Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.912

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ HR-інструмент для збору інформації про шукачів роботи за даними відкритих ресурсів ”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ – 30.00.00.000

Студент
ІПЗ-41 _____ /Ілля КУЗЬМЕНКО/

Науковий керівник
асист. _____ /Кирило КАДОМСЬКИЙ/

Консультант
з питань нормоконтролю
фахівець _____ /Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри
д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2021

випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії

професор, доктор техн. наук Андрій БОНДАРЧУК

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО
Зав. кафедри програмних систем і технологій
_____(Олексій БИЧКОВ)
(підпис) (прізвище та ініціали)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ

РОБОТУ СТУДЕНТУ

Кузьменку Іллі Євгенійовичу

1. Тема бакалаврської роботи “ HR-інструмент для збору інформації про шукачів роботи за даними відкритих ресурсів ” керівник проекту (роботи) Кадомський Кирило Костянтинович
затверджені наказом вищого навчального закладу від “ 11 ” листопада 2020 р. № 6

2. Строк подання студентом роботи 30 травня 2021 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій направлених на збір та обробку даних.

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)
1. Огляд методів і технологій
2. Алгоритм оцінки релевантності оголошень
3. Проектування
4. Реалізація кросплатформеного hr-інструменту для збору інформації про шукачів роботи за даними відкритих ресурсів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Приклад перетворення тексту у вектор (рис. 1.2, ст. 25)
2. Візуалізація косинусних відстаней (рис. 1.3, ст. 27)

3. Структура даних Pandas (рис. 4.5, ст. 41)
4. Приклад розташування слів у векторному просторі (рис. 4.6, ст. 44)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1 Огляд методів та технологій	Кадомський К.К.	07.03.2021	12.04.2021
Розділ 2 Алгоритм оцінки релевантності оголошень	Кадомський К.К.	13.04.2021	14.04.2021
Розділ 3 Проєктування	Кадомський К.К.	18.04.2021	23.04.2021
Розділ 4 Реалізація кросплатформеного hr-інструменту для збору інформації про шукачів роботи за даними відкритих ресурсів	Кадомський К.К.	25.04.2021	27.05.2021

7. **Дата видачі завдання** 09 жовтня 2020 р.

Керівник _____ (Кирило КАДОМСЬКИЙ)

Завдання прийняв до виконання _____ (Ілля Кузьменко)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	28.11.2020	виконано
2	Аналіз існуючих концепцій та алгоритмів	09.01.2021	виконано
3	Вивчення концепцій збору та обробки даних.	10.02.2021	виконано
4	Розробка алгоритмічної моделі.	20.03.2021	виконано
5	Опис розробленого алгоритму.	25.03.2021	виконано
6	Розробка та тестування програмного забезпечення.	05.04.2021	виконано
7	Оформлення пояснювальної записки та презентації	01.05.2021	виконано
8	Затвердження пояснювальної записки роботи завідувачем кафедри.	29.05.2021	виконано

Студент – бакалавр _____(Ілля КУЗЬМЕНКО)

Керівник роботи _____(Кирило КАДОМСЬКИЙ)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 57 с., 12 рис., 1 табл., 1 додат., 18 джерел.

Тема: HR-інструмент для збору інформації про шукачів роботи за даними відкритих ресурсів.

Об'єкт дослідження: процес збору та структуризації даних з відкритих ресурсів пошуку роботи, представлених у вигляді неструктурованих даних.

Мета роботи: розробка інструменту для спеціаліста HR відділу для оптимізації часу та якості пошуку претендентів на посаду.

Предмет дослідження: технологія вилучення даних з відкритих ресурсів, досліджуються методи вилучення, обробки та оцінці даних відповідно до запиту.

Результати дослідження:

Досліджено можливості застосування технологій веб-скрапінгу та обробки натуральної мови для вирішення задач обробки та впорядкування неструктурованих текстових даних. Запропоновано методи реалізації просунутого пошуку за неструктурованими даними та покращення роботи розроблених алгоритмів.

Висновок

В результаті досліджень було отримано алгоритм, що застосовується для збору даних про шукачів роботи з відкритих ресурсів, що дозволяє реалізувати їх подальшу обробку, структуризацію та виконання пошуку за зібраними даними.

АВТОМАТИЗОВАНИЙ ЗБІР ДАНИХ, ВЕБ-СКРАПІНГ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ІНТЕЛЕКТУАЛЬНИЙ ПОШУК, ОБРОБКА НЕСТРУКТУРОВАНИХ ДАНИХ, ОЦІНКА РЕЛЕВАНТНОСТІ, ПРОСУНУТИЙ ПОШУК.

SUMMARY

Final qualifying bachelor's thesis: 57 pp., 12 figs., 1 table, 1 append, 18 sources.

Topic: HR-tool for collecting information about employees on the basis of open resources.

Object of research: collection and structuring of data from open seek employment resources, presented in the form of unstructured data.

Purpose of research: development of a tool for the HR department specialist to optimize the time and quality of seek employment.

Subject of research: technology of data extraction from open resources, methods of data extraction, processing and evaluation according to the request are investigated.

Results of the research:

Possibilities of application of web scraping and natural language processing technologies for solving problems of processing and ordering of unstructured text data are investigated. Methods of realization of the advanced search on unstructured data and improvement of work of the developed algorithms are offered.

Conclusion

As a result of research, an algorithm was obtained that is used to collect data on employees seekers from open resources, which allows them to further process, structure and perform a search on the collected data.

AUTOMATED DATA COLLECTION, WEB-SCRAPING, SOFTWARE, INTELLECTUAL SEARCH, PROCESSING OF UNSTRUCTURED DATA, EVALUATION OF RELAYS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ОПМ – обробка природної мови;

NLP – Natural-language processing;

БЗ – база знань;

ІБД – інтегрована база даних;

ІІТ – інтелектуальні інформаційні технології;

ЕОД – електронний обмін даними;

НМ – нейронна мережа;

ООП – об’єктно орієнтовне програмування;

ПК – персональний комп’ютер;

РНМ – рекурентна нейронна мережа;

СРС – система розпізнавання символів;

ШІ – штучний інтелект;

ШНМ – штучні нейронні мережі.

ВСТУП

Створення обчислювальних машин дало початок для розвитку багатьох прикладних наук. Стрімкого росту зазнала інформатика, яка фактично виросла з класу теоретичних фундаментальних дисциплін та значно розширила практичні сфери свого застосування. Однією з перспективних галузей сучасної інформатики на сьогодні є нейроінформатика.

Нейроінформатика – це підрозділ інформатики, що відноситься до аналізу та переробки інформації, базується на використанні моделей штучного нейрона та побудові на їх основі нейронних мереж.

Розвиток штучних нейронних мереж тісно пов'язаний з біологією. Штучний нейрон – це спрощена модель біологічного нейрона. Нейронні мережі прекрасно підходять для автоматичного збору, обробки та уніфікації різноманітної інформації. За допомогою даної методології було прийнято рішення автоматизувати робочий процес спеціалістам з управління персоналом.

Спеціаліст з управління персоналом – людина публічна. Він розмовляє з претендентами при прийомі на роботу, допомагає керівництву у вирішенні кадрових проблем, спілкується зі співробітниками і втілює в життя корпоративну політику. Для пошуку робітників часто доводиться звертатися до різних ресурсів пошуку роботи, з різними правилами заповнення анкетної інформації. Подібні процеси займають досить багато робочого часу тому було б добре автоматизувати даний процес. Ідея в тому, щоб розробити певний інструмент, який збиратиме дані з різних відкритих ресурсів для пошуку роботи, приведе дані в зручний для перегляду та сприйняття форму в межах однієї платформи.

РОЗДІЛ 1
ОГЛЯД МЕТОДІВ І ТЕХНОЛОГІЙ

На сьогодні для виконання задач активно розробляються методи та технології, які допомагають реалізовувати додатки на високому рівні програмування, не вкладаючи великих інвестицій в розробку. Для виконання задачі збору даних варто використати метод вилучення даних з веб ресурсів — веб-скрапінг.

1.1 Веб-скрапінг

Веб-скрапінг — це метод цілеспрямованого автоматизованого вилучення інформації з веб-сайтів. Дозволяє отримувати нетабличні або погано структуровані дані з веб-сайтів і перетворювати їх у придатний для використання структурований формат, такий як файл csv або база даних.

На відміну від інших видів скрапінгу, веб-скрапінг отримує базовий HTML-код, а разом із ним і дані. Основний принцип роботи полягає в тому, що: автоматизований код виконує GET-запити на сервер, а відповідь, HTML-документ, аналізує та шукає потрібну інформацію та перетворює її в необхідний формат.

Для реалізації HR-інструменту веб-скрапінг дозволяє організувати автоматизований збір даних про шукачів роботи.

1.1.1 Використання та сфера застосування

Застосування веб-скрапінгу досить обширно, для прикладу розглянемо деякі з них:

- відстеження тенденцій на ринку шляхом вилучення даних з інтернет-магазинів;
- збір даних для маркетингових досліджень;
- створення баз даних на основі відкритих ресурсів;
- інтеграція даних з декількох джерел;
- моніторинг інформаційних погоди;

- збір урядових даних;
- збір коментарів та інших дискурсів для аналізу;
- тощо.

Також веб-скрапінг використовується в неправомірних цілях, включаючи шахрайство, хакерські атаки та крадіжку вмісту захищеного авторським правом. Веб-скрапери часто імітують звичайну поведінку користувачів через що виникають складнощі з їх блокуванням та можуть без особливих зусиль заволодіти будь-якими даними. Веб-ресурс, на який націлений скрепер, може зазнати фінансових збитків, особливо якщо це бізнес, який покладається на моделі конкурентних цін або угоди з розподілом контенту.

1.1.2 Техніки виконання

Веб-скрапінг — це процес автоматичного видобування даних або збору інформації із інтернету. Сучасні рішення для вишкрібання веб-сторінок варіюються від спеціальних, що вимагають людських зусиль, до повністю автоматизованих систем, які здатні перетворювати цілі веб-сайти в структуровану інформацію.

Людське копіювання та вставка

Найпростіша форма веб-скрапінгу — це ручне копіювання та вставка даних з веб-сторінк в текстовий файл або електронну таблицю. Іноді навіть найкращі технології веб-скрапінгу не може замінити ручну роботу людини, а саме копіювання та вставка, іноді це може бути єдиним дієвим рішенням, коли веб-сайти для скрапінгу прямо встановлюють захист для збереження даних від автоматичного збору даних.

Регулярні вирази

Простий, але потужний підхід до вилучення інформації з веб-сторінок може базуватися на команді UNIX grep або засобах роботи з регулярними виразами мов програмування (наприклад, Perl або Python).

Програмування HTTP

Статичні та динамічні веб-сторінки можна отримати, розмістивши HTTP-запити на віддаленому веб-сервері за допомогою програмування сокетів.

Розбір HTML сторінок

Багато веб-сайтів мають великі колекції сторінок, які динамічно генеруються з базового структурованого джерела, такого як база даних. Дані тієї самої категорії зазвичай кодуються за шаблоном. При добуванні даних програма, яка виявляє такі шаблони в певному джерелі інформації, витягує його вміст і переводить у реляційну форму, називається обгорткою.Алгоритми генерації обгортки припускають, що вхідні сторінки системи відповідають загальному шаблону і їх можна легко ідентифікувати з точки зору загальної схеми URL. Більше того, деякі напівструктуровані мови запитів даних , такі як XQuery та HTQL, можуть використовуватися для синтаксичного аналізу HTML-сторінок і отримання та заміни вмісту сторінки.

Розбір DOM дерева

Використовуючи повноцінний веб-браузер, такий як Google Chrome або елемент керування браузера Mozilla Firefox, програми можуть отримувати динамічний вміст, генерований клієнтськими сценаріями. Ці елементи керування браузером також аналізують веб-сторінки в дереві DOM, на основі яких програми можуть отримувати частини сторінок.

Вертикальна агрегація

Є кілька компаній, які розробили специфічні вертикально-специфічні платформи. Ці платформи створюють і відстежують безліч «ботів» для конкретних цілей без «людини у циклі» (без участі людини). Підготовка передбачає створення бази знань для всієї вертикалі, а потім платформа автоматично створює ботів для аналізу інтернет ресурсів. Надійність платформи вимірюється якістю інформації, яку вона отримує (зазвичай кількість полів) та її масштабованістю (наскільки швидко вона може масштабувати до сотень або тисяч сайтів).

Розпізнавання семантичних анотацій

Сторінки, що обробляються, можуть охоплювати метадані або семантичні розмітки та анотації, які можна використовувати для пошуку конкретних фрагментів даних. Якщо анотації вбудовані на сторінки, як це робить Microformat , цей прийом можна розглядати як особливий випадок аналізу DOM дерева. В іншому випадку анотації, організовані в семантичний шар , зберігаються та управляються окремо від веб-сторінок.

Аналіз веб-сторінок за допомогою комп'ютерного зору

Зусилля використовують машинне навчання та комп’ютерний зір, які намагаються ідентифікувати та витягти інформацію з веб-сторінок, інтерпретуючи сторінки візуально, як це може зробити людина.

1.1.3 Синтаксичний аналізатор Java

Jsoup — це бібліотека Java для роботи з реальним HTML. Він забезпечує дуже зручний API для отримання URL-адрес та вилучення та обробки даних, використовуючи методи DOM HTML5 та селектори CSS.

Jsoup реалізує специфікацію WHATWG HTML5 і аналізує HTML до того самого DOM, що і сучасні браузери.

За допомогою даної технології можна:

- збирати та аналізувати HTML з URL-адреси, файлу або рядка;
- знаходити та витягувати дані, використовуючи обхід по DOM дереву або CSS селектори;
- маніпулювати з елементами HTML, атрибутами та текстом;
- очищувати вміст, поданий користувачем із білого списку, щоб запобігти атакам XSS;
- вивести охайний HTML код.

Jsoup — це проект з відкритим кодом, який поширюється за ліцензією MIT . Вихідний код доступний на GitHub .

Висновок: для виконання задачі знадобляться техніки веб-скрапінгу, такі як розбір дерева DOM та розбір HTML сторінок, які робляться за допомогою отримання вмісту відправляючи запит GET. Перевагами такої технології є її ефективність, якість виконання збору даних, та відносна простота реалізації даних методів. Недоліком такого підходу є захищеність на отримання певних даних користувача, шляхом приховування даних від неавторизованих веб-користувачів, таким чином веб-скрапер не зможе отримати певні дані. Також, є проблема з отриманням певних сторінок, які захищені від копіювання HTML-коду або в HTML-сторінці використовують підміну тегів. Для вирішення цих проблем, необхідно реалізовувати додатковий скрапер, який буде

оцінювати веб-сторінку за допомогою комп’ютерного зору, а після аналізувати та визначати дані.

1.2 Обробка природної мови

Для реалізації застосунку необхідно буде реалізувати структуризацію даних та просунутий пошук для цього знадобляться техніки обробки текстів та природної мови.

Обробка природної мови (англ. Natural-language processing, NLP) — це один із напрямків штучного інтелекту та лінгвістики де програмне забезпечення обробляє природну мову. Тут є багато додатків, таких як аналіз тональності тексту, переклад мови, виявлення плагіату, фальшивих новин, граматичних помилок тощо. Аналіз природної мови використовує алгоритми обчислювальної лінгвістики та теорії штучного інтелекту, вирішує проблеми обробки взаємодії людських комунікативних актів та обчислювальних систем.

1.2.1 Використання та сфера застосування

Зазвичай люди не замислюються про хитросплетіння власних мов. Це інтуїтивна поведінка, яка використовується для передачі інформації та значення за допомогою семантичних сигналів, таких як слова, знаки чи жести. Відомо, що мову легше вивчати в підлітковому віці, оскільки це повторювана, інтуїтивна та навчена поведінка - подібно до ходьби.

Однак, комп’ютерам надзвичайно складно обробляти велику кількість даних з неструктурованими даними. Ось чому машинне навчання та штучний інтелект привертають стільки уваги. І в міру того, як ШІ стає все більш досконалим, зростатиме і обробка природних мов. Ось кілька яскравих прикладів використання даних технологій.

Фільтри електронної пошти

Фільтри електронної пошти — це одне з найпростіших додатків NLP онлайн. Почалося з фільтрів спаму, розкриття певних слів або фраз, що сигналізують про спам-повідомлення. Але фільтрація покращилася, як і ранні адаптації ОПМ.

Одне з найпоширеніших, новіших додатків NLP міститься в класифікації електронної пошти Gmail. Система розпізнає, чи електронні листи належать до однієї з трьох категорій (основна, соціальна чи рекламна) залежно від їх змісту.

Розумні помічники

Розумні помічники, такі як Siri від Apple, розпізнають закономірності в мові завдяки розпізнаванню голосу, а потім виводять значення та надають корисну відповідь. Ми звикли до того, що можемо сказати "Гей, Сірі", задати питання, і вона розуміє, що ми сказали, і відповідає відповідними відповідями на основі контексту. І ми звикаємо бачити, як Siri у нашому домі та повсякденному житті, коли ми керуємо такими елементами, як термостат, вимикачі світла, автомобілями, побутовими пристроями тощо.

Результати пошуку

Пошукові системи використовують NLP для виведення відповідних результатів на основі подібної поведінки пошуку або намірів користувача, тому пересічна людина знаходить те, що їй потрібно, не будучи майстром пошукових термінів.

Наприклад, Google не тільки передбачає, які популярні пошукові запити можуть застосовуватися до вашого запиту, коли ви починаєте вводити текст, але він розглядає всю картину та розпізнає те, що ви намагаєтесь сказати, а не тільки точні пошукові слова.

Інтелектуальні тексти

Такі речі, як автовиправлення, автозаповнення та передбачення тексту, настільки звичні для наших смартфонів, що ми сприймаємо їх як належне. Автозаповнення та інтелектуальний текст схожі на пошукові системи тим, що вони передбачають речі, щоб сказати, виходячи з того, що ви вводите, закінчуючи слово або речення. А автовиправлення іноді навіть змінює слова, щоб загальне повідомлення мало більше сенсу. Інтелектуальний текст адаптується до наших особистих мовних примх, чим довше ми його використовуємо.

Мовний переклад

За допомогою NLP онлайн-перекладачі можуть точніше перекладати мови та представляти граматично правильні результати. Це нескінченно корисно при спробі спілкуватися з кимось іншою мовою. Мало того, але при перекладі з іншої мови на вашу власну зараз інструменти розпізнають мову на основі введеного тексту та перекладають її.

Цифрові телефонні дзвінки

Іноді спілкуючись з представниками розвинених компаній, які хочуть налагодити роботу з клієнтами за допомогою інтелектуальної телефонії ми можемо чути: "цей дзвінок може бути записаний для навчальних цілей", але рідко ми замислюємось, навіщо це. Ці записи можуть бути використані для навчальних цілей. Автоматизовані системи направляють дзвінки клієнтів до представника служби або онлайн-ботів, які відповідають на запити клієнтів корисною інформацією. Це практика NLP, яку застосували багато компаній, у тому числі великі провайдери телекомунікацій.

Аналітика тексту та аналіз даних

За допомогою аналізу тексту неструктуровані текстові дані перетворюються у значущі дані для аналізу з використанням різних лінгвістичних, статистичних та машинних методів навчання.

Аналіз настроїв є корисним інструментом для брендів - особливо якщо у них велика клієнтська база. Аналіз цих взаємодій може допомогти брендам визначити, наскільки добре працює маркетингова кампанія, або відстежувати тенденції, що стосуються споживачів, перш ніж вони вирішать, як реагувати або покращувати сервіс для кращого досвіду роботи з клієнтами.

Додатковими способами, за допомогою яких NLP допомагає в аналізі тексту, є вилучення ключових слів та пошук структури або шаблонів у неструктурованих текстових даних.

У цифровому світі існує величезна кількість застосувань NLP, і цей список буде збільшуватись у міру того, як бізнес та галузі сприймуть і побачать його значення.

1.2.2 Попередня обробка тексту

Неопрацьовані дані містять числове значення, розділові знаки, спеціальні символи тощо. Ці значення можуть перешкоджати роботі моделі, тому спочатку нам потрібно перетворити необроблені дані у значущі дані, які також називаються попередньою обробкою тексту. Це можна зробити наступними способами.

Видалення інформаційного шуму

Зазвичай інформаційний шум можна визначити як верхній колонтитул, нижній колонтитул, HTML, XML, дані розмітки. Оскільки дані цього типу не мають значення і не надають жодної інформації, тому видаляти цей тип шумних даних обов'язково. Дані розмітки HTML, XML може бути видалений бібліотекою BeautifulSoup (Python) або Jsoup (Java).

Обробка текстових даних

Вхідним матеріалом для обробки природної мови є тексти. Збір даних для цього відбувається з різних джерел. Перш ніж дані можна буде використовувати для аналізу їх необхідно обробити та очистити інакше можна втратити важливу інформацію в ході аналізу.

Найбільш поширені методи обробки даних у ОПМ:

Токенізація

Токенізація розбиває необроблений текст на елементарні одиниці, що називаються лексемами тобто токени. Ці маркери допомагають зрозуміти контекст або розробити модель для ОПМ. Токенізація допомагає інтерпретувати значення тексту, аналізуючи послідовність слів та дає змогу працювати зі словом як з окремою сутністю, при цьому знаючи його контекст.

Існують різні методи та бібліотеки для токенізації. NLTK, Gensim, Keras — це деякі бібліотеки, які можна використовувати. Але дані бібліотеки можуть не підтримувати роботу з деякими мовами, в такому випадку потрібно проводити адаптацію алгоритму для стемінгу.

Токенізація може бути виконана як для окремих слів та словоформ, так і для речень. Якщо текст розділено на слова за допомогою певної техніки, це називається токенізацією слів, а розділення зроблене для речень, називається токенізацією речень.

Доступні різні техніки токенізації, які можуть бути застосовані залежно від мови та цілей для яких розробляється модель. Нижче наведено декілька методів токенізації, що використовуються в ОПМ:

- Токенізація пропусків
- Словникова токенізація
- Токенізація на основі правил
- Токенізація регулярними виразами
- Токенізація Penn TreeBank (семантична та синтаксична)
- Токенізація SpaCy
- Токенізація Moses

Видалення стоп-слів

Стоп-слова — це слова в тексті, які не додають ніякого значення реченню і їх вилучення не вплине на зміст тексту. Вони вилучаються зі словника для зменшення статистичної похибки при обробці та зменшення обсягу даних.

Стемінг

Стемінг відноситься до грубого евристичного процесу, який відрубує кінці слів в надії досягти цієї мети правильно більшу частину часу і часто причетний до видалення деривaційних й формотворчих афіксів.

Наприклад слова «хата, хатина, хатинка» в результаті роботи будуть приведені до форми «хат», в той же час слова «стіл, стільчик, стілець» до форми «стіл», що є прикладом видалення деривaційних афіксів.

Нормалізація

Коли ми нормалізуємо природний мовний ресурс, ми намагаємось зменшити кількість випадкових слів у ньому, наближаючи його до певного стандарту — звичайного розподілу. Це допомагає зменшити кількість зайвої інформації, з якою повинен працювати комп'ютер, отже, покращує ефективність.

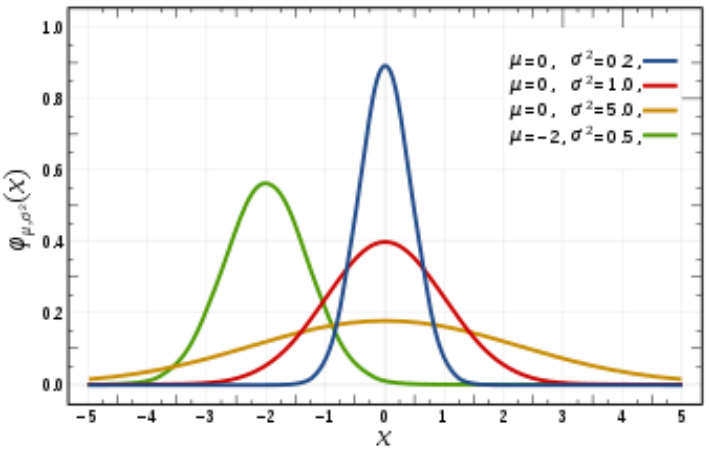

Рис.1.1 Приклад нормального розподілу

Лематизація

Лематизація, як правило, стосується правильних дій із використанням словникового запасу та морфологічного аналізу слів, як правило, метою видалення лише

флективних закінчень та повернення основи або словникової форми слова, яка відома як лема.

Класифікація частин мови

Розбір текстів за допомогою класифікації на частини мов поділяють слова за певними ознаками. Такі групи називають розрядами частин мови або лексико-граматичними розрядами. Інформація яку несе граматичне значення слова може бути важливою для навчання моделей та вирішення певних задач. Поділ на частини мови відбувається за основними принципами: смисловий, морфологічний, синтаксичний, словотворчий.

Та інші

1.2.3 Векторизація текстових даних

Машинне навчання використовує числові атрибути кожне з яких може виконувати деякі математичні операції, такі як множення матриць, скалярний добуток тощо. Тому для того щоб обчислювальні пристрої могли обробляти текстові данні їх потрібно привести що числового вигляду.

Для перетворення текстових даних у числові можна використовувати наступні методи:

Bag of words

Це базова модель, яка використовується в обробці природної мови. Називається торбою слів, оскільки будь-який порядок слів у документі відкидається, дана техніка лише повідомляє нам, чи слово в документі присутнє чи ні. Тут кожне речення є окремим документом, якщо ми складаємо список слів таким чином, що одне слово має зустрічатися лише один раз.

Розглянутий підхід використовує поняття N-грам. В свою чергу N-грами використовують в обробці природної мови для потреб передбачення на основі ймовірнісних моделей.

Таким чином, процес перетворення тексту у вектор називається векторизацією. За допомогою мови Python та функції CountVectorizer ми можемо перетворити текстовий документ в розріджену матрицю, більшою частиною якої є нулі, підрахунку слів типу NumPy.int64.

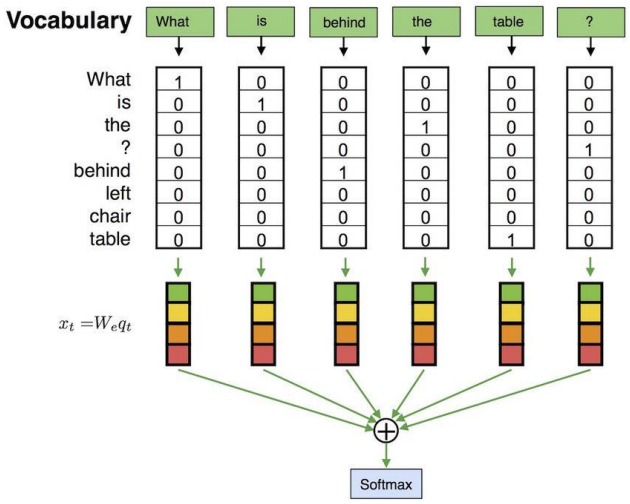

Рис. 1.2 Приклад перетворення тексту у вектор

TF-IDF

TF-IDF використовується для зважування слів відносно документа. Термін англійською розшифровується «Term Frequency — Inverse Document Frequency».

TF (частота термінів) — обчислює частоту слів у кожному документі корпусу. Частоту термінів для кожного документа можна розрахувати за формулою (1.1).

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \quad (1.1)$$

IDF (обернена частота даних) — використовується для обчислення ваги рідкісних слів у всіх документах корпусу. Слова, які рідко трапляються в корпусі, мають високий бал IDF. Формула IDF

$$idf(w) = \log(\frac{N}{df_i}) \quad (1.2)$$

Поєднуючи ці два, ми отримуємо оцінку TF-IDF (***ω***) для слова в документі корпусу (1.3).

$$\omega_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}) \quad (1.3)$$

Принцип роботи нескладно пояснити на прикладі в якому кожне з речень буде виступати окремим документом у якому, для прикладу, за змінну А виступатиме речення «Короп живе в річці», а змінну В – «Камбала живе в морі»:

Таблиця 1.1

Терміни	TF		IDF	TF*IDF	
	A	B		A	B
Короп	¼	0	log(2/1) = 0.3	0.043	0
Камбала	0	¼	log(2/1) = 0.3	0	0.043
Живе	¼	¼	log(2/2) = 0	0	0
В	¼	¼	log(2/2) = 0	0	0
Річці	¼	0	log(2/1) = 0.3	0.043	0
Морі	0	¼	log(2/1) = 0.3	0	0.043

З даної таблиці видно, що TF-IDF загальних слів дорівнює нулю, що світчить про незначущість даних слів. З іншого боку слова «Короп», «Камбала», «Річці», «Морі» є ненульовими і мають більше значення.

Косинус подібності

Перш ніж розуміти Word2Vec, визначимо спочатку, що таке косинус подібності, оскільки Word2Vec для пошуку найбільш близького за сенсом слова використовує косинус подібності. Подібність косинусів не лише говорить про схожість двох векторів, але й перевіряє ортогональність вектора. Подібність косинусів представлена формулами (1.4):

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} \qquad (1.4)$$

Де θ (тета) — це кут між двома векторами, якщо кут близький до нуля, можна сказати, що вектори дуже схожі один на одного, а якщо тета дорівнює 90°, можна сказати, що вектори ортогональні один одному (ортогональний вектор, не пов'язаний один з одним) і, якщо тета дорівнює 180°, можна сказати, що обидва вектори протилежні один одному, як показано на малюнку.

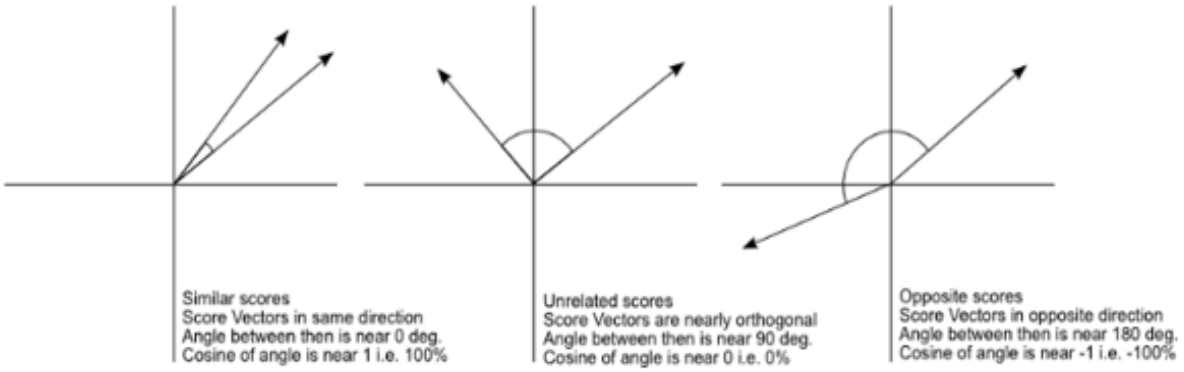

Рис.1.3 Візуалізація косинусної відстані

Подібність косинусів не тільки виявляє схожість між вектором, але також ігнорує підрахунок частоти слів. Отже, часто вживані слова впливатимуть на результат моделі, кут між обома словами буде хибно близьким.

Google Word2Vec

Це техніка глибокого навчання з двошаровою нейронною мережею. Google Word2Vec збирає вхідні дані з великих дата сетів (розглянемо на прикладі даних від Google) і перетворює у векторний простір. Google Word2Vec в основному попередньо навчається на наборі даних Google.

Word2Vec в основному розміщує слово у об'єктному просторі таким чином, що їх розташування визначається їх значенням, тобто слова, що мають подібне значення, об'єднуються в кластери і відстань між двома словами також має однакове значення. Розглянемо приклад, наведений нижче:

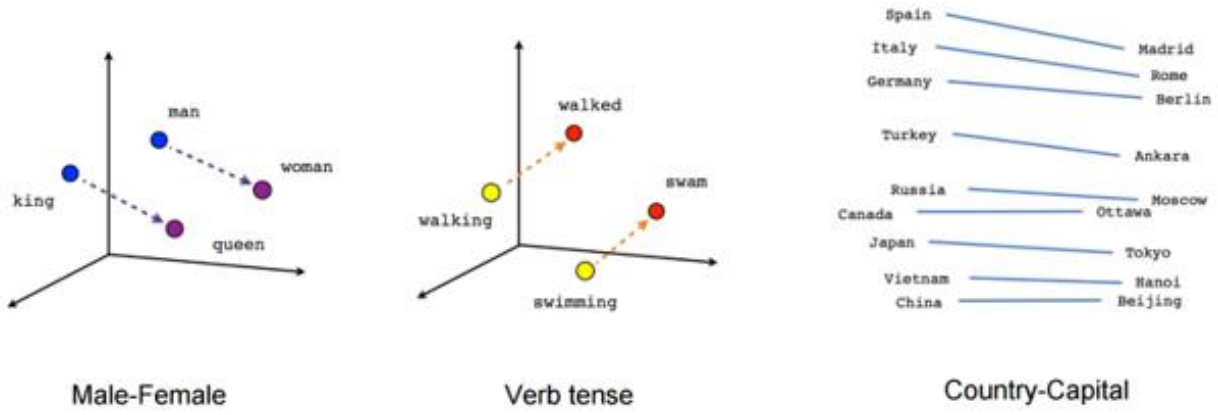

Рис.1.4 Особливості представлення об'єктів на векторній площині

На рисунку, на графіку «Male-Female», ми спостерігаємо, що відстань між «чоловіком» і «жінкою» така ж, як відстань між «королем» (чоловіком) і «королевою» (жінкою). Відстань між «королевою» та «жінкою» і відстань між «королем» та «чоловіком» однакові (король та чоловік, королева та жінка представляють однакове порівняння статі, отже вони повинні бути рівновіддалені).

Word2Vec не лише відображає значущі слова, але також відображає граматику, яку можна побачити на графіку «Verb tense». На цьому графіку дієслова «ходьба» і «плавання» та час дієслова є рівною відстанню один від одного.

Так само він може відобразити країну з її столицею, як показано на графіку «Country-Capital».

Висновок: для виконання задачі розумного пошуку знадобиться використовувати векторні представлення слів, для створення синонімічних форм та знаходження близьких за значенням слів, щоб отримати відсортований список з необхідними оголошеннями.

В свою чергу для знаходження контексту, вилучення досвіду роботи, освіти та власних умінь, з тексту резюме, також знадобиться користуватися регулярними виразами та векторними представленнями слів.

РОЗДІЛ 2
АЛГОРИТМ ОЦІНКИ РЕЛЕВАНТНОСТІ ОГОЛОШЕНЬ

Для вирішення задачі необхідно визначати, які оголошення найкраще відповідатимуть запиту. Задача оцінки оголошення, полягає в тому, щоб визначити чи має людина досвід роботи, освіту або певні власні якості, які передає в запиті шукач. Для вирішення цієї задачі, спершу необхідно вирішити, які данні підлягають оцінці. В даному випадку це текст оголошення в якому йдеться про досвід, освіту, особливі досягнення та власні якості. В першу чергу, необхідно обробити дані для досягнення кращого результату.

2.1 Обробка даних

Для виконання оцінювання, потрібно лематизувати дані безпосередньо оголошення. Таким чином отримуємо словоформи, з якими простіше працювати завдяки тому, що слова з різними закінченнями, префіксами але з однаковим сенсом, споріднені за значенням набувають початкової форми.

Також необхідно очистити дані від стоп-слів, які не впливають на роботу алгоритму, такі як: сполучник, прийменник, частка. Для ще більшої ефективності роботи алгоритму, крім самостійних частин мови, можна позбутися від слів кількість згадок про яких завищена і вони так само втрачають зміст для роботи алгоритму.

Наступним кроком буде обробка запиту, його також треба привести в початкову форму для налагодження виконання програми. Для більшої ефективності потрібно отримати близькі за значенням слова, яким надати певну вагу, яка буде використана для присвоєння оцінки релевантності для оголошення.

Даний алгоритм є лінійний і продемонстрований на блок-схемі на рисунку 2.1 нижче.

Рис. 2.1 Блок-схема алгоритму очистки та обробки даних.

2.2 Оцінка відповідності даних

Виконавши обробку даних, можна перейти до задачі пошуку за отриманими даними за допомогою синонімічних формам відповідно до спаду значення кута косинусної відстані між словами.

Задача пошуку, виконується завдяки синонімізації текстових даних в пошуковому запиті. В свою чергу синонімізація реалізовується з використанням моделі Word2Vec, яка містить векторні представлення слів та описує відношення між ними, завдяки чому

можна знайти пов’язані та протилежні за значенням слова. В реалізованому програмному рішенні пошуковий алгоритм надає найвищі бали словам з пошукового запиту, а подібним за значенням — менші, які розташовуються в порядку спадання коефіцієнтів залежності використаної векторної моделі. Завершальним етапом пошуку є порівняння обробленого запиту з зібраними даними.

Висновок: даний алгоритм дозволяє виконати оцінку важливості резюме шукача роботи, до запиту шляхом очищення даних, отримання слів схожих за значенням, та присвоєння їм коефіцієнтів важливості.

Недоліком такого рішення є те, що обробка даних виконується недостатньо швидко, це можна вирішити за допомогою векторного представлення тексту, це має зекономити час на обчислення і, відповідно, на сам пошук. Також недоліком є те, що не враховується контекст слів і на оцінку може впливати використання їх в іншому контексті, який не відповідає вимогам. Дану проблему також можна вирішити за допомогою семантичного аналізу тексту. Таким чином можна спершу отримати дані, які відповідають контексту, а вже потім проводити оцінювання.

РОЗДІЛ 3
ПРОЄКТУВАННЯ

Для реалізації інструменту, в першу чергу необхідно спроектувати виконання та реалізацію програми, визначитися з архітектурними рішеннями і обрати фреймворки, бібліотеки які знадобляться для вирішення поставленої задачі.

3.1 Проєктування та архітектурні рішення реалізації веб-скраперу

Першою частиною збір даних з відкритих ресурсів пошуку роботи. Так як мова йде про збір даних в великих об’ємах, то добре було б автоматизувати процес вилучення вмісту пошукових сайтів робітників та роботодавців. Для цього нам потрібно реалізувати веб-скрапер, який зробить цю роботу якісно та за короткий проміжок часу, чого б не змогла зробити людина. Для реалізації я вирішив використати переваги мови Java для того щоб зробити застосунок максимально продуктивним та гнучким для розширення завдяки парадигмам ООП. За допомогою бібліотеки Jsoup, яка допоможе реалізувати парсинг HTML та розбору DOM дерева. Останнім кроком реалізації даної частини буде збереження даних в БД для подальшого аналізу, обробки та використання.

3.2 Проєктування та архітектурні рішення обробки та аналізу тексту

Другою частиною реалізації буде аналіз зібраної інформації, для цього було прийнято рішення використовувати можливості інтерпретованої мови Python з очевидних причин, а саме: наявність зручних інструментів для аналізу та обробки даних, простота та швидкість написання коду. В першу чергу для обробки даних їх необхідно передати в зручний для обробки об’єкт. Pandas DataFrame оптимально підходить для

виконання задачі, даний об’єкт наділений можливостями індексування та маніпулювання даними. Таким чином є можливість виконувати роботу з великими наборами даних за допомогою декількох рядків коду, не використовуючи складні конструкції з циклами та з безліччю перевірок. Також для аналізу використано техніки стемінгу та векторизації текстових даних за допомогою бібліотеки Gensim та натренованих моделей векторного представлення Word2Vec та Glove, для реалізації пошуку контексту і знаходження та оцінки досвіду роботи.

3.3 Проєктування та архітектурні рішення для представлення даних

Останнім кроком буде адаптація та представлення даних, яка буде реалізована за допомогою чат-ботів, таким чином буде вирішена проблема мультиплатформності. В даному випадку обрано пакет PyTelegrsmBot який допомагає реалізувати, чат бот для Telegram месенджера.

На основі зібраної інформації щодо архітектурних рішень побудовано діаграму класів, яка описує статичне представлення структури моделі та відображає декларативні елементи програмного рішення. В діаграмі основним класом вилучення даних виступає клас Base Scrapper, наслідуваний класами JobsUA Scrapper та WorkUA Scrapper. Клас PostgreSQL відповідає за збереження та читання даних з БД. Data preprocessor — обробляє дані для подальшого аналізу та обробки тексту. Lang Tool — пероводить операції з обробки текстових даних та зберігає її. Клас Telebot в свою чергу відповідає за керування чат-ботом на платформі месенджера Telegram.

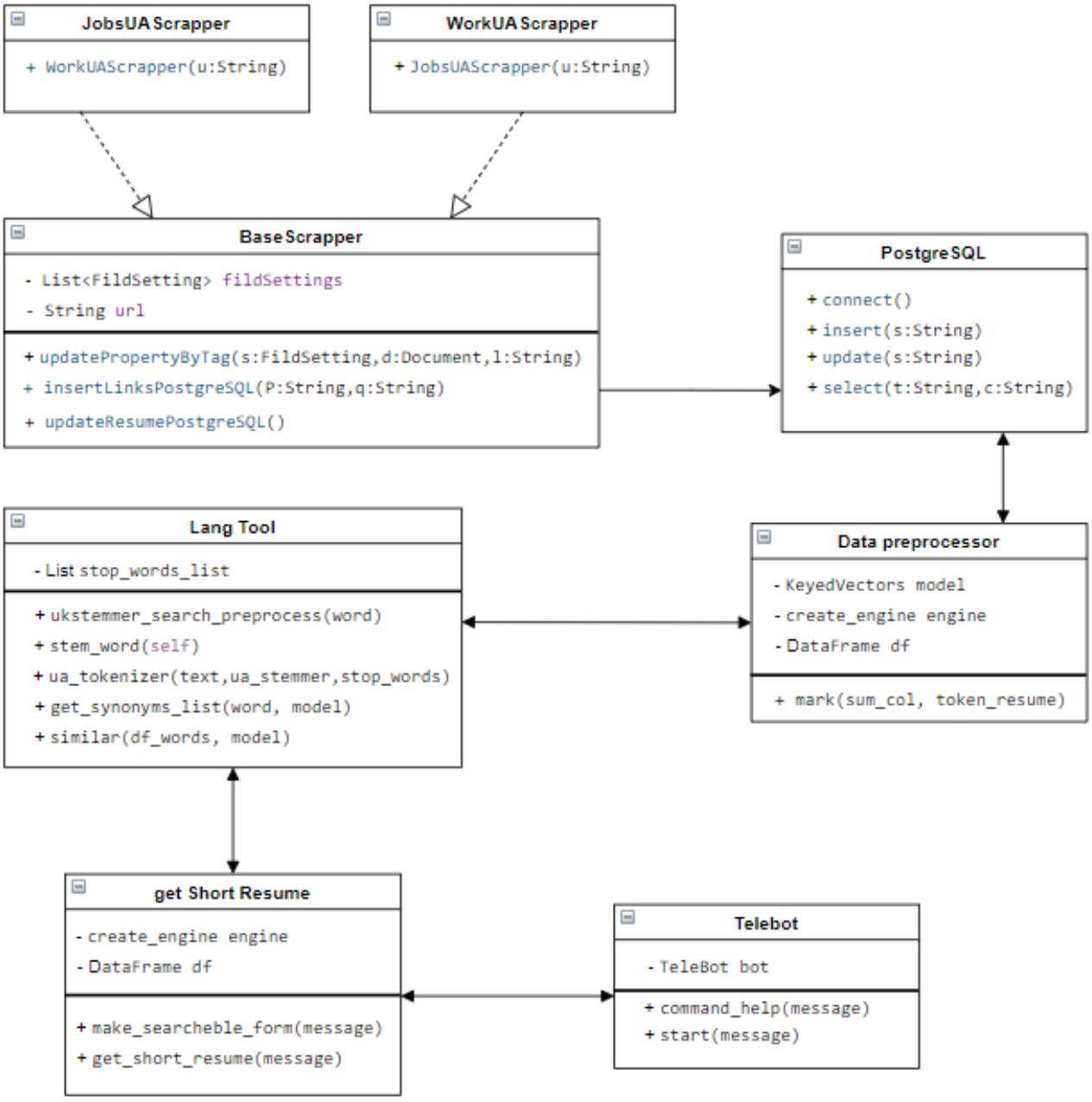

Рис. 3.1 Діаграма класів

На діаграмі компонентів зображені основні компоненти для реалізації HR-інструменту.

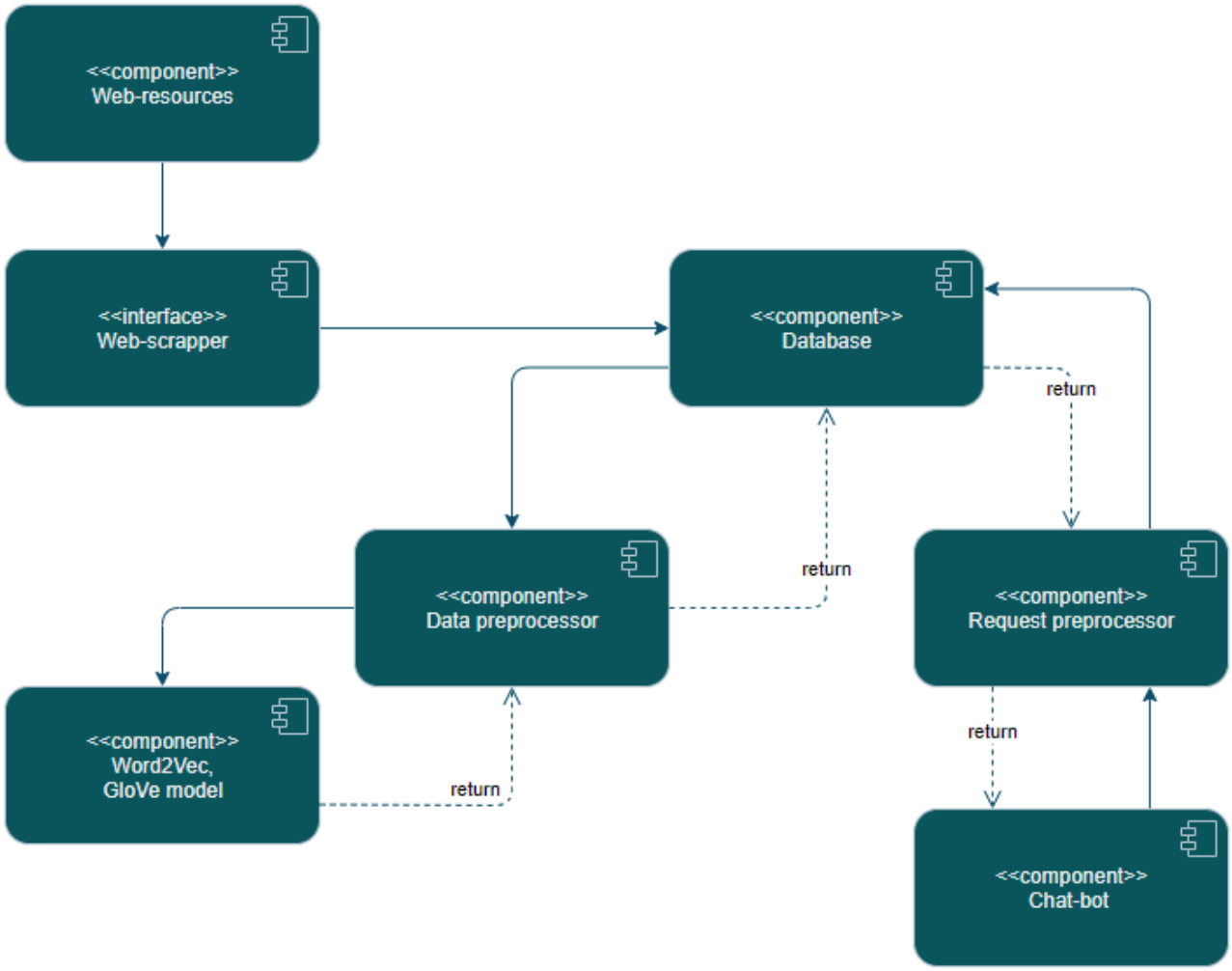

Рис. 3.2 Діаграма компонентів

Висновок: обрано основні шляхи для виконання завдання для збору даних, обрано архітектурні рішення та бібліотеку для її реалізації. Визначено шляхи реалізації інтелектуального пошуку, підібрано необхідний перелік інструментів. Обрано інструмент для розробки чат-бота для виводу інформації.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ КРОСПЛАТФОРМЕНОГО HR-ІНСТРУМЕНТУ ДЛЯ ЗБОРУ ІНФОРМАЦІЇ ПРО ШУКАЧІВ РОБОТИ ЗА ДАНИМИ ВІДКРИТИХ РЕСУРСІВ

4.1 Обґрунтування вибору інструментальних засобів

Для розробки інструменту було обрано мови програмування Java та Python. Розробка веб-скрапера виконано мовою Java з використанням бібліотеки Jsoup, для обробки даних, синонімізації слів та виконання пошуку виконано мовою Python з використанням бібліотеки Gensim та Regex. Також, для реалізації кросплатформеності прийнято рішення використати можливості чат-ботів.

Зберігання даних реалізовано за допомогою реляційної бази даних PostgreSQL, переваги даного рішення полягають в тому, що немає обмеження на максимальний розмір БД, також, має потужні і надійні механізми транзакцій та реплікації і легко масштабується.

Задля збереження та контрою версій додатку використовується система контролю версій Git. Підхід Git до зберігання даних схожий на набір станів файлової системи. Кожен раз, коли ви зберігаєте стан свого проекту в Git, система запам'ятовує, як виглядає кожен файл в цей момент, і зберігає посилання на цей знімок. Ця технологія є невід’ємною частиною розробки масштабованих застосунків.

Даний перелік інструментальних засобів є найбільш зручним та практично корисним для виконання поставленої задачі. Використання мови програмування Java є гарним рішенням для виконання скрапінгу, завдяки швидкості виконання операцій, що дозволяє реалізувати швидкий збір даних, а мовою Python реалізовані зручні пакети для аналізу даних, що дозволяє заощадити час на розробку.

4.2 Технічне та програмне забезпечення

Даний інструмент може бути запущений на будь-якій з платформі на яку можна завантажити месенджер з підтримкою чат-ботів.

4.3 Програмна реалізація веб-скрапера

Згідно задачі, необхідно реалізувати збір даних з відкритих ресурсів та зберегти їх. Для цього, був проведений аналіз для вибору інструментів для виконання задачі. Було виявлено, що з найпопулярніших ресурсів, можна вилучати дані за допомогою завантаження HTML сторінки та розбору DOM дерева.

В ході розробки, за допомогою бібліотеки Jsoup, було реалізовано об’єкт скрапера, який в параметрах спершу отримує назву масиву даних який збирає, потім назву HTML тегу та третім параметром CSS тег, який необхідно виділити. Також, останнім кроком, реалізовано збереження даних в БД. Далі розглянемо більш детально.

4.3.1 Завантаження сторінок з оголошеннями

В функції insertLinksPostgreSQL(String path, String cssQuery) реалізовано наповнення бази даних посиланнями на резюме. В параметрах приймає посилання на сторінку з якої потрібно збирати дані та CSS селектор для знаходження посилань. Для того щоб отримати і проаналізувати HTML-документ з інтернет мережі та знайти в ньому дані використовується метод Jsoup.connect(String url), який в параметрах створить нове з’єднання і відправить GET-запит. Якщо під час отримання URL-адреси виникне

помилка, метод поверне повідомлення IOException, яке потрібно обробити належним чином.

Після вилучення посилань на резюме виконано запис даних в PostgreSQL, який виконується функцією insert(sql) об’єкта PostgreSQL.

```
public void insertLinksPostgreSQL(String path, String cssQuery) {
        Document doc = Jsoup.connect(this.url +
path).userAgent("Chrome/86.0.4240.198").get();
        Elements nameLink = doc.select(cssQuery);
        for (Element workersList : nameLink) {
            String link = workersList.attr("abs:href");
            String sql = "INSERT INTO USERS (LINK) VALUES ('" + link + "') ON CONFLICT
DO NOTHING;";
            System.out.println(sql);
            PostgreSQL.insert(sql);
        }
        System.out.println("USERS table, column LINK, INSERT SQL complete.");
}
```

4.3.2 CSS селекціонування

Для виконання CSS селекціювання, наприклад, як в функції updatePropertyByTag(FildSetting setting, Document document, String link), яка є складовою об’єкта BaseScrepper, використано функції об’єктів Element та Elements. Тут, за допомогою функції select(cssQuery), яка в параметрі приймає CSS тег для знаходження елементу. Завдяки тому, що об’єкти Elements підтримують jquery, є можливість робити потужні та надійні запити.

Тут метод select(cssQuery) спочатку відкидає всю розмітку HTML у якому відсутній CSS-тег, який передається в параметрах. Потім отриманий текст необхідно очистити від рядків які не віповідають нашому запиту та очистити для збереження в БД, за допомогою getElementsByTag(tagName) та регулярного виразу.

Після отримання підготовленого тексту, викликається метод updateSQL(colName, fildValue, link), який в першому рядку визначає назву колонки в яку потрібно записати отримані дані, а другим передає безпосередньо самі дані.

```
private void updatePropertyByTag(FildSetting setting, Document document, String link){
    Elements domList = document.select(setting.cssQuery);
    for (Element list : domList) {
        String fildValue = list.getElementsByTag(setting.tagName).text().replace("'",
"''");
        if (!setting.pattern.isEmpty() && !fildValue.isEmpty()) {
            Pattern pattern = Pattern.compile(setting.pattern);
            Matcher matcher = pattern.matcher(fildValue);
            matcher.find();
            fildValue = matcher.group();
        }
        updateSQL(setting.colName, fildValue, link);
    }
}
```

4.3.3 Запис до DB

Базою даних в даній роботі виступає PostgreSQL, в першу чергу було сформовано поля для резюме користувача. Для зручності створено 8 колонок, а саме:

- Link — посилання на резюме користувача.
- Name — ім’я користувача з резюме.
- Date — дата публікації резюме.
- City — місто в якому готовий працювати робітник.
- Age — вік робітника.
- Resume — поле вільного запису, фактично текст резюме.
- Profession — шукана професія.
- Index — індексація резюме, допоможе в подальшому в процесі аналізу.

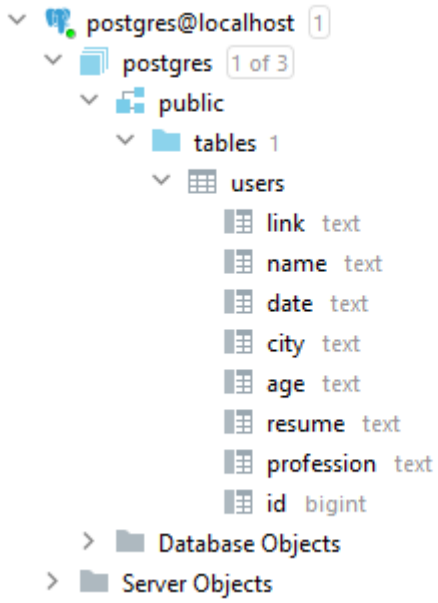

Рис. 4.1 Вигляд таблиці USERS.

Для запису в БД функція updateSQL(String colName, String fildValue, String link) формує SQL запит та викликає функцію update() об’єкту PostgreSQL.

```
protected void updateSQL(String colName, String fildValue, String link){
    String sql = "UPDATE USERS SET " + colName + " = '" + fildValue + "' where LINK =
'" + link + "';";
    System.out.println(sql);
    PostgreSQL.update(sql);
}
```

Функція update(String sql) власне виконує з’єднання з БД, розміщення даних у відповідній колонці, зберігає оновлений стан таблиці та закриває з’єднання.

```
c = connect();
c.setAutoCommit(false);
stmt = c.createStatement();
stmt.executeUpdate(sql);
c.commit();
stmt.close();
c.close();
```

4.3.4 Результати виконання веб-скрапера

Розроблений додаток було застосовано для виконання задачі збору неструктурованих даних в БД . Це дозволило ефективно отримувати дані для функціонування розроблюваного інструменту.

В ході виконання, скрапер звітує про виконання роботи та спершу повідомляє про збереження посилань на оголошення в мережі Інтернет, приклад наведено на скріншоті внизу.

```
C:\Users\admin\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.1.1
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/6863671/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/7078750/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/6987634/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/7078729/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/5394040/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/7078744/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/6986670/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/7078749/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/6231389/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/7078748/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/7078739/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/4814876/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/5405444/') ON CONFLICT DO NOTHING;
INSERT INTO USERS (LINK) VALUES ('https://www.work.ua/resumes/6917579/') ON CONFLICT DO NOTHING;
USERS table, column LINK, INSERT SQL complete.
```

Рис. 4.2 Логування виконання SQL запитів з заповнення посилань на резюме.

Після цього програма починає збирати окремо дані з різних резюме та наповнювати за посиланнями таблицю у відповідні колонки. Також є можливість контролювати запити, які виконує база даних.

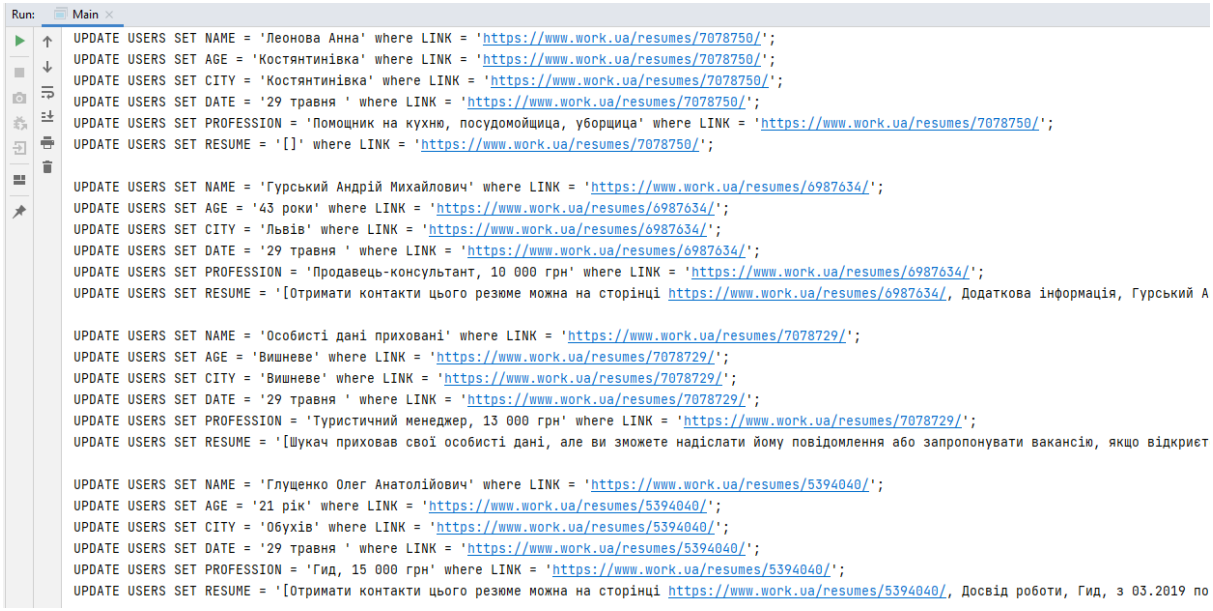

Рис. 4.3 Логування виконання SQL запитів з заповнення полів відповідно до посилань.

Після завершення виконання в таблиці USERS з’явилися записи з відповідними даними, які необхідно буде обробити для реалізації оцінки досвіду роботи та пошуку за запитом. Результат роботи продемонстровано на картинці.

	link	name	date	city	age	resume	profession	id
1	https://www.work.ua/resumes/6933912/	Авдеева Алина Александров…	25 березня	Павлоград	22 роки	[Продавец-консультант, з…	Специалист по раб…	1
2	https://jobs.ua/cv-menedjer-po-perso…	Анотольевич Кирилл	14 квітня	Вінниця	47 років	Опис Организация работы …	Менеджер по персо…	2
3	https://jobs.ua/cv-detektiv-1156754	Нестер Максим	15 березня	Харків	23 роки	Досвід роботи Криминалис…	Детектив	3
4	https://jobs.ua/cv-upravlyayuschiy-z…	Вакулюк Максим	24 березня	Київ	44 роки	Опис РЕЗЮМЕ Вакулюк Макс…	Управляющий загор…	4
5	https://www.work.ua/resumes/6816549/	Антонов Волидимир Андрійо…	14 квітня	Павлоград	20 років	[Освіта, ХНУРЭ, КН, 122,…	Сотрудник для люб…	5
6	https://www.work.ua/resumes/5547733/	Azubuike Cynthia	15 квітня	Запоріжжя	24 роки	[Досвід роботи, English …	English teacher, …	6
7	https://www.work.ua/resumes/5812595/	Курочка Андрей Александро…	14 квітня	Кривий Ріг	30 років	[Досвід роботи, Мастер, …	Инженер по строит…	7
8	https://www.work.ua/resumes/4356351/	Понтус Инна Николаевна	7 квітня	Одеса	26 років	[Досвід роботи, Senior B…	Менеджер по прода…	8
9	https://www.work.ua/resumes/5230423/	Федоренко Виктория	13 квітня	Київ	33 роки	[Досвід роботи, Няня, з …	Оператор 1С	9
10	https://www.work.ua/resumes/5925792/	Rybka Tanya	13 квітня	Київ	40 років	[Досвід роботи, Руководи…	Руководитель PR, …	10
11	https://www.work.ua/resumes/6844089/	Гордейчук Євген Миколайов…	5 квітня	Чернівці	18 років	[Мене звати Євген, мені …	Менеджер по робот…	11
12	https://www.work.ua/resumes/6422003/	Тютюнник Андрей Игоревич	14 квітня	Одеса	21 рік	[Досвід роботи, Член бри…	Сотрудник для люб…	12
13	https://www.work.ua/resumes/4270486/	Правдзіва Галина Олексіїв…	13 квітня	Київ	48 років	[Провідний економіст, з …	Провідний економі…	13
14	https://www.work.ua/resumes/6958759/	Романенко Анастасия Алекс…	7 квітня	Конотоп	18 років	[Освіта, Конотопський фа…	Разнорабочий, 7 0…	120
15	https://www.work.ua/resumes/4733720/	Веклич Олег	14 квітня	Запоріжжя	46 років	[Досвід роботи, Руководи…	Руководитель снаб…	121
16	https://www.work.ua/resumes/721027/	Сківка Евген	13 квітня	Одеса	34 роки	[Бренд-менеджер, з 08.20…	Керівник відділу …	14
17	https://www.work.ua/resumes/6969855/	Березовская Анастасия	13 квітня	Київ	19 років	[Досвід роботи, Официант…	Оператор call-цен…	15
18	https://jobs.ua/cv-tehnchniy-direkto…	Уруський Іван Михайлович	7 квітня	Івано-Фра…	61 рік	Досвід роботи технічний …	Професійне резюме…	16
19	https://jobs.ua/cv-analitik-po-zakup…	Echkenko Andrii	14 квітня	Харків	55 років	Опис ЦЕЛЬ получение долж…	Аналитик по закуп…	17
20	https://www.work.ua/resumes/6933911/	Соболев Эдуард Васильевич	25 березня	Київ	28 років	[Освіта, Школа №183, Кие…	Продавец-консульт…	19
21	https://www.work.ua/resumes/6975463/	Янчук Валерія Дмитрівна	15 квітня	Чернівці	18 років	[Освіта, ЧНУ, Юридичний …	Менеджер по робот…	20
22	https://jobs.ua/cv-tehnolog-z-vprova…	Шилко Сергій	14 квітня	Київ	45 років	Досвід роботи технолог к…	Професійне резюме…	21
23	https://jobs.ua/cv-chertejnik-pomosc…	Ткаченко Александр	14 квітня	Київ	40 років	Досвід роботи Преподават…	Професійне резюме…	22
24	https://www.work.ua/resumes/1830113/	Островский Сергей Василье…	14 квітня	Вінниця	41 рік	[Додаткова інформація, Н…	Региональный пред…	23
25	https://www.work.ua/resumes/4562198/	Маслюк Александр Иванович	7 квітня	Дніпро, К…	33 роки	[Досвід роботи, Руководи…	Финансовый директ…	24

Рис. 4.4 Вигляд таблиці після виконання роботи додатку.

4.4 Аналіз отриманих даних та адаптація для виводу

Для виконання задачі аналізу необхідно визначити дані в зручному для обробки класі. Для цього найбільш зручним рішенням є бібліотека Pandas яка має відкритий код, що забезпечує високопродуктивні, прості у використанні структури даних та засоби аналізу даних для мови програмування Python.

Основні типи даних якими оперує дана Pandas — це Series та DataFrame. Ці об’єкти представляють собою певну таблицю, з якою зручно працювати. Структура даних DataFrame містить розмічені осі (рядки та стовпці). Арифметичні операції виконуються як за мітками рядків, так і за стовпцями. Можна розглядати як контейнер, схожий на словник, для об’єктів серії. Первинна структура даних про Pandas продемонстрована нижче.

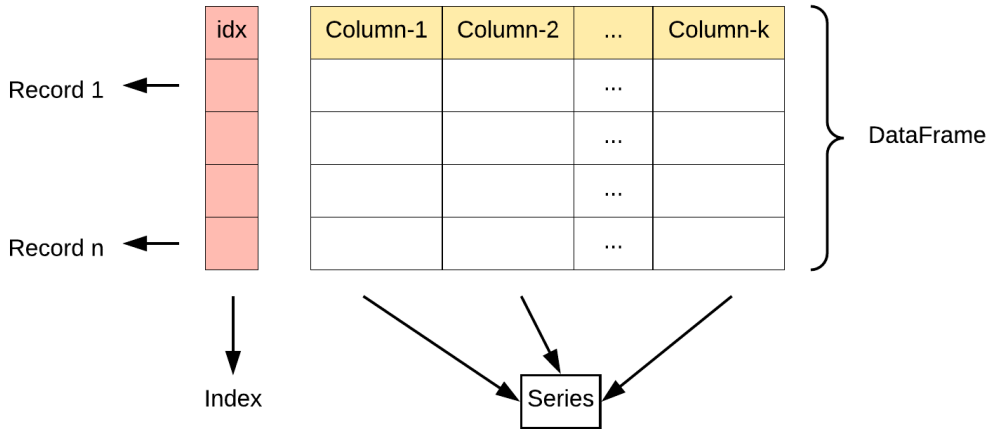

Рис. 4.5 Структура даних бібліотеки Pandas

4.4.1 Отримання даних в структуру даних Pandas DataFrame.

Для того, щоб отримати клас DataFrame з даними з БД використана функція класу read_sql_query(sql, con), першим параметром якої виступає SQL запит для знаходження необхідної таблиці, а другим — параметри з’єднання.

```
engine = create_engine('postgresql://postgres:1JvVeb3q@localhost:5432/postgres')
df = pd.read_sql_query('select * from USERS', con=engine)
```

В наслідок роботи функції ми отримуємо фрейм даних за допомогою якого зручно працювати та обробляти дані.

4.4.2 Приведення типів даних

Для подальшої роботи з даними, необхідно використати приведення типів, в даному випадку, веб-скрапер заповнює стовпці “resume” і ‘profession’ як список і для зручності його необхідно переформатувати в рядковий тип даних. Приклад приведення типів даних наведено нижче.

```
df['resume'] = df['resume'].astype(str)
df['profession'] = df['profession'].astype(str)
```

4.4.3 Лематизація, токенізація, стемінг та стоп-слова

Для виконання задачі знадобиться алгоритм стемінгу, даний алгоритм буде окремим для кожної мови. Наприклад для групи романських мов у бібліотеці nltk (Natural Language Toolkit) є готові стемери які можна використовувати, вони вже налагоджені і протестовані. Для слов’янських — необхідно скористатись алгоритмом, якого поки немає в бібліотеці nltk.

Ці алгоритми базуються на правилах, за якими можна скорочувати слово. Правила скорочення слів в свою чергу підлягають різні частини мови. Одне й те саме скорочення застосовується для всіх граматичних форм одного й того самого слова, незалежно від року, числа, відмінка й часу.

У класі UkrainianStemmer зібрано перелік закінчень для різних частин мови, такі як: іменник, дієслово, прикметник, дієприкметник, рефлексивне дієслово. Приклад закінчень наведено нижче.

```
self.word = word
self.vowel = r'аеиоуюяіїє'
self.perfectiveground = r'(ив|ивши|ившись|ыв|ывши|ывшись((?<=[ая])(в|вши|вшись)))$'
self.reflexive = r'(с[яьи])$'
self.adjective =
r'(ими|ій|ий|а|е|ова|ове|ів|є|їй|єє|еє|я|ім|ем|им|ім|их|іх|ою|йми|іми|у|ю|ого|ому|ої)$'
self.participle = r'(ий|ого|ому|им|ім|а|ій|у|ою|ій|і|их|йми|их)$'
self.verb = r'(сь|ся|ив|ать|ять|у|ю|ав|али|учи|ячи|вши|ши|е|ме|ати|яти|є)$'
self.noun =
r'(а|ев|ов|е|ями|ами|еи|и|ей|ой|ий|й|иям|ям|ием|ем|ам|ом|о|у|ах|иях|ях|ы|ь|ию|ью|ю|ия|ь
я|я|і|ові|ї|ею|єю|ою|є|еві|ем|єм|ів|їв|ю)$ '
self.rvre = r'[аеиоуюяіїє]'
self.derivational =
r'[^аеиоуюяіїє][аеиоуюяіїє]+[^аеиоуюяіїє]+[аеиоуюяіїє].*(?<=о)сть?$'
```

Для проведення токенізації даних крім алгоритму стемінгу, для підвищення ефективності роботи необхідно прибрати стоп-слова, які у реченнях виконують роль зв'язку. Такі слова здебільшого не змінюють сенс речення, тому їх можна прибрати. Спершу було взято готовий список стоп-слів, а потім список було розширено, шляхом видалення слів, які не несуть змісту для виконання задачі.

На етапі коли проведено стемінг слів, які будуть використані для реалізації просунутого пошуку та зібрано стоп-слова, можна проводити токенізацію. Даний процес очистить тексти від спец-символів та стоп-слів і поверне дані, які містять тільки важливу інформацію. Таким чином відбувається підвищення точності та оптимізація часу на виконання алгоритму оцінювання релевантності оголошень. Приклад реалізації токенізатора для української мови наведено в коді нижче.

```
def ua_tokenizer(text, ua_stemmer=True, stop_words=None):
    if stop_words is None:
        stop_words = stop_words_list
    tokenized_list = []
    text = re.sub(r"""['’"`]""", '', str(text))
    text = re.sub(r"""([0-9])([\u0400-\u04FF]|[A-z])""", r"\1 \2", str(text))
    text = re.sub(r"""([\u0400-\u04FF]|[A-z])([0-9])""", r"\1 \2", str(text))
    text = re.sub(r"""[\-.,:+*/_]""", ' ', str(text))
    for word in nltk.word_tokenize(text):
        if word.isalpha():
            word = word.lower()
            if ua_stemmer is True:
                word = UkrainianStemmer(word).stem_word()
            if word not in stop_words:
                tokenized_list.append(word)
    return tokenized_list
```

4.4.4 Вкладання слів Word2Vec, GloVe, LexVec

З виконанням задання пошуку споріднених за значеннм слів, добре справляються методики мовного моделювання та навчання ознак в обробці природної мови, в яких слова або фрази зі словника відображують у вектори дійсних чисел, такі медоди називають вкладанням слів (word embedding). Дана методологія дає математичне вкладення з простору з багатьма вимірами, по одному на слово, до неперервного векторного простору набагато нижчої розмірності.

Дані моделі формують за допомогою нейронні мережі, яка навчається на зібраному корпусі даних. Корпуси можуть бути зібрані на будь-яких даних і саме від даних на яких навчається модель сильно залежить які слова попадуть у векторний простір в безпосередній близькості.

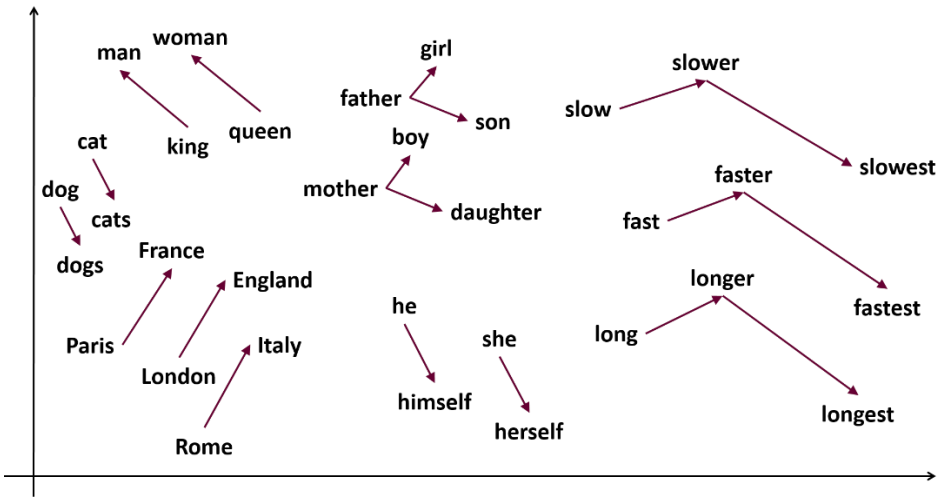

Рис. 4.6 Приклад розташування слів у векторному просторі

На базі зібраних корпусів новин, статей, художньої літератури, законів та юридичних текстів учасники спільноти «lang-uk» обчислили найпоширеніші ембедінги: Word2Vec (та його покращену версію LexVec) й GloVe та опублікувати ці моделі, тому що їх обчислення займає доволі багато часу та серверних ресурсів.

Були створені окремі моделі для кожної категорії текстів з 300d векторами. Також можна завантажити лематизовану версію тих самих корпусів.

Корпус	300d as is	300d lowercase	300d lemmatized	300d lemmatized lowercase
Художня література	LexVec 105MB Word2Vec 98MB GloVe 99MB	LexVec 100MB Word2Vec 93MB GloVe 94MB	LexVec 53MB Word2Vec 51MB GloVe 51MB	LexVec 53MB Word2Vec 50MB GloVe 50MB
Новини	LexVec 329MB Word2Vec 328MB GloVe 324MB	LexVec 296MB Word2Vec 296MB GloVe 292MB	LexVec 160MB Word2Vec 161MB GloVe 159MB	LexVec 156MB Word2Vec 157MB GloVe 155MB
Уберкорпус	LexVec 536MB Word2Vec 530MB GloVe 526MB	LexVec 483MB Word2Vec 480MB GloVe 476MB	LexVec 297MB Word2Vec 295MB GloVe 294MB	LexVec 291MB Word2Vec 289MB GloVe 288MB

Рис. 4.7 Зібрані моделі корпусів командою «lang-uk»

Першим кроком у впровадженні моделі є навчання моделі, а для навчання моделі просто виконуємо команду наведену нижче.

```
model = KeyedVectors.load_word2vec_format('news.lowercased.lemmatized.word2vec.300d', binary=False)
```

Де KeyedVectors — це пакет, який присутній всередині Gensim, а 'news.lowercased.lemmatized.word2vec.300d' — це набір даних українською мовою зібраний групою ентузіастів, який використовується тут для навчання.

Тепер нам доведеться знайти найбільш подібний елемент, який присутній у наборі даних із вибраним нами словом. Наприклад, ми маємо знайти всі подібні слова, пов'язані зі словами «менеджер», «юрист» в наборі даних новин:

```
model.most_similar("менеджер")
```

```
[('менеджмент', 0.6241483092308044),
 ('консультант', 0.6234381794929504),
 ('директор', 0.5953992009162903),
 ('топ-менеджер', 0.5875224471092224),
 ('юрисконсульт', 0.5569397807121277),
 ('маркетолог', 0.5499082803726196),
 ('маркетинг', 0.5348953008651733),
 ('продюсер', 0.5267871022224426),
 ('сео', 0.5251652598381042),
 ('управлінець', 0.5206174254417419)]
```

Рис.4.8 Матриця подібності 1

```
model.most_similar("юрист")
```

```
[('правник', 0.7351112961769104),
 ('правознавець', 0.643578290939331),
 ('економіст', 0.6146543622016907),
 ('консультант', 0.611090362071991),
 ('адвокат', 0.6066429615020752),
 ('фінансист', 0.6062204241752625),
 ('спеціаліст', 0.5755605101585388),
 ('психолог', 0.5691010355949402),
 ('фахівець', 0.5683552622795105),
 ('міжнародник', 0.5618101954460144)]
```

Рис.4.9 Матриця подібності 2

Тут функція most_similar обчислює подібність косинусів між кожними словами та виводить список в порядку спадання.

4.4.5 Реалізація функції для отримання наближених за значенням слів

Для реалізації використовується векторизована модель Word2Vec зібрана на корпусі новин українською. Для цього емпіричним методом було підібрано найбільш підходящий ембединг, який давав найкращі результати при роботі з назвами професій.

```
for word in df_words:
        synonyms_turple = model.most_similar(positive=[word])
        for synonyms in synonyms_turple:
            if synonyms[1] > 0.62:
                new_list_profession += synonyms[0].split()
```

Для покращення роботи даного алгоритму необхідно перенавчити векторні моделі на більшому корпусі даних, які стосуються професій, власних навичок, освітніх закладів.

4.5 Вивід даних за допомогою чат-ботів

Дану частину можна реалізувати за допомогою будь-якого чат боту, для дипломної роботи було прийнято рішення використовувати бібліотеку PyTelegramBot для реалізації чат-бота в месенджері Telegram.

Чат-бот працює в режимі запит – відповідь. У запиті користувач повинен сформувати запит з назвою професії та отримати відповідь з резюме п’ятьох шукачів роботи. Розділ «Ім’я» містить гіперпосилання, яке відправляє користувача на першоджерело резюме. Дані про ім’я шукача можуть бути приховані налаштуванням приватності на сайтах де вони адмініструють свої оголошення. Також для того, щоб спеціалісту з HR було зручно орієнтуватися, представлено дані про вік, шукану посаду або професію та досвід роботи, якщо такі є. Приклад виконання програми наведено нижче. Також в додатку А є діграма прецендентів де описано взаємодію акторів з додатком.

Для покращення роботи даного інструменту, потрібно додати шари фільтрів, для ще більш точного селекціювання робітників. Можна додати пошук за певним містом, або наближеними містам, досвідом роботи, освітою, власними якостями або навичками.

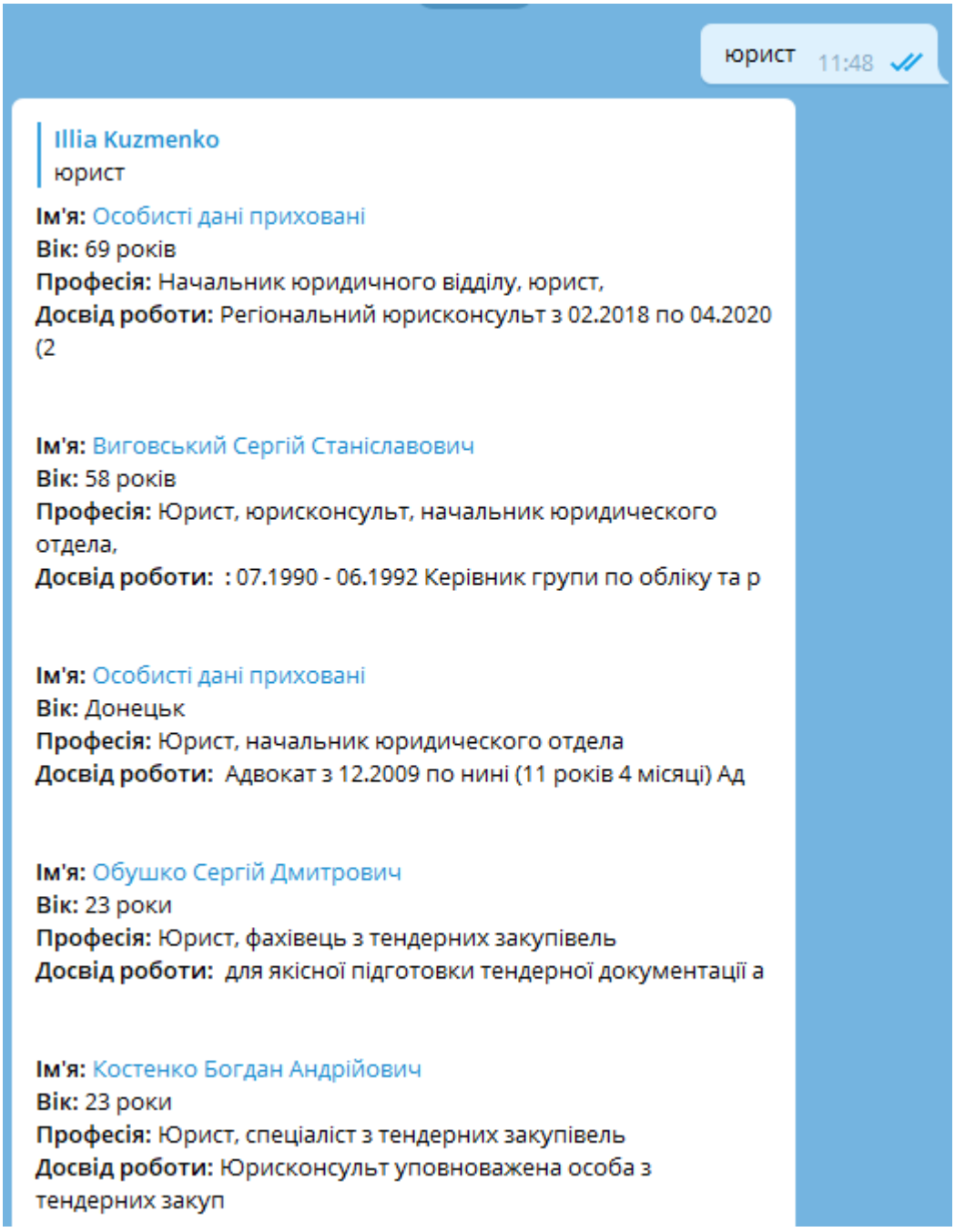

Рис.4.10 Приклад виконання програми за запитом з назвою професії

Висновок: згідно поставленої задачі, було виконано збір даних з відкритих ресурсів за допомогою технологій веб-скрапінгу, а саме аналіз DOM дерева. Дані зібрані з відкритих ресурсів збережено в БД за відповідними колонками. Обдумано рішення для покращення роботи веб-скраперу.

Зібрані дані проаналізовано відповідно до поставленої задачі. Визначено та реалізовано методи для обробки даних резюме. Спроектовано та виконано оцінку відповідності між шуканою професією та змістом резюме. Обдумано методи покращення точності для просунутого пошуку та оцінки оголошень відповідно до поставленої задачі.

Дані взаємодію з додатком реалізовано в чат-боті, який працює в режимі запит – відповідь. Для розширення можливостей інструменту продумано можливість обробки запитів на стороні чат-боту. Додаток протестовано та визначено ефективність роботи.

ВИСНОВКИ

Застосування методів веб-скрапінгу відкриває великі можливості при роботі з відкритими ресурсами. За допомогою комбінації різних технік вилучення контексту підвищується ефективність вилучення даних незважаючи в обхід захисту ресурсів від скрапінгу. Даний метод разом з БД, дають можливість використовувати його в великому спектрі задач, від моніторингу цін на авіаквитки до контролю за рухом курсів валют. Відповідно до поставленої задачі обрано основні шляхи для виконання збору даних, обрано архітектурні рішення та бібліотеку для її реалізації

Отримані дані в наслідок веб-скрапінгу були проаналізовані відповідно до поставленої задачі. Визначено та реалізовано методи для обробки даних резюме. Спроектовано та реалізовано алгоритм, який дозволяє виконати оцінку важливості резюме шукача роботи, до запиту, шляхом очищення даних, отримання слів схожих за значенням та присвоєння їм коефіцієнтів важливості.

Проаналізовано недоліки такого рішення, а саме те, що обробка даних виконується недостатньо швидко, це можна вирішити за допомогою векторного представлення тексту, це має зекономити час на обчислення і, відповідно, на сам пошук. Проведено дослідження ефективності роботи різних видів моделей векторних представлень, зібраних на різних корпусах даних. Проаналізовано можливість збору власної моделі навченої на корпусі даних, який найкраще відповідає поставленій задачі. Також недоліком реалізації є те, що не враховується контекст слів і на оцінку може впливати використання їх в іншому контексті, який не відповідає вимогам. Дану проблему також можна вирішити за допомогою семантичного аналізу тексту. Таким чином можна спершу отримати дані, які відповідають контексту, а вже потім проводити оцінювання.

Для реалізації кросплатформності прийнято рішення використовувати чат-боти в режимі запит - відповідь. В роботі продемонстровано роботу інструменту за допомогою

бібліотеки PyTelegramBot, яка допомагає реалізовувати чат-боти для популярного месенджеру Telegram. Розглянуто можливість розширення функціоналу, шляхом додавання більшої кількості пошукових запитів, які при належному налаштуванні такі боти можуть обробляти їх на сервері та отримувати відповідь, яка задовольнить спеціаліста з управління персоналом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ryan Mitchell Web Scraping with Python: Collecting Data from the Modern Web, 1st Edition, 2015. – 256с.

2. Kevin Sahin The Java Web Scraping Handbook, 2020. – 115с.

3. Karthiek Reddy Bokka, Shubhangi Hora, Tanuj Jain, Monicah Wambugu Deep Learning for Natural Language Processing: Solve your natural language processing problems with smart deep neural networks, 2019. – 372с.

4. Charu C. Aggarwal Neural Networks and Deep Learning: A Textbook, 2019. – 520с.

5. Steven Bird, Ewan Klein and Edward Loper Natural Language Processing with Python, 2009. – 504с.

6. Ющук І.П. Українська мова 10 клас 2018р. – 208с.

7. Tom Dotz, Tom Hoobyar, Susan Sanders NLP: The Essential Guide to Neuro-Linguistic Programming, 2013. – 480с.

8. Штучна нейронна мережа - Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/штучна_нейронна_мережа.

9. Стемінг та лематизація – NLP Stanford Education. – [Електронний ресурс]. – Режим доступу: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

10. Обробка природної мови – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/обробка_природної_мови.

11. Liddy E.D. Natural Language Processing. In Encyclopedia of Library and Information Science, 2nd Ed. NY 2001. — 10с.

12. Бабенко Т. В., Сушко С. О. Про ентропію української мови // Науково-практичний журнал «Захист інформації». — 2012. — № 3. — 105с.

13. Диковицкий В. В., Шишаев М. Г. Обробка текстів природньої мови в моделях пошукових систем // Збірник наукових трудів. — 2010. — 30с.

14. Іванов О. В. Класичний контент-аналіз та аналіз тексту: термінологічні та методологічні відмінності / Іванов Олег Валерійович // Вісник Харківського національного університету імені В. Н. Каразіна, Харків: Видавничий центр ХНУ імені В. Н. Каразіна, 2013. — № 1045. — 72с.

15. Ланде Д. В., Снарский А. А., Безсуднов И. В. Интернетика: Навигация в сложных сетях: модели и алгоритмы. — М.: Либроком, 2009. — 264с.

16. Anders Søgaard, Ivan Vulić, Sebastian Ruder Cross-Lingual Word Embeddings (Synthesis Lectures on Human Language Technologies), 2019. — 134с.

17. Йоганнес Гелльріх Word Embeddings: Reliability & Semantic Change, 2019. — 192с.

18. Manning C. D. An Introduction to Information Retrieval / Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. – Cambridge University Press, 2009. – P. 22–36.

ДОДАТКИ

Додаток А. Діаграма прецендентів.