**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
*(код, найменування спеціальності)*
освітньо-професійної програми Інженерія програмного забезпечення
*(назва)*

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Єгор МОТРУК
*(підпис)*

Виконав: здобувач вищої освіти група ПДМ-62
Єгор МОТРУК

Керівник: Наталія ТРІНТІНА
*к.т.н., доцент*

Рецензент:
*науковий ступінь, вчене звання*
_____
Ім'я, ПРІЗВИЩЕ

**Київ 2024**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Мотруку Єгору Олеговичу

1. Тема кваліфікаційної роботи: Розробка системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму

керівник кваліфікаційної роботи Наталія ТРІНТІНА к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, автоматизовані методи оцінки, навчання нейронної мережі на основі датасету.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз сучасних методів автоматизованого оцінювання кандидатів.
2. Проведення аналізу готових систем автоматизації.
3. Розробка власного алгоритму оцінювання кандидатів.

5. Перелік графічного матеріалу: *презентація*

1. Обгрунтування необхідності впровадження системи оцінювання
2. Порівняння існуючих систем
3. Алгоритм оцінки кандидатів
4. Вхідні данні системи
5. Алгоритм роботи проектованої системи

6. Дата видачі завдання«19» жовтня 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз наявної науково-технічної літератури | 19.10-05.11.23 | |
| 2 | Вивчення матеріалів для впровадження системи оцінювання | 06.11-12.11.23 | |
| 3 | Дослідження сучасних систем | 13.11-19.11.23 | |
| 4 | Аналіз особливостей впливу автоматизованої системи на ринок праці | 20.11-26.11.23 | |
| 5 | Дослідження технологій машинного навчання | 27.11-03.12.23 | |
| 6 | Застосування машинного навчання за допомогою датасету | 04.12-10.12.23 | |
| 7 | Оформлення роботи: вступ, висновки, реферат | 11.12-20.12.23 | |
| 8 | Розробка демонстраційних матеріалів | 21.12-29.12.23 | |

Здобувач вищої освіти _____ Єгор МОТРУК
*(підпис)*

Керівник
кваліфікаційної роботи _____ Наталія ТРІНТІНА
*(підпис)*

# РЕФЕРАТ

Пояснювальна записка: 83 с., 15 рисунків, 8 таблиць, 16 джерел.

*Мета роботи* – підвищені точності ранжування великих об’ємів резюме кандидатів відповідно до запиту рекрутерів та збільшенні швидкості обробки датасету резюме відповідно до одиниці часу.

*Об’єкт дослідження* – процес автоматизованого оцінювання кандидатів для роботодавців в процесі найму.

*Предмет дослідження* – моделі та методи автоматизованого оцінювання кандидатів на основі нейронних мереж.

*Короткий зміст роботи:* Дипломний проект спрямований на аналіз методів машинного навчання для оцінки та ранжування даних, зокрема застосування їх для створення системи оцінки кандидатів для потенційних роботодавців. Основною ціллю є оптимізація процесу найму шляхом впровадження автоматизованого сортування кандидатів на основі їхніх резюме та можливість швидкого пошуку серед них за допомогою ефективних запитів.

Дослідження включає в себе використання методів машинного навчання, зокрема, алгоритмів ранжування даних. Новизною є розробка математичної моделі та методу ранжування документів, які інтегруються в структуру системи оцінювання потенційних працівників.

КЛЮЧОВІ СЛОВА: СОРТУВАННЯ, РЕЗЮМЕ, АНАЛІЗ ДАНИХ, МАТЕМАТИЧНА МОДЕЛЬ, МАШИННЕ НАВЧАННЯ, КАНДИДАТ, РЕКРУТЕР, PYTHON.

# ABSTRACT

Explanatory note: 83 pp., 15 figures, 8 tables, 16 sources.

The purpose of the work - improve the accuracy of ranking large volumes of candidate resumes in accordance with the request of recruiters and to increase the processing speed of the resume dataset per unit of time.

Object of research - the process of automated candidate evaluation for employers in the hiring process.

Subject of research - models and methods for automated candidate evaluation based on neural networks.

Summary of work: The diploma project is aimed at studying machine learning methods for data analysis and classification, in particular, their use to create a candidate evaluation system for potential employers. The main goal is to improve the recruitment process by introducing automated sorting of candidates based on their resumes and the ability to quickly search among them using effective queries. The research includes the application of machine learning methods, in particular data ranking algorithms. The innovation of the project is the creation of a mathematical model and a method for ranking documents that are integrated into the structure of the potential employee evaluation system.



KEYWORDS: SORTING, RESUME, DATA ANALYSIS, MATHEMATICAL MODEL, MACHINE LEARNING, CANDIDATE, RECRUITER, PYTHON.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Інтерфейс програмування додатків;

СУБД - Система управління базами даних;

БД - База даних;

ПЗ - Програмне забезпечення;

LTR - Вивчення ранжування;

ML - Машинне навчання;

CV - Історія життя;

UML - Об'єднана мова моделювання;

MART - Багатократне додавання регресійних дерев.

# ВСТУП

Процес найму відповідних працівників у наш час є одним з найважливіших питань для багатьох як великих так і локальних компаній. Особливо це стосується позицій з великою кількістю кандидатів, де вибір може бути подібним до пошуку голки в купі сіна. У таких випадках традиційні методи найму можуть бути дорогими та затратними. Тому не дивно, що технології, спрямовані на полегшення цього процесу, стають дуже популярними. За допомогою баз даних кандидатів та пошукових систем, рекрутер може попередньо відібрати невелику кількість підходящих кандидатів з великої кількості, для подальшого оцінювання.

Важливо відзначити, що ці технології не призначені для заміщення людей, але вони можуть зробити процес вибору в рази ефективнішим та швидшим для рекрутера. Інформаційні системи, що в тій чи іншій ступені автоматизують процеси цієї галузі дозволяють значно спростити рутинні завдання, такі як аналіз резюме та вилучення інформації, щоб рекрутер міг зосередитися на більш складних завданнях. З іншого боку, зі зростанням важливості соціальних мереж, багато компаній також використовують "соціальний рекрутинг", щоб привертати кандидатів через платформи, такі як LinkedIn.

Отже, для повного використання потенціалу кандидатів важливо використовувати розумні стратегії пошуку та рейтингування, щоб максимально оптимізувати процес пошуку відповідних кандидатів на необхідні позиції та процес взаємодії з ними.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Загальне визначення системи оцінювання кандидатів

В умовах ринкової економіки запорукою успішного функціонування та процвітання підприємства є правильно підібрані кадри. Саме трудові ресурси відіграють основну роль у виробничої діяльності підприємства [1].

Під підбором персоналу розуміється процес пошуку та залучення фахівців на ринку праці, у яких є необхідні досвід, знання та кваліфікація. Успішний підбір персоналу сприяє збільшенню прибутку, продуктивності, і навіть лояльності співробітників. В разі, якщо підбір здійснюється неправильно, це неодмінно тягне за собою зниження ефективності трудової діяльності, високу змінність кадрів, появі конфліктів у колективі, а також демотивацію співробітників [2].

Процедура підбору складається з ряду послідовних і взаємопов'язаних елементів [2]:

- Планування персоналу;
- Розробка критеріїв підбору (особистісної специфікації);
- Пошук та залучення кандидатів;
- Відбір персоналу;
- Прийом персоналу.

Розглянемо трактування поняття «відбір персоналу». Так, згідно визначення відбір – це оцінка претендентів на вакантну посаду за допомогою цілеспрямованої діяльності встановлення відповідності параметрів працівника (здібностей, властивостей, мотивації) вимога посади та робочого місця [2].

Згідно визначення відбір - управлінські заходи, за допомогою яких організація з низки кандидатів на вакантну посаду обирає одного чи кількох найбільш відповідних за наявними професійно-моральним критеріям (моделі компетенцій) [2].

Відбір персоналу є процесом раціонального вибору на основі вивчення та оцінки професійних та особистісних якостей претендентів, тих із них, хто найкращим чином відповідає її вимогам, та придатності до виконання обов'язків на певному робочому місці чи посаді [2].

Більш узагальненим поняттям відбору персоналу можна назвати наступне: відбір персоналу – це процес вивчення психологічних та професійних якостей кандидата з метою встановлення його придатності для виконання обов'язків, а також вибір із сукупності претендентів найбільш відповідного з урахуванням відповідності його кваліфікації, спеціальності, особистих якостей та здібностей характеру діяльності, інтересам організації та самого працівника [3]. З визначення випливає, що сама по собі процедура відбору є досить складною у порівнянні з іншими етапами підбору.

Одним із найбільш трудомістких та відповідальних процесів у відборі персоналу є оцінка професіоналізму майбутніх співробітників компанії. Це вимагає від менеджерів суттєвого досвіду, професіоналізму та знання сучасних методів оцінки, щоб оцінити навички потенційного працівника.

Під оцінкою розуміється цілеспрямований процес порівняння характеристик працівника та вимог посади або робочого місця, що дозволяє отримати інформацію для подальших управлінських рішень.

Завдання служби персоналу, яка здійснює оцінку кандидатів при прийомі на роботу, полягає в тому, щоб відібрати такого працівника, який у стані досягти очікуваного організацією результату. Фактично оцінка при прийомі - це одна з форм попереднього контролю якості людських ресурсів організації.

Однак необхідно також брати до уваги те, що як би ретельно не була проведено оцінку кандидата, в її результатах завжди присутня значна похибка, оскільки досконало вивчити людину, виміряти її можливості та потенціал просто нереально. За підсумками оцінки можна зробити висновок лише про ступінь ймовірності відповідності кандидата вакантній посаді. І завжди в частці похибки маячить людський фактор - тяжко прогнозований і практично некерований [3].

Основною проблемою оцінки кандидатів є суб'єктивізм, незважаючи на велику кількість розроблених та використовуваних методів оцінки. У всьому різноманітті різних методів оцінки кандидатів не існує такого, який був би універсальним, єдиним, ефективним і доцільним для всіх підприємств.

У зв'язку з цим перед нами стає питання про те, який метод чи методи оцінки використовувати, який метод даватиме достовірну та достатню інформацію про потенційного співробітника, а також буде найбільш рентабельним, є важливим у системі управління персоналом.

На сьогоднішній день у практиці українських компаній використовуються різні методи оцінки кандидатів, які можна умовно класифікувати на традиційні та нетрадиційні. Розглянемо кожну групу методів оцінки кандидатів більш детально.

До традиційних методів оцінки кандидатів відносяться співбесіда, інтерв'ю, анкетування, тестування, резюме та центри оцінки.

*Резюме* – документ, що містить інформацію про навички, досвід роботи, освіту та іншу інформацію про претендента на вакантну посаду. У ньому претендент надає відомості, які він побажав повідомити про себе роботодавцю. Ознайомившись з резюме, роботодавець приймає рішення про доцільність проведення детальних зустрічей із кандидатом [4].

*Анкетування* – це процедура проведення опитування у письмовій формі з допомогою заздалегідь підготовлених бланків, тобто анкет [4]. Анкетування використовується для ознайомлення із кандидатом на вакансію. У анкеті найчастіше претенденту пропонують заповнити такі дані: прізвище, ім'я, по батькові, адреса прописки, адреса проживання, контактний телефон, освіта, відомості про минулі місця роботи, інформація про те, звідки отримано інформацію про вакансію та інше.

*Співбесіда* досі залишається найбільш широко застосовуваним методом оцінки кандидатів. Співбесіда проводиться з усіма здобувачами, у яких роботодавець попередньо зацікавлений. Під співбесідою розуміється процес, під час якого представник компанії особисто зустрічається з кандидатом на заміщення вакантної посади та спілкується з ним протягом певного періоду часу.

На відміну від співбесіди, яка насправді є бесідою двох сторін, *інтерв'ю* полягає в опитуванні претендента на вакансію представником роботодавця. Інтерв'ю – бесіда, що дає уявлення про досвід та професійні якості претендента. Як метод оцінки кандидатів існує давно, але останніми роками було розроблено кілька його різновидів (у рамках віднесення інтерв'ю до традиційних методів оцінки кандидатів):

- Структуроване інтерв'ю. Вважається одним із самих найпоширеніших серед інтерв'юерів. При підготовці питань зазвичай використовують самі пункти, як у анкеті. Воно проводиться для визначення відповідності даних, зазначених претендентом на резюме, дійсності.
- Ситуаційне або кейс-інтерв'ю. Дозволяє оцінити здатності претендента при вирішенні конкретних завдань відповідно до ситуації. Питання інтерв'юера базуються на тому, як кандидат діє у зазначених обставинах. Отримані відповіді порівнюються з еталонними. Залежно від цього формується думка про рівень підготовленості претендента та наявність професійного досвіду.
- Проективне інтерв'ю. Проводиться оцінка кандидата на на підставі його коментарів до дій вигаданих людей у різних ситуаціях. Для кожного інтерв'ю вибирається відповідна модель, яка допоможе охарактеризувати претендента відповідно до вимог роботодавця. Науково доведено, що, аналізуючи вчинки інших людей, людина оцінює їхні дії, виходячи зі свого досвіду. Даний вид інтерв'ю розкриває психологічний вигляд претендента, показує, якими були б його дії, якби він потрапив у схожу ситуацію.
- Поведінкове інтерв'ю. Розкриває здатність кандидата приймати відповідальні рішення для усунення проблем, що виникають у процесі виконання поставлених перед ним завдань. Основною функцією даного інтерв'ю, є виявлення у здобувача здібності адекватно реагувати на робочі моменти Підходить для оцінки професійних якостей претендентів.
- Групове інтерв'ю. Дозволяє швидко оцінити велику кількість кандидатів на відповідність посади, головними критеріями якою є товариськість та

доброзичливість. Проводиться в присутності кількох кандидатів. У проведенні можуть приймати участь кілька менеджерів із персоналу.

Загалом інтерв'ювання дозволяє зібрати максимально повну інформацію про кандидата, що позитивно позначається на об'єктивності прийняття рішення про наймання претендента.

*Тестування*. Тестування включає два види оцінки: психологічні тести та професійні тести. Психологічне тестування кандидата дозволяє оцінити його лідерські здібності, можливість управління підприємством, колективом, спрогнозувати успішність його роботи у команді. За допомогою професійного тестування визначаються професійні знання, додаткові вміння та навички, що дозволяють виконати службові обов'язки [4].

Варто зазначити, що тести є найоб'єктивнішими методами оцінки кандидатів, оскільки виключається наявність емоційної складовою, яка може вплинути на суб'єктивність інтерв'юера. До того ж існує безліч тестових методик, за допомогою яких є можливість оцінити як інтелектуальні здібності кандидата, так і особистісні характеристики. І хоча валідність та надійність тестів не абсолютна, і методика не може дати стовідсотковий правильний результат, при використанні кількох варіантів роботодавець можемо отримати достатньо стійкий результат.

*Центри оцінки* –різновид групової співбесіди, що активно набирає популярність у наш час. В Україні центри оцінки стали використовувати на початку 1990-х років. Даний метод оцінки відмінно підходить для роботи в ситуаціях масового підбору персоналу, коли потрібно з великої кількості кандидатів вибрати кращого у найкоротший період часу. Цей метод заснований на спостереженні спеціально навчених асесорів (оцінювачів) за поведінкою кандидатів при виконанні ними різних завдань. Зміст завдань відображає основні аспекти та проблеми діяльності у межах тієї чи іншої посади.

До нетрадиційних методів оцінки кандидатів належать стресове або шокове інтерв'ю, brainteaser-інтерв'ю, графологія, соціоніка, фізіогноміка, лінгвістичний аналіз мовлення.

*Стресове інтерв'ю* проводиться для визначення рівню стресової стійкості та конфліктності кандидата. Під час інтерв'ю задаються питання, спрямовані на виведення людини з комфортного стану та спонукання до конфлікту. Найчастіше використовуються підступні питання, до яких заздалегідь важко підготуватися. В даному випадку звертається увага не так на правильність відповіді, але в психоемоційний стан претендента [4]. Метод стресового інтерв'ю дозволяє виявити такі якості кандидата, як стресостійкість, комунікабельність, гнучкість поведінки.

*Brainteaser - інтерв'ю* як метод оцінки кандидатів полягає в тому, що кандидатам необхідно дати відповідь на складне запитання або розв'язати логічне завдання. Мета такого методу – перевірити аналітичне мислення та творчі здібності претендента.

*Графологія:* кандидат на вакантну посаду пише текст, після чого відбувається розбір його щодо наявності помилок, аналіз почерку. Такий метод у наш час слабко розвинений у Україні, швидше за все, через труднощі пояснення результатів або через відсутність спеціалізованого програмного забезпечення.

*Лінгвістичний аналіз мовлення* заснований на аналізі формулювань, оцінної складової слів та виразів. Головна перевага методу полягає в тому, що він дозволяє уникнути соціально бажаних відповідей, оскільки людина неспроможна постійно контролювати форму промови. Провівши лінгвістичний аналіз мови, фахівець із підбору персоналу може зрозуміти особливості мислення кандидата, прийняття рішень, мотивації у робочих відносинах.

Таким чином, існує велика кількість різних методів оцінки кандидатів Для встановлення ступеня відповідності кандидатів вимогам може використовуватися цілий комплекс різних методів, вкладених у всебічну оцінку кандидатів.

Необхідність використання різних методів в оцінці кандидатів пов'язана з тим, що жоден із запропонованих методів щодо окремо не дає вичерпної інформації, на підставі якої можна було б ухвалити рішення про прийом на роботу. Тільки доповнюючи результати, отримані за допомогою одного методу, даними, зібраними з за допомогою інших методів можна розраховувати на те, що відібрані

кандидати максимально відповідатимуть встановленим критеріям відбору та повністю влаштують організацію [4].

Важливо пам'ятати, що в результаті оцінки кандидата на вакантну посаду застосовувані оціночні технології мають бути побудовані таким чином, щоб кандидат був оцінений:

- Об'єктивно – незалежно від чиїхось приватної думки або окремого судження;
- Надійно – відносно вільно від впливу ситуативних факторів (настрою, погоди, минулих успіхів чи невдач);
- Достовірно – оцінюватись має реальний рівень володіння навичками та компетенціями (наскільки успішно людина справляється зі своїм функціоналом);
- З можливістю прогнозу – оцінка має давати дані про те, з якими видами діяльності та на якому рівні людина здатна потенційно впоратися [4].

Сукупність вибраних методів оцінки кандидатів на вакантні посади якраз і буде системою оцінки кандидатів. Система оцінки кандидатів – це сукупність взаємозалежних прийомів та дій, спрямованих на отримання інформації про відповідність кандидатів по заданим критеріям [4].

Таким чином, під оцінкою кандидатів розуміється процес зіставлення характеристик претендента з вимогами роботодавця до посади. Для оцінки кандидатів використовують різні методи оцінки. У наш час інформаційних технологій більшість сучасних компаній намагаються максимально автоматизувати цей процес, що і складає завдання даної роботи.

## 1.2 Технології та програмні засоби, використовувані в системах автоматизованого оцінювання кандидатів

У сучасному світі, де конкуренція на ринку праці надзвичайно висока, роботодавці виявляють ростучий інтерес до використання систем автоматизованого оцінювання кандидатів для ефективного та об'єктивного відбору

талановитих фахівців. Розглянемо ключові технології та програмні засоби, що використовуються в таких системах, зокрема в контексті процесів найму.

### 1.2.1 Методи аналізу тексту та природної мови (NLP)

Методи аналізу тексту та природної мови (NLP) стали необхідною складовою в сфері автоматизованого оцінювання кандидатів, відіграючи ключову роль у вдосконаленні процесу відбору та відборі найкращих талантів для компаній. NLP використовується для інтелектуального аналізу та розуміння текстової інформації, дозволяючи системам розпізнавати не лише слова, а й їхній контекст, семантику та виразність.

Однією з ключових функцій NLP є виявлення ключових компетенцій та навичок, вказаних кандидатами у своїх резюме та супровідних листах. Системи використовують алгоритми, що автоматично виділяють та оцінюють важливі терміни, що дозволяє роботодавцям швидко і точно виділити релевантні дані.

Помітною особливістю NLP є його здатність аналізувати мовний стиль та виразність кандидатів. Це важливо для розуміння якісних аспектів спілкування, таких як чіткість висловлення, адаптабельність та здатність ефективно спілкуватися в командному середовищі.

Додатково, системи NLP розуміють семантичний зв'язок між словами та фразами, що полегшує їхню здатність визначати контекст та точніше інтерпретувати інформацію. Це дозволяє не лише виявляти ключові слова, а й аналізувати їхню взаємодію в конкретному контексті.

Окрім цього, застосування NLP у відборі кандидатів дозволяє автоматизовано порівнювати інформацію із резюме кандидатів з вимогами конкретної вакансії. Це допомагає роботодавцям швидко відсіювати несумісних кандидатів та концентрувати увагу на тих, хто найкраще відповідає потребам компанії.

В цілому, методи аналізу тексту та NLP виявляються невід'ємними інструментами для ефективного та об'єктивного відбору перспективних співробітників, допомагаючи підвищити якість та ефективність процесу найму.

### 1.2.2 Машинне навчання та алгоритми рекомендацій

Машинне навчання (ML) та алгоритми рекомендацій стали ключовими елементами в сучасних системах автоматизованого оцінювання кандидатів, дозволяючи роботодавцям швидше та ефективніше вибирати ідеальних працівників для своїх команд.

Однією з ключових функцій машинного навчання в системах оцінювання є використання алгоритмів для прогнозування придатності кандидатів. На основі аналізу великих обсягів даних про успішних та неуспішних працівників системи можуть створювати моделі, які передбачають ймовірність успіху нових кандидатів.

Алгоритми рекомендацій використовуються для персоналізованого підбору вакансій та підтримки користувачів у прийнятті рішень. Шляхом аналізу попередніх взаємодій з платформою рекрутингу, системи можуть рекомендувати кандидатів, які найкраще відповідають конкретним вимогам та побажанням роботодавців.

Машинне навчання використовується для аналізу даних про вакансії, кандидатів та процеси найму. Алгоритми можуть виявляти закономірності та тренди, які допомагають роботодавцям оптимізувати стратегії пошуку талантів, пристосовувати вимоги до ринкових реалій та ефективніше конкурувати за кращих кандидатів.

Системи, що використовують машинне навчання, можуть покращувати свою продуктивність та точність з часом. Вони можуть вчитися на основі реакцій користувачів на рекомендації, аналізуючи, які кандидати були вибрані або відхилені, та адаптуючи свої алгоритми для кращої відповідності уподобанням роботодавців.

Машинне навчання також допомагає системам виявляти не лише формально зазначені навички, а й ті, що можуть бути виокремлені з досвіду кандидата. Аналізуючи кар'єрний шлях та досягнення, системи можуть пропонувати більш докладні та релевантні рекомендації.

Використання машинного навчання та алгоритмів рекомендацій дозволяє системам автоматизованого оцінювання кандидатів стати більш адаптованими та ефективними, сприяючи точнішому відбору та підбору перспективних фахівців для роботодавців.

### 1.2.3 Тести та асесмент-центри

Тести та асесмент-центри грають важливу роль у сучасних системах автоматизованого оцінювання кандидатів, забезпечуючи об'єктивність та структурованість при визначенні технічних та міжособистісних навичок кандидатів.

Системи використовують онлайн-тести для об'єктивної оцінки технічних знань кандидатів. Ці тести можуть включати питання різного рівня складності, від базових до більш продвинутих, щоб визначити рівень експертизи та придатності до конкретної посади.

Деякі системи використовують асесмент-центри, де кандидатам пропонуються симуляції реальних робочих сценаріїв. Це може включати в себе вирішення завдань або ситуацій, що відтворюють реальні виклики та обставини, з якими кандидати можуть стикатися на робочому місці.

Системи також використовують тести та асесмент-центри для вимірювання міжособистісних навичок та адаптивності кандидатів. Це може включати тести на лідерство, ефективність комунікації, спроможність до співпраці та інші аспекти, важливі для успіху в командному середовищі.

Деякі системи надають автоматизований зворотний зв'язок кандидатам після проходження тестів або асесмент-центрів. Це дозволяє кандидатам отримати більше інформації про свої сильні та слабкі сторони, що сприяє більш прозорому та справедливому процесу оцінювання.

Результати тестів та асесмент-центрів часто інтегруються з іншими етапами виробничого процесу найму. Це може включати в себе автоматизовану фільтрацію кандидатів, створення рейтингів придатності або навіть використання результатів для персоналізованих рекомендацій.

Тести та асесмент-центри стають необхідним інструментарієм для об'єктивного та структурованого відбору кандидатів, сприяючи якісному відбору та підбору талановитих фахівців для компаній у сучасному конкурентному ринку праці.

### 1.2.4 Інтеграція з платформами рекрутингу

Інтеграція з платформами рекрутингу є стратегічно важливою частиною сучасних систем автоматизованого оцінювання кандидатів, забезпечуючи зручний та ефективний процес найму для роботодавців та кандидатів.

Системи автоматизованого оцінювання можуть автоматично розміщувати вакансії на різних платформах рекрутингу та отримувати резюме кандидатів, які відповідають вимогам вакансій. Це дозволяє роботодавцям швидко заповнювати вакансії та отримувати доступ до різноманітних профілів.

Інтеграція дозволяє автоматично синхронізувати дані між системою автоматизованого оцінювання та платформами рекрутингу. Це сприяє уніфікації інформації, що дозволяє роботодавцям та рекрутерам працювати з актуальною та консистентною базою даних.

Системи можуть автоматично розсилати сповіщення та запрошення на співбесіди кандидатам, використовуючи інформацію з платформ рекрутингу. Це полегшує процес планування та управління календарями для всіх сторін, зменшуючи адміністративні труднощі.

Інтеграція дозволяє збирати дані про ефективність рекрутингових кампаній. Аналітика та звітність з платформ рекрутингу можуть бути автоматично інтегровані в систему оцінювання, надаючи роботодавцям стратегічні дані для оптимізації своїх стратегій найму.

Інтеграція з платформами рекрутингу дозволяє створювати персоналізовані рекомендації для кандидатів на основі їхнього взаємодії з різними вакансіями та рекрутинговими заходами. Це підвищує ефективність та точність відбору.

Інтеграція з платформами рекрутингу в системи автоматизованого оцінювання кандидатів робить найм ефективнішим, прискорює процес та

забезпечує більшу прозорість для всіх учасників, сприяючи успішному залученню та відбору талановитих фахівців.

### 1.2.5 Відео-інтерв'ю та аналіз мови тіла

Використання відео-інтерв'ю та аналізу мови тіла в системах автоматизованого оцінювання кандидатів додає нові можливості для глибшого розуміння професіоналізму та особистих якостей кандидатів.

Відео-інтерв'ю надає можливість роботодавцям та рекрутерам взаємодіяти з кандидатами в режимі реального часу. Це дозволяє отримати більше інформації про комунікативні навички, виразність та професійну зовнішність кандидата.

Системи автоматизованого оцінювання можуть використовувати технології аналізу мови тіла для оцінки невербальної комунікації кандидата. Виявлення жестів, міміки та інших невербальних елементів може допомогти розкрити додаткову інформацію про емоційний стан та взаємодію.

Відео-інтерв'ю дозволяють оцінювати навички публічних виступів та ефективність спілкування. Рекрутери можуть оцінювати якість відповідей, виразність та логічність висловленого.

Додатково, інтелектуальні алгоритми можуть бути використані для аналізу мови тіла та виявлення паттернів, які можуть бути невидимі для людського ока. Це може включати аналіз тону голосу, частоти жестів та інших аспектів невербальної комунікації.

Системи можуть зберігати записи відео-інтерв'ю для подальшого огляду та порівняння. Це дозволяє роботодавцям більш об'єктивно оцінювати кандидатів та враховувати різні аспекти їхньої професійної та особистісної якості.

Зберігання та обробка відео-інтерв'ю повинні дотримуватися стандартів безпеки та конфіденційності, щоб забезпечити захист особистих даних кандидатів.

Використання відео-інтерв'ю та аналізу мови тіла в системах автоматизованого оцінювання кандидатів робить найм більш ефективним та об'єктивним, надаючи більше можливостей для деталізованого вивчення професійних та особистісних якостей кандидатів.

## 1.3 Аналіз сучасних тенденцій системи оцінки кандидатів

Сфера оцінки кандидатів перебуває у постійному розвитку та вдосконалення. Однією з головних тенденцій розвитку системи оцінки кандидатів є цифровізація, а саме використання різних digital - технологій в оцінці кандидатів.

Одним із прикладів сервісів з автоматизації процедури оцінки кандидатів в Україні є програма підбору персоналу Test-Dome (Рисунок 1.1). Це онлайн-платформа для проведення тестів, спрямованих на оцінювання технічних навичок кандидатів. Призначена для використання в рекрутменті та відборі кандидатів, особливо у технічних галузях, таких як програмування та розробка. Ключові особливості цього сервісу можна представити у вигляді наступних критеріїв:

- Різноманітність тестів: Test-Dome пропонує широкий вибір тестів, що охоплюють різні професійні галузі та технічні стеки.
- Практичні тести: Тести можуть включати завдання з реальних сценаріїв роботи, що дозволяє кандидатам демонструвати свої навички на практиці.
- Аналітика та звіти: Після завершення тестів рекрутери отримують докладні звіти та аналітику з результатами кандидатів.
- Кастомізація тестів: Рекрутери можуть створювати власні тести або модифікувати існуючі відповідно до конкретних вимог.
- Оцінка м'яких навичок: Окрім технічних аспектів, платформа дозволяє також оцінювати м'які навички кандидатів, такі як комунікація та рішення проблем.
- Ефективне фільтрування кандидатів: Test-Dome дозволяє рекрутерам ефективно фільтрувати кандидатів, оцінюючи їхні навички на основі результатів тестів. Це може значно скоротити час, потрібний для відбору та оцінки кандидатів.
- Система рейтингу кандидатів: Платформа може надавати рейтинг кандидатів на основі їхніх результатів у тестах, що допомагає рекрутерам швидше визначити найбільш обіцяних кандидатів.

- Підтримка багатомовності: Test-Dome може бути використана для тестування кандидатів у різних країнах та регіонах, підтримуючи багатомовність тестів та інтерфейсу.

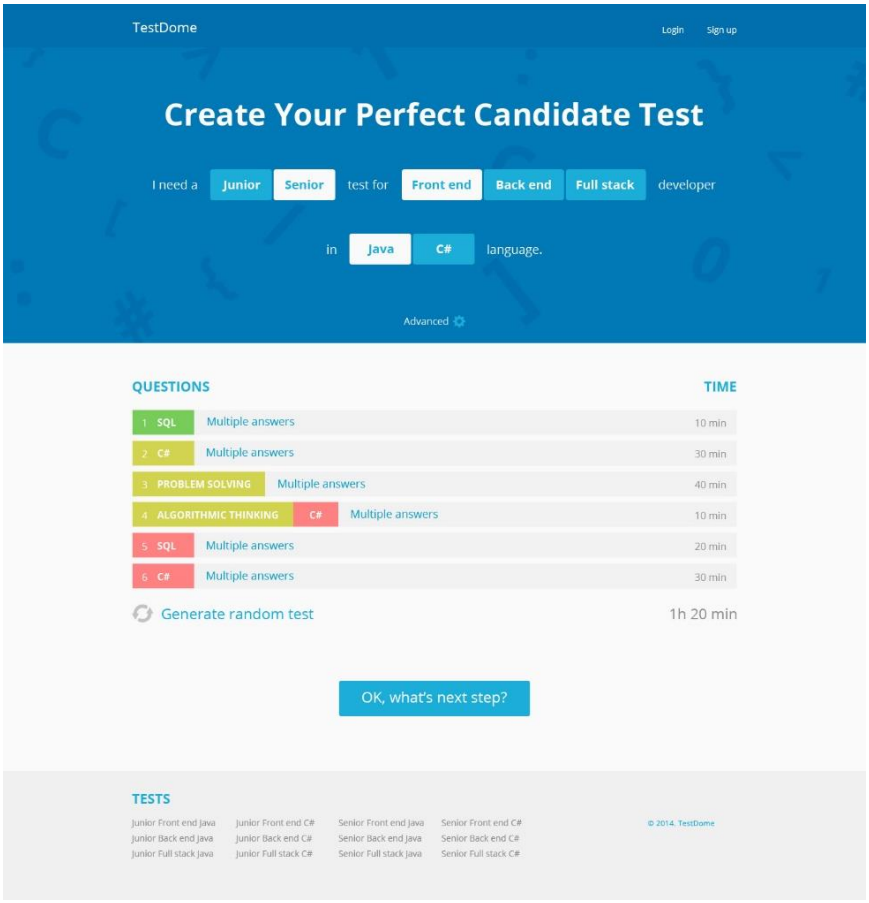



Рис. 1.1 Інтерфейс користувача Test-Dome

Розглянемо ще один інструмент автоматизації процедури оцінки кандидатів на Українському ринку, а саме eSkill (Рисунок 1.2). eSkill - це платформа для тестування навичок, яка надає широкі можливості для оцінки кандидатів у різних областях. Платформа дозволяє створювати та адмініструвати тести, оцінюючи різні компетенції, такі як технічні навички, мовні здібності, аналітичні вміння та інші. Ключові особливості цього сервісу можна представити у вигляді наступних критеріїв:

- Різноманітність тестів: Платформа має великий вибір тестів для різних професій і галузей, що дозволяє створювати настроєні тести для конкретних позицій.

- Аналіз результатів: eSkill надає докладні звіти та аналіз результатів тестів, що допомагає рекрутерам ефективно оцінювати кандидатів.
- Адаптивність: Платформа може адаптувати тести до різних рівнів складності, враховуючи потреби конкретної позиції.
- Бібліотека завдань: eSkill надає бібліотеку різноманітних завдань, які можна використовувати для створення тестів. Це дозволяє рекрутерам вибирати завдання, які найкраще відповідають конкретним потребам та вимогам певної позиції.
- Множинні типи тестів: Платформа підтримує різні типи тестів, включаючи тестування множинного вибору, завдання на виконання, а також завдання, пов'язані із реальними сценаріями роботи.
- Аналітика та звіти: eSkill забезпечує інструменти аналітики, які дозволяють ретельно аналізувати результати тестів, включаючи висновки щодо сильних та слабких сторін кандидатів.
- Персоналізовані тести: Рекрутери можуть створювати персоналізовані тести, враховуючи конкретні потреби та вимоги компанії.
- Інтеграція з іншими системами: Платформа може інтегруватися з іншими системами управління кадрами та іншими інструментами рекрутингу, що дозволяє забезпечити плавніший процес відбору.

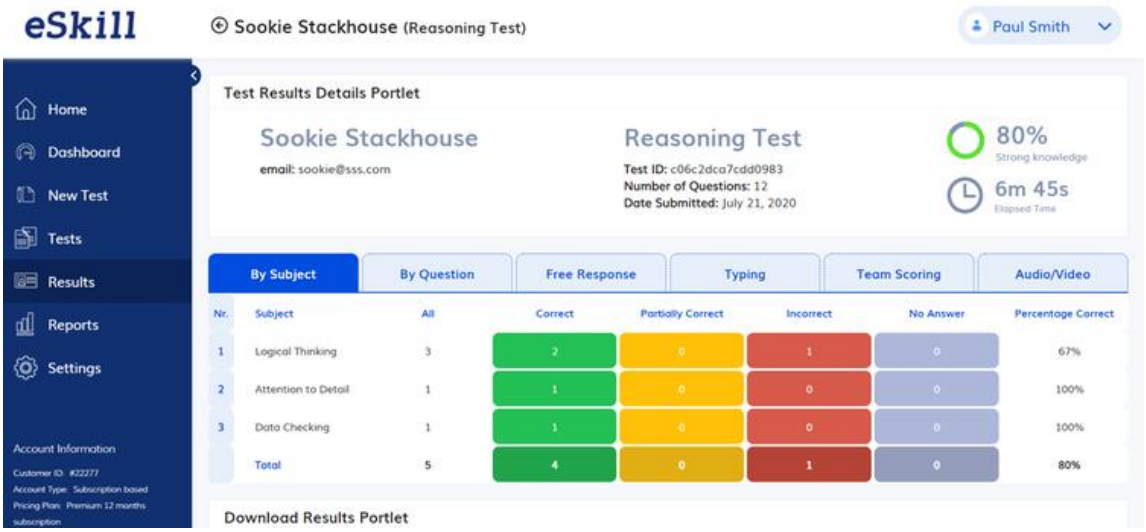



Рис. 1.2 Інтерфейс користувача eSkill

В якості ще одного інструменту автоматизації процедури оцінки кандидатів на Українському ринку можна представити Vervoe (Рисунок 1.3). Vervoe - це платформа для оцінки навичок, яка дозволяє підприємствам та рекрутерам створювати, адмініструвати та оцінювати завдання, щоб ефективно визначати навички та здатності кандидатів. Основна ідея полягає в тому, щоб перевіряти навички не тільки через резюме, але і через практичне виконання завдань, що може бути ближче до реальних умов роботи. Ключові особливості цього сервісу можна представити у вигляді наступних критеріїв:

- Створення тестів та завдань: Рекрутери можуть створювати різні типи завдань, включаючи відео відповіді, кодування, написання текстів та інші, щоб оцінити різні аспекти навичок кандидатів.
- Машинне навчання: Використовуючи машинне навчання, платформа може навчитися розпізнавати якісні відповіді та оцінювати їх згідно з певними критеріями.
- Звіти та аналітика: Після завершення тестів, рекрутери отримують докладні звіти та аналітику про результати кандидатів.
- Адаптивні тести: Vervoe може адаптувати тести залежно від відповідей кандидатів, що робить оцінку більш індивідуалізованою.
- Оцінка м'яких навичок: Крім технічних аспектів, Vervoe також дозволяє оцінювати м'які навички, такі як комунікація, рішення проблем, співпраця та інші.
- Забезпечення доступу до тестів: Кандидати можуть виконувати тести в зручний для них час та місце, а це може покращити доступність для рекрутингу на різних ринках.
- Опції відображення: Рекрутери можуть використовувати опції відображення, такі як анонімність кандидатів, щоб забезпечити більше об'єктивності в процесі оцінювання.
- Інтеграції з іншими системами: Vervoe може інтегруватися з іншими системами управління кадрами та іншими інструментами рекрутингу для оптимізації робочих процесів.

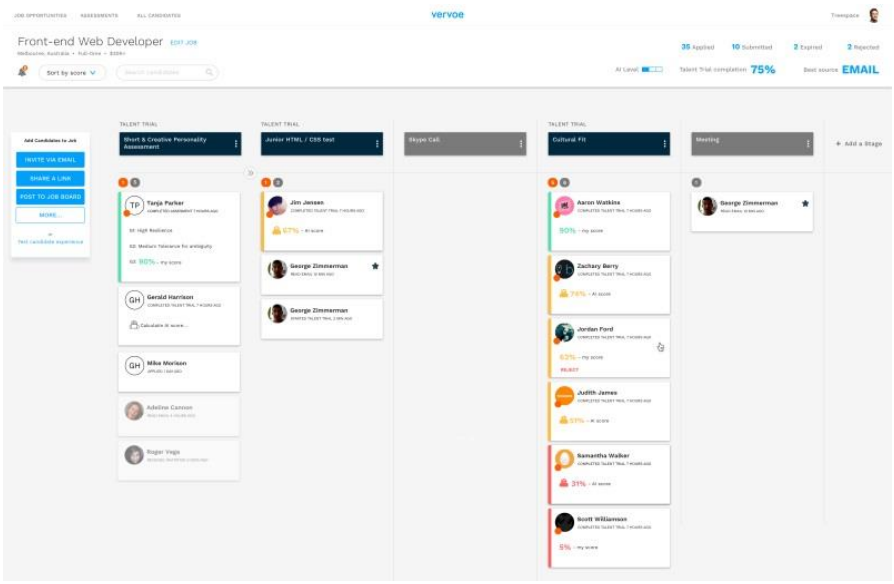



Рис. 1.3 Інтерфейс користувача Vervoe

Зрештою, серед проаналізованих аналогів можна визначити тенденцію автоматизації системи оцінки кандидатів, а саме використання онлайн-тестування, яке є інструментом первинної оцінки кандидатів та дозволяє оцінити претендента за заданими критеріями відбору (це можуть бути знання, здібності, мотивація кандидата та інше). Онлайн-тестування використовується для економії часу як фахівця з підбору персоналу, так і самого кандидата. В випадку використанні систем подібного типу немає потреби запрошувати кандидата в офіс для заповнення тесту вручну, він може зробити це у зручний час та в комфортне місце.

Для онлайн-опитувальників компанії можуть використовувати самостійно створені тести, наприклад з використанням Google-форм. Тести такого типу більше підійдуть для виявлення професійних компетенцій, рівня знань у потрібній сфері. Такі тести найчастіше перевіряються вручну в компанії силами самого спеціаліста з підбору персоналу[7].

Для психологічних тестів використовуються готові методики. У випадку з ними краще вдатися до послуг компаній, що спеціалізуються на онлайн-тестуваннях. Для оцінки кандидатів найбільше широко застосовуються тестові

сервіси SHL (вербальні та числові тести), Talent Q (числовий, вербальний та логічні тести) та Kenexa (логічні, технічні, вербальні та обчислювальні тести).

Згідно проаналізованої інформації різні компанії використовують різні способи автоматизації процесів оцінки кандидатів Розглянемо кілька прикладів.

Компанія Unilever для відбору кандидатів активно використовує інтелектуальні ігри та штучний інтелект. Відбір кандидатів відбувається за наступним алгоритмом:

1) Кандидати знаходять інформацію про актуальні вакансії компанії Unilever через Інтернет та соціальні мережі, відправляють свої дані у відділ підбору персоналу;
2) За допомогою платформи Phymetrics кандидати близько 20 хвилин грають у «розумні» ігри;
3) Якщо отримані результати виявляються достатніми для обраної кандидатами позиції, то кандидати проходить інтерв'ю через програму HireVue, де їх відповіді записуються і потім аналізуються з за допомогою технології штучного інтелекту;
4) Після успішного проходження попередніх етапів кандидати запрошуються до офісу компанії, щоб провести повноцінний робочий день, за результатами якого менеджер визначає, підходять кандидати на позиції чи ні [1].

Іншим прикладом використання digital-технологій в оцінці кандидатів може послужити практика компанії Nestle, у якій активно використовуються алгоритми онлайн-тестування для оцінки кандидатів. Тести компанії Nestle використовуються для первинної оцінки претендентів, зокрема, навичок розуміння числових та тестових даних. Компанія використовує випробування CEB SHL. Психометричні SHL-тести – один з digital-інструментів для відбору персоналу, що дозволяє відсіяти невідповідних кандидатів ще до співбесіди. SHL-тести не перевіряють знання претендентів, вони оцінюють їхні інтелектуальні здібності [2].

Таким чином, основною тенденцією у системі оцінки кандидатів є застосування автоматизованих методів оцінки, які вбудовуються у вже існуючі системи. Автоматизовані методи оцінки кандидатів дозволяють оптимізувати час

спеціаліста з підбору персоналу, що сприяє підвищенню пропускної здатності HR-відділу та відповідно гарантує кращий результат процесу підбору кандидату на відповідну посаду.

За статистикою [3], більшість рекрутерів та HR працюють за універсальною схемою, такою як, наприклад, публікація актуальних вакансій на job-порталах, пошук співробітників по базах, затвердження їх профілів із замовниками/роботодавцями, організація співбесіди, тестування тощо. На кожному з цих етапів існує великий шанс втрати кандидата по причинам втрати великої кількості часу на «ручну» і часто невиправдану працю (заклад кандидатів на базу, переписування інформації, пошук комунікаційної стрічки з кандидатом у листуванні тощо).

По ходу справи щось видаляється, не завантажується або пропадає в невідомому напрямку, т.к. завжди є місце людського чинника, з яким, на жаль, нічого не вдієш. У свою чергу автоматизація процесів оцінювання кандидатів ці ризики виключає – все зберігається дбайливо та «по поличках». Наприклад, ніякий форс-мажор не знищить важливий лист із ще важливішим резюме в ньому.

Автоматизована система контролює весь процес найму – від звернення потенційного кандидату до управління графіком співбесід з відповідними претендентами. Це робить рекрутинг прозорим і більш оперативним, фокусуючи увагу рекрутера на тих людях, які дійсно заслуговують на увагу.

## 1.4 Визначення переваг та недоліків програмного забезпечення предметної галузі

Після аналізу програмного забезпечення, що реалізує автоматизацію предметної галузі, можна визначити його переваги та недоліки. Наведемо їх у таблиці 1.1.

Таблиця 1.1

Переваги та недоліки програмного забезпечення предметної галузі

| Назва | Test-Dome | eSkill | Vervoe |
|---|---|---|---|
| Розробник | Test Dome Company | eSkill Company | Vervoe Company |
| Переваги | Різноманітність тестів, Практичні завдання, Ефективний відбір кандидатів, Аналітика та звіти, Підтримка багатомовності. | Різноманітність тестів, Адаптивність та гнучкість, Тестування онлайн, Аналітика та звіти, Інтеграція з іншими системами. | Широкий спектр завдань, Практичні тести, Адаптивність, Оцінка м'яких навичок, Зручний процес тестування. |
| Недоліки | Вартість, Неспецифічність для деяких галузей, Можливість неточності, Недостатня оцінка м'яких навичок. | Вартість, Обмеження в оцінці м'яких навичок, Можливість неточності, Потреба в Інтернет-з'єднанні. | Вартість, Обмеження в оцінці великих груп кандидатів, Можливість неточності, Потреба в Інтернет-з'єднанні. |

Як можна зробити висновок – основними недоліками існуючих систем є вартість, що робить їх впровадження недоцільним для невеликих локальних компаній та можливість неточності результатів оцінювання. Тож під час розробки

програмного продукту слід зосередитись на низьких витратах на систему та максимізації точності результатів оцінювання.

## 1.5 Формулювання завдання

Проаналізувавши предметну галузь, можна визначити завдання, що стоїть перед нами, а саме розробка системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму**.** На підставі цього, перед інформаційною системою, що розробляється ставиться мета: скорочення витрат часу менеджером із персоналу на отримання та аналіз інформації з кадрового плану, резюме та даних кандидатів, автоматизоване формування звітності, а також зниження ймовірність помилок під час процесу підбору кандидатів.

Відповідно до цього, проектована інформаційна система повинна реалізовувати наступний функціонал:

1. Можливість завантаження датасету кандидатів
2. Можливість навчання моделі нейронної мережі на основі датасету
3. Можливість ведення запиту рекрутера щодо параметрів необхідного кандидата
4. Можливість автоматичної класифікації кандидатів за наведеними параметрами
5. Автоматизований аналіз отриманих результатів.
6. Виведення рекрутеру необхідної кількості резюме, найбільш відповідних його критеріям пошуку.

Таким чином, потрібно розробити інформаційну систему, яка б дозволила автоматизувати початкові процеси оцінювання кандидатів для роботодавців та значно скоротила об’єм роботи менеджеру з персоналу за рахунок скорочення часу на введення та обробку інформації та могла взаємодіяти з великими об’ємами даних для аналізу.

# 2 АНАЛІЗ МЕТОДІВ АВТОМАТИЗОВАНОГО ОЦІНЮВАННЯ ТА ФОРМУВАННЯ АЛГОРИТМУ РОБОТИ

## 2.1 Загальні принципи автоматизації оцінювання кандидатів

Зазвичай автоматизація оцінювання кандидатів означає оптимізацію процесу. У нашому випадку завдання може бути сформульоване як підвищення ефективності пошуку кандидатів. Ефективність у даній галузі виявляється через знаходження найбільш підходящих вакансії кандидатів із мінімальними витратами ресурсів.

Таким чином, система пошуку кандидатів має зіставляти кандидатів та відкриті позиції (вакансії). Очевидно, що і те, й інше можна представити у вигляді набору навичок. Якщо обидві ці сутності (вакансії та кандидатів) представити через навички, ми можемо порівняти, наскільки кандидат відповідає відкритій позиції з точки зору набору навичок. Саме такий підхід застосовує і рекрутер на першому етапі пошуку: у нього є позиція, описана у вигляді набору навичок, і він намагається знайти такий же набір у кандидатах, використовуючи внутрішні бази даних або зовнішні сервіси. Це частина процесу відбору кандидатів, яку потенційно можна автоматизувати без істотного погіршення якості найму.

З погляду формулювання завдання все просто, але виникає багато питань: як висловити відповідність навичок для кандидата, як оцінити, наскільки сильно набір навичок відповідає конкретній людині, де взяти цих кандидатів.

Основне завдання тут – знайти ефективний спосіб відображення відповідності кандидатів та навичок. Іншими словами, ми маємо зробити репрезентацію кандидатів через набір навичок. Відповідно, на виході ми очікуємо на деякий вектор, який описує положення кандидата у просторі навичок.

## 2.2 Метод One-Hot Encoding

Принцип даного методу в тому, щоб уявити кандидатів та їх навички у вигляді деякої матриці, де значення 1 - має навичку, 0 - не має навичку (Рисунок 2.1). Розглянемо математичну модель цього методу:

Припустимо, у нас є множина категорій *C = {C1,C2,…,Cn}*. Якщо у нас є категоріальний об'єкт, який відповідає категорії *Ci*, то його One-Hot Encoding виглядає так (2.1):

$$\text{One-Hot}(c_i) = [0, 0, \ldots, 1, \ldots, 0] \tag{2.1}$$

де всі елементи вектора рівні 0, окрім *i*-го, який рівний 1.

Наприклад, якщо у нас є три категорії *C = {A,B,C}*, то їх One-Hot Encoding може виглядати так (2.2):

$$\begin{aligned} \text{One-Hot}(A) &= [1, 0, 0] \\ \text{One-Hot}(B) &= [0, 1, 0] \\ \text{One-Hot}(C) &= [0, 0, 1] \end{aligned} \tag{2.2}$$

Цей метод часто використовується при роботі з машинним навчанням та обробці природної мови для подання категоріальних змінних у вигляді, зручному для алгоритмів машинного навчання.

Цей підхід простий, але має ряд недоліків. Основна проблема даного підходу в тому, що навички в отриманому за його допомогою просторі будуть ортогональні один одному, і ми не зможемо порівняти їхню схожість між собою. У поставленому завданні, швидше за все, не так важливо розрізняти такі навички, як Java7 і Java8, при цьому добре б відрізнити їх від інших навичок, абсолютно не пов'язаних з

позицією Java-девелопера. При такому підході Java7 від Java8 відрізнятиметься також, як Java7 від Python.

Крім цього, недоліком цього підходу є те, що ми не можемо відрізнити специфічні навички від популярних, які поширені по всій нашій вибірці. Це вноситиме певний шум у наш пошук і заважатиме розрізняти кандидатів і виділяти схожих.

Можна трохи модифікувати цей метод — використовувати не бінарні оцінки, а виважені, що ґрунтуються на частоті народження у вибірці загалом і в окремих документах. Але в цьому випадку ми, як і раніше, не зможемо оцінити, наскільки схожі навички між собою.

| | skill1 | skill2 | ... | skillN-1 | skillN |
|---|---|---|---|---|---|
| User 1 | 1 | 0 | ... | 1 | 0 |
| User 2 | 1 | 0 | ... | 1 | 0 |
| ... | 0 | 1 | ... | 0 | 1 |
| User N | 0 | 0 | ... | 1 | 0 |

Рис. 2.1 Метод One-Hot Encoding на прикладі оцінювання кандидатів

## 2.3 Метод матричної факторизації

Представлення кандидатів у просторі, де кожна навичка — це координата простору, є надмірною, оскільки частина навичок майже не відрізняються одна від одної. Відповідно, можна зійти згрупувати навички в деякі фактори/компоненти/латентні ознаки. Одним із підходів, що дозволяють знаходити такі компоненти, є група методів матричної факторизації (Рисунок 2.2). Розглянемо математичну модель цього методу:

Нехай $R$ - це матриця оцінок кандидатів розмірністю $m \times n$, де $m$ - кількість роботодавців, $n$ - кількість характеристик (навичок або властивостей кандидатів).

Метод матричної факторизації намагається апроксимувати цю матрицю як добуток двох менших матриць $P$ і $Q$, де $P$ - матриця роботодавців розмірністю $m \times k$ і $Q$ - матриця характеристик кандидатів розмірністю $k \times n$, де $k$ - це кількість факторів.

В такому випадку матриця факторів P (2.3):

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mk} \end{bmatrix} \quad (2.3)$$

Матриця факторів Q (2.4):

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{k1} & q_{k2} & \cdots & q_{kn} \end{bmatrix} \quad (2.4)$$

Отже, матрична факторизація може бути представлена як: $R \approx PQ$, де $\approx$ позначає апроксимацію. Задача полягає в знаходженні оптимальних матриць $P$ і $Q$, щоб якнайкраще апроксимувати початкову матрицю $R$, яка містить оцінки роботодавців для кандидатів.

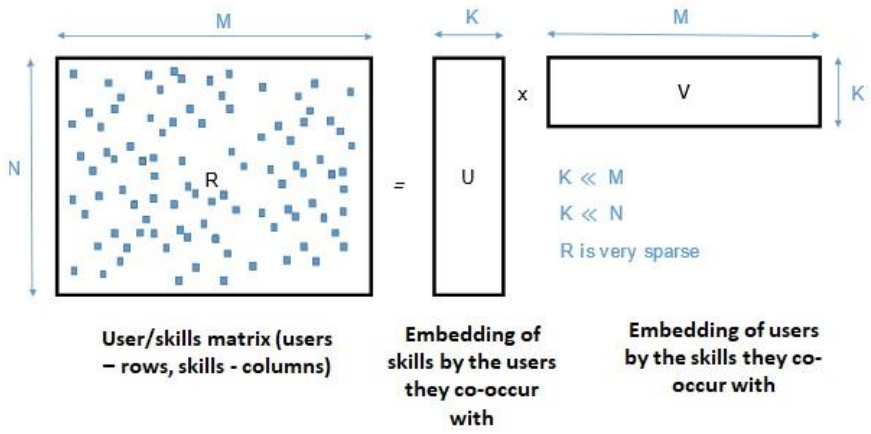



Рис. 2.2 Метод матричної факторизації на прикладі оцінювання кандидатів

В нашому випадку оцінювання кандидатів матрицю User-Skills ми представляємо у вигляді двох матриць, добуток яких наближатиме вихідну матрицю. Перша матриця представляє латентні ознаки для кожного з кандидатів у сконструйованому латентному просторі. Друга матриця - латентні ознаки навичок (skills embedding). Для простоти можна цей простір сприймати як простір інтересів кандидатів — кожного кандидата можна описати за допомогою його відповідності тим чи іншим інтересам, так само й кожну навичку можна описати через величину того, наскільки вона задовольняє той чи інший інтерес кандидата.

Плюс цього підходу в тому, що він стійкіший до популярних навичок, знижує розмірність і дозволяє оцінювати відповідність між навичками. Тобто, репрезентації у просторі зменшеної розмірності зберігають деякі залежності між сутностями, які спостерігалися у вихідному просторі. Мінус — зазвичай потрібно чимало даних, щоб зробити гарну репрезентацію. Крім того, ми не можемо відстежити складні залежності між навичками та кандидатами.

Щоб вивчити більш складні репрезентації кандидатів та навичок, потрібні потужніші методи, і група таких методів може ґрунтуватися на глибокому навчанні.

## 2.4 Колаборативна фільтрація з глибоким навчанням

У рекомендаційних системах часто використовують колаборативну фільтрацію, засновану на нейронних мережах. Принцип застосування такий самий, що й у матричній факторизації — конструювання простору латентних ознак та поміщення в цей простір кандидатів та їх навичок. Тільки відбувається це вже в ході навчання моделі нейронної мережі, яка вивчає латентне подання кожної сутності, вирішуючи при цьому, наприклад, завдання прогнозування релевантності навички для кандидата (у supervised постановці завдання), або ітеративно коригуючи латентний простір таким чином, щоб навички, що часто зустрічаються разом у кандидатів знаходилися близько один до одного в отриманому просторі, а навички, які рідко разом описують одного фахівця, були б далекі один від одного в цьому просторі (не supervised постановка завдання). Отримані уявлення кандидатів та навичок називають ембедингами. Розглянемо детальніше математичну модель цього методу:

Введемо матрицю представлення кандидатів $C$ розмірності $m \times d$, де $m$ - кількість кандидатів, а $d$ - розмірність векторів, що представляють кандидатів.

Також введемо матрицю представлення роботодавців $E$ розмірності $n \times d$, де $n$ - кількість роботодавців.

Оцінка придатності кандидата $c_i$ для роботодавця $e_j$ може бути обчислена як скалярний добуток векторів кандидата і роботодавця (2.5):

$$s_{ij} = C_i \cdot E_j^T \tag{2.5}$$

де $C_i$ - вектор, який представляє кандидата $c_i$, $E_j$ - вектор, який представляє роботодавця $e_j$, а $T$ позначає транспонування.

Визначимо функцію втрат, яка вимірює різницю між прогнозованими та фактичними оцінками придатності. Мета - мінімізувати цю втрату під час тренування моделі (2.6):

$$L = \frac{1}{2}\sum_{i,j}(r_{ij} - s_{ij})^2 \quad (2.6)$$

де $r_{ij}$ - фактична оцінка придатності, а $s_{ij}$ - прогнозована оцінка придатності. Використовуючи глибокий навчальний підхід, оптимізуємо параметри матриць $C$ і $E$ для мінімізації функції втрат. Використовуються методи оптимізації, такі як стохастичний градієнтний спуск або алгоритми оптимізації на основі зворотного поширення помилок.

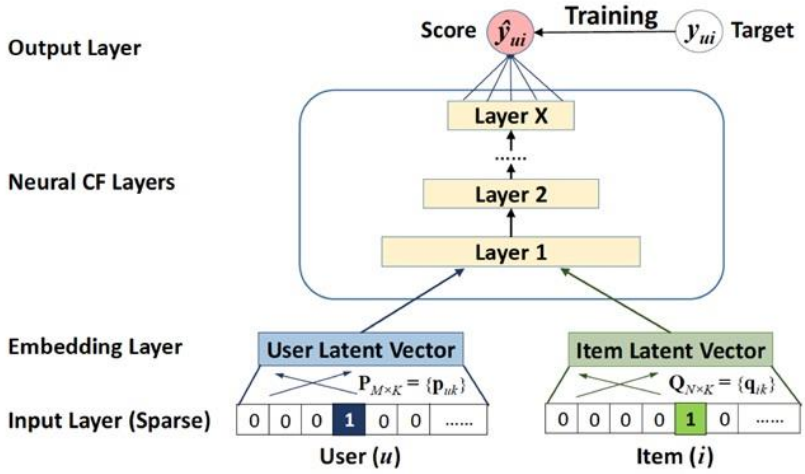



Рис. 2.3 Метод колаборативної фільтрації з глибоким навчанням

Цей метод дозволяє вирішити ті завдання, які раніше не піддавалися. З'являється більше можливостей отримання простору, враховує специфіку конкретних даних, можна використовувати різні архітектури мереж, що у результаті дозволяє відбивати складніші взаємозв'язку між навичками і кандидатами. Але є одна проблема — потрібно чимало даних, щоб отримати гарну виставу.

## 2.5 Метод K – Means

Метод K – means (Рисунок 2.4) — це алгоритм кластеризації, що використовується в оцінюванні кандидатів для групування їх на основі схожих характеристик та вмінь. Основною метою є розділення набору даних на K кластерів, де кожен кластер має свій центр.

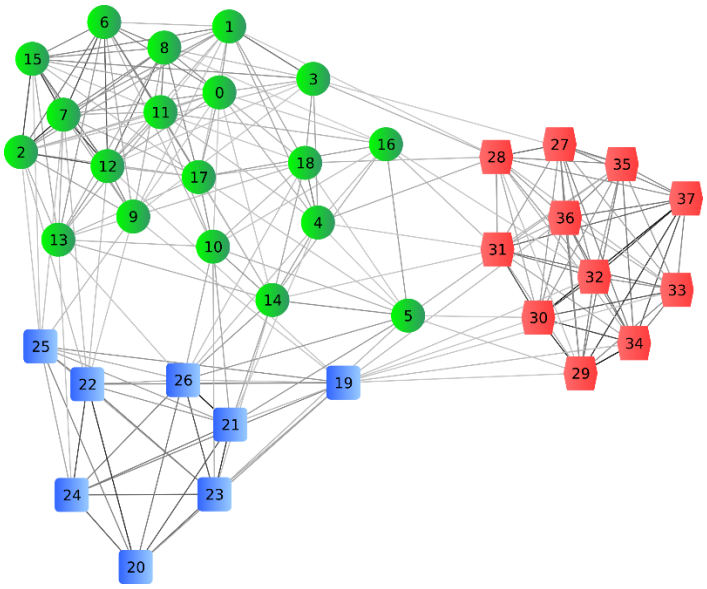



Рис. 2.4 Метод K – Means

Цей метод є потужним інструментом в оцінюванні кандидатів, де важливо виділити групи зі схожими характеристиками. Цей алгоритм кластеризації розділяє кандидатів на K груп, забезпечуючи можливість більш ефективного та об'єктивного оцінювання.

Метод починається з визначення попередньої кількості кластерів (K), ініціалізації початкових центрів кластерів та вибору критерію відстані між кандидатами та центрами. Наступною є ітеративна процедура, де кожен кандидат призначається до того кластеру, центр якого знаходиться найближче. Після цього оновлюються центри кластерів на основі призначених кандидатів. Розглянемо детальніше математичну модель цього методу:

Припустимо, у нас є набір даних $X = \{x_1, x_2, ..., x_n\}$, де $x_i$ - це кандидати з їхніми характеристиками. Також маємо $K$ кластерів, і кожен кластер має свій центр $c_k$. Першим кроком проведемо ініціалізацію центрів кластерів (2.7):

$$c_k^{(0)} \text{ для } k = 1, 2, ..., K \tag{2.7}$$

Кожна точка $x_i$ призначається до кластеру $k$, якщо відстань між $x_i$ та $c_k$ є мінімальною (2.8):

$$\min_k ||x_i - c_k||^2 \tag{2.8}$$

Новий центр кластеру $c_k$ обчислюється як середнє арифметичне всіх точок, призначених до цього кластеру (2.9):

$$c_k^{(t+1)} = \frac{1}{|S_k|} \sum_{i \in S_k} x_i \tag{2.9}$$

де $S_k$ - множина точок, призначених до кластеру $k$.

Кроки 2 і 3 повторюються до збіжності, коли призначення точок до кластерів та центри кластерів залишаються стабільними або до досягнення максимальної кількості ітерацій. Цей процес мінімізує функціонал внутрішньокластерної варіації і призводить до ефективного розділення кандидатів на групи за схожістю їхніх характеристик.

Цей метод відзначається своєю ефективністю, забезпечуючи можливість групувати кандидатів зі схожими навичками та характеристиками. Дозволяє виокремити групи кандидатів для об'єктивного оцінювання, що робить його корисним інструментом в рекрутингу та оцінюванні персоналу.

Цей метод може бути використаний в різних областях, таких як рекрутинг, оцінка компетенцій та кар'єрне консультування, де важлива систематизація процесу відбору та оцінювання кандидатів.

## 2.6 Методи репрезентації графів

Ще одна група методів, що дозволяє розв'язувати задачі репрезентації сутностей - репрезентація графів.

Загалом подання взаємозв'язку кандидатів та навичок у вигляді графа здається досить природним. Наприклад, можна уявити граф, де вузли – кандидати, а зв'язки – наявність спільних навичок у кандидатів. Або граф навичок, де зв'язки – приналежність навичок до одного кандидата. Інший варіант - граф кандидати-навички - двомодальний граф, де кожен вузол належить або до однієї сутності, або до іншої. Але більшість методів графової репрезентації працює з одномодальними графами, тому зазвичай двомодальні графи слід трансформувати у граф, де вузли представлені одним видом сутностей.

Особливі характеристики графа, важливі з погляду репрезентації структури графа - однорідність і структурна схожість вузлів.

Вузли - наприклад кандидати можуть бути чимось схожі між собою, перебувати в одному ком'юніті, розділяти спільні інтереси, працювати в одній компанії або мати інші однакові характеристики - за це відповідає характеристика однорідності. З іншого боку, вузли різних груп можуть об'єднуватися тим, що відіграють у своїх групах однакову роль — лідери, помічники лідерів, зберігачі інформації, комунікатори, аутсайдери. Якби ми хотіли порівняти два графи, ми могли б зрозуміти, що лідери в одному графі відіграють ту ж роль, що й лідери в іншій — це те, що називається структурною схожістю.

Методи графової репрезентації так чи інакше намагаються конструювати простір з урахуванням як однорідності, і структурної еквівалентності графа.

### 2.6.1 Графова факторизація

Насамперед розглянемо метод, заснований на графовій факторизації (Рисунок 2.5).

Принцип дії даного методу аналогічний матричній факторизації: необхідно трансформувати граф в матрицю перетинів, тобто. якщо два кандидати мають

загальні навички – 1, якщо ні – 0. На жаль, при використанні подібних моделей ми втрачаємо структурну еквівалентність вузлів.

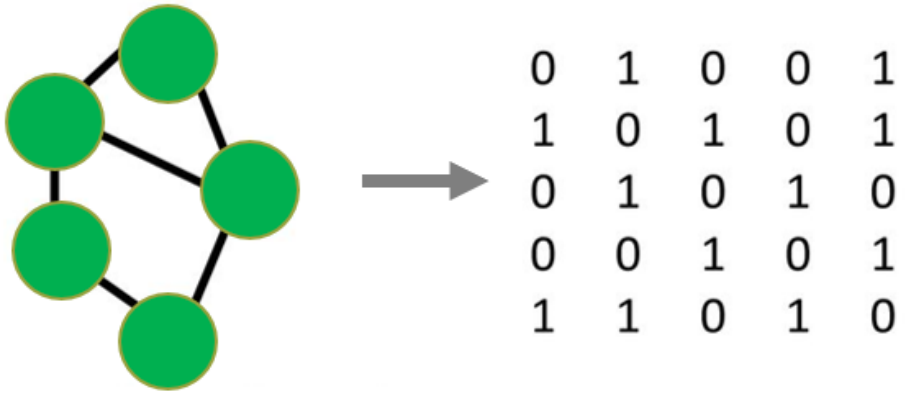



Рис. 2.5 Метод графової факторизації

### 2.6.2 Методи, що засновані на нейронних мережах

Ця група методів заснована на випадковому виділенні підграфів (частин графа, які самі розглядаються як окремі графи) та навчанні спроможності передбачати скупчення вузлів у підграфі. Іншими словами, ми виділяємо деякі шматочки графа і намагаємося вивчити модель передбачати вузлом підграфа те, що буде навколо нього, які зв'язки у нього з іншими вузлами. Способів виділяти підграфи у великому графі також може бути багато. Така модель дозволяє робити репрезентацію як інформації про структурну еквівалентність вузлів, так і їх приналежність до деяких загальних груп. Ця група методів дуже схожа на методи, які застосовуються для репрезентації текстів - w2v (skip-gram), doc2vec.

Крім того, у навчанні репрезентації бере участь інформація про загальну структуру графа та характеристики вузла. Основним нововведенням даних методів є те, що модель нормалізує значення кожного вузла таким чином, щоб позиція в латентному просторі двох вузлів була тим ближче, чим схожі структурні ролі цих вузлів у підграфі.

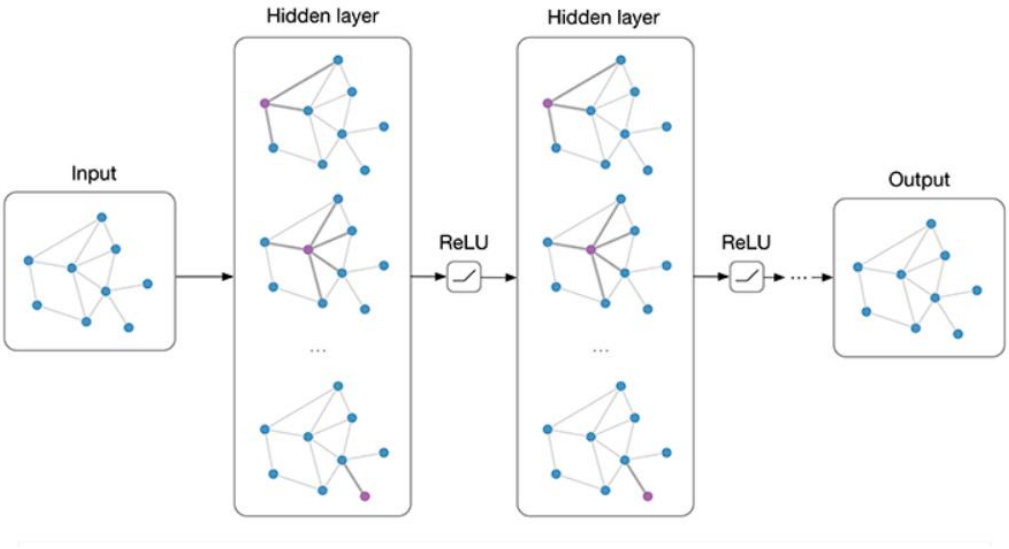
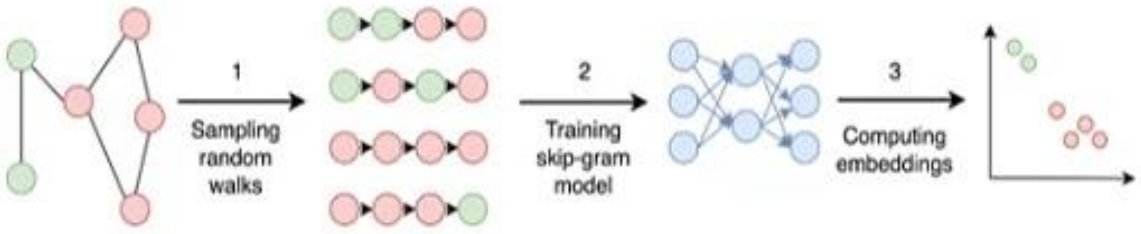



Рис. 2.6 Алгоритм репрезентації на основі нейронної мережі

## 2.7 Вибір алгоритму оцінювання

Після аналізу типів методів, якими можна виконати поставлену задачу, необхідно безпосередньо визначити алгоритм, на основі якого проектована система буде проводити оцінювання кандидатів.

Для виконання поставленого завдання було обрано алгоритм Word2Vec ( Рисунок 2.7 )




.

Рис. 2.7 Принцип роботи алгоритму Word2Vec

Word2Vec - це алгоритм векторного представлення слів, розроблений для конвертації слів або термінів у вектори числового простору. Це основний

інструмент для роботи з векторними представленнями слів в області обробки природної мови (Natural Language Processing - NLP). В контексті автоматизованого оцінювання кандидатів, Word2Vec може бути використаний для перетворення текстових даних (наприклад, описів проектів чи навичок кандидата) у вектори, які можна використовувати для порівняння та ранжування. Розглянемо математичну модель цього алгоритму:

Нехай:

$V$ - словник (всі унікальні слова в корпусі);

$N$ - розмір словника;

$D$ - розмірність вектору прихованого шару;

$W$ - матриця ваг, що визначає вхідні ваги між вхідним словом і прихованим шаром;

$W^i$ - матриця ваг, що визначає ваги між прихованим шаром і виходом;

$x$ - вектор входу (вектор векторного представлення слова);

$h$ - вектор прихованого шару;

$y$ - вектор виходу (вектор векторного представлення слова в контексті).

Тоді ймовірність входження слова $w_t$ у контекст слова $w_c$ визначається логістичною функцією softmax (2.10):

$$P(w_t|w_c) = \frac{e^{y_t}}{\sum_{i=1}^{N} e^{y_i}} \tag{2.10}$$

де $y_t$ - це $t$-ий елемент вектору у для слова $w_t$.

Логістична функція втрат для нього виглядає так (2.11):

$$\mathcal{L} = -\sum_{t=1}^{N} \sum_{c \in C_t} \log P(w_t|w_c) \tag{2.11}$$

де $C_t$ - множина контекстних слів для слова $w_t$.

Ваги матриць $W$ та $W^i$ оновлюються за допомогою градієнтного спуску зі стохастичним градієнтом (2.12):

$$\theta_t = \theta_t - \eta \frac{\partial \mathcal{L}}{\partial \theta_t} \tag{2.12}$$

де $\theta_t$ - параметри для слова $w_t$, $\eta$ - швидкість навчання.

Основні ідеї алгоритму Word2Vec можна представити як:

1. Локальність семантики - Word2Vec базується на припущенні, що семантично близькі слова часто зустрічаються разом у тексті. Наприклад, слова "Java" і "Джава" мають схожі контексти та можуть взаємозамінюватися у реченнях.
2. Контекстне перетворення - Word2Vec використовує контекстне навчання, де модель прогнозує ймовірність зустрічі слова в конкретному контексті або навпаки. Для цього використовується нейронна мережа з одним прихованим шаром.
3. Continuous Bag of Words (CBOW) та Skip-Gram - Word2Vec має дві основні варіації: CBOW, яка намагається передбачити слово на основі його контексту, і Skip-Gram, яка на вході отримує слово і намагається передбачити його контекст. Обидві варіації дозволяють отримувати векторні представлення слів.
4. Векторні представлення - Після тренування моделі кожному слову привласнюється унікальний вектор у просторі чисел. Ці вектори можна використовувати для порівняння семантичної подібності між словами.
5. Аналогії та віднімання слів - Вектори можна використовувати для вирішення аналогій, що вказує на те, що векторні представлення слов дійсно ухвалюють семантичні зв'язки.

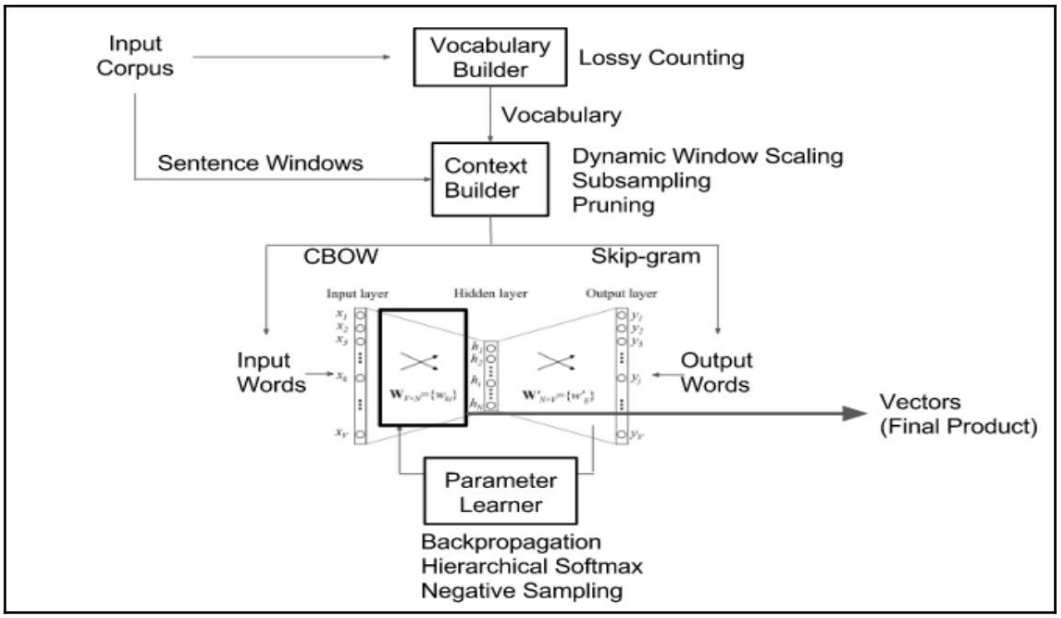



Рис. 2.8 Блок - схема роботи алгоритму Word2Vec

У контексті оцінювання кандидатів Word2Vec може використовуватися для векторизації текстових описів навичок чи проектів кандидата. Потім ці вектори можуть бути використані для порівняння та ранжування кандидатів за схожістю їхніх навичок до запитів рекрутерів. Використання Word2Vec у процесі найму може значно полегшити пошук та відбір кандидатів, особливо при аналізі великої кількості текстової інформації.

# 3 РОЗРОБКА АЛГОРИТМІЧНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ РЕЗУЛЬТАТІВ

## 3.1 Аналіз та вибір інструменту реалізації програмного забезпечення

В рамках дипломної роботи є потреба написання програмного продукту**,** який буде програмно реалізовувати процес оцінювання кандидатів для роботодавців в процесі найму**.** Для написання обумовленного програмного продукту необхідно вибрати інструмент реалізації, а саме мову програмування від якої залежатиме швидкість написання та роботи проектованого програмного продукту.

На сьогоднішній день існує велика кількість різних мов програмування і в кожного з них своя сфера застосування, але все ж для проведення аналізу на вибір кращої мови для написання проектованого програмного продукту, необхідно вибрати серверні мови, що найбільш актуальні для виконання поставленої задачі.

Розглянемо мови програмування що надають інструментарій для виконання поставленого завдання та обгрунтуємо переваги їх використання у контексті нашої розробки:

1) Python - Використання Python у розробці системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму на основі нейронної мережі має кілька обґрунтованих переваг:

1. Широкі можливості для обробки даних: Python має багатий екосистем та бібліотеки для роботи з обробкою даних, машинним навчанням та нейронними мережами. Наприклад, бібліотеки такі як TensorFlow, PyTorch та scikit-learn дозволяють легко реалізовувати та тренувати моделі нейронних мереж для завдань оцінювання кандидатів.
2. Зручність та читабельність коду: Python відомий своєю зручністю та читабельністю коду, що робить його ідеальним вибором для розробки складних систем. Це особливо важливо в області машинного навчання, де розробка та підтримка моделей може бути складною.

3. Велика та активна спільнота: Python має велику та активну спільноту розробників, що сприяє доступності різноманітних рішень, бібліотек та допомоги. Для задач нейронного мережування це робиться набагато простіше завдяки наявності великої кількості ресурсів та прикладів коду.
4. Інтеграція з іншими технологіями: Python легко інтегрується з різноманітними технологіями та інструментами, що може бути важливим для системи, яка використовує дані з різних джерел, таких як GitHub API, та проводить комплексний аналіз кандидатів.
5. Гнучкість та швидкість розробки: Python є мовою з високим рівнем абстракції, що полегшує розробку та експерименти з різними моделями. Це особливо важливо для задач, пов'язаних із штучним інтелектом, де експериментування з різними конфігураціями може бути ключовим.

Узагальнюючи, використання Python у розробці системи автоматизованого оцінювання кандидатів на основі нейронних мереж обгрунтоване його зручністю, доступністю та гнучкістю для реалізації складних завдань у галузі навчання машин та обробки даних.

2) C++ - Існує декілька аспектів, які можуть обґрунтовувати використання C++ у розробці системи автоматизованого оцінювання кандидатів для роботодавців на основі нейронних мереж:

1. Швидкодія: C++ відомий своєю високою швидкодією та ефективністю виконання, що робить його відмінним вибором для важких обчислювальних завдань, таких як навчання нейронних мереж. Використання C++ може покращити продуктивність системи, зокрема для великих обсягів даних чи складних алгоритмів.
2. Доступність бібліотек та інструментів: Існує кілька популярних бібліотек для роботи з нейронними мережами, які мають обгортки для C++, такі як TensorFlow та PyTorch. Використання цих бібліотек може значно спростити розробку та інтеграцію нейронних мереж у систему
3. Можливість оптимізації та паралельного виконання: C++ надає широкі можливості для оптимізації коду та використання механізмів паралельного

виконання. Це стає важливим в контексті розробки системи, яка обробляє великі об'єми даних та вимагає високої продуктивності.

4. Контроль над ресурсами: C++ надає великий контроль над ресурсами системи, такими як пам'ять та процесор. Це може бути корисним для оптимізації та управління ресурсами в системі, особливо у великих масштабах.
5. Широке використання у великих проектах: C++ традиційно використовується великими проектами та системами, і він може бути вибраним в разі інтеграції системи оцінювання кандидатів у вже існуючий корпоративний код або середовище.

Узагальнюючи, використання C++ у розробці системи оцінювання кандидатів може бути обґрунтованим, якщо пріоритетом є висока швидкодія, ефективність виконання та контроль над ресурсами.

3) Java – Використання Java у розробці системи автоматизованого оцінювання кандидатів для роботодавців на основі нейронних мереж може бути обґрунтовано з кількох причин:

1. Переносність: Java відома своєю переносністю, що дозволяє використовувати код на різних платформах без модифікацій. Це може бути важливо, особливо якщо система має працювати на різних операційних системах.
2. Широке застосування в корпоративному середовищі: Java широко використовується у корпоративних проектах, і вона може бути вибрана як технологічний стек для системи, що інтегрується в існуючі корпоративні структури.
3. Велика кількість бібліотек і фреймворків: Java має обширну екосистему бібліотек і фреймворків, які полегшують розробку та інтеграцію різноманітних компонентів, включаючи роботу з нейронними мережами.
4. Масштабованість: Java позначена своєю здатністю масштабувати великі системи. Для систем, які обробляють великі обсяги даних та мають велику кількість користувачів, це може бути ключовою перевагою.

5. Безпека: Java відома своєю високою рівні безпеки. Це може бути важливим аспектом для систем, які обробляють конфіденційні дані, такі як дані кандидатів.
6. Широкий вибір інструментів для розробки: Java надає розробникам широкий вибір інструментів для розробки, включаючи різноманітні інтегровані середовища розробки (IDE), системи управління версіями, інструменти для тестування та інші.

Враховуючи ці переваги, використання Java може бути обґрунтованим, особливо якщо важливі аспекти такі як переносність, масштабованість та безпека є пріоритетами для системи автоматизованого оцінювання кандидатів.

4) C# - Використання C# у розробці системи автоматизованого оцінювання кандидатів на основі нейронної мережі може бути обґрунтоване наступним чином:

1. Інтеграція з технологіями Microsoft: C# є мовою програмування, що інтегрується з технологіями Microsoft, такими як .NET Framework або .NET Core. Це важливо, якщо система пов'язана з іншими Microsoft-орієнтованими рішеннями.
2. Ефективна розробка для платформи Windows:C# часто використовується для розробки десктопних та веб-застосунків для платформи Windows. Якщо система призначена для внутрішнього користування в організації, це може забезпечити ефективну розробку та інтеграцію.
3. ASP.NET для розробки веб-застосунків: Якщо система вимагає веб-інтерфейсу, C# в поєднанні з ASP.NET може забезпечити потужні інструменти для розробки веб-застосунків та їхню ефективну інтеграцію з іншими компонентами.
4. Розширені можливості мови та зручний синтаксис: C# надає розробникам зручний та сучасний синтаксис, а також широкий набір вбудованих бібліотек та інструментів. Це може полегшити розробку та підтримку системи.

5. Підтримка для мови LINQ: Language-Integrated Query (LINQ) у C# дозволяє виразно та декларативно працювати з даними, що може полегшити роботу з об'єктами даних та запитами до баз даних.
6. Велика спільнота та розширені ресурси: C# має велику та активну спільноту розробників, а також багато ресурсів для навчання та підтримки. Це може бути корисним при розробці та удосконаленні системи.

З урахуванням цих аспектів, використання C# може бути обґрунтованим, особливо якщо система пов'язана з екосистемою Microsoft або вимагає розробки для платформи Windows.

Для аналізу характеристик можливих для використання технологій, складемо порівняльні таблиці для проведення порівняння за такими характеристиками як: Парадигми, Типізація, Керування пам'яттю та Об'єктно - орієнтованими можливостями.

Таблиця 3.1

Порівняльна характеристика за парадигмами

| Можливість | C# | Java | C++ | Python |
|---|---|---|---|---|
| Імперативна | + | + | + | + |
| Обїєктно-орієнтована | + | + | + | + |
| Функціональна | +/- | +/- | +/- | + |
| Рефлексивна | +/- | +/- | + | + |
| Узагальна | + | + | + | + |
| Логічна | - | - | - | - |
| Декларативна | +/- | - | +/- | + |

Таблиця 3.2

Порівняльна характеристика за типізацією

| Можливість | C# | Java | C++ | Python |
|---|---|---|---|---|
| Статична типізація | + | + | - | - |
| Динамічна типізація | + | - | + | + |
| Явна типізація | +/- | + | - | +/- |
| Неявна типізація | + | - | + | + |
| Неявне приведення | - | + | + | + |

Таблиця 3.3

Порівняльна характеристика за можливостями керування пам’яттю

| Можливість | C# | Java | C++ | Python |
|---|---|---|---|---|
| Створення об’єктів на стеку | + | - | - | - |
| Некеровані вказівники | + | - | - | - |
| Ручне керування пам’яттю | + | - | - | - |
| Збирання сміття | + | + | + | + |

Таблиця 3.4

Порівняльна характеристика за функціональними можливостями

| Можливість | C# | Java | C++ | Python |
|---|---|---|---|---|
| Декларації чистоти функцій | - | - | - | - |
| First class функції | + | - | + | + |
| Анонімні функції | + | + | + | + |
| Лексичні замкнення | + | + | + | + |
| Часткове застосування | - | - | + | + |
| Карування | + | - | + | + |

Таблиця 3.5

Порівняльна характеристика за об’єктно – орієнтованими можливостями

| Можливість | C# | Java | C++ | Python |
|---|---|---|---|---|
| Інтерфейси | + | + | + | + |
| Мультиметоди | +/- | - | - | - |
| Міксини | - | + | + | + |
| Множинне спадкування | - | - | - | + |

Щоб зробити вибір мови програмування для створення програмного продукту для реалізації автоматизації процесу оцінювання кандидатів для роботодавців в процесі найму, потрібно зрозуміти, можливості якої мови можуть задовольнити поточну потребу.

За статистикою та проведеним аналізом, було з’ясовано, що найбільше з наведених мов програмування для написання логіки для реалізації наведеного програмного забезпечення буде доречно використання Python, так як ПЗ створене на ньому простіше у написанні, має багато документації і має великий вибір бібліотек, які допоможуть створити якісний програмний продукт а також що має для нас критичну перевагу має швидший час виконання за аналоги.

## 3.2 Визначення алгоритму роботи системи

Перш ніж переходити до програмної реалізації системи, необхідно сформувати безпосередньо алгоритм, за яким вона буде працювати.

Основним завданням проектованої системи є автоматизація процесу оцінювання кандидатів для роботодавців в процесі найму. Розробимо алгоритм дій, що будуть виконувати поставлену задачу:

1) Використання Github API для формування датасету - Здійснюється взаємодія з GitHub API для отримання деталей користувачів, їх мов програмування та іншої важливої інформації. Формування датасету резюме на основі цих даних для подальшого навчання моделі.

2) Створення embedding для резюме користувачів - Для кожного користувача, отриманого з GitHub API чи, формується embedding на основі його резюме. Використання методу векторного представлення Word2Vec, для створення embedding. Цей метод дозволяє перетворити інформацію про підписки користувачів в векторні представлення, які можна використовувати для подальшої обробки.
3) Формування запиту рекрутерів - Рекрутери формують запити, визначаючи набір навичок, які вони шукають у потенційних кандидатах. Запити можуть містити мови програмування, технології, інструменти та інші ключові слова, що характеризують необхідні навички.
4) Пошук відповідності навичок і резюме - Здійснюється пошук резюме, які містять навички, вказані в запиті рекрутера. Використовуються інструменти, що аналізують тексти описів резюме, щоб визначити, чи відповідають вони запиту. Для виконання використовуються embedding резюме, отриманих на попередньому кроці.
5) Агрегація embedding резюме в запиті - Отримані embedding резюме, що відповідають запиту рекрутера, об'єднуються в один вектор запиту. Це може включати усереднення або інші методи агрегації векторів.
6) Визначення подібності між кандидатами та запитом - Вимірюється відстань між вектором запиту та векторами користувачів, які були отримані на попередньому кроці. Це може включати використання метрик відстані, таких як косинусна відстань, щоб визначити ступінь подібності.
7) Сортування та представлення результатів - Кандидати сортуються за зростанням відстані від вектору запиту. Результати представляються рекрутерам у вигляді відсортованого списку користувачів, які найбільше відповідають їхнім запитам.

Наведений алгоритм (Рисунок 3.1) повністю реалізує поставлену задачу, а саме автоматизацію процесу оцінювання кандидатів для роботодавців в процесі найму.

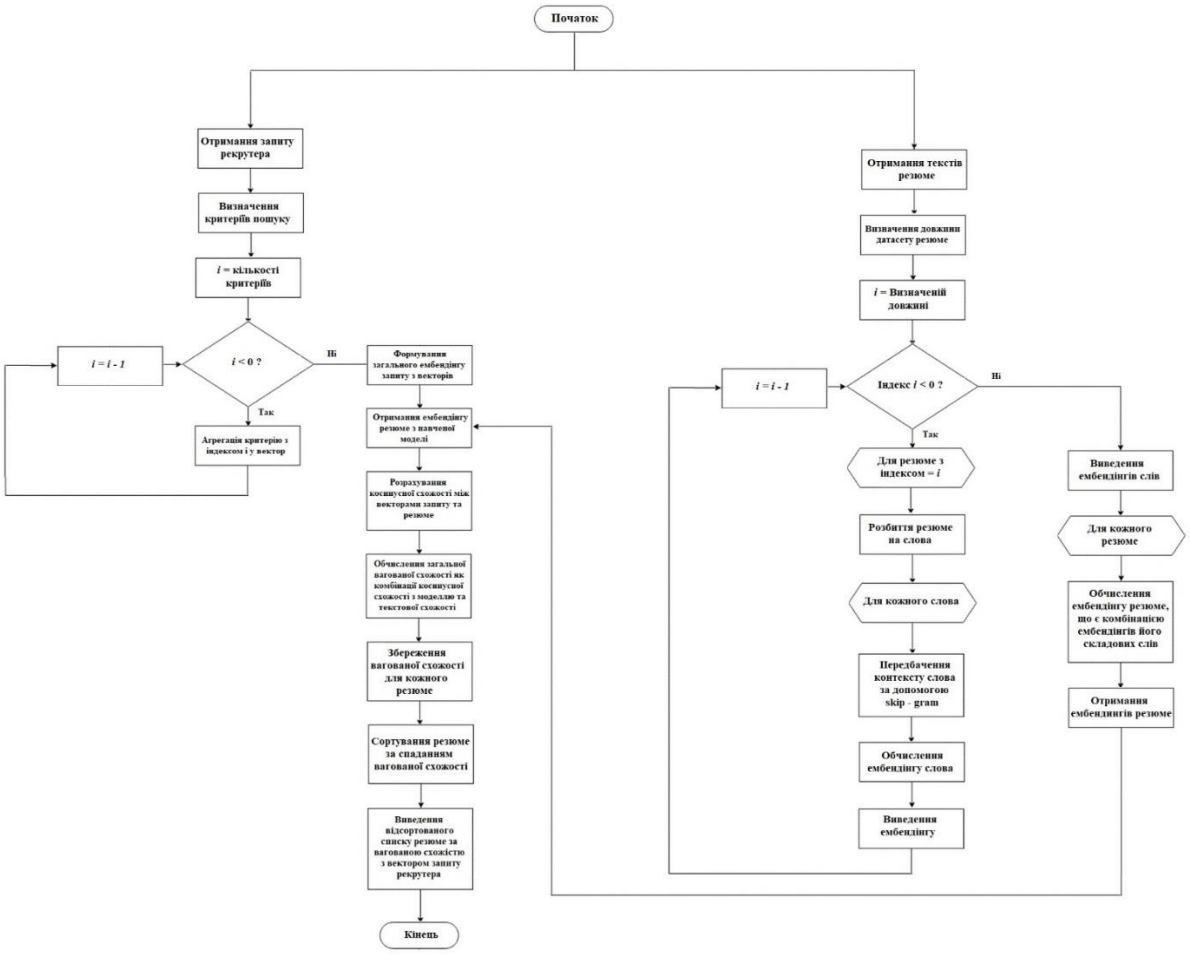



Рис. 3.1 Алгоритм роботи проектованої системи

## 3.3 Попередня обробка даних та ознаки відповідності

Для того, щоб проектована система мала можливість працювати на реальних датасетах – було прийнято рішення не створювати датасет вручну з критеріями, які будуть наперед відомі, а формувати датасет в процесі роботи системи за допомогою джерела вхідних даних, що будуть схожі по структурі. Припустимо, що вакансії, на які ми хотіли б розглядати кандидатів, — це IT-вакансії. Отже, нам потрібне джерело даних, де багато інформації про IT-фахівців, ці дані можна було б використовувати з юридичної точки зору (тобто ми не порушували б закон використовуючи ці дані для потреб системи), також не менш важливим критерієм є те, щоб доступ до цих даних був дешевий, бажано безкоштовний.

Всім цим вимогам відповідає GitHub, датасет користувачів якого ми будемо використовувати як джерело даних для формування резюме. Для отримання даних скористаємось GitHub API та GitHub Archive, за допомогою яких GitHub полегшує доступ до даних для розробників, завдяки чому в нас не буде потреби парсувати сторінки ресурсу.

Після отримання даних, система згідно розробленого алгоритму перетворить їх у вектори ознак, на яких і буде навчатись розроблена нейронна мережа. Початковий вигляд даних може виглядати так (Таблиця 3.6):

Таблиця 3.6

Початковий вигляд вхідних даних

| Датасет | Резюме | Критерії |
|---|---|---|
| Dataset 1 | 3145 | JavaScript, Jquery, React |
| Dataset 2 | 4876 | Java, C#, Python, Досвід роботи, Web |
| Dataset 3 | 5938 | C, C#, Python, C++ |
| Dataset 4 | 6971 | C#, Ruby, Bigdata |

Для кожного резюме, яке входить до зазначених чотирьох датасетів, система розрахує відповідні векторні ознаки. У Таблиці 3.7 вказано кількість векторних ознак, що асоціюються з кожним з даних наборів.

Таблиця 3.7

Кількість векторів ознак резюме у наведених датасетах

| Датасет | Резюме | Вектори |
|---|---|---|
| Dataset 1 | 3145 | 70281 |
| Dataset 2 | 4876 | 144391 |
| Dataset 3 | 5938 | 164144 |
| Dataset 4 | 6971 | 186171 |

Для отримання результату відповідності кандидата система використовує ознако - орієнтований підхід для отримання моделі рейтингу. Модель навчається на результатах ознак з вхідних даних, після чого на основі кількості співпадання ознак формує коефіцієнт відповідності для кожного резюме в датасеті.

Ознаки відповідності - це ключові характеристики запиту-документа, які залежать від обох: як від самого запиту, так і від кожного резюме в датасеті. Ці ознаки вказують, наскільки ефективно кожне резюме відповідає конкретному запиту.

На виході кожен роботодавець чи рекрутер отримає необхідну йому кількість резюме, які навчена модель визначила за найбільш підходящі щодо вхідних даних запиту.

## 3.4 Навчання моделі

Для практичного значення реалізованого підходу система окрім великого датасету резюме повинна мати інформації щодо необхідних критеріїв найму для кожного рекрутера, бо неможливо визначити коефіцієнт необхідності того чи іншого кандидата, не маючи уяви щодо необхідностей рекрутера. Тобто для повноти роботи проектованої моделі на вхід окрім датасету вона повинна приймати критерії співробітника, якого шукає рекрутер. Також як показує практика часто рекрутеру необхідно визначити не 1 людину, що найбільш підходить за його критеріями, а наприклад 10 кращих резюме. Для цього проектована модель буде приймати на вхід коефіцієнт n, що буде визначати яку кількість резюме, найбільш відповідаючих вимогам рекрутера він отримає в результаті роботи системи.

Загальний вигляд вхідних даних у реалізованій системі можна представити так (Рисунок 3.2):

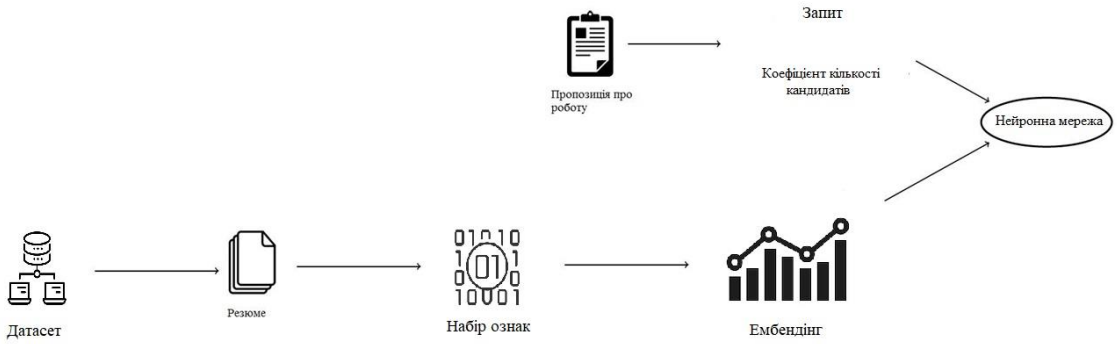



Рис. 3.2 Вхідні дані проектованої системи

Внаслідок попередньої обробки формується набір, який нараховує близько 9 тисяч векторів ознак, що відображають характеристики резюме кандидатів. Результат навчання дозволить нам надсилати у систему запит з відповідними критеріями пошуку та коефіцієнтом кількості.

Фактичні результати щодо найму, тобто інформація про те, чи був кандидат прийнятий чи ні, буде доступна рекрутеру, система надасть йому 10 найкращих резюме та коефіцієнти їх відповідності наведеній вакансії.

У той самий час, набір даних також виявляє неоднорідність у тому розумінні, що він включає інформацію про різних користувачів із різними, можливо, неточними характеристиками. Отже, є можливість, що два резюме з дуже схожими критеріями два однакові запити можуть видати різний результат через знаходження у критеріях невідповідних символів, пробілів тощо. Отже є вірогідність, що результат навчання моделі на такому резюме може бути не до кінця коректним.

Ще однією особливістю попередньо обробленого набору даних є те, що сама попередня обробка може призвести до помилок і пропущених значень, оскільки моделі аналізу ґрунтуються на технологіях машинного навчання і не є абсолютно точними. Інформація, яка доступна в оригінальній, зрозумілій для людини формі, може бути неправильно класифікована. Аналогічно, автоматичне формування запитів базується на попередньо визначених

евристичних правилах, тому важлива інформація про певні вакансії може бути відсутня.

Отже, прийняття рішення щодо найму має кілька недоліків, що робить процес навчання більш складним. Проте використання нами датасету значного розміру повинно перекрити ці проблеми.

Як і в багатьох випадках у сфері машинного навчання, завдання знаходження значного набору даних, який би був придатний для навчання, є найскладнішим в аспекті рекрутингу. Реальні резюме осіб, які шукають роботу, включають особисту інформацію і часто підпадають під конфіденційність. Однак, те, що ще рідше доступно, - це дані, які можна використовувати як основу для контрольованого навчання (supervised learning). Тому раніше прийняте рішення з генерування датасету резюме на основі даних про користувачів, що надає GitHub API стає ще більш релевантним.

## 3.5 Програмна реалізація проектованої системи

### 3.5.1 Ініціалізація датасету для навчання

Для навчання будь якої нейромережі нам потрібно ініціалізувати набір вхідних даних, на основі яких власно наша нейронна мережа буде робити аналіз та навчатись, залишилося вирішити, звідки брати дані щодо наших умовних кандидатів.

Припустимо, що вакансії, на які ми хотіли б розглядати кандидатів, — це IT-вакансії. Отже, нам потрібне джерело даних, де багато інформації про IT-фахівців, ці дані можна було б використовувати з юридичної точки зору (тобто ми не порушували б закон використовуючи ці дані для потреб системи), також не менш важливим критерієм є те, щоб доступ до цих даних був дешевий, бажано безкоштовний.

Всім цим вимогам відповідає GitHub, датасет користувачів якого ми будемо використовувати як джерело даних для формування резюме. Для отримання даних скористаємось GitHub API та GitHub Archive, за допомогою яких GitHub полегшує доступ до даних для розробників, завдяки чому в нас не буде потреби парсувати сторінки ресурсу.

GitHub можна назвати соціальною мережею для обміну кодом та знаннями. Тут є всі необхідні для нашої системи дані: дані про користувача, дані про репозиторії, з приналежністю їх до певних користувачів, мовами, якими вони написані, текстовим описом і тегами, що вказують на напрям, технології репозиторію, також є зв'язок між учасниками у вигляді підписки один на одного, підписки користувачів на репозиторії, інших учасників, та багато іншої інформації.

Для використання даних GitHub зробимо припущення, що репозиторій відображає навички користувача, що його створив. Тобто представим профілі користувачів у GitHub як резюме для навчання нашої моделі.

Проведемо програмну реалізацію даного кроку для формування нашого датасету:

```
import requests

def get_user_data(username, token):
    url = f"https://api.github.com/users/{username}"
    headers = {"Authorization": f"Bearer {token}"}

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error: {response.status_code}")
        return None
```

```
def get_user_repositories(username, token):
    url = f"https://api.github.com/users/{username}/repos"
    headers = {"Authorization": f"Bearer {token}"}

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error: {response.status_code}")
        return None


github_token = "sdgs687sgdg7sg8sgd7gs86dsgdg876"
user_data_list = []
user_repositories_list = []

for username in usernames:
    user_data = get_user_data(username, github_token)
    user_repositories = get_user_repositories(username,
github_token)

    if user_data and user_repositories:
        user_data_list.append(user_data)
        user_repositories_list.append(user_repositories)
```

У даній частині коду ми за допомогою функцій «get_user_data» та «get_user_repositiries» у циклі реалізуємо наповнення датасету інформацією щодо користувачів, який буде представлено як умовних кандидатів та їх навичок, які буде представлено як мітки для ранжування.

### 3.5.2 Створення ембендінгу для елементів датасету

Коли датасет для навчання моделі сформовано, наступним кроком згідно визначеного алгоритму є створення ембендінгу для кожного елементу його датасету. Як було визначено, для векторного представлення буде використано метод Word2Vec на основі нейронних мереж. Для програмної реалізації цього методу використаємо бібліотеку Gensim для Python, що надає відповідні інструменти.

```
from gensim.models import Word2Vec

def train_word2vec_model(resume_texts):
    # resume_texts - список текстів резюме з репозиторіїв

    # Розділіть кожне резюме на слова (токени)
    tokenized_resume_texts = [resume.split() for resume in
resum_texts if code]

    # Створення та навчання Word2Vec модель
    model = Word2Vec(sentences=tokenized_resume_texts,
vector_size=100, window=5, min_count=1, workers=4)

    return model
resume_texts = [...]
# Навчаємо Word2Vec модель
word2vec_model = train_word2vec_model(resume_texts)
# Отримуємо векторне представлення (embedding) для кожного резюме
repo_embeddings = {resume_name: word2vec_model.wv[resume_name] for
resume_name in word2vec_model.wv.index_to_key}
```

Реалізована функція отримує на вхід текст резюме, що було отримано в результаті виконання функції get_user_repositiries, розподілює його на токени, ініціалізує навчання моделі та отримання векторного представлення для кожного резюме.

### 3.5.3 Реалізація пошуку відповідностей

Для того, щоб реалізувати пошук відповідностей датасету кандидатів, необхідно мати запит роботодавця, тобто розуміти, за якими кретиріями він шукає кандидатів. Для цього реалізуємо функцію, що буде отримувати на вхід характеристики, що шукає роботодавець у потенційному кандидаті, у нашому випадку це JavaScript, React, Bigdata розробник з досвідом роботи більше 5 років.

```
query_skills = ['JavaScript','React','BigData','більше 5 років
досвіду' ]
def get_query_embedding(query_skills, word2vec_model):
    # query_skills - список навичок, які шукає рекрутер

    # Створіть вектор запиту, агрегуючи вектори навичок
    query_embedding = sum(word2vec_model.wv[skill] for skill in
query_skills if skill in word2vec_model.wv.index_to_key)

    return query_embedding
```

Наведена функція get_query_embedding приймає список навичок, які шукає рекрутер, та Word2Vec модель (word2vec_model). Для кожної навички, яка зустрічається в моделі, вектор цієї навички додається до вектора запиту. В якості результату роботи функції отримуємо вектор, який представляє запит рекрутера в просторі векторів Word2Vec.

Наступним кроком реалізуємо пошук відповідностей між векторами запиту та резюме кандидатів.

```
def find_matching_repositories(query_embedding, resume _embeddings):

    # Розрахунок косинусної схожості між вектором запиту та
векторами резюме
    similarities = {resume_name:
word2vec_model.wv.similarity(resume_name, query_embedding) for
resume_name in resume_embeddings}
```

```
    # Сортування резюме за зменшенням схожості
    sorted_resumes = sorted(similarities.items(), key=lambda x:
x[1], reverse=True)

    return sorted_ resumes
```

Функція find_matching_repositories приймає на вхід вектор запиту рекрутера (query_embedding) та словник резюме з їхніми векторами (resume_embeddings). Використовуючи Word2Vec модель, розраховується косинусна схожість між вектором запиту та векторами резюме. Результат - відсортований список резюме за спаданням схожості з вектором запиту.

Останнім кроком необхідно знайти для умовного роботодавця чи рекрутера необхідну йому кількість кандидатів, які за результатами роботи моделі будуть визначені як найбільш відповідні до його потреб. Для цього реалізуємо функцію, що буде отримувати на вхід результат роботи нейронної мережі та параметр top_n, що буде визначати яку саме кількість найбільш підходящих за запитом роботодавця кандидатів потрібно повернути у результаті роботи системи.

```
def present_results(sorted_resumes, top_n=10):
    # top_n - кількість найкращих результатів для представлення
    # Виведення топ-н результатів або використання їх для подальшої
обробки
    top_resumes = sorted_resumes[:top_n]

    for resume_name, similarity in top_results:
        print(f":Candidate {resume_name}, Similarity: {similarity}")
    return top_results
```

Функція present_results приймає відсортований список пар (назва резюме кандидату, схожість) та параметр top_n, який визначає кількість найкращих результатів для виведення. Виводить або повертає топ-н результатів, включаючи назву резюме кандидата та рівень схожості з вектором запиту.

### 3.5.4 Недостатня вибірка

На випадок невалідності більшої частини вхідних даних реалізуємо недостатню вибірку для вхідного датасету. Важливо відзначити, що ми не будемо зменшувати обсяг вибірки для тестування, оскільки ми хочемо, щоб наша модель добре працювала з асиметричним розподілом класів.

Для виконання поставленої задачі виконаємо наступні дії :

- Використаємо 5-кратну перехресну перевірку на навчальному наборі.
- На кожній із складок використаємо субдискретизацію.
- Помістимо модель на тренувальні складки та перевіримо на перевірочному згині.

Imbalanced-Learn - це модуль Python, який допомагає збалансувати набори даних, які сильно спотворені чи зміщені у бік певних класів. Це допомагає у класах передискретизації, які зазвичай передискретизуються чи недодискретизуються. Якщо коефіцієнт дисбалансу вище, вихідні дані зміщуються у бік класу із найбільшою кількістю вибірок.

Цей метод промаху відноситься до групи стратегій недостатньої вибірки, які вибирають вибірки на основі відстані між екземплярами класу більшості та меншості.

У наведеному нижче коді ми створюємо гнучку функцію, яка може виконувати сітку або рандомізований пошук за заданим оцінювачем та його параметрами з недостатньою/передискретизацією або без нього та повертає кращий оцінювач разом із показниками продуктивності:

```python
def get_model_best_estimator_and_metrics(estimator, params, kf=kf, X_train=X_train,
                                         y_train=y_train, X_test=X_test,
                                         y_test=y_test, is_grid_search=True,
                                         sampling=NearMiss(), scoring="f1",
                                         n_jobs=2):
    if sampling is None:
        pipeline = make_pipeline(estimator)
    else:
        # зробити конвеєр надмірної/недобірки
        pipeline = make_pipeline(sampling, estimator)
    # отримуємо назву оцінювача
    estimator_name = estimator.__class__.__name__.lower()
    # будування параметрів для сітки/випадкового пошуку cv
    new_params = {f'{estimator_name}__{key}': params[key] for key in params}
    if is_grid_search:
        # пошук у сітці замість випадкового пошуку
        search = GridSearchCV(pipeline, param_grid=new_params, cv=kf, scoring=scoring, return_train_score=True,
                              n_jobs=n_jobs, verbose=2)
    else:
        # рандомізований пошук
        search = RandomizedSearchCV(pipeline, param_distributions=new_params,
                                    cv=kf, scoring=scoring, return_train_score=True,
                                    n_jobs=n_jobs, verbose=1)

    search.fit(X_train, y_train)
    cv_score = cross_val_score(search, X_train, y_train, scoring=scoring, cv=kf)
    # робимо прогнози на основі тестових даних
    y_pred = search.best_estimator_.named_steps[estimator_name].predict(X_test)
    # обчислюємо показники: запам'ятовування, точність, оцінка F1 тощо.
    recall = recall_score(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    y_proba = search.best_estimator_.named_steps[estimator_name].predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    auc = roc_auc_score(y_test, y_proba)
    # повертаємо найкращу оцінку разом із показниками
    return {
        "best_estimator": search.best_estimator_,
        "estimator_name": estimator_name,
        "cv_score": cv_score,
        "recall": recall,
        "accuracy": accuracy,
        "f1_score": f1,
        "fpr": fpr,
        "tpr": tpr,
        "auc": auc,
    }
```

Оскільки для навчання нашої моделі ніколи не буває достатньо даних, виключення частини її для перевірки призводить до недообучення. Це призводить до ризику втратити важливих закономірностей/тенденції в наборі даних, занижені навчальні дані, що збільшує помилку, викликану передбачуваністю.

Отже, нам необхідна стратегія, що забезпечить достатню кількість даних для навчання моделей, при цьому залишаючи достатньо інформації для проведення ефективної перевірки.

Функція cross_val_score() використовує перекрестну перевірку для визначення оцінок, які ми використовуємо у наведеній вище функції. Функція зроблена гибкою. Наприклад, якщо ми маємо потребу виконати пошук за набором моделей LogisticRegression з недостатнім вибором, буде виконана наступна частина коду:

```
# Створюємо таблицю для кривої ROC
## Створюємо дані
res_table = pd.DataFrame(columns=['classifiers', 'fpr', 'tpr', 'auc'])

rfc_results = get_model_best_estimator_and_metrics(
    estimator=RandomForestClassifier(),
    params={
        'n_estimators': [50, 100, 200],
        'max_depth': [4, 6, 10, 12],
        'random_state': [13]
    },
    sampling=None,
    n_jobs=3,
)
res_table = res_table.append({'classifiers': rfc_results["estimator_name"],
                              'fpr': rfc_results["fpr"],
                              'tpr': rfc_results["tpr"],
                              'auc': rfc_results["auc"]
                              }, ignore_index=True)
```

Якщо є потреба  вимкнути недостатню вибірку, ми маємо можливість просто застосовати параметр None до sampling параметру get_model_best_estimator_and_metrics() функції.

Якщо нам необхідно побудувати криву ROC для декількох моделей, ми можемо запустити наведений вище код для декількох моделей та їх параметрів. Головне відредагувати ім'я класифікатора, щоб розрізняти моделі з передискретизацією та недостатньою вибіркою.

У якості тесту вдалося запустити п'ять різних моделей з недостатньою вибіркою, код побудови кривої ROC, використовуючи нашу res_table наведено далі:

```
# Будуємо криву ROC для недостатньої вибірки
res_table.set_index('classifiers', inplace=True)
fig = plt.figure(figsize=(17, 7))

for j in res_table.index:
    plt.plot(res_table.loc[j]['fpr'],
             res_table.loc[j]['tpr'],
             label="{}, AUC={:.3f}".format(j, res_table.loc[j]['auc']))

plt.plot([0, 1], [0, 1], color='orange', linestyle='--')
plt.xticks(np.arange(0.0, 1.1, step=0.1))
plt.xlabel("Positive Rate(False)", fontsize=15)
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel("Positive Rate(True)", fontsize=15)
plt.title('Analysis for Oversampling', fontweight='bold', fontsize=15)
plt.legend(prop={'size': 13}, loc='lower right')
plt.show()
```

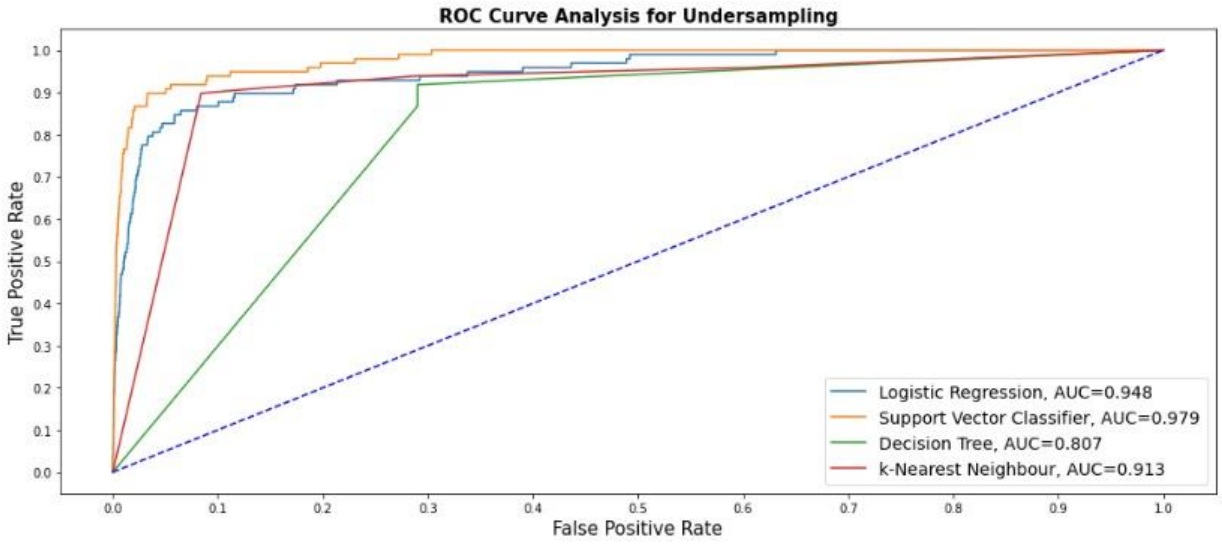



Рис. 3.3 Результат побудови кривої ROC

## 3.6 Тестування результатів роботи реалізованої системи

Для оцінки результату роботи реалізованого алгоритму проведемо тестування його цільової функції, а саме автоматизованого оцінювання кандидатів для роботодавців. З цією метою проведемо навчання моделі на датасеті у 5000 користувачів та сформуємо 8 рекрутингових запитів різноманітного змісту. Результати тестування наведено у таблиці 3.8.

Таблиця 3.8

Результати тестування запитів до реалізованої системи

| Номер | Запит | Результат |
|---|---|---|
| 1. | Володіння Python (мінімум 80%), досвід у веб-розробці (мінімум 2 роки). | Кандидат 3674: 90% Python, 3 роки досвіду у веб-розробці.<br>Кандидат 673: 85% Python, 2.5 роки досвіду у веб-розробці.<br>Кандидат 4672: 92% Python, 4 роки досвіду у веб-розробці. |
| 2. | Участь у відкритих проектах (останній внесок не більше 6 місяців тому), великий інтерес до нових технологій. | Кандидат 2752: Востаннє вносився в відкриті проекти 5 місяців тому, активний інтерес до нових технологій.<br>Кандидат 2790: Початківець у відкритих проектах, інтерес до нових технологій.<br>Кандидат 648: Активно залучений до відкритих проектів, останній внесок - тиждень тому. |
| 3. | Спеціалізація у машинному навчанні.<br>Досвід роботи з нейронними мережами (мінімум 1 проект). | Кандидат 1574: Спеціалізується на машинному навчанні, працював з нейронними мережами у декількох проектах.<br>Кандидат 3644: Має досвід у роботі з нейронними мережами та брав участь у проектах, пов'язаних із штучним інтелектом.<br>Кандидат 722: Експерт у сфері нейронних мереж, активно публікується та брав участь у дослідженнях. |

| | | |
|---|---|---|
| 4. | Досвід роботи з базами даних (SQL та NoSQL).<br>Володіння мовою програмування Java. | Кандидат 672: Досвід роботи з MySQL та MongoDB, володіє Java.<br>Кандидат 4891: Робив проекти із використанням SQL Server та Cassandra, володіє Java.<br>Кандидат 1263: Експерт у використанні PostgreSQL та CouchDB, володіє Java. |
| 5. | Знання фреймворка React.js.<br>Досвід роботи з системами контролю версій (Git). | Кандидат 1874: Досвід роботи з React.js, активно використовує Git.<br>Кандидат 3178: Спеціалізується на React.js, має великий досвід роботи з Git.<br>Кандидат 3098: Повний стек React.js, досвід роботи з Git. |
| 6. | Досвід роботи з Docker та контейнеризацією.<br>Володіння мовою програмування Go. | Кандидат 1902: Використовує Docker у всіх своїх проектах, має досвід роботи з Go.<br>Кандидат 101: Має досвід із Docker, працював з Go в певних проектах.<br>Кандидат 2864: Експерт у Docker, також володіє Go. |
| 7. | Досвід у використанні та оптимізації високонавантажених веб-систем.<br>Навички у моніторингу та | Кандидат 1732: Оптимізував великі веб-системи, використовує інструменти моніторингу.<br>Кандидат 625: Досвід із високонавантаженими веб-системами, використовує інструменти аналізу продуктивності. |

| | | |
|---|---|---|
| | аналізі продуктивності. | Кандидат 32: Експерт у високонавантажених системах, активно використовує інструменти моніторингу. |
| 8. | Володіння мовами Scala та Kotlin. Розробка мікросервісів у середовищі Kubernetes. | Кандидат 3671: Володіє Scala та Kotlin, розробляє мікросервіси на Kubernetes. Кандидат 2390: Досвід роботи з Scala та Kotlin, робив проекти у Kubernetes. Кандидат 4172: Експерт у Scala та Kotlin, активно використовує Kubernetes. |

Наведені приклади враховують різноманітні критерії пошуку, які можуть бути використані рекрутерами для відбору кандидатів на основі впровадженого датасету. За результатами можна визначити, що система виконала задачу, тому результат тестування можна вважати успішним.

Проведемо також тестування технічних характеристик для визначень можливостей системи у швидкодії, точності тощо. Результати тестування наведено у таблиці 3.9.

Таблиця 3.9

Результати тестування технічних характеристик реалізованої системи

| Номер | Критерій тестування | Тест | Результат |
|---|---|---|---|
| 1. | Швидкодія та Масштабованість | Система успішно обробляє 1000 запитів в секунду при великому обсязі вхідних даних | Середній час відповіді залишається стабільним, а система легко масштабується для обробки більшої кількості запитів |

| | | | |
|---|---|---|---|
| 2. | Точність та Ранжування | Запити рекрутерів на різних етапах процесу найму порівнюються з реальними профіілями кандидатів | Модель ефективно ранжує кандидатів відповідно до їхньої відповідності запитам, а точність рекомендацій залишається відповідній 99.82% |
| 3. | Робота із Збереженими Даними | Система правильно оновлює інформацію користувачів та їхніх репозиторіїв, використовуючи GitHub API та GitHub Archive | Оновлення даних відбувається швидко та безперервно, забезпечуючи актуальну інформацію для кожного користувача |
| 4. | Стабільність та Відновлення | Система успішно відновлюється після аварій та переривань з'єднання з GitHub API | Відновлення відбувається автоматично, і система продовжує свою роботу після відновлення доступу до даних |

# ВИСНОВОК

У ході виконання даної роботи проведено проектування та реалізацію алгоритму автоматизованого оцінювання кандидатів для роботодавців в процесі найму, що на основі навченої нейронної мережі проводить автоматизований відбір кандидатів з датасету резюме на основі вхідних даних рекрутера.

В ході виконання теоретичної частини роботи було проведено аналіз предметної галузі. Надано загальне визначення системи оцінювання кандидатів, проведено аналіз сучасних тенденцій системи оцінки кандидатів, проведено визначення переваг та недоліків програмного забезпечення предметної галузі та формулювання завдання.

У другій частині роботи проведено аналіз методів автоматизованого оцінювання та формування алгоритму роботи, проаналізовано загальні принципи автоматизації оцінювання кандидатів, розглянуто існуючі методи та підходи оцінювання, обрано алгоритм, що буде використано під час виконання роботи.

У третій частині роботи було проведено розробку алгоритмічного програмного забезпечення та тестування результатів. Наведено аналіз та вибір інструменту реалізації програмного забезпечення, визначено алгоритм роботи системи, описано попередню обробку даних та ознаки відповідності, описан процес навчання моделі, проведено програмну реалізацію та тестування проектованої системи.

Після проведення тестування системи дані було визначено, що вона успішно виконує процес оцінювання кандидатів та може бути використана роботодавцями в процесі найму.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Андреас М. Введення в машинне навчання за допомогою Python. Керівництво для фахівців по роботі з даними / Мюллер Андреас // М .: Альфакнига, 2017. - 487с.

2. Бенгфорт Б. Прикладний аналіз текстових даних на Python. Машинне навчання та створення додатків обробки природної мови / Б. Бенгфорт // 2019. - 368 с.

3. Каллан, Р. Нейронні мережі: Короткий довідник / Р. Каллан. - М .: Вільямс І.Д., 2017. - 288 с.

4. Ніколенко С. Глибоке навчання / С. Ніколенко, А. Кадурін, Е. Архангельська // 2018. - 480 с.

5. Плас Д. Python для складних завдань. Наука про дані і машинне навчання. Керівництво / Плас Джейк Вандер // , 2018. - 759 с.

6. Рашка С. Python і машинне навчання / С. Рашка // ДМК Пресс, 2017. - 418 с.

7. Редько В.Г. Еволюція, нейронні мережі, інтелект: Моделі і концепції еволюційної кібернетики / В.Г. Редько // М .: Ленанд, 2019. - 224 с.

8. Хайкін С. Нейронні мережі: повний курс / С. Хайкін // М .: Діалектика, 2019. - 1104 с.

9. Шолль Ф. Глибоке навчання на Python / Ф. Шолль // 2018. - 400 с.

10. How to Digitize Texts with Open-Source Command-Line Optical Character Recognition (OCR) Software [Електронний ресурс] / Режим доступу: https://hdw.artsci.wustl.edu/articles/154

11. Karpathy A. et al. Convolutional Neural Networks for Visual Recognition [Електронний ресурс]. Режим доступу: http://www.cs231n.stanford.edu

12. The MNIST DATABASE of handwritten digits [Електронний ресурс]. Режим доступу : http://yann.lecun.com/exdb/mnist/

13. Kay A. Tesseract: Open-Source Optical Character Recognition Engine [Електронний ресурс] / Режим доступу: http://www.linuxjournal.com/article/9676

14. Nielsen MA Neural Networks and Deep Learning [Електронний ресурс]. Режим доступу: http://www.neuralnetworksanddeeplearning.com

15. OCRopy: Python-based tools for document analysis and OCR [Електронний ресурс] / Режим доступу: https://github.com/tmbdev/ocropy

16. Theodoridis S., Koutroumbas K., Pattern Recognition. New York: Elsevier Science, 2008 [Електронний ресурс]. Режим доступу: https://books.google.com/books?id=QgD-3Tcj8DkC

# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

**Магістерська робота**

**Розробка системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму**

Виконав: Студент групи ПДМ-62 Мотрук Є. О.

Керівник: к.т.н., доц., доцент кафедри ІПЗ Трінтіна Н.А.

Київ - 2023

2

## МЕТА РОБОТИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

**Мета роботи:** підвищені точності ранжування великих об'ємів резюме кандидатів відповідно до запиту рекрутерів та збільшенні швидкості обробки датасету резюме відповідно до одиниці часу.

**Об'єкт дослідження:** процес автоматизованого оцінювання кандидатів для роботодавців.

**Предмет дослідження:** моделі та методи автоматизованого оцінювання кандидатів на основі нейронних мереж.

3

## ОБГРУНТУВАННЯ НЕОБХІДНОСТІ СИСТЕМИ ОЦІНЮВАННЯ КАНДИДАТІВ

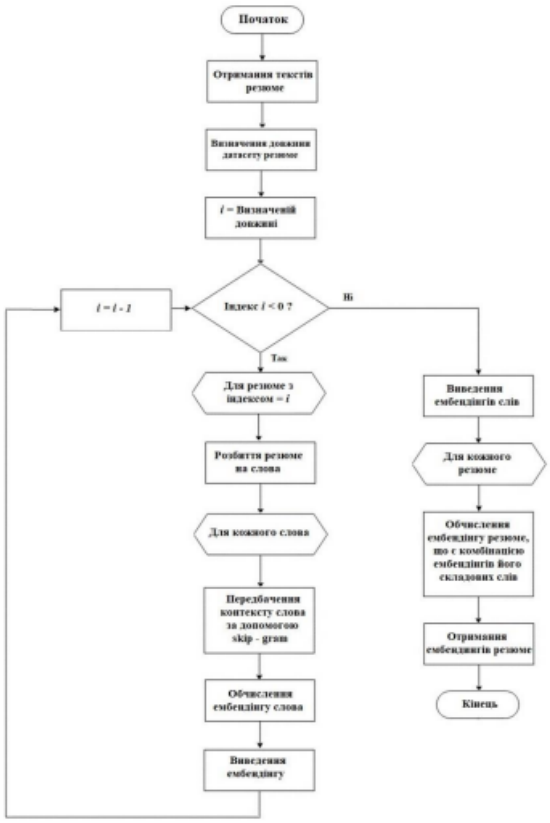

4

## РЕЗУЛЬТАТИ ПОРІВНЯЛЬНОГО АНАЛІЗУ ІСНУЮЧИХ СИСТЕМ

| Назва | Test-Dome | eSkill | Vervoe | Розроблена система |
|---|---|---|---|---|
| Переваги | Різноманітність тестів, Практичні завдання, Ефективний відбір кандидатів, Аналітика та звіти, Підтримка багатомовності. | Різноманітність тестів, Адаптивність та гнучкість, Тестування онлайн, Аналітика та звіти, Інтеграція з іншими системами. | Широкий спектр завдань, Практичні тести, Адаптивність, Оцінка м'яких навичок, Зручний процес тестування. | Автоматизована обробка великого датасету кандидатів, класифікація кандидатів за наведеними параметрами з великим відсотком точності, автоматизований аналіз результатів класифікації щодо запиту рекрутера |
| Недоліки | Вартість, Неспецифічність для деяких галузей, Низький відсоток точності, Неможливість обробки резюме. | Вартість, Неможливість обробки резюме, Низький відсоток точності, Потреба в Інтернет-з'єднанні. | Вартість, Неможливість обробки резюме, Низький відсоток точності, Потреба в Інтернет-з'єднанні. | Недостатня маштабованість, Відсутність вбудованого тестування. |

5

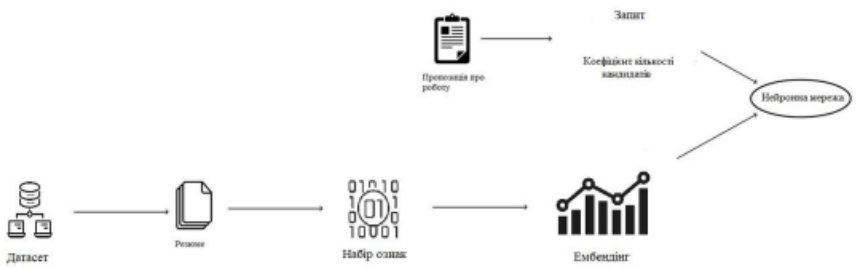
## АЛГОРИТМ НАВЧАННЯ МОДЕЛІ




6

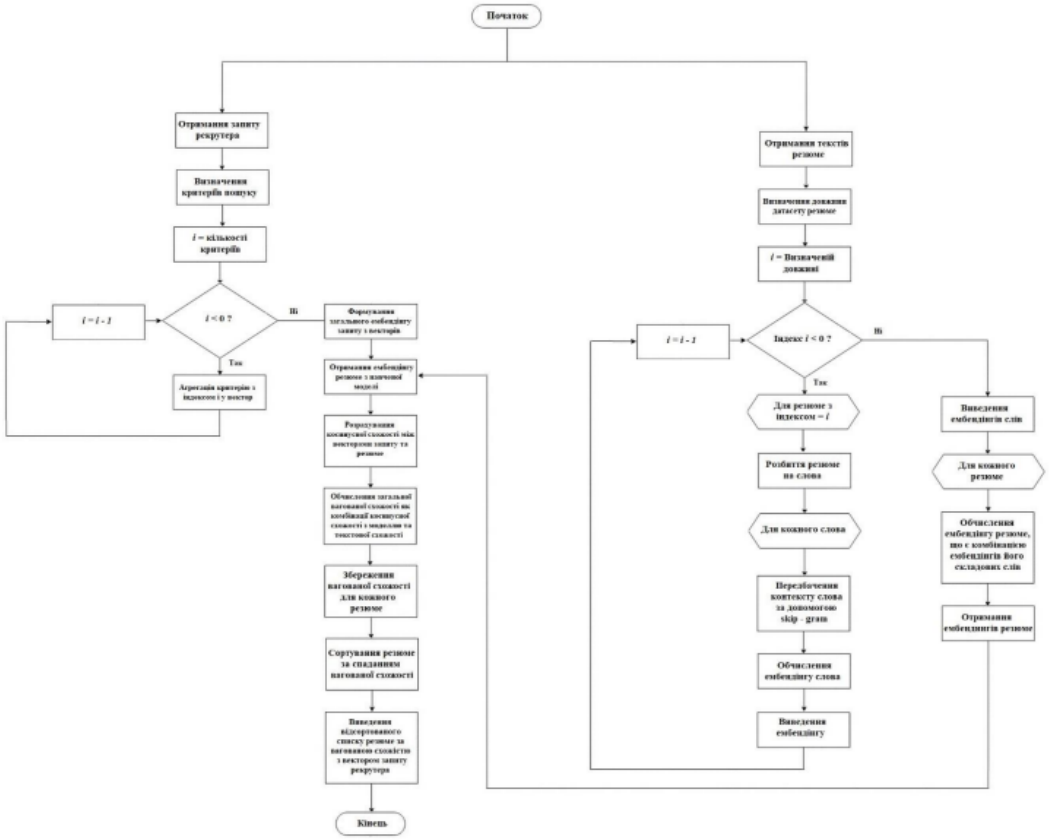
## ПРИКЛАД ВХІДНИХ ДАНИХ СИСТЕМИ

| Датасет | Резюме | Критерії |
|---|---|---|
| **Dataset 1** | 3145 | JavaScript, Jquery, React |
| **Dataset 2** | 4876 | Java, C#, Python, Досвід роботи, Web |
| **Dataset 3** | 5938 | C, C#, Python, C++ |
| **Dataset 4** | 6971 | C#, Ruby, Bigdata |



## ЗАГАЛЬНИЙАЛГОРИТМ РОБОТИ ПРОЕКТОВАНОЇ СИСТЕМИ

7




## МАТЕМАТИЧНА МОДЕЛЬ РЕАЛІЗОВАНОГО АЛГОРИТМУ

8

$D$ - словник унікальних слів у текстах резюме.
$R_i$ - резюме користувача $i$.
$q$ - вектор запиту рекрутера в просторі
$v_{wij}$ - векторне представлення слова $w_{ij}$.
$v_{Ri}$ - векторне представлення резюме $R_i$. $\text{sim}_{\text{Word2Vec}}$
$\text{sim}_{\text{text}}(q,t_i)$ - косинусна схожість з текстового опису

$v_Q$ - векторне представлення запиту $Q$ (рекрутерські навички).
$\|q\|$ та $\|r_i\|$ - норми векторів $q$ та $r_i$
$weight_{Word2Vec}$ - вага для косинусної схожості Word2Vec,
$weight_{text}$ - вага для косинусної схожості текстового опису,
$(q,r_i)$ - косинусна схожість з Word2Vec,

Загальну модель навчання моделі можна представити за формулою:

$$\text{maximize} \prod_{i=1}^{N} \prod_{j=1}^{M_i} P(w_{ij}|R_i;\theta)$$

Отримання векторного представлення для кожного резюме виконується за формулою:

$$v_{R_i} = \frac{1}{M_i} \sum_{j=1}^{M_i} v_{w_{ij}}$$

Створенняя вектору запиту рекрутера відбувається за формулою:

$$v_Q = \sum_{k=1}^{K} v_{\text{skill}_k}$$

Косинусна схожість між векторами Word2Vec розраховується за формулою:

$$\text{sim}_{\text{Word2Vec}}(q, r_i) = \frac{q \cdot r_i}{\|q\| \cdot \|r_i\|}$$

Загальну ваговану схожість між резюме та запитом рекрутера можна представити за формулою:

$$\text{sim}_{\text{total}}(q, r_i, t_i) = \text{weight}_{\text{Word2Vec}} \times \text{sim}_{\text{Word2Vec}}(q, r_i) + \text{weight}_{\text{text}} \times \text{sim}_{\text{text}}(q, t_i)$$

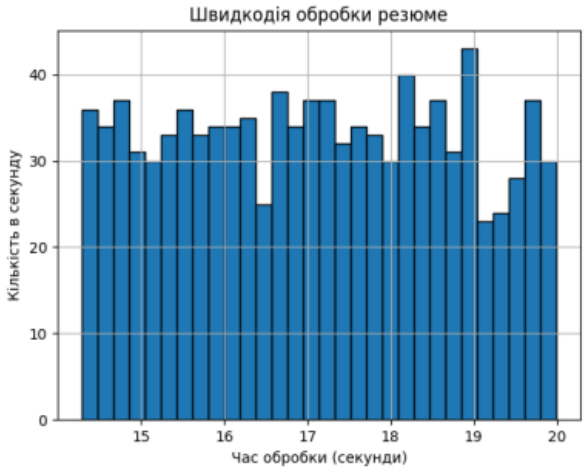
# РЕЗУЛЬТАТ ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

9

| Номер | Запит | Результат |
|---|---|---|
| 1. | Володіння Python (мінімум 80%), досвід у веб-розробці (мінімум 2 роки). | Кандидат 3674: 90% Python, 3 роки досвіду у веб-розробці.<br>Кандидат 673: 85% Python, 2.5 роки досвіду у веб-розробці.<br>Кандидат 4672: 92% Python, 4 роки досвіду у веб-розробці. |
| 2. | Участь у відкритих проектах (останній внесок не більше 6 місяців тому), великий інтерес до нових технологій. | Кандидат 2752: Востаннє вносився в відкриті проекти 5 місяців тому, активний інтерес до нових технологій.<br>Кандидат 2790: Початківець у відкритих проектах, інтерес до нових технологій.<br>Кандидат 648: Активно залучений до відкритих проектів, останній внесок - тиждень тому. |
| 3. | Спеціалізація у машинному навчанні. Досвід роботи з нейронними мережами (мінімум 1 проект). | Кандидат 1574: Спеціалізується на машинному навчанні, працював з нейронними мережами у декількох проектах.<br>Кандидат 3644: Має досвід у роботі з нейронними мережами та брав участь у проектах, пов'язаних із штучним інтелектом.<br>Кандидат 722: Експерт у сфері нейронних мереж, активно публікується та брав участь у дослідженнях. |
| 4. | Досвід роботи з базами даних (SQL та NoSQL).<br>Володіння мовою програмування Java. | Кандидат 672: Досвід роботи з MySQL та MongoDB, володіє Java.<br>Кандидат 4891: Робив проекти із використанням SQL Server та Cassandra, володіє Java.<br>Кандидат 1263: Експерт у використанні PostgreSQL та CouchDB, володіє Java. |
| 5. | Знання фреймворка React.js. Досвід роботи з системами контролю версій (Git). | Кандидат 1874: Досвід роботи з React.js, активно використовує Git.<br>Кандидат 3178: Спеціалізується на React.js, має великий досвід роботи з Git.<br>Кандидат 3098: Повний стек React.js, досвід роботи з Git. |
| 6. | Досвід роботи з Docker та контейнеризацією.<br>Володіння мовою програмування Go. | Кандидат 1902: Використовує Docker у всіх своїх проектах, має досвід роботи з Go.<br>Кандидат 101: Має досвід із Docker, працював з Go в певних проектах.<br>Кандидат 2864: Експерт у Docker, також володіє Go. |

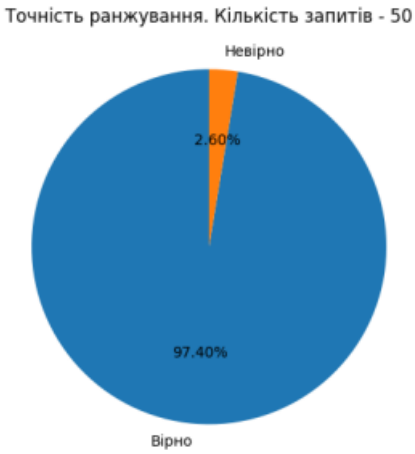
10

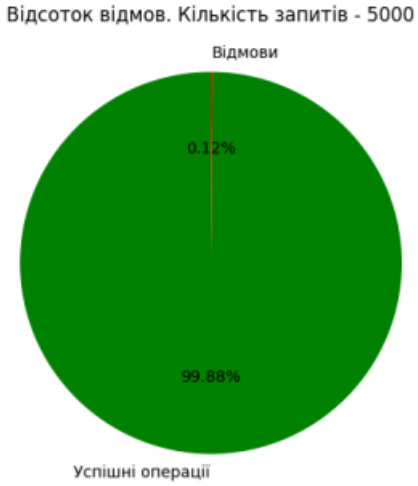
# РЕЗУЛЬТАТ ТЕСТУВАННЯ ШВИДКОДІЇ ОБРОБКИ РЕЗЮМЕ



11

# ТОЧНІСТЬ РАНЖУВАННЯ ТА ВІДСОТОК ВІДМОВ




# ВИСНОВКИ

12

- У ході виконання даної роботи проведено проектування та реалізацію алгоритму автоматизованого оцінювання кандидатів для роботодавців в процесі найму, що на основі навченої нейронної мережі проводить автоматизований відбір кандидатів з датасету резюме на основі вхідних даних рекрутера.
- В ході виконання теоретичної частини роботи було проведено аналіз предметної галузі. Надано загальне визначення системи оцінювання кандидатів, проведено аналіз сучасних тенденцій системи оцінки кандидатів, проведено визначення переваг та недоліків програмного забезпечення предметної галузі та формулювання завдання.
- У другій частині роботи проведено аналіз методів автоматизованого оцінювання та формування алгоритму роботи, проаналізовано загальні принципи автоматизації оцінювання кандидатів, розглянуто існуючі методи та підходи оцінювання, обрано алгоритм, що буде використано під час виконання роботи.
- У третій частині роботи було проведено розробку алгоритмічного програмного забезпечення та тестування результатів. Наведено аналіз та вибір інструменту реалізації програмного забезпечення, визначено алгоритм роботи системи, описано попередню обробку даних та ознаки відповідності, описан процес навчання моделі, проведено програмну реалізацію та тестування проектованої системи.

## ПУБЛІКАЦІЯ ТА АПРОБАЦІЯ РОБОТИ



**Статті:**

1. Трінтіна Н.А., Мотрук Є.О. Розробка системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму // Київ: Журнал «Телекомунікаційні та інформаційні технології». Подано до друку

**Тези:**

2. Трінтіна Н.А., Мотрук Є.О. Розробка системи автоматизованого оцінювання кандидатів для роботодавців в процесі найму // Науково-практична конференція «Проблеми комп'ютерної інженерії» – Київ : ДУІКТ, 2023. – С. 187-188.

# ДЯКУЮ ЗА УВАГУ!