
- POS – тег частини мови;
- tag – варіант розширеного тегу частин мови;
- dep – розширені залежності токенів.

Тож подивимося на результат POS-тегування на прикладі того ж речення (рисунок 4.4).

TEXT	LEMMA	POS	TAG	DEP		
Mr.	Mr.	PROPN	NNP	nsubj		
and	and	CCONJ	CC	cc		
Mrs.	Mrs.	PROPN	NNP	compound		
Dursley	Dursley	PROPN	NNP	conj		
,	,	PUNCT	,	punct		
of	of	ADP	IN	prep		
number	number	NOUN	NN	pobj		
four	four	NUM	CD	nummod		
,	,	PUNCT	,	punct		
Privet	Privet	PROPN	NNP	compound		
Drive	Drive	PROPN	NNP	appos		
,	,	PUNCT	,	punct		
were	be	AUX	VBD	ccomp		
proud	proud	ADJ	JJ	acomp		
to	to	PART	TO	aux		
say	say	VERB	VB	xcomp		
that	that	SCONJ	IN	mark		
they	-PRON-	PRON	PRP	nsubj		
were	be	AUX	VBD	ccomp		
perfectly	perfectly	ADV	RB	advmod		
normal	normal	ADJ	JJ	acomp		
,	,	PUNCT	,	punct		
thank	thank	VERB	VBP	ROOT		
you	-PRON-	PRON	PRP	doobj		
very	very	ADV	RB	advmod		
much	much	ADV	RB	advmod		
.	.	PUNCT	.	punct		
Mr.	Mr.	PROPN	NNP	ROOT		
.	.	PROPN	NNP	punct		
Dursley	Dursley	PROPN	NNP	nsubj		
was	be	AUX	VBD	ROOT		
the	the	DET	DT	det		
director	director	NOUN	NN	attr		
of	of	ADP	IN	prep		
a	a	DET	DT	det		
firm	firm	NOUN	NN	pobj		
called	call	VERB	VBN	acl		
Grunnings	Grunnings	PROPN	NNP	oprd		

Рисунок 4.4 – POS-тегування тексту

Після проведення тегування частин мови перейдемо до створення функції для автоматичного видалення непотрібної інформації, такої як стоп-слова та розділові знаки, з кожного речення.

Ми почнемо з імпорту необхідних нам англійських моделей з `sraCu`, а також рядкового модуля `Python`, який містить корисний список усіх розділових знаків, які ми можемо використовувати в `string.punctuation`. Ми створимо змінні, що містять розділові знаки та стоп-слова, які ми хочемо

видалити, та синтаксичний аналізатор, який запускає введення через модуль англійської мови spaCy. Потім ми створимо функцію, яка приймає речення як вхідні дані та обробляє речення в токени, виконуючи видалення стоп-слів. Як результат отримаємо ряд слів без вживання, наприклад, таких слів, як “the”, “a”, “an” тощо (рисунок 4.5).

```
Source
['Mr', 'and', 'Mrs', 'Dursley', 'of', 'number', 'four', 'Privet', 'Drive', 'were', 'proud', 'to', 'say',
'that', 'they', 'were', 'perfectly', 'normal', 'thank', 'you', 'very', 'much', 'Mr', 'Dursley', 'was',
'the', 'director', 'of', 'a', 'firm', 'called', 'Grunnings', 'which', 'made', 'drills', 'Nothing', 'like',
'this', 'man', 'had', 'ever', 'been', 'seen', 'on', 'Privet', 'Drive', 'He', 'was', 'tall', 'thin', 'and',
'very', 'old', 'judging', 'by', 'the', 'silver', 'of', 'his', 'hair', 'and', 'beard', 'which', 'were',
'both', 'long', 'enough', 'to', 'tuck', 'into', 'his', 'belt', 'His', 'blue', 'eyes', 'were', 'light',
'bright', 'and', 'sparkling', 'behind', 'half', 'moon', 'spectacles', 'and', 'his', 'nose', 'was',
'very', 'long', 'and', 'crooked', 'as', 'though', 'it', 'had', 'been', 'broken', 'at', 'least', 'twice',
'This', 'man', 's', 'name', 'was', 'Albus', 'Dumbledore', 'Albus', 'Dumbledore', 'didn', 't', 'seem',
to', 'realize', 'that', 'he', 'had', 'just', 'arrived', 'in', 'a', 'street', 'where', 'everything', 'from',
'his', 'name', 'to', 'his', 'boots', 'was', 'unwelcome', 'Potter', 'wasn', 't', 'such', 'an', 'unusual',
'name', 'He', 'was', 'sure', 'there', 'were', 'lots', 'of', 'people', 'called', 'Potter', 'who', 'had',
'a', 'son', 'called', 'Harry', 'True', 'I', 'haven', 't', 'introduced', 'meself', 'Rubeus', 'Hagrid',
'Keeper', 'of', 'Keys', 'and', 'Grounds', 'at', 'Hogwarts']

Without stop words
['Mr', 'Mrs', 'Dursley', 'number', 'Privet', 'Drive', 'proud', 'perfectly', 'normal', 'thank',
'Mr', 'Dursley', 'director', 'firm', 'called', 'Grunnings', 'drills', 'like', 'man', 'seen',
'Privet', 'Drive', 'tall', 'thin', 'old', 'judging', 'silver', 'hair', 'beard', 'long', 'tuck',
'belt', 'blue', 'eyes', 'light', 'bright', 'sparkling', 'half', 'moon', 'spectacles', 'nose', 'long',
'crooked', 'broken', 'twice', 'man', 's', 'Albus', 'Dumbledore', 'Albus', 'Dumbledore', 'didn', 't',
'realize', 'arrived', 'street', 'boots', 'unwelcome', 'Potter', 'wasn', 't', 'unusual', 'sure',
'lots', 'people', 'called', 'Potter', 'son', 'called', 'Harry', 'True', 'haven', 't', 'introduced',
'meself', 'Rubeus', 'Hagrid', 'Keeper', 'Keys', 'Grounds', 'Hogwarts']
```

Рисунок 4.5 – Результат видалення стоп-слів

Тепер спробуємо вирахувати матрицю частотності зустрічальності для всіх наших слів у тексті. Частина результату зображена на рисунку 4.6.

```
{'Mr': {'mr': 2}, 'Mrs': {'mr': 1}, 'Dursley': {'dursley': 2},
'number': {'number': 1}, 'Privet': {'privet': 2}, 'Drive': {'drive': 2},
'proud': {'proud': 1}, 'perfectly': {'perfectli': 1},
'normal': {'normal': 1}, 'thank': {'thank': 1},
'director': {'director': 1}, 'firm': {'firm': 1}, 'called': {'call': 1},
'Grunnings': {'grun': 1}, 'drills': {'drill': 1}, 'like': {'like': 1},
'man': {'man': 1}, 'seen': {'seen': 1}, 'tall': {'tall': 1},
'thin': {'thin': 1}, 'old': {'old': 1}, 'judging': {'judg': 1},
'this': {'this': 1}, 'on': {'on': 1}, 'and': {'and': 1},
'very': {'very': 1}, 'old': {'old': 1}, 'judging': {'judg': 1},
'both': {'both': 1}, 'long': {'long': 1}, 'enough': {'enough': 1},
'to': {'to': 1}, 'tuck': {'tuck': 1}, 'into': {'into': 1}, 'his': {'his': 1},
'belt': {'belt': 1}, 'His': {'his': 1}, 'blue': {'blue': 1}, 'eyes': {'eyes': 1},
'were': {'were': 1}, 'light': {'light': 1}, 'bright': {'bright': 1},
'and': {'and': 1}, 'sparkling': {'sparkling': 1}, 'behind': {'behind': 1},
'half': {'half': 1}, 'moon': {'moon': 1}, 'spectacles': {'spectacles': 1},
'and': {'and': 1}, 'his': {'his': 1}, 'nose': {'nose': 1}, 'was': {'was': 1},
'very': {'very': 1}, 'long': {'long': 1}, 'and': {'and': 1}, 'crooked': {'crooked': 1},
'as': {'as': 1}, 'though': {'though': 1}, 'it': {'it': 1}, 'had': {'had': 1},
'been': {'been': 1}, 'broken': {'broken': 1}, 'at': {'at': 1}, 'least': {'least': 1},
'twice': {'twice': 1}, 'This': {'this': 1}, 'man': {'man': 1}, 's': {'s': 1},
'name': {'name': 1}, 'was': {'was': 1}, 'Albus': {'albus': 1}, 'Dumbledore': {'dumbledore': 1},
'didn': {'didn': 1}, 't': {'t': 1}, 'seem': {'seem': 1}, 'to': {'to': 1},
'realize': {'realize': 1}, 'that': {'that': 1}, 'he': {'he': 1}, 'had': {'had': 1},
'just': {'just': 1}, 'arrived': {'arrived': 1}, 'in': {'in': 1}, 'a': {'a': 1},
'street': {'street': 1}, 'where': {'where': 1}, 'everything': {'everything': 1},
'from': {'from': 1}, 'his': {'his': 1}, 'name': {'name': 1}, 'to': {'to': 1},
'his': {'his': 1}, 'boots': {'boots': 1}, 'was': {'was': 1}, 'unwelcome': {'unwelcome': 1},
'Potter': {'potter': 1}, 'wasn': {'wasn': 1}, 't': {'t': 1}, 'such': {'such': 1},
'an': {'an': 1}, 'unusual': {'unusual': 1}, 'name': {'name': 1}, 'He': {'he': 1},
'was': {'was': 1}, 'sure': {'sure': 1}, 'there': {'there': 1}, 'were': {'were': 1},
'lots': {'lots': 1}, 'of': {'of': 1}, 'people': {'people': 1}, 'called': {'called': 1},
'Potter': {'potter': 1}, 'who': {'who': 1}, 'had': {'had': 1}, 'a': {'a': 1},
'son': {'son': 1}, 'called': {'called': 1}, 'Harry': {'harry': 1}, 'True': {'true': 1},
'I': {'i': 1}, 'haven': {'haven': 1}, 't': {'t': 1}, 'introduced': {'introduced': 1},
'meself': {'meself': 1}, 'Rubeus': {'rubeus': 1}, 'Hagrid': {'hagrid': 1},
'Keeper': {'keeper': 1}, 'of': {'of': 1}, 'Keys': {'keys': 1}, 'and': {'and': 1},
'Grounds': {'grounds': 1}, 'at': {'at': 1}, 'Hogwarts': {'hogwarts': 1}]
```

Рисунок 4.6 – Матриця частотності

Тож, тепер перейдемо до вирішення задачі NER за допомогою графів та стандартних методів бібліотек, які вже використовують методи нейронних мереж. Як уже говорилося раніше, сутність може складатися з декількох токенів, наприклад «Білий дім», тому необхідно навчити синтаксичні аналізатори враховувати ці особливості, тому що без урахування таких прикладів суб'єкти та об'єкти будуть визначатися лише як окремі слова. spaCy вже має вбудований швидкий статистичний метод NER з візуалізацією іменованих об'єктів та призначає мітки для токенів. За замовчуванням у spaCy вже є стандартні мітки – ім'я людей, локація, числа та номери, організації, об'єкти. Тож спробуємо отримати іменовані сутності за допомогою реалізованих нами методів (рисунок 4.7).

```
Verbs: ['say', 'thank', 'call', 'mal
Dursley      PERSON
number four  CARDINAL
Privet Drive GPE
Dursley      PERSON
Grunnings    ORG
Privet Drive WORK_OF_ART
half-moon    QUANTITY
Albus Dumbledore PERSON
Albus Dumbledore PERSON
Harry       PERSON
Rubeus Hagrid PERSON
Keeper of Keys ORG
Grounds     FAC
```

Рисунок 4.7 – Результат роботи NER

Спробуємо за допомогою візуалізації у тексті відобразити такий результат (рисунок 4.8).

Mr. and Mrs. **Dursley PERSON**, of **number four CARDINAL**, **Privet Drive GPE**, were proud to say that they were perfectly normal, thank you very much.

Mr. **Dursley PERSON** was the director of a firm called **Grunnings ORG**, which made drills.

Nothing like this man had ever been seen on **Privet Drive WORK\_OF\_ART**.

He was tall, thin, and very old, judging by the silver of his hair and beard, which were both long enough to tuck into his belt.

His blue eyes were light, bright, and sparkling behind **half-moon QUANTITY** spectacles and his nose was very long and crooked, as though it had been broken at least twice.

This man's name was **Albus Dumbledore PERSON**.

**Albus Dumbledore PERSON** didn't seem to realize that he had just arrived in a street where everything from his name to his boots was unwelcome.

Potter wasn't such an unusual name.

He was sure there were lots of people called Potter who had a son called **Harry PERSON**.

True, I haven't introduced meself.

**Rubeus Hagrid PERSON**, **Keeper of Keys ORG** and **Grounds FAC** at Hogwarts.

Рисунок 4.8 – Приклад роботи NER

Таке відображення вже виглядає краще для сприйняття. Тепер для необхідно зробити вилучення пар сутностей. Ми вже знаємо, що для побудови графа знань найважливішими є вузли та ребра між ними. У якості вузлів виступають сутності, присутні у реченнях вибірки, а ребра це відношення, що з'єднують ці об'єкти друг з другом. Тож, такі пари для нас є суб'єкт “person”, та об'єкт, наприклад, “Dursley”. За допомогою бібліотек `networkx` створюємо фрейми для таких пар та за допомогою `matplotlib` візуалізуємо це у вигляді графу. Спробуємо спочатку візуалізувати граф із суб'єктом “ORG” (рисунок 4.9).

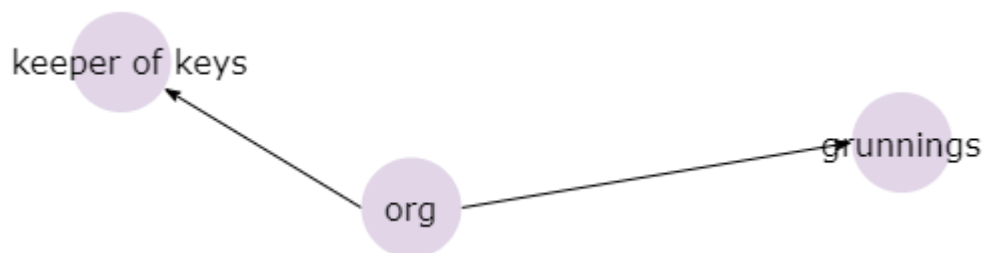


Рисунок 4.9 – Візуалізація графу

Тепер спробуємо візуалізувати графи для суб'єкту “person” та його об'єктів, та для суб'єкту “GPE” (рисунок 4.10, 4.11).

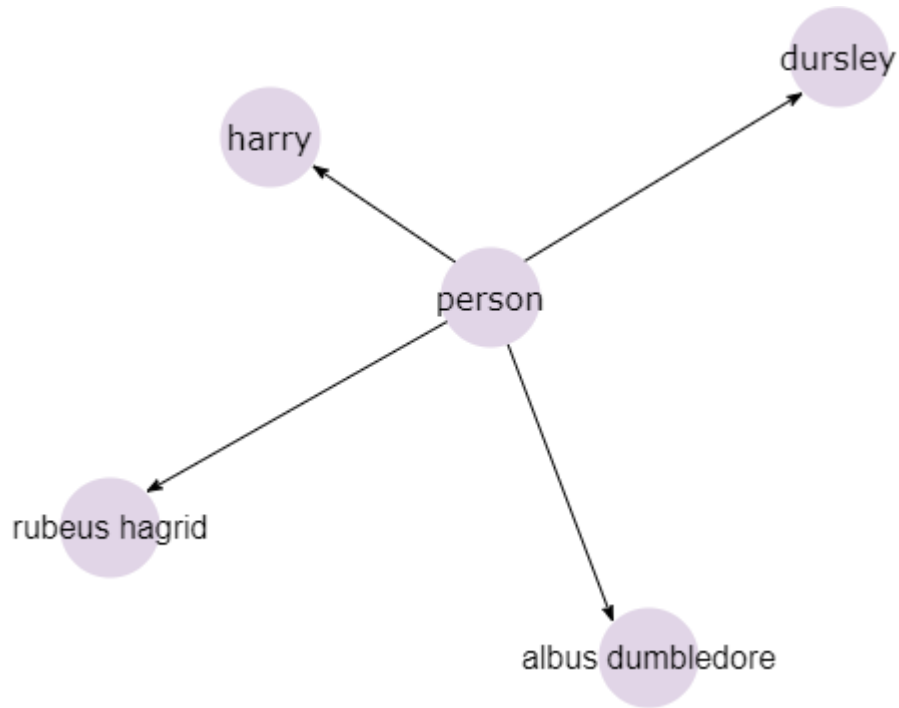


Рисунок 4.10 – Візуалізація графу

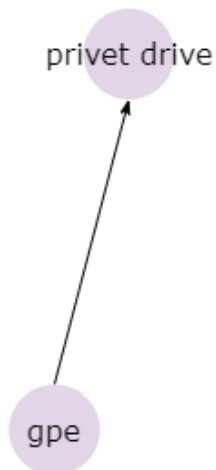


Рисунок 4.11 – Візуалізація графу

Тож, тепер спробуємо візуалізувати графі зі всіма суб'єктами та об'єктами, які класифікувала NER (рисунок 4.12).

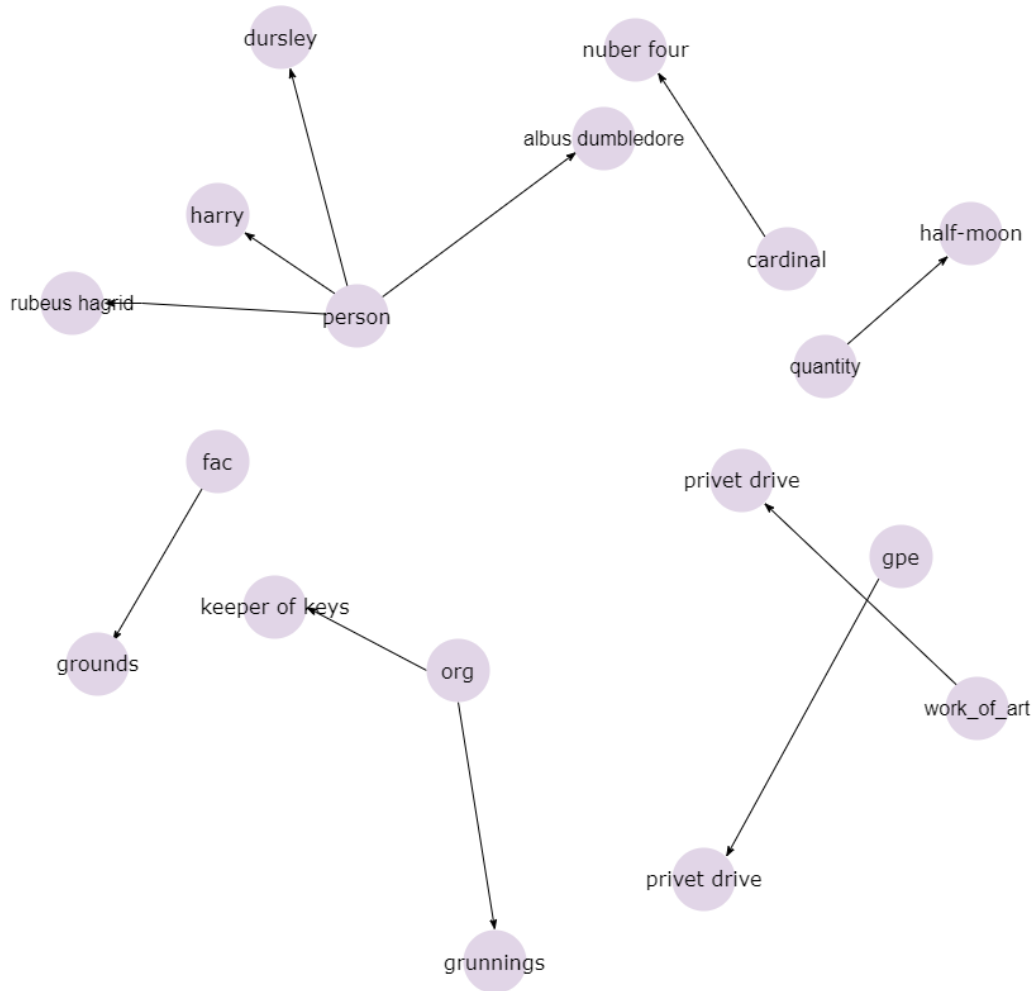


Рисунок 4.12 – Візуалізація графу зі всіма суб'єктами та об'єктами

Як ми бачимо за результатом, побудований граф відобразив усі суб'єкти та об'єкти, які ми мали на виході вирішення задачі NER. У деяких випадках ми отримали не дуже вірні результати, наприклад як із міткою “work\_of\_art” для назви вулиці, але такі методи зараз тільки починають стрімко розвиватись із шаленим темпом розвитку нашої мови, кількістю інформації у мережах інтернету.

## ВИСНОВКИ

В ході виконання даної кваліфікаційної роботи було проведено теоретичні дослідження в галузі обробки природномовних текстів. Була розглянута технічна література із різних джерел, на основі якої було отримано інформацію про актуальні завдання в цій галузі, вивчено різні методи та підходи до рішень.

Однією з найпопулярніших і найактуальніших завдань на сьогодні є завдання вилучення інформації (Information Extraction). Для цього напряду також були вивчені та описані в пояснювальній записці відомі підзадачі та підходи до вирішення. Однією з таких завдань є видобування іменованих сутностей (Named entity recognition), яка була взята як основа для подальшого теоретичного і практичного досліджень. NER є одним із базових завдань, без якого неможливо вирішити решту завдань вилучення інформації з тексту, наприклад завдання NED або NEL. Завдання NER було детально розглянуто з технічної та теоретичної точки зору – були визначені основні підходи до вирішення, основні переваги та недоліки, складності у реалізації. NER хоч і є основним завданням у напрямі ІЕ, через свої складності та особливості, в основному не використовується повсюдно та ізольовано від інших завдань, пов'язаних із вилученням інформації.

Основними підходами до вирішення не тільки NER, а й задач обробки текстів загалом є методи, засновані на нейронних мережах, а також графи знань. Були проведені теоретичні дослідження також у цих напрямках – вивчені та описані відомі архітектури, методи навчання, стадії розробки. Для реалізації завдання NER за допомогою нейронних мереж найбільше підходить архітектура рекурентних нейромереж (RNN), проте в силу своїх недоліків – вкрай невелика тривалість запам'ятовування попередньої інформації, складність у навчанні та проблема зникаючого градієнта – цю

архітектуру не використовують повсюдно для цих задач. Тому для вирішення NER за допомогою нейромереж було обрано та описано архітектуру мережі LSTM. Крім нейромережевого підходу, був також докладно описаний підхід графів знань. Даний підхід разом із реалізованими методами нейронних мереж і був обраний як програмне вирішення завдання вилучення іменованих сутностей.

Для програмної реалізації була обрана мова програмування Python, а також, через свої переваги, середовище розробки Google Colab. Для реалізації графів були також задіяні бібліотеки spaCy (для роботи з обробкою природної мови), nltk (для побудови графа знань) та matplotlib (для візуалізації графа). Також, у якості навчальної вибірки був обраний датасет з відкритого доступу, що містить близько 4300 речень зі статей Вікіпедії про фільми. Підсумковий граф знань створений із витягнутих сутностей (пар суб'єкт-об'єкт) та предикатів (зв'язок між сутностями), де вузли представляють об'єкти, а ребра чи з'єднання між вузлами становлять відносини між вузлами.

Як результат, була отримана візуалізація графа з усіма його сутностями та відносинами, що трохи ускладнює візуальне розуміння графа. Тому було прийнято рішення для візуалізації графа використовувати лише по кілька важливих зв'язків.

Вирішення задачі NER за допомогою як нейронних мереж, так і за допомогою графів є вкрай корисним у період активного розвитку напряму обробки природних мов, а також збільшенням кількості інформації на просторах інтернету, наприклад, для аналітиків для проведення статистичного аналізу за будь-якими даними текстового документа.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Understanding Evolution and Future of Natural Language Processing. URL: <https://www.xenonstack.com/blog/evolution-of-nlp> (дата звернення 20.04.2022).
2. F. Dieter, Ş. Umutcan, A. Kevin Knowledge Graphs Methodology, Tools and Selected Use Cases, 2020. P. 147.
3. Shaoxiong J., Shirui P., Cambria E. A Survey on Knowledge Graphs: Representation, Acquisition and Applications, 2020, P. 1-4.
4. Harmelen F., Lifschitz V., Porter B. Handbook of Knowledge Representation. The Netherlands, AE Amsterdam, Elsevier, 2008. P.1035.
5. Knowledge extraction from unstructured texts. URL: <https://medium.com/heuritech/knowledge-extraction-from-unstructured-texts-c279e3b8f92f> (дата звернення 20.04.2022).
6. Mayank K. Domain-Specific Knowledge Graph Construction / Kejriwal Mayank., 2017. P. 107.
7. Chen P., Cuzzocrea A., Kara O. Knowledge Graph and Semantic Computing, Knowledge Computing and Language Understanding, 2007. P. 142.
8. Challenges of Knowledge Graphs, From Strings to Things – An Introduction. URL: <https://medium.com/@sderymail/challenges-of-knowledge-graph-part-1-d9ffe9e35214> (дата звернення 25.04.2022).
9. Yadav V., Bethard S., A survey on recent advances in named entity recognition from deep learning models, 2018. Proc. 27th ICCL. P.2145-2158.
10. Knowledge Graph – A Powerful Data Science Technique to Mine Information from Text (with Python). URL: <https://www.analyticsvidhya.com/blog/2019/10/how-to-build-knowledge-graph-text-using-spacy/> (дата звернення 25.04.2022).

11. Sang E. F., De Meulder F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. Antwerpen: University of Antwerp, 2003. P.1-2.
12. Al-Moslmi T. et al. Named entity extraction for knowledge graphs: A literature overview, 2020. IEEE Access. T. 8. P.32862-32881.
13. Nadeau D., Sekine S. A survey of named entity recognition and classification, 2007. Lingvisticae Investigationes, vol. 30, no. 1. P.3-26.
14. Jithesh V., Sagayaraj M. J., Srinivasa K. G. LSTM recurrent neural networks for high resolution range profile based radar target classification, 2017. IEEE. P.1-6.
15. Ehrlinger L., Wöß W. Towards a Definition of Knowledge Graphs. Leipzig, Germany, Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems – SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16), 2016. P.5.
16. Gomez-Perez J. M., Denaux R., Garcia-Silva A. A Practical Guide to Hybrid Natural Language Processing. Combining Neural Models and Knowledge Graphs for NLP. Springer. 2020. P.281.
17. Рябова Н. В., Громак О. В. Дослідження та застосування графів знань та нейромережових моделей для обробки природномовних текстів.: тез. докл. 12-ї Міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління». Харків: ХНУРЕ, 2022. Том 2. С. 15.
18. Aoran L., Xinmeng W., Wenhuan W. A Survey of Relation Extraction of Knowledge Graphs. / Ed. S. Jensen, Cyrus S., Xiaochun Y. Chengdu, China, August 3, 2019. P.52-66.

19. Li J. et al. Knowledge Graph and Semantic Computing. Language, Knowledge, and Intelligence: Second China Conference, CCKS 2017, Hangzhou, China, 2019, P.160.
20. Kejriwal M. Domain-Specific Knowledge Graph Construction. Springer, 2019, P.12-15.
21. Zavala R., Martínez P., Segura-Bedmar I. A Hybrid Bi-LSTM-CRF model for Knowledge Recognition from eHealth documents, 2018. SEPLN. P.65-70.
22. Sagot B. A multilingual collection of CoNLL-U-compatible morphological lexicons.: Proceedings of the Eleventh International Conference on Language Resources and Evaluation, 2018. P.1861.
23. Bodyanskiy E., Ryabova N., Zolotukhin O. Multilayer Adaptive Fuzzy Probabilistic Neural Network In Classification Problems Of Text Documents – Radio Electronics, Computer Science, Control Volume: 1(32), 2015, P. 39-45.
24. Бодянский Е.В., Рябова Н.В., Золотухін О.В. Классификация текстовых документов с помощью нейронной сети встречного распространения с контролируемым обучением // Науково-технічний журнал «Біоніка інтелекту» – 2014. – № 1(82). – С. 3-7.
25. Shatalov O., Ryabova N. Named Entity Recognition Problem for Long Entities in English Texts, 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), 2021, Volume 1, pp. 76-79, doi: 10.1109/CSIT52700.2021.9648768.