ЗМІСТ

Перелік умовних позначень

ASR(acoustic speech recognition) – акустичне розпізнавання мови

CTS(conversational telephone speech) - розмовна телефонна мова

DER(Diarization Error Rates) - Частота помилок діаризації

FA(False Alarm) - Помилковий сигнал тривоги

LSTM(Long short-term memory) – довга короткосрочна пам’ять

MFCC - меп-кепстральні коефіцієнти

MSCD(MeanSquared Cosine Distances) - Середні квадратні відстані косинусів

PLDA(Probabilistic Linear Discriminant Analysis) - Імовірнісний лінійно-дискримінаційний аналіз

RNN – рекурентна нейрона мережа

TPM - точність розпізнавання мови

VAD(Voice Activity Detector) - Детектор голосової активності

Вступ

Актуальність теми дипломної роботи. Коли невелика компанія, що починає свою діяльність, має намір рости, їй доводиться наймати працівників. Тому що компанія потрібен високоякісний штат, роботодавець повинен взяти інтерв'ю з багатьма кандидатами, і, отже, вона може потрібно багато часу, щоб зібрати необхідні дані. Звичайно, є ще один важливий попит який час займає процес найму, який повинен бути якомога коротшим або темпи зростання компанії, які повинні йти якомога швидше. Отже, компанія має дві основні вимоги та потреби для досягнення балансу між ними. Це інтуїтивне уявлення про найму, від якого ця проблема конкретного дослідження носить свою назву. Проблема найму - це лише абстрактна модель реального процесу найму. Зрозуміло, що Проблема найму не охоплюватиме кожен аспект реальних процесів найму, але вона досліджує деякі важливі параметри за спрощеною математичною моделлю. З іншого боку, заява проблеми - як ми побачимо - дасть загальне математичне запитання з багатьма можливими додатки; це актуально в багатьох випадках, коли потрібно приймати рішення в умовах невизначеності. Проблема пошуку найкращого кандидата послідовності представляє просту модель для прийняття рішення прийняття в умовах невизначеності, тому що особа, що приймає рішення - у якийсь момент - повинна вибрати одну з них з цієї послідовності без опитування кандидатів, які можуть прийти після обраної. Так, особа, що приймає рішення, повинна максимізувати ймовірність вибору найкращого. Також рішення найму або звільнення кандидата приймається в режимі онлайн, і це безвідклично. Ця проблема перша змодельована як відома проблема, яка називається проблемою секретаря [3]. У проблемі секретаря, роботодавець шукає лише одного кандидата на одну посаду секретаря. Секретар проблема добре вивчена і має безліч розширень. Одним із важливих розширень є випадок, коли роботодавець шукає багато працівників для зростання своєї компанії – це явище і є проблемою найму.

Наразі є багато сервісів для пошуку роботи задля облегшення обов'язків рекрутингових компаній. Їх суть полягає у тому, что на сторінці є перегляд вакансій

з детальним описом тієї чи іншої вакансії, контактами компанії та місця її знаходження. Якщо брати ІТ-сферу, у якій рекрутинг показав себе найбільше, то всі джерела є електронними. Тобто, це вебсайти або вебплатформи такі як: djinni, dou, тощо. Усі ці платформи мають майже однаковий потенціал та можливості. Єдине, що їх відрязнає між собою – це вакансії та дизайн та зручність у використанні. Тож, ми маємо рекрутинг як ключ до пошуку кандидатів з однієї сторони, і кандидатів, які шукають нове місце роботи на спеціалізованих платформах, з іншої сторони. І досі ніхто не запропонував спробувати об'єднати ці дві сміжні культури у один єдиний проект, який буде вигідний усім трьом сторонам цього процесу(включно з компаніями).

Проаналізувавши поточні сервіси та їх можливості, було виявлено, що є доволі просте рішення щодо покращення пошуку роботи: це запис технічного інтерв'ю кандидата та використання його для декількох вакансій. На поточний момент таке рішення використовується тільки у декількох іноземних компаніях, і виключно на закритому внутрішньому ринку праці(у межах цієї самої компанії). Це стане не тільки корисним для рекрутерів, компаній, кандидатів, але й задля вебплатформи також. Таке рішення привлече за собою нових кліентів на нових кандидатів. Це дасть змогу оживити ІТ ринок працю в Україні на початку, і можливо розкрити цю ідею у інших.

Метою роботи є підвищення ефективності сервісу анонімного найму ІТ співробітників по заданим параметрам.

Завданням роботи є розробити новий метод оцінки технічного рівня кандидата і інтегрувати розроблений метод у веб-додаток.

Методи дослідження: теоретичні: узагальнення теоретичних даних, аналіз наукових джерел, порівняння, синтез – присвячених проблемі дослідження з метою накопичення науково-теоретичного підґрунтя для визначення сутності рекрутингу як сфери розвитку людського капіталу.

Наукова новизна одержаних результатів: наукова новизна полягає у впровадженні нових методів для найму персоналу за допомогою удосконалених алгоритмів розбору записів технічних інтерв'ю.

Об’єктом є процес обробки технічного інтерв’ю кандидата.

Предметом дослідження є зменшення часу на закриття позиції.

Теоретична та практична цінність роботи. Теоретична цінність цієї роботи полягає у поглибленному вивченні методів пошуку роботи в сфері IT з точки зори соціологічної теорії управління та розроблення рекомендацій щодо покращення методів вебсервісів для анонімного пошуку роботи.

Практичне значення отриманих результатів для дипломної роботи: дає змогу монетизувати данні рішення, допомогти украхнському ринку праці у підборі персоналу за більш короткий час. Такой, таке рішення має потягнути за собою більше можливостей для кандидатів.

Структура роботи: Дипломна робота складається зі вступу, трьох розділів, що поділені на підрозділи, висновків та рекомендацій, списку використаних джерел та додатків. Загальний обсяг роботи – 60 сторінок (з них 57 основного тексту). Список використаних джерел містить 38 найменувань.

1 АНАЛІЗ СЕРВІСІВ ПОШУКУ РОБОТИ НА РИНКУ ІТ ПРАЦІВНИКІВ

1.1 Аналіз ІТ ринку

Протягом останніх років ринок ІТ-послуг в Україні стрімко розвивається і на сучасному етапі охоплює понад 150 тис. працівників галузі інформаційних технологій. У країні діють декілька тисяч компаній, де працюють висококваліфіковані фахівці з якісною технічною освітою, які надають послуги з розробки програмного забезпечення (ПЗ), реалізують сервіси ПЗ. Вплив ІТ-сектору на розвиток економіки є безперечним, проте його оцінювання вимагає відповідної статистичної бази. Водночас ринок інформаційних технологій України й досі залишається недостатньо дослідженим через низку проблем, пов’язаних зі статистичними спостереженнями. Дослідженню питань функціонування сфери інформаційних технологій та її впливу на економіку України присвячено праці таких вітчизняних вчених, як О. Бабанін, С. Войтко, В. Глухов, А. Маслов, А. Могилова, С. Пиріг, І. Седікова, Л. Федулова, М. Чайковська, А. Чухно та ін. Однак значна частина питань, які стосуються ринку ІТ та перспектив його розвитку в Україні, залишаються недостатньо розкритими та потребують подальшого опрацювання, особливо крізь призму статистичного моніторингу цих процесів. В Україні ІТ посідає третє місце серед орієнтованих на експорт індустрій, поступаючись лише аграріям та металургам. За підсумками 2016 р. ринок ІТ-послуг збільшився до 2,9 млрд дол. США, що становить 4% ВВП країни [2]. Незважаючи на кризові явища в економіці України, приріст ІТсектору в 2017 р. становив близько 7%. За прогнозами експертів, якщо галузі не заважати розвиватися, то до 2020 року вона може зрости вдвічі [3]. В Україні у 2016 р. налічувалося близько 100 тис. фахівців цієї галузі, що на 12% більше, ніж у 2015 р. Вони генерують дохід понад 3 млрд дол. США, пропонуючи сервіси з різним обсягом доданої вартості й

охоплюючи увесь життєвий цикл розробки ПЗ. Зростання ринку відображається і на підвищенні кількості вакансій (на 40%). У 2015 р. майже 30 тис. осіб пройшли через різноманітні курси та ІТ-програми у спеціалізованих ІТ-школах [4]. За результатами 2016 р. 12 українських компаній потрапили в рейтинг 100 кращих постачальників послуг аутсорсингу The Global Outsourcing 100 [5], попереднього року таких компаній було десять. Це свідчить, що український ринок інформаційних технологій є конкурентоспроможним і має перспективи для подальшого розвитку. Водночас, на думку експертів з консалтингової компанії PricewaterhouseCoopers, індустрія ІТ-послуг в Україні у будь-якому випадку зростатиме, однак через систематичні проблеми відбудеться зниження темпів після 2019 року. Ключовими елементами ІТ-послуг в Україні, за даними фахівців PricewaterhouseCoopers [12], є такі:

• ІТ-підтримка та аутсорсинг – різноманітні послуги для підтримки, розміщення, оновлення, інтеграції та налаштовування ІТ-продуктів, розроблених третіми сторонами.

• Custom Application Development – послуги під час повного або частини циклу розробки програмного забезпечення, що дозволяють забезпечити додаткову вартість шляхом розробки продукту, хоча право на інтелектуальну власність повністю належить клієнтам.

• ІТ-консалтинг та оцифровування – частина CAD- або окремих сервісів, що складаються з послуг консалтингу.

• Аутсорсинг R&D бізнес-процесів – послуги з великою доданою вартістю, що представляють суміш ІТ-консалтингу та CAD, але з невизначеними межами та постійними повтореннями циклу розробки ПЗ.

За розрахунками аналітиків State of European Tech [6], у 2018 р. в Україні працювало 184 700 розробників програмного забезпечення, що на 12 тисяч більше, ніж у 2017 р. Таких спеціалістів в Україні понад утричі більше, ніж у Білорусі – 54 200 осіб, але менше, ніж у Польщі – 279 800 осіб та у Німеччині – майже 851 000. Отже, Україна увійшла в топ-10 країн Європи за кількістю розробників ПЗ. Усього ж в Україні нараховують 4 тисячі ІТ-компаній. Інвестиції у їх роботу в 2018 р.

склали 290 млн дол. США. Найбільше офісів ІТ-підприємств розташовано у м. Києві, Київській, Харківській, Львівській, Дніпропетровській та Одеській областях. Більшість українських ІТпідприємств реалізують бізнес-модель розробки програмного забезпечення на замовлення іноземних клієнтів, за якої усі інтелектуальні права є власністю замовника (аутсорсинг-розробки).

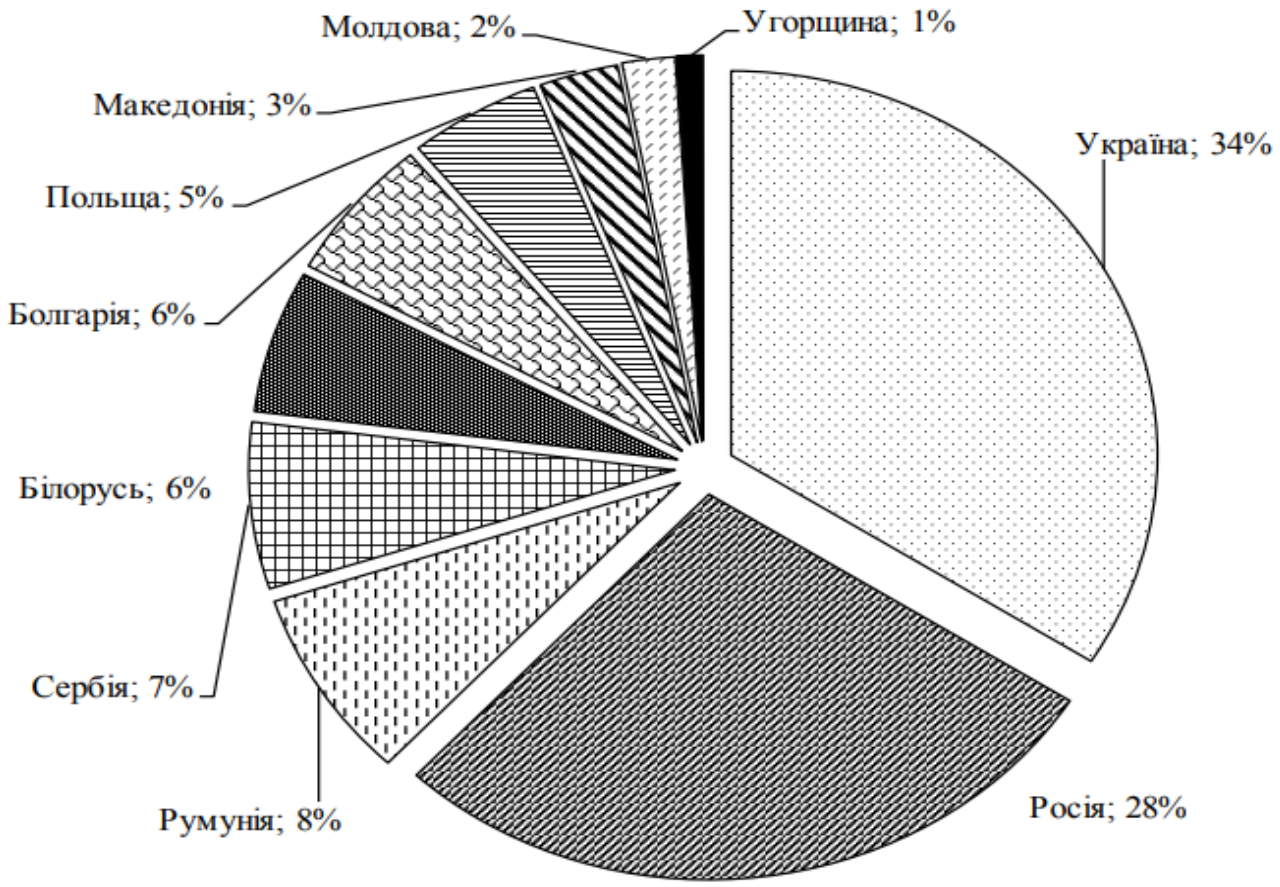


Рис 1.1. Частка країн за показником обсягу замовлень.

Це пов’язано, насамперед, з низькою дієвістю українського законодавства із захисту інтелектуальної власності, а відтак, низькою привабливістю українського ринку та використанням переважно імпортного ПЗ в Україні [7]. Для України основними ринками експорту є Сполучені Штати та Східна Європа. На рис. 1 (за даними [6]) показані відомості щодо часток країн у загальному обсязі замовлень з розробки програмного забезпечення за 2016 р.

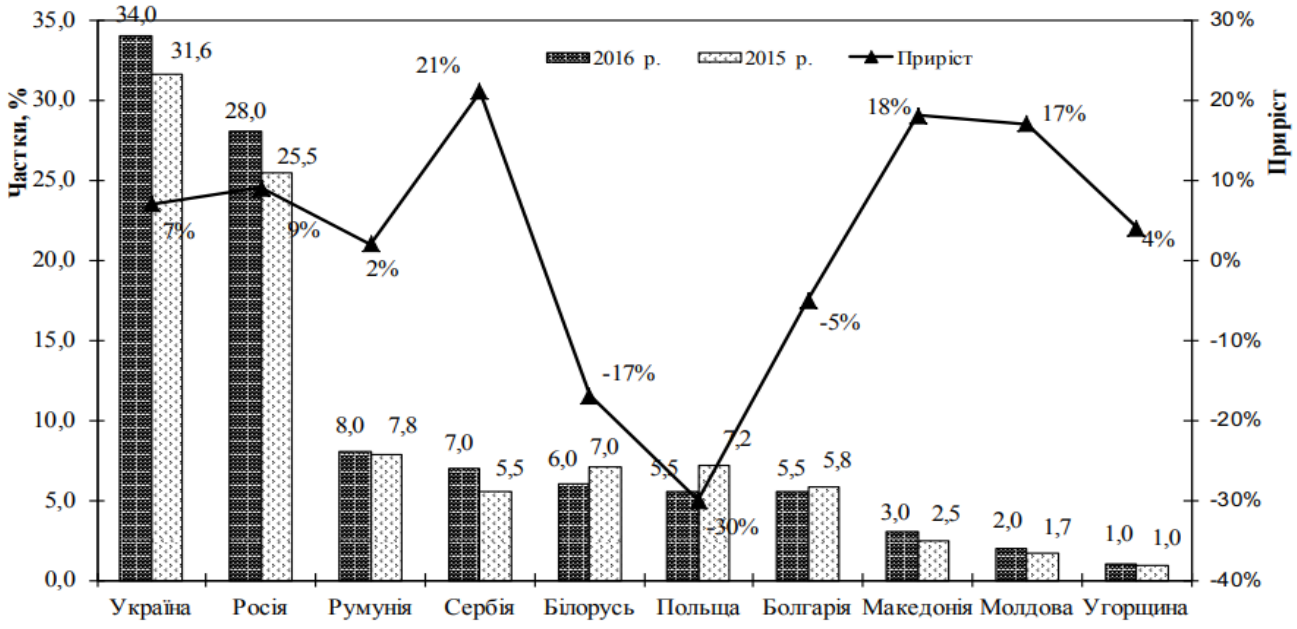


Рис. 1.2. Порівняння часток країн за обсягом замовлень із попереднім роком

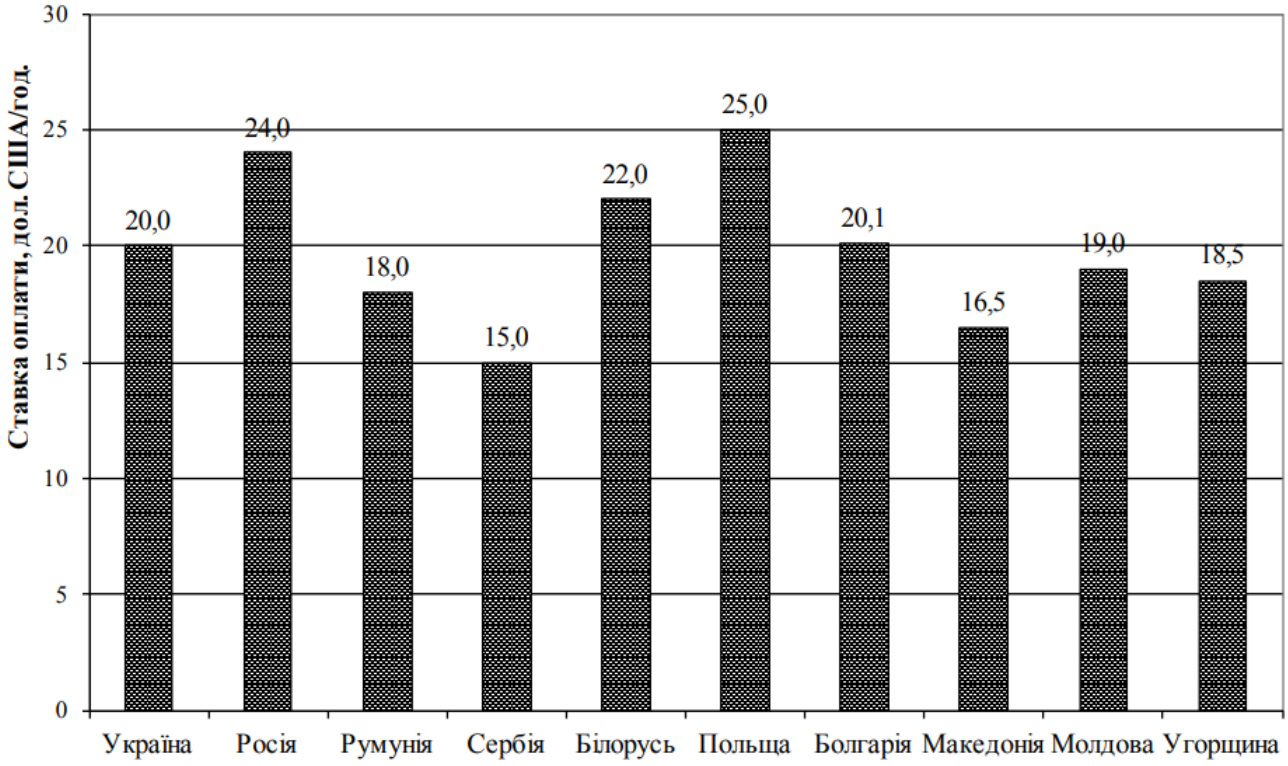


Рис. 1.3. Ставки оплати праці розробників ПЗ за країнами

Як бачимо, Україна знаходиться на першому місці (понад третина замовлень) та за останній рік мала приріст 7% (рис. 2, за даними [6]). На рис. 3 представлені

дані щодо оплати (у дол. США за год.) для основних країн-розробників ПЗ. Як видно з рис. 3, оплата розробників з України є дещо нижчою (20 дол. / год.), ніж у Польщі (25 дол. / год.), Росії (24 дол. / год.) та Білорусі (22 дол. / год.), але не найнижчою у Європі. Цей показник разом з високим рівнем працівників і якісною освітою приводить до того, що Україна залишається лідером ринку IT-фріланса Східної Європи. Але метою української IT-індустрії наразі є не тільки збільшення об'ємів надання послуг з аутсорсингу, а й перехід до розробки та виробництва власних продуктів. За даними IT Ukraine Association, 72% ITспеціалістів – чоловіки і 28% – жінки. Середній вік українського спеціаліста 30 років, а середня зарплата у 2017 р. склала 1600 дол. США [9].

Значна кількість IT-спеціалістів співпрацюють з компаніями як фізичні особи-підприємці й самостійно сплачують податки. Абсолютна більшість з них перебуває на спрощеній системі оподаткування і сплачує єдиний податок. Обсяг його надходжень зростав у середньому на 58.8% протягом 2013–2017 рр. і становив 3.2 млрд грн у 2017(рис 1.4).

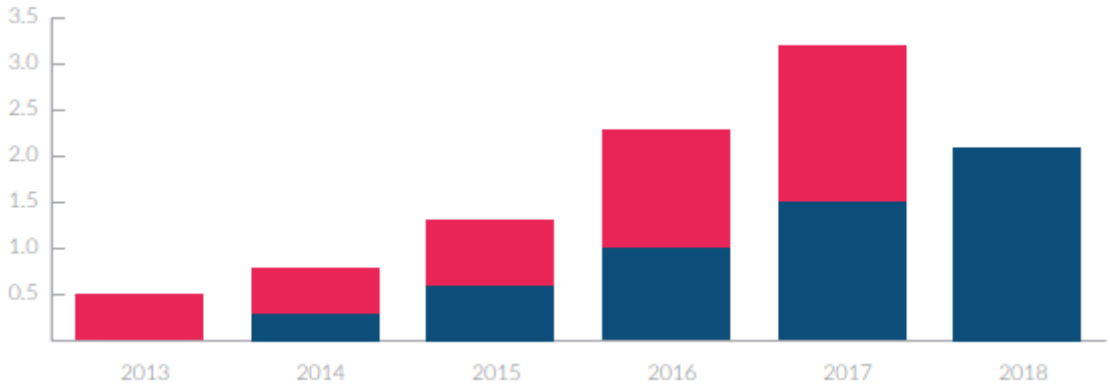


Рис. 1.4. Динаміка надходжень єдиного податку (I та II півріччя), млрд грн

Слід також зазначити, що єдиний податок, який сплачують ФОП, на 100% залишається в місцевих бюджетах за місцем реєстрації, тоді як ПДФО розподіляють серед місцевих бюджетів різного рівня на 75%, а податок на прибуток на 90% надходить до центрального. Тож можна сказати, що працівники IT-сфери сприяють наповненню місцевих бюджетів і розвитку регіонів. Так, надходження єдиного податку від представників IT-галузі у м. Києві 2017 року становили лише 28% від

загальних, і ще 10 областей перевищили планку в 2%,8 що свідчить про більш рівномірний розподіл, ніж, наприклад, податку на прибуток.

1.2 Проблеми пошуку роботи на ринку IT

На мій погляд, в наймі є кілька загальних проблем, незалежно від області. Головна проблема - протилежність інтересів. Кожен учасник хоче максимізувати прибуток. Тобто роботодавець хоче отримати максимум праці за меншу оплату, працівник хоче багато одержувати і мало робити. І це нормально, так і має бути. По суті, договір найму фіксує якісь середні умови, які часто не влаштовують жодну зі сторін, але обидві сторони миряться з цим компромісом. Але смиренність не їсти прийняття, і у всіх залишаються деякі сумніви. Здавалося б, просте рішення - чесно озвучити взаємні вимоги і далі вже, підписавши договір, не сумніватися.

Але, раптово, друга проблема - кваліфікаційна. Роботодавець хоче знайти того, хто буде здатний ефективно вирішувати необхідні завдання, а працівник хоче займатися тим, що йому цікаво. Тут корінь взаємних сумнівів знову в чесності і усвідомленості. Чи про всі вимоги говорить роботодавець і чесний працівник в тому, чи відповідають його здібності та інтереси потребам і інтересам роботодавця?

Здавалося б, відкритість врятує світ, так будемо ж чесні і в цьому питанні. Але виникає третя проблема, соціальна. Всім працівникам неприємно усвідомлювати, що вони отримують менше, ніж колеги, сусіди, знайомі знайомих і т.д. А подекуди ще є і профспілки, або навіть колективний несвідомий / ненавмисний змову (це коли велика частина фахівців без явної домовленості завищує планку очікуваної своєї зарплати або занижують планку вартості найманої праці). Як підсумок - відчуття несправедливості, часто взаємне. Та до того ж роботодавець впевнений, що платить занадто багато, а працівник упевнений, що йому недоплачують. І всі учасники процесу, звичайно, в курсі того, що саме так і буде, а тому намагаються виторгувати переваги заздалегідь (ага, запросити більше, може отримаєш приблизно стільки, скільки хотів). Ситуацію періодично посилює

четверта проблема - дефіцит. Іноді для роботодавця потрібних фахівців немає поблизу, або не вистачає коштів на оплату висококласного працівника, або є дефіцит розуміння того, які взагалі фахівці потрібні, або ці фахівці на ринку є, але нарозхват настільки, що очікуваний прибуток від фахівця нижче тієї ціни, за яку його вдасться переманити. Це теж породжує масу перекосів. А поруч пасеться п'ята проблема - буває ж і відверта брехня, чому всі учасники процесу змушені бути обережним, перевіряти один одного, закладати у вартість ризики. Прикладів того, як ці проблеми проявляються і вирішуються, якщо вирішуються взагалі, вагон і маленький візок. Ідеальних рішень немає (хоча, подейкують, у минулі часи єдина по всій країні ставка плюс-мінус надбавки і обов'язковий розподіл, з одного боку мали свої позитивні сторони). Процес співбесіди не набагато кращий. Запросивши когось до себе додому, ви зобов'язані цій людині ввічливістю. Коли ви востаннє запрошували когось у вашому домі до вас у гості, і ви просто сиділи там, ігноруючи їх чи не розмовляючи з ними? Напевно, ніколи, бо це незручно, грубо, і людина, мабуть, піде і не захоче повертатися. Чому компанії вважають, що те саме не стосується їх? Приходьте в наші офіси, зустрічайтесь із працівниками, відповідайте на наші запитання, щоб дізнатись, чи підходите ви до нашої компанії, і як тільки ви поїдете, можливо, ви нам відповісте. Якщо вам пощастить через пару тижнів, ви отримаєте автоматичний електронний лист із системи, коли HR закриє запит на роботу, повідомляючи вас, що ми вибрали когось, крім вас самого. Якщо вам справді пощастить, рекрутер насправді передзвонить вам, щоб повідомити, що вони перейшли до іншого кандидата. Цей дзвінок займає 2 хвилини, він повинен бути стандартним, не винятком. Мені шкода, занадто зайнятий, занадто багато кандидатів, занадто багато резюме, знову копія. Ви очікуєте, що кандидат передзвонить вам, з’явиться вчасно, одягнеться належним чином, відповість на всі ваші запитання і буде професійним у будь-який час під час прийому на роботу, але компанія може поводитися з кандидатом, як хоче, особливо якщо вони не вибрани.

1.3 Аналіз існуючих платформ

Наприкінці серпня dou.ua провели опитування щодо пошуку роботи і зібрали 4008 анкет від IT-фахівців. Тож, давайте подивимося, де програмісти (і не тільки вони) знаходять роботу, які канали пошуку вважають найефективнішими та чи слідкують за ринком праці(рис. 1.6).

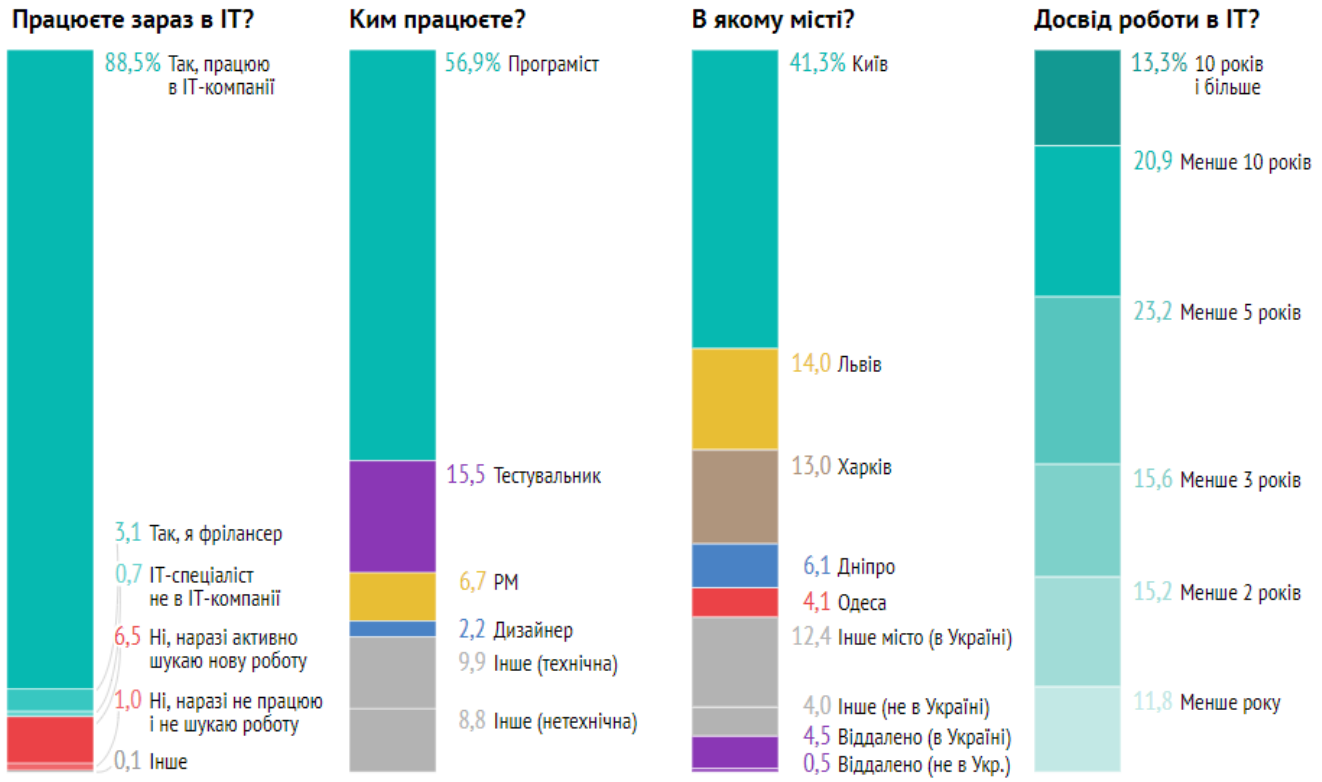


Рис. 1.6. Потрет учасників опитування

Отже, щодо каналів пошуку роботи в IT-сфері, то найрезультативнішими виявились особисті рекомендації: 26% опитаних відповіли, що знайшли поточне місце роботи за рекомендацією друга або колеги; а ще 4% запросив знайомий рекрутер. Наступним за результативністю є LinkedIn: за допомогою цієї мережі працевлаштувались 16% опитаних. Профільні сайти також активно сприяють у пошуках роботи в IT: на Джині та DOU знайшли роботу 12% та 10% опитаних відповідно. Дещо менш ефективним способом пошуку роботи в IT виявилися сайти

пошуку роботи Work.ua (7%) та Rabota.ua (7%). Курси та навчальні центри при компанії надали роботу 6% опитаних. Ще для 6% успішною виявилася спроба надіслати резюме на сайт компанії. А 2% відповіли, що з ними зв’язався незнайомий рекрутер із запрошенням на роботу через скайп, особисту пошту або за допомогою телефонного дзвінка.

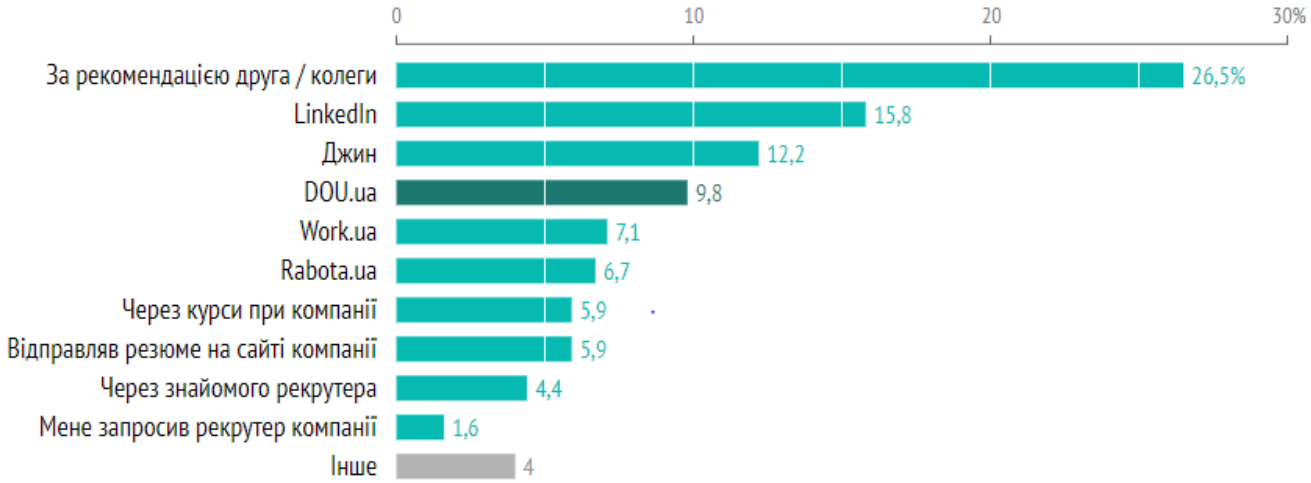


Рис. 1.7. Як люди знайшли поточне місце роботи

Якщо ж розглядати окремо за спеціальностями, то програмісти частіше за інших знаходять роботу через LinkedIn (18%) та Джин (14%) (див. рис 1.7.) . Курси при компанії, а також відправка резюме на сайт компанії — канали пошуку роботи, які ефективніше працюють для тестувальників, і трохи менш ефективні для всіх інших спеціальностей. Так, курси або навчальний центр допомогли в працевлаштуванні 11% тестувальників, і 7% тестувальників влаштувалися на роботу після відправки резюме на сайт компанії. Менеджери проектів частіше за інших знаходять роботу за рекомендацією: 31% знайшли поточну роботу за рекомендацією друга або колеги, і ще 7% — через знайомого рекрутера. Також частіше за інших менеджери проектів знаходять роботу на DOU (12%). На сайтах пошуку роботи частіше знаходять роботу дизайнери, а також спеціалісти інших технічних або нетехнічних спеціальностей.

Незалежно від того, шукають опитані наразі роботу чи вже мають постійне місце, вони досить активно стежать за ринком праці. Так, більшість респондентів

відповіли, що читають зарплатні опитування й переглядають вакансії на DOU, а ще відповідають рекрутерам на LinkedIn. Найуважніше за зарплатними новинами стежать тестувальники: 85% тестувальників відповіли, що читають зарплатні опитування на DOU, а 43% читають зарплатну розсилку Джина. Дизайнери активно моніторять вакансії та відгукуються на ті, що їх зацікавили найбільше: 63% дизайнерів переглядають вакансії на DOU; 37% переглядають вакансії на інших сайтах і 24% відгукуються на вакансії, які їх особливо зацікавили.

Найактивніше на співбесіди ходять програмісти: 16% програмістів зазначили, що відвідують співбесіди, аби бути у формі, і ще 5% — для того, щоб отримати надбавку до зарплати. Також частіше за інших обирали варіанти про співбесіди респонденти зі Львова: 20% респондентів зі Львова сказали, що ходять на співбесіди, аби бути у формі, і 8% — для того, щоб отримати надбавку до зарплати. Для порівняння: в Києві такі відповіді дали 14% і 4% відповідно. І новачки, і ті, хто вже давно в IT, активно стежать за ринком праці. Відрізняються лише способи. Так, якщо новачки активно переглядають свіжі вакансії на всіх доступних платформах, то корифеї відповідають рекрутерам в LinkedIn, читають зарплатну розсилку Джина, відгукуються на вакансії, які їх особливо зацікавили, і відвідують співбесіди.

1.4 Аналіз анонімних платформи

Застосування анонімних заявок на роботу (або сліпого найму) для боротьби з дискримінацією за наймом привертає увагу та інтерес. Результати польових експериментів (рис. 1.8) та пілотних проектів у європейських країнах (тут розглядаються Франція, Німеччина, Нідерланди та Швеція), Канаді та Австралії проливають світло на їхній потенціал щодо зменшення деяких дискримінаційних бар'єрів для найму меншин та інших груп, що знаходяться в неблагополучному положенні. Але хоча цей підхід може досягти своїх основних цілей, слід враховувати також важливі застереження.

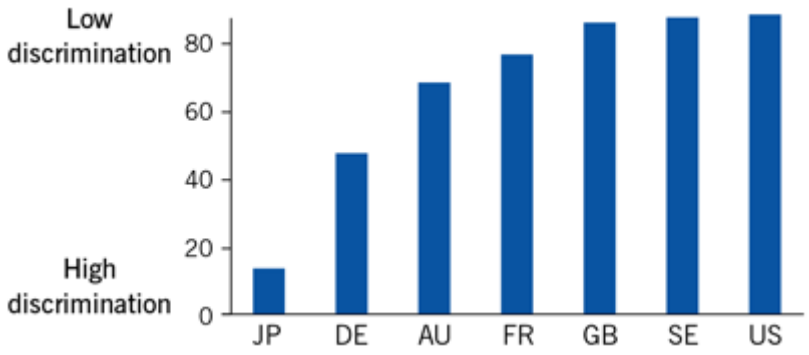


Рис. 1.8. Країни по антидискримінійній політиці

Плюси анонімного пошуку роботи:

- Анонімні заявки на роботу можуть запобігти дискримінації на початковому етапі перевірки набору.
- Анонімні заявки на роботу можуть підвищити рівень пропозиції роботи для кандидатів меншин.
- Анонімні заявки на роботу сигналізують про рішуче прагнення роботодавця зосередитись виключно на навичках та кваліфікації.
- Стандартизовані анонімні форми заявок на роботу є ефективним методом реалізації
- Порівнянність претендентів на роботу може зрости із використанням анонімних заявок на роботу.

Мінуси анонімних платформ:

- Анонімні заявки на роботу можуть зменшити дискримінацію лише тоді, коли дискримінація висока.
- Анонімні заявки на роботу можуть просто відкласти дискримінацію на потім у процесі найму.
- Набір сліпих може зірвати інші позитивні заходи, що сприяють збільшенню різноманітності, і може обмежити можливості для позитивних дій.
- Субоптимальне впровадження анонімних процедур подання заявок на роботу може бути дорогим, трудомістким, трудомістким та схильним до помилок.

- Інформація, що стосується конкретного контексту, може бути інтерпретована невигідно, якщо особа кандидата невідома.

Дискримінація є не лише несправедливою та потенційно дорогою для людей, які її зазнають, але також призводить до великих економічних витрат для суспільства. Хоча дискримінація існує на багатьох ринках світу, найбільша увага приділяється дискримінації на ринку праці. Ключовим бар'єром є доступ до робочих місць. Вражаюче різні ставки зворотного дзвінка після первинних заявок на роботу були задокументовані для заявників з меншин або інших груп, що знаходяться в неблагополучному положенні, таких як іммігранти та жінки.

Але що, якби характеристики, що визначають статус групи меншин, були невідомі вербувальникам? Очевидно, дискримінація повинна стати неможливою. Анонімні заявки на роботу (або процедури набору сліпих) застосовують цю просту і зрозумілу ідею на практиці. Тепер, коли вони були протестовані в декількох європейських країнах, Канаді та Австралії, можна оцінити деякі можливості та межі цього нового інструменту політики. Незважаючи на те, що це може здатися неінтуїтивним, основна гіпотеза полягає в тому, що менша кількість інформації може призвести до кращого вибору (і результатів) - принаймні на початковій стадії процесу найму. Дискримінація на ринку праці приймає різні форми, але, як видається, вона найчастіша при наймі на роботу, ніж, наприклад, як компенсація. Ступінь дискримінації при прийомі на роботу була задокументована в багатьох дослідженнях, які можна широко розділити на аудиторські та заочні дослідження. Хоча аудиторські дослідження, в яких використовуються відповідні пари учасників з однаковими характеристиками, за винятком одного (наприклад, етнічної приналежності), давно використовуються для документування поширеної дискримінації, вони зазнають критики, оскільки заявники з різних груп можуть виявитися (інакше) не повністю ідентичними роботодавцям . Кореспондентські дослідження вирішують це занепокоєння шляхом вимірювання дискримінації на основі вигаданих заявників паперу. Їх результати є настільки ж переконливими, і

дослідження, як правило, документують суттєву дискримінацію на початковому етапі найму у багатьох країнах, яка до того ж зберігається з часом.

1.5 Методи покращення процессу найму в Україні

Анонімні заявки на роботу можуть усунути або зменшити деякі дискримінаційні бар'єри для прийому на роботу, з якими стикаються кандидати з меншин та інших груп, що знаходяться в неблагополучному положенні. При ефективному впровадженні анонімні заявки на роботу вирівнюють умови для доступу до робочих місць, переносячи фокус на навички та кваліфікацію. Проте анонімні заяви на роботу не слід розглядати як універсальний засіб, який застосовується в будь-якому контексті або може запобігти будь-якій формі дискримінації. Зменшення або усунення дискримінації за наймом може мати великі переваги. Окрім встановлення рівності можливостей та сигналізації прагнення роботодавця зосередитись виключно на навичках та кваліфікації, очікується, що зменшення дискримінації збільшить різноманітність на робочому місці. Крім того, якщо рішення про набір персоналу базуються виключно на навичках та кваліфікації, результати повинні автоматично відповідати меті фірми наймати найпродуктивніших робітників. Також можуть бути додаткові наслідки, наприклад, на шляху кар'єрного росту та заробітної плати працівників, які потенційно дискриміновані. Анонімні заявки на роботу можуть бути практичним методом досягнення цих переваг. Анонімні процедури вже давно застосовуються в інших сферах. Наприклад, вчені вже давно використовують подвійні чи сліпі процедури в експериментальних дослідженнях або для рецензування робіт інших дослідників. Сліпі прослуховування симфонічних оркестрів продемонстрували сильний вплив на гендерний склад. Цей досвід показує, що для досягнення запланованих результатів можна прийняти рішення або відібрати анонімно. Практики розкриття заявок на роботу значно різняться між різними ринками праці. Європейські та

північноамериканські країни працюють порівняно добре з точки зору загальної антидискримінаційної політики та законодавства (див. Ілюстрацію), але це не гарантує рівності у всіх аспектах життя. Тип інформації, яка може бути законно запитана у заявах на роботу, варіюється в широких межах. Тоді як явне резервування роботи для особи певної раси чи статі було незаконним у США з 1960-х років, в багатьох азіатських країнах вимагають дуже детальної особистої інформації. Наприклад, китайські оголошення про роботу часто розділяються за статтю, а південнокорейські форми заявок можуть містити питання щодо таких особистих питань, як звички куріння та пиття, зріст і вага, група крові та фінансовий стан.

На даний момент на українському ринку є такі рішення анонімного пошуку роботи для IT – спеціаліста:

- Djinni.co
- Jobla.ua
- Skyworker.io

Ці сервіси дійсно користуються попитом на українському IT-ринку. Але пропоную розглянути новий метод та технологію для покращення цих сервісів та рекрутингу у цілому. Він доволі тісно пов’язаний з відео-конференцією, але має на увазі трохи інше. Це - метод запису техніч-ного інтерв’ю.

Уявіть, що на одне інтерв'ю кандидат витрачає від 1 до 3 годин часу. І це не враховуючи час на очікування, час на те щоб домовитися про це інтер-в'ю. І навіть після витраченного часу роботодавець не можете бути впевненим, що та чи інша особа під-ходить на проект, бо є шанс знайти кращого канди-дата. У такому разі роботодавець буде шукати далі. Даний метод пропонує записувати та зберігати тех-нічні інтерв'ю кандидатів для подальшого їх викори-стання. Це допоможе зберегти час на найм тим, що технічний керівник зможе відразу продивитися інтерв'ю кандидата та прийняти рішення: чи треба запрошувати його на більш детальну співбесіду.

1.6 Опинування кореспондентів

У опитуванні(рис 1.9) приймали участь програмісти різних рівнів та різного напрямку. Майже всі зазначають, що процес є достатньо затяжним і тривалим. Є компаніх з великим досвідом за своїми плечами і професіональними кадрами, які дозволяють зробити цей процес набагато швидшим. Але їх достатньо мало.

Компания	Начало общения	HR-собес	Тех. Собес	Результат	На I этап	На II этап	На III этап	Итого
luxoft.com	12 авг 2019	21 авг 2019	27 авг 2019	18 сен 2019	9	6	22	37
think-cell.com	10 авг 2019	12 авг 2019	-	-	2			
Upland	22 авг 2019	27 авг 2019	30 авг 2019	-	5	3		
NPO Vertikal	4 сен 2019	5 сен 2019	-	-	1			
MCG	2 сен 2019	4 сен 2019	-	-	2			
Ignite	16 сен 2019	17 сен 2019	18 сен 2019	24 сен 2019	1	1	6	8
GlobalLogic	13 сен 2019	16 сен 2019			3			
AMC Bridge	26 сен 2019	30 сен 2019	-	-	4			
Lohika					0	10	1	11
3dlook					5	5	28	38
Stamavi					1	12		
Dataart	1 сер 2019	5 сер 2019	14 сер 2019	27 сер 2019	4	9	13	26
Beetroot	12 сер	16 сер		27 сер	4		11	15
Enapps	30 сер		2 вер.	12 вер.		3	10	13
Echo	4 вер.		5 вер.	10 вер.		1	5	6
COAX	3 вер.	5 вер.	10 вер.	17 вер.	2	5	7	14
AnvilEight	18 вер.	23 вер.	23 вер.	24 вер.		5	1	6
StartupSoft	20 вер.	23 вер.	24 вер.	26 вер.	3	1	2	6
Preply	9 вер.	11 вер.	16 вер.	26 вер.	2	5	10	17
DataRobot	19 лип.	29 лип.			10			
CaptainGrowth	29 лип.	2 сер	8 сер		4	6	1	11
Probegin	16 вер.	19 вер.	19 вер.	4.10.2019	3	0	15	18
SportLabs	7 сер	-	17 сер	23 сер		10	6	16
eTeam	19 вер.	20 вер.	24 вер.	4 жов	1	4	10	15
SoftServe	3 чер	18 чер	19 чер	27 чер	15	1	8	24
ElifTech	27 тр	29 тр	12 чер	24 чер	2	14	12	28
CloudSimple	13 чер	14 чер	16 чер	20 чер	1	2	4	7
Essentia One	18 січ.	31 січ.	31 січ.	31 січ.		13	0	13
Datafbrest	2 вер. 2019	-	10 вер. 2019	-		8		
StartUpSoft	11 вер. 2019	17 вер. 2019	24 вер.2019	26 вер. 2019	6	7	2	15
Gameloft	12 июль 2019	14 июля 2019	18 июля 2019	24 июля 2019	2	4	6	12
WeSoftYou	7 вер. 2018	-	-	14 вер. 2018			7	7
Introlab System	5 февраля 2019	-	10 февраля	18 февраля		5	8	13
Lexico Telecom	25 вер. 2018	28 вер. 2018	4 жовтня 2018	6 жовтня 2018	3	6	2	11
DataObrii	7 июля 2019	14 июля 2019	19 июля 2019	5 августа	7	5	16	28
Sintez Technologie	17 июня 2019	-	9 июля 2019	11 июля		22	2	24
Soft-Industry	20 травня 2018	-	25 травня 2018	31 травня 2018		5	6	11

Рис. 1.9 . Опитування програмістів щодо часу найму.

Висновки до розділу

Ринок ІТ є дуже виликим і коорисним для економіки України. Він приносить кошти, інвесторів. Через це бюджет поповлюється налогами, що є корисним для кожного громадянина. Але щоб якось покращити і стимулювати цей ринок треба проаналізувати його проблеми і знайти ріщення. Одне з таких рішень – метод запису технічного інтерв'ю задля повторного його використання. Це дозволить кандадитам витрачати меньше часу і добавить гнучкість у процесі найму. Слід за цим, компаніям буде легче закривати свої позиції.

2. ПІДВИЩЕННЯ ЕФФЕКТИВНОСТІ СЕРВІСУ ДЛЯ ОБРОБКИ ТЕХНІЧНОГО ІНТЕРВ'Ю КАНДИДАТА

2.1 Загальний опис проекту

Як визначено, для покращеня найму і процесів пошуку роботи, потрібно організувати платформу, яка може витримувати великі нагрузки. У перспективі, вона вкраде частку ринку у конкурентів. Окрім цього, вона має здійснювати увесь потрібний функціонал і бути зручною у використанні. Задля організації метода запису технічного інтерв'ю треба продумати момент зберігання тих самих інтерв'ю, як воно буде оброблятися і як краще організувати алгоритми оптимизацію обробки відео. Сервіс матиме назву «Interview.top». Ця назва ідеально підкреслює сенс платформи та її призначення. Для створення логотипу були задійсненні професійні дизайнери. Виглядає він наступним чином:

Рис. 2.1. Логотип проекту «Interview.top»

2.2 Обгрунтування мов программування для бекенду

Для створення нашого продукту мова буде йти як і про Back-end розробку, так і про Front-end. Слід ретельно обгрунтувати вибір мов програмування для обоїх частин проекту, бо від цього залежатиме успіх проекту.

Для бекенд розробки найпопулярніші наступні мови програмування:

- PHP - скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення. PHP — мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег. Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в C або Pascal. Важливою перевагою PHP є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте хоч би що робили розробники PHP, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність PHP достатня для створення цілком серйозних веб-застосунків.
- Java - об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Оскільки Java програми виконуються віртуальною машиною Java, то це призводить до дещо нижчої швидкодії, порівняно з самого початку скомпільованими у машинний код програмами (наприклад, написаними на C++). Проте за останнє десятиліття розробники віртуальної

машини значно пришвидшили цей процес, тож в даний час програми на Java не надто поступаються аналогам на C++. Іншою проблемою Java є те, що для низькорівневого програмування (для роботи з апаратним забезпеченням) все ж необхідно використовувати модулі написані на інших більш пристосованих для цього мовах програмування (той же C++). Для стандартних задач, як то робота із портами комп'ютера, для Java уже існують готові сторонні native-бібліотеки. Програми, які використовують платформно залежні модулі уже не є настільки портабельними і потребують, щоб дані модулі були реалізовані для різних систем. Ще однією з ґрунтовних проблем Java є безпека Java аплетів — спеціальних програм на Java, що можуть вбудовуватися у веб-сторінки. Щоправда останні так і не набули значної популярності в інтернеті, поступившись іншим аналогічним технологіям, як то флеш технологія.

- Ruby - це інтерпретована, повністю об'єктно-орієнтована мова програмування з чіткою динамічною типізацією. Мова вирізняється високою ефективністю розробки програм і увібрала в себе найкращі риси Perl, Java, Python, Smalltalk, Eiffel, Ada і Lisp. Ruby поєднує в собі Perl-подібний синтаксис із об'єктно-орієнтованим підходом мови програмування Smalltalk. Також деякі риси запозичено із мов програмування Python, Lisp, Dylan та CLU. Не вимагає попереднього оголошення змінних, хоча для інтерпретатора бажано, щоб змінній присвоювалось порожнє значення nil (тоді інтерпретатор знає, що ідентифікатор вказує на змінну, а не на ім'я методу). Може динамічно завантажувати розширення, якщо це дозволяє операційна система. Містить автоматичний прибиральник сміття. Він працює для всіх об'єктів Ruby, в тому числі для зовнішніх бібліотек
- Python - часи, коли використання Python обмежувалося десктопними додатками, канули в лету. Сьогодні ця мова займає почесне місце в царстві бекенд веб-розробки. Більшість сучасних програмістів, які займаються бекенд, застосовують Python в якості заміни PHP. Ця мова міцно утримує лідируючі позиції і з часом стає тільки краще. Python популярний, легкий і має досить низьку криву вивчення. Python це досить гнучкий мова. Також його

можна розглядати як своєрідний «пропуск» до вивчення інших мов програмування. Серед фреймворків Python обов'язково слід згадати Django. Він відрізняється надійністю і здатний задовольнити всі ваші потреби в тому що стосується бекенд. Саме цю мову программування буде обрано для розробки даного проекту.

2.2.1 Мова програмування Python

Python з технічної точки зору, є об'єктно-орієнтованою мовою програмування високого рівня з інтегрованою динамічною семантикою, перш за все для розробки веб-додатків та активно використовується для Data Science. Вона є надзвичайно привабливою в області швидкого розробки додатків, оскільки Python пропонує динамічну типізацію та динамічні можливості прив'язки.

Python є відносно простим, тому його легко впізнати, оскільки він вимагає унікального синтаксису, який фокусується на читабельності. Розробники можуть читати код Python набагато простіше, ніж інші мови. Це, у свою чергу, знижує витрати на підтримку та розвиток програм, оскільки дозволяє командам працювати спільно без значних бар'єрів у мові та досвіді.

Крім того, Python підтримує використання модулів і пакетів, що означає, що програми можуть бути розроблені в модульному стилі, а код може бути повторно використаний в різних проектах. Як тільки ви розробили потрібний модуль або пакет, його можна масштабувати для використання в інших проектах, і їх легко імпортувати або експортувати.

Однією з найбільш перспективних переваг Python(рис. 2.1) є те, що стандартна бібліотека та інтерпретатор доступні безкоштовно, як у двійковій, так і у вихідній формі. Не існує жодної ексклюзивності, оскільки Python і всі необхідні інструменти доступні на всіх основних платформах. Таким чином, це привабливий варіант для розробників, які не хочуть турбуватися про високі витрати на розробку.

Рис. 2.2 – Логотип Python

За даними [8]:

– У 2016 році Python обігнав R на Kaggle, головній платформі для змагань з Data Science;
– У 2017 році він обігнав R на щорічному опитуванні KDNuggets з найбільш використовуваних інструментів Data Science;
– У 2018 році 66% вчених з даних повідомили, що використовують Python щодня, що робить його інструментом номер один для Data Science спеціалістів.

2.3 Обгрунтування вибору мови программування для фронтенду

Найбільш популярною мовою програмування для фронтенду э JavaScript. Мова програмування JavaScript призначений для створення інтерактивних HTML-документів. Це об'єктно-орієнтована мова розробки вбудованих додатків, що виконуються як на стороні клієнта, так і на стороні сервера. Синтаксис мови схожий на синтаксис мови Java - тому його часто називають Java-подібним. Клієнтські програми виконуються браузером перегляду Web-документів на машині користувача, серверні додатки виконуються на сервері.

При розробці обох типів додатків використовується загальний компонент мови, званий ядром і включає визначення стандартних об'єктів і конструкцій

(змінні, функції, основні об'єкти і засіб LiveConnect взаємодії з Java-аплетами), і відповідні компоненти доповнень мови, містять специфічні для кожного типу додатків визначення об'єктів.

Конструкції мови сценаріїв JavaScript вбудовуються в сторінки HTML і виконуються (інтерпретуються) під керуванням браузера при завантаженні сторінок, а також при здійсненні користувачем певних дій над об'єктами, розташованими в цих сторінках.

Рис. 2.3. Логотип JavaScript

Як щодо проблем JavaScript? У цієї мови є серйозні недоліки, його примхи, протиріччя і обмеження можуть бути вкрай неприємними для того, хто тільки починає вивчати програмування. На щастя, існують сучасні рішення, які можуть згладити більшість недоліків JavaScript.

Розберемо три основні претензії, що пред'являються до цієї мови. Найважливішою концепцією, яку початківці програмісти освоюють дуже рано, є ідея змінних, контейнерів, які зберігають інформацію під час роботи програми. Проблема JavaScript полягає в тому, що ця мова дуже вільно і неакуратно звертається зі змінними. Він дозволяє робити те, що не виглядає правильним, і ігнорує очевидні нестиковки. Його недбалість здатна перетворювати незначні помилки в катастрофи, що порушують роботу програм. Хоча JavaScript толерантно ставиться до багатьох негативних речей, якщо скомбінувати JavaScript з якісним

редактором коду і з правильними додатковими інструментами, можна створити середовище розробки, схожу на ті, якими володіють інші сучасні мови програмування. В результаті розробка на JavaScript виявляється такою ж зручною, як і на інших мовах, і, природно, інші його переваги нікуди не діваються.

Об'єктно-орієнтоване програмування (ООП) - це підхід до моделювання та організації коду. Якщо методики ООП застосовуються правильно, вони допомагають програмісту створювати простий і добре організований код. Крім того, ООП спрощує повторне використання важливих функціональних можливостей програм.

JavaScript сумно відомий відсутністю підтримки об'єктно-орієнтованого програмування. Насправді, JavaScript-розробники традиційно обходять цей недолік, використовуючи всілякі дивні конструкції. Ці конструкції можуть щось означати для того, хто вже вивчав ООП (і навіть для того, хто не вивчав, а просто скопіював в свій код якийсь шаблон і звик з ним працювати). Але якщо ви - абсолютно нова людина в програмуванні, використання подібних сумнівних конструкцій, що реалізують базові концепції програмування - це абсолютно неправильно.

На щастя, існують гарні рішення, що забезпечують підтримку ООП в JavaScript. Моє улюблене рішення такого роду - це TypeScript - опенсорсний проект, запущений Microsoft в 2012 році. TypeScript є чимось на зразок поліпшеною різновиди JavaScript, яка підтримує ООП (а також багато інших корисних можливості на зразок суворої перевірки типів).

Хтось може сказати, що ми говоримо про JavaScript, і що TypeScript - це, все ж, не JavaScript. І, насправді, це так. Але ось одна цікава деталь. Код пишуть на TypeScript, а потім конвертують його в JavaScript перед тим, як він буде виконуватися. Це дозволяє користуватися всім кращим із світів TypeScript і JavaScript. Якщо ви виберете TypeScript, то у вас буде сучасна мова програмування, на якому ви зможете писати свій код, і та найширша підтримка, якою користується звичайний JavaScript. І, що найприємніше, перетворення TypeScript-коду в JavaScript проводиться автоматично. Звичайно, в створеному комп'ютером JavaScript-коді використовуються, для відображення в ньому концепцій,

реалізованих засобами TypeScript, досить громіздкі конструкції, але це нічого не змінює. Програміст вивчає концепції ООП, а готовий код правильно працює, без проблем обробляючи засобами сучасних комп'ютерів.

Для того щоб отримати доступ до більш широкого, ніж є в мові, набору функціональних можливостей (і не винаходити велосипеди), JavaScript-програмістам потрібно використовувати бібліотеки і фреймворки сторонніх розробників. Вибір правильних інгредієнтів, які використовуються при створенні якогось проекту - це не так просто, як може здатися на перший погляд. Мова йде, зокрема, про те, що обрані додаткові інструменти повинні правильно вирішувати поставлені перед ними завдання, потрібно, щоб у програміста була б упевненість в тому, що вони ще довго будуть користуватися підтримкою своїх розробників, потрібно, щоб вони не конфліктували один з другом.

З якоїсь різновидом цієї проблеми стикаються і розробники, що використовують інші мови. Однак треба зазначити, що не всі мови страждають від цієї проблеми так само сильно, як JavaScript. Для того щоб стати серйозним програмістом у відкритому для всіх JavaScript-світі, потрібно зібрати власний набір інструментів розробки. При цьому кожен з можливих варіантів вибору настільки складний і багатогранний, що зрозуміти, чи підходить вам, скажімо, якась бібліотека, можна тільки дуже добре її вивчивши (а коли ви її вивчите, може виявитися так, що вона вже втратить актуальність і на її місце прийде щось нове, яке притягує, можливо - на короткий час, загальний інтерес).

Яким би сумним все це не було, ці проблеми, насправді, не впливають на новачків. Якщо хтось вивчає програмування з використанням JavaScript, то йому краще всього триматися в стороні від фреймворків і бібліотек, чи йде мова про щось широко відомому, начебто jQuery, Angular, React або Vue, або про щось винайденому в той момент, коли було написано цю пропозицію, або в ту секунду, коли була опублікована ця стаття. Звичайно, новачок, добре освоївши фундаментальні речі, ймовірно, захоче ознайомитися, як мінімум, з одним з популярних додаткових інструментів. Але це - вже зовсім інша історія.

2.4 Вибір фронтенд фреймворку та бібліотек для реалізації запису інтерв’ю.

Якщо відштовхуватися від популярності і поширеності інструментів фронтенд-розробки, то ось - п'ять найбільш помітних JavaScript-фреймворків і бібліотек:

- React
- Vue
- Angular
- Ember
- Backbone.js

2.4.1 React

В JavaScript-світі React - це, безумовно, лідер. У цій бібліотеці використовуються ідеї реактивного програмування, вона вводить в фронтенд-розробку і безліч власних концепцій. Для того щоб гнучко використовувати React в розробці веб-проектів, потрібно вивчити безліч додаткових інструментів.

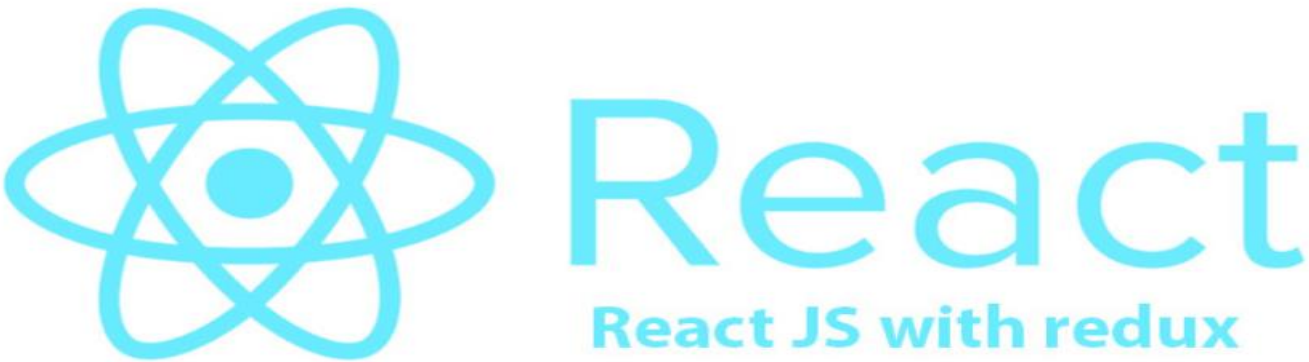


Рис 2.4. Логотип React фреймворку

Ось, наприклад, далеко не вичерпний список подібних інструментів, представлений бібліотеками, які можна використовувати спільно з React. Це - Redux, MobX, Fluxy, Fluxible, RefluxJS. У React-розробці, крім того, можна використовувати jQuery AJAX, Superagent, Axios і Fetch API.

2.4.2 Angular

На конференції AngularConnect 2019 команда розробників Angular зробила заяви, які дозволяють вважати вихід Angular 9 поворотною точкою в розвитку цього фреймворка. Зокрема, планується зробити компілятор Angular Ivy стандартним засобом, доступним для всіх додатків. Основні переваги цієї технології полягають в тому, що її застосування дозволяє прискорити процес розробки, зменшити розмір додатків, підвищити їх продуктивність і надійність.Сучасний Angular - це просунутий модульний фреймворк для фронтенд-розробки. Раніше для підключення Angular на сторінку досить було просто додати в її HTML-код відповідний тег, тепер же розробник може імпортувати в свій проект необхідні йому модулі Angular.Angular відомий своєю гнучкістю. Саме тому все ще актуальні версії Angular 1.x. Однак багато розробників в наші дні користуються Angular 2+ через MVC-архітектури фреймворка, яка значно змінилася в бік архітектури, заснованої на компонентах.Тому, хто хоче користуватися Angular, доведеться, при освоєнні цього фреймворка, зіткнутися з деякими труднощами. Так, для створення Angular-додатків практично обов'язково використовувати TypeScript. Хоча це і ускладнює роботу з Angular, у такого стану справ є свої плюси. Зокрема, це підвищує надійність додатків за рахунок просунутого контролю типів, це дає програмісту додаткові кошти розробки.

Рис. 2.5. Логотип Angular

Елементи Angular:

- Інструменти командного рядка, що спрощують створення заготовок додатків.
- Набір модулів, які можна використовувати при розробці Angular-проектів: @ angular / common, @ angular / compiler, @ angular / core, @ angular / forms, @ angular / http, @ angular / platform-browser, @ angular / platform-browser - dynamic, @ angular / router і @ angular / upgrade.
- Бібліотека Zone.js, яка дозволяє управляти контекстами виконання коду.
- Мови TypeScript і CoffeeScript.
- Обмін даними в Angular-додатках можна організувати з використанням RxJS і спостережуваних (Observable) об'єктів.
- Angular Augury - засіб для налагодження Angular-додатків.
- Технологія Angular Universal, призначена для організації серверного рендеринга Angular-додатків.

Angular - це повноцінний JS-фреймворк, що дає сучасному веб-програмісту все необхідне для продуктивної роботи. На Angular варто звернути увагу тим, кому хотілося б отримати в своє розпорядження великий набір стандартних засобів і звести до мінімуму використання сторонніх бібліотек.

2.4.3. Vue

Vue - це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Рис. 2.6. Логотип VueJs

При роботі з Vue логіка компонента, його макет і стилі зберігаються в одному файлі. Так само, за винятком стилів, матеріали проектів зберігаються і в React. Взаємодія компонентів в Vue забезпечується за допомогою об'єктів, які зберігають властивості і стан компонентів. Цей підхід теж, ще до того, як він з'явився в Vue, був використаний в React.

Vue, що ріднить його з Angular, дозволяє змішувати HTML-розмітку і JavaScript-код. Для того щоб інтегрувати дані компонента в шаблон, потрібно використовувати директиви Vue. Такі, як v-bind або v-if.

Одна з причин, по якій Vue варто розглядати як гідну альтернативу React, полягає в тому, як тут організовано управління станом додатки. У React-проектах, при використанні зв'язки React + Redux, у міру зростання розмірів додатки

ускладнюються і процедури, необхідні для управління його станом. Це може звестися до того, що програмісту, замість роботи над самим додатком, доводиться витрачати чимало часу на налаштування механізмів Redux. В Vue для управління станом використовується бібліотека Vuex. Вона схожа на Flux і створена спеціально для Vue. Працювати з нею набагато зручніше, ніж з Redux.

Якщо ви намагаєтеся зробити вибір між Vue і Angular, то причини, за якими можна віддати перевагу Vue, можна звести до того, що Angular, в порівнянні з Vue, виглядає переусложнённим великомасштабним проектом, в природі якого закладено прагнення обмежувати розробника. Vue ж набагато простіше ніж Angular і не так сильно обмежує програмістів. Ще одна перевага Vue перед Angular і React полягає в тому, що для роботи з цим фреймворком не доведеться вчити нову мову.

2.4.4. Ember.js

Ember схожий на Backbone і Angular. Це, крім того, один з найстаріших JavaScript-фреймворків. Але, незважаючи на це, він, в плані можливостей, не відстає від своїх молодших конкурентів. Наприклад, він підтримує технологію відслідковуються змін властивостей, яка спрощує спостереження за змінами стану програми і полегшує візуалізацію цих змін.

Рис. 2.7. Логотип Ember

Ember володіє досить-таки хитромудрої архітектурою, яка дозволяє швидко створювати величезні клієнтські програми. У ньому реалізовані типові MVC-ідеї. Ember-додатки будуються з адаптерів, компонентів, контролерів, допоміжних об'єктів, моделей, маршрутів, сервісів, шаблонів, утиліт, доповнень.

Одна з найбільш цікавих можливостей Ember - інструменти командного рядка (Ember CLI). Ці інструменти допомагають фронтенд-розробникам продуктивно працювати. За допомогою Ember CLI можна створювати не тільки шаблони проектів, але і заготовки контролерів, компонентів і інших сутностей, з яких будуються програми.

У розпорядженні того, хто займається фронтенд-розробкою з використанням Ember, виявляються такі основні можливості:

- Ember CLI - інструмент командного рядка для швидкого прототипування застосувань і для управління залежностями.
- Стандартний сервер розробника Ember.
- Ember Data - бібліотека для роботи з даними.
- Handlebars - движок шаблонів, який використовується в Ember-додатках.
- QUnit - фреймворк для тестування Ember-проектів.
- Ember Inspector - інструменти розробника для Chrome і Firefox.
- Ember Observer - каталог доповнень для Ember CLI.

Мабуть, Ember можна назвати недооціненим фреймворком, але незважаючи на це він відмінно підходить для створення складних веб-проектів.

2.4.5. Backbone.js

Backbone - це JavaScript-фреймворк, заснований на архітектурі, подібної MVC. Скажімо, те, що в MVC називається «контролером» (Controller) в Backbone називається «поданням» (View). Уявлення Backbone можуть використовувати різні

системи шаблонізації. Наприклад - Mustache, Handlebars, jQuery-tmpl. У Backbone-проектах можна використовувати і сторонні бібліотеки. Треба відзначити, що єдиною жорсткої залежністю Backbone є бібліотека Underscore.js.

Рис 2.8 Логотип Backbone

Backbone - це легкий у використанні фреймворк, який дозволяє швидко розробляти односторінкові додатки. Серед допоміжних засобів, які використовуються спільно з Backbone.js, можна відзначити такі, як Chaplin, Marionette, Thorax.

Якщо вам потрібно розробити додаток, з яким будуть працювати користувачі, що належать до різних груп, то для поділу моделей можна скористатися колекціями (масивами) Backbone. У моделях, колекціях, маршрутах і уявленнях Backbone можна користуватися подіями.

Серед особливостей екосистеми Backbone можна відзначити наступні:

- Бібліотека Backbone включає в себе події, моделі, колекції, уявлення і маршрутизатор.
- Бібліотека Underscore.js, від якої залежить Backbone, містить набір допоміжних функцій, що допомагають писати крос-браузерні JS-код.
- При розробці Backbone-додатків можна використовувати різні системи шаблонізації.
- Засіб командного рядка Backbone CLI спрощує розробку додатків.
- Бібліотеки Marionette, Thorax і Chaplin допомагають створювати додатки, що відрізняються особливими архітектурними рішеннями.

Мабуть, Backbone не можна назвати серйозним конкурентом для інших розглянутих в цьому матеріалі інструментів фронтенд-розробки. Однак цей фреймворк користується певною популярністю серед розробників. Тому цілком може виявитися так, що Backbone - це саме те, що вам потрібно.

2.5 Розпізнавання мови: принципи та застосування

Розділення дикторів(Speaker diarization) - це процес розділення вхідного звукового потоку на однорідні сегменти відповідно до ідентичності мовця. Він відповідає на запитання "хто говорив коли" в режимі багатомовного спілкування. Він має широкий спектр програм, включаючи пошук мультимедійної інформації, аналіз поворотів динаміків та обробку звуку. Зокрема, межі спікерів, створені системами діаризації, можуть значно покращити акустику.

Типова система діаризації спікерів зазвичай складається з чотирьох компонентів:

- сегментація мови, де вхідний звук сегментується на короткі секції, які, як передбачається, мають один динамік,а немовні розділи відфільтровуються;

- вбудовування аудіо вилучення, де із сегментованих розділів витягуються такі особливості, як MFCC(меп-кепстральні коефіцієнти) [13], фактори динаміків [14] або і-вектори [15, 16, 17];
- Кластеризація, де визначається кількість доповідачів,і витягнуті аудіовклади кластеризовані в ці колонки; і, за бажанням,
- ресегментація [18], де результати кластеризації додатково уточнюються для отримання остаточних результатів діаризації

В останні роки вбудовані аудіо вбудовання на основі нейронних мереж (d-vectors) широко застосовуються в додатках для перевірки динаміків [19, 20, 21, 22, 23], часто значно перевершуючи раніше найсучасніші техніки, засновані на і-векторах. Однак більшість із цих програм належать до текстової залежності перевірки динаміків, де вбудовані колонки витягуються із конкретних виявлених. На відміну від цього, для діаризування мовця потрібні незалежні від тексту вбудовування, які працюють на довільну мову.

У цій роботі досліджується незалежний від тексту підхід до діаризації динаміків на основі d-вектора. Ми використовуємо роботу [23] для навчання тоді модель перевірки ораторів, що не залежить від тексту, на основі LSTM поєднати цю модель з нещодавньою роботою в непараметричній спектральній алгоритм кластеризації для отримання найсучаснішого режиму діаризації динаміків система. У той час як кілька авторів досліджували за допомогою нейронної мережі вкладань для завдань діаризації, їх робота в основному зосереджена на використання DNN з прямим зворотним зв'язком для безпосереднього проведення діаризації. Наприклад, [26] використовує вбудовування DNN, навчені збитків, натхненних PLDA. В навпаки, у нашій роботі використовуються RNN (зокрема LSTM [27]), які краще фіксують послідовний характер звукових сигналів, а наші узагальнені наскрізна архітектура навчання безпосередньо імітує реєстрацію та перевірку логіка виконання.

Було кілька спроб застосувати спектральну кластеризацію [28] до проблеми діаризації спікерів [29, 15]. Однак до знання авторів, наша робота перша, що поєднує LSTM d-векторні вкладення зі спектральною кластеризацією. Крім того, як частина

нашого алгоритму спектральної кластеризації ми представляємо нову послідовність етапів уточнення матриці спорідненості, які діють на зменшення шуму спорідненості матриці, і мають вирішальне значення для успіху нашої системи. Буде описано:

1 Як може бути адаптована модель незалежної від тексту перевірки ораторів, заснована на LSTM, навчена в рамках [23] представити необроблені аудіодані та підготувати їх до кластеризації.
2 Чотири різні алгоритми кластеризації та обговорити плюси та мінуси кожного в контексті діаризації оратора, що завершується с модифікований алгоритм спектральної кластеризації.
3 Результати експерименту та висновки

2.5.1 Диарізація за допомогою d-векторів

Компанія Wan нещодавно представikf LSTM [27] вбудовану мережу дикторів як для залежних від тексту, так і для незалежних від тексту дикторів верифікація [23]. Їх модель тренується на сегментах фіксованої довжини витягнуте з великого корпусу довільної мови. Вони це показали за допомогою вбудовуваних d-векторів, створені такими мережами, зазвичай значно перевершують і-вектори в двоступеневій програмі реєстрації та перевірки. Тепер ми описуємо, як цю модель можна модифікувати для цілей діаризації спікерів. Блок-схема нашої системи діаризації наведена на рис. 2.5.1. В цій системі аудіосигнали спочатку трансформуються у кадри ширини 25 мс і крок 10 мс, і енергія вимірювання лог-мель-фільтра 40 витягуються з кожного кадру як мережевий вхід. Ми будуємо розсувні вікна фіксованої довжини на цих рамах та запускають LSTM мережі на кожному вікні. Тоді виводиться останній кадр LSTM використовується як d-векторне представлення цього розсувного вікна.

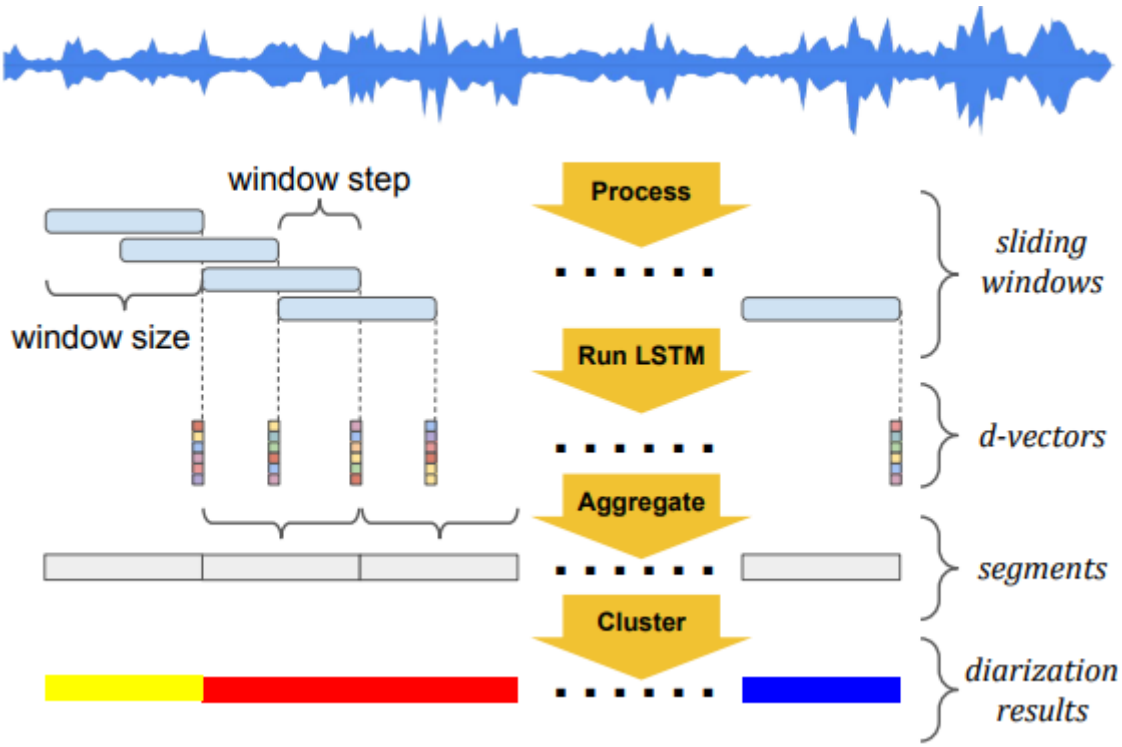


Рис. 2.5. Блок-схема системи діаризації на основі d-вектора.

На рисунку 2.5.1 видно, як використовується детектор голосової активності (VAD) для визначення мовлення сегменти аудіо, які далі поділяються на менші сегменти, що не перекриваються, використовуючи максимальну довжину сегмента (наприклад, 400 мс у наших експериментах), що визначає часову роздільну здатність результатів діаризації. Для кожного сегмента – відповідний d-вектори спочатку нормуються L2, а потім усереднюються, щоб утворити вкладання сегмента. Вищевказаний процес служить для зменшення довільної довжини аудіовходу у послідовність вкладених фіксованих довжин. Тепер можно застосувати а алгоритм кластеризації для цих вбудовувань для того, щоб визначити кількість унікальних колонок та призначити кожну частину аудіосистеми конкретному оратору.

2.5.2. Кластеризація

Представио чотири алгоритми кластеризації, які використовує наша система диарізації. Варто звернути особливу увагу на алгоритм спектральної офлайн-кластеризації, який значно перевершував альтернативні підходи в експериментах. Зауважимо, що алгоритми кластеризації можна розділити на дві категорії відповідно до часу очікування:

- Онлайн кластеризація: негайно видається мітка спікера. Як тільки сегмент стане доступним, не бачачи майбутніх сегментів.
- Офлайн кластеризація: Мітки спікерів створюються після того, як доступні вбудовування всіх сегментів.

Офлайн-алгоритми кластеризації зазвичай перевершують онлайн-кластеризацію алгоритмів завдяки додатковій контекстній інформації, доступній у налаштування офлайн. Крім того, остаточний етап резегментації може бути лише застосовуватися в режимі офлайн. Тим не менш, вибір між онлайн та офлайн залежить насамперед від характеру програми - де система призначена для розгортання. Наприклад, чутливі до затримки програми, такі як аналіз відео в реальному часі.

Наївна онлайн-кластеризація - це прототип алгоритму онлайн-кластеризації. Ми застосовуємо поріг щодо подібності між вбудованими сегментами. Для узгодження з узагальненою наскрізною навчальною архітектурою [23], косинусова подібність використовується як наша метрика подібності. У цьому алгоритмі кластеризації кожен кластер представлений як центроїд усіх відповідних вкладених елементів. Коли новий сегмент вбудовування стає доступним, він обчислюється як схожість із центроїдами в усі існуючі кластери. Якщо всі вони менше порога, тоді створити новий кластер, що містить лише це вбудовування; в іншому випадку додати це вбудовування до найбільш подібного кластеру та оновлення центроїда. Онлайн кластеризація посилань - це метод онлайн-кластеризації, розроблено для вдосконалення наївного підходу. Він оцінює кластерні розподіли ймовірностей та

моделює їх підструктуру на основі отриманих векторів вкладання так далеко. Технічні деталі описані в окремій роботі [30].

Як і в багатьох системах діаризації [31, 15, 32], було інтегровано алгоритм кластеризації K-Means у нашу систему. Зокрема, використано K-Means ++ для ініціалізації [33]. Щоб визначити кількість мовців ek, використано “ elbow(лікоть)” похідних умовного Середнього Відстань косинусів у квадраті(MSCD) між кожним вбудовуванням у свій кластерний центроїд:

$$\tilde{k} = \arg\max MSCD`(k), k \geq 1 \qquad (1)$$

Алгоритм спектральної кластеризації складається з наступних етапів:

1. Побудувати матрицю спорідненості A, де A_{ij} - подібність косинусів між вбудовуванням i-го та j-го сегментів, коли $i \neq$ j, а діагональні елементи встановлюються на максимальне значення в кожному рядку: $A_{ij} = max_{i \neq j} A_{ij}$
2. Застосувати наступну послідовність операцій доробки на матриці спорідненості A:
    - Розмиття Гауса із стандартним відхиленням σ;
    - Порогове обмеження по ряду: для кожного рядка встановіть елементи менше, ніж p-процентиль цього рядка, до 0;
    - Симетризація: рядку: $Y_{ij} = \max(X_{ij}, X_{ji})$
    - Дифузія: $Y = XX^T$
    - Максимальна нормалізація по ряду: $Y_{ij} = X_{ij}/max_k X_{ik}$

Ці уточнення впливають як на згладжування, так і на зменшення шуму даних у просторі подібності, як показано на рис. 2.5.2, і є вирішальними до успіху алгоритму. Уточнення базуються про часову локальність мовних даних - суміжне мовлення сегменти повинні мати подібні вбудовування, а отже і подібні значення в матриці спорідненості. Тепер ми пропонуємо інтуїцію кожної з цих операцій: розмиття Гауса діє, щоб згладити дані та зменшити ефект викидів. Порогове порогове значення служить для нульове споріднення між вкладеннями, що належать

двом різним ораторам. Симетризація відновлює симетричність матриці що є вирішальним для алгоритму спектральної кластеризації. Етапи дифузії черпають натхнення з алгоритму Diffusion Maps [34] і служать для загострення зображення, що дає чіткий результат межі між ділянками належності матриці спорідненості для різних динаміків. Нарешті, максимальна нормалізація по рядках служить для масштабування спектра матриці, щоб уникнути небажаних ефектів масштабу під час подальшої спектральної крок кластеризації.

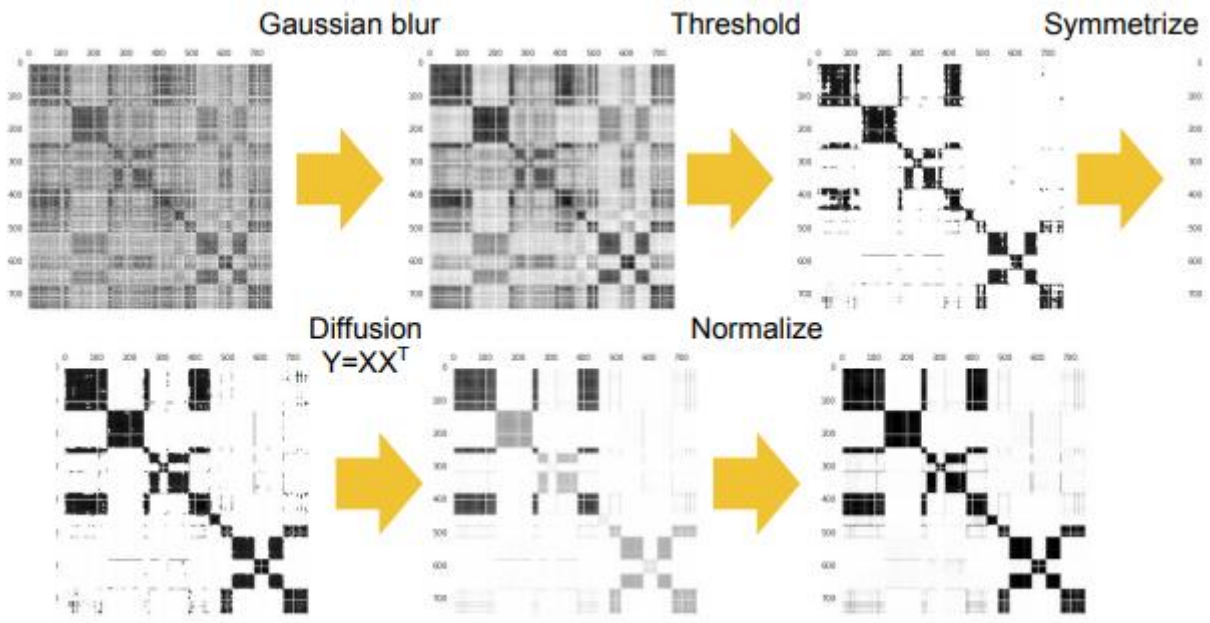


Рис 2.6. Операції уточнення матриці спорідненості

3. Після того, як були застосовані всі операції з доробки, виконайте власне розкладання на вишуканій матриці спорідненості. Нехай n власні значення: λ1> λ2> · · ·> λn. Ми використовуємо максимальне власний розрив для визначення кількості кластерів $\tilde{k}$:

$$\tilde{k} = \arg\max \frac{Y_k}{Y_{k+1}}, 1 \leq k \leq n \qquad (2)$$

4. Нехай власні вектори, що відповідають найбільшим власним значенням ek, будуть v_1, v_2, · · ·, $v_{\tilde{k}}$. Змінемо вкладання і-го сегмента на відповідну

розмірність у цих власних векторах: $e_i = [v_{1i}, v_{2i}, \cdots, v_{\widetilde{ki}}.]$. Використаємо ті самі K-засоби алгоритму далі.

Аналіз мовних даних є надзвичайно складною проблемою,і звичайні алгоритми кластеризації, такі як K-Means, часто працюють погано. Це пов’язано з низкою невдалих властивостей, властивих мовним даним, до яких належать:

- Негауссові розподіли: Мовні дані часто не є гауссовими. У цьому параметрі центроїд скупчення (центральний до Кластеризація K-Means) не є достатнім представленням.
- Кластерний дисбаланс: у мовних даних це часто буває так, коли один спікер буде говорити часто, тоді як інші говорять рідко. У цьому налаштуванні K-Means може неправильно розділити великі скупчення в кілька менших скупчень.
- Ієрархічна структура: Спікери поділяються на різні групи відповідно до статі, віку, акценту тощо. Ця структура є проблематичною, оскільки різниця між чоловіком та жінкою динамік набагато більший, ніж різниця між двома акустичними системами. Це ускладнює K-Means розрізнення кластерів, що відповідають групам, та кластерів, що відповідають окремим динамікам. На практиці це часто змушує K-Means неправильно кластеризувати всі вставки, що відповідають динамікам чоловічої статі, в один кластер, а всі вкладиші, що відповідають жіночим динамікам, в інший.

Проблеми, викликані цими властивостями, не обмежуються кластеризацією KMeans, але є ендемічними для більшості параметричних алгоритмів кластеризації. На щастя, ці проблеми можна пом'якшити, використовуючи непараметричний алгоритм кластеризації на основі з'єднань, такий як спектральна кластеризація.

Висновки до розділу

Для розробки новітнього продукту для запису інтерв’ю було проаналізовані мови програмування, їх переваги та недоліка. Була вибрана найоптимальніша мова, а саме – мова програмування Python. Вона є доволі простою у порівнянні з іншими мовами. Це можно інтерпретувати як і недолік, і як перевагу. Але важливо те, що на неї є попит і знайти додаткових розробників у команду буде не тяжко і затрачено меньше коштів. Також, як вагому перевагу, можно виділити високу швидкіть розробки проекту. Для мови программування у розробки фронтенду був обраний JaveScript, бо він є майже повноцінним лідером у фронтенд розробці. У якості фреймворків вибрані Django і VueJS. Швидка розробка на даних фреймфорках і висока взаємодія цих фреймфорків зіграла вирішаючу роль у виборі. Для аналізу алгоритма для обробки співбесіди проведено експерименти на чотирьох алгоритмах кластеризації в поєднанні з обома i-векторами та d-векторами. Зокрема, поєднано d-векторний звук на основі LSTM вбудовування з нещодавньою роботою в непараметричну кластеризацію для отриманняультрасучасна система діаризації спікерів. Така конфігурація цілком задовільняє потреби нашої ідеї і дозволяє приступати до розробки проекту.

3. МОДЕЛЮВАННЯ СЕРВІСУ ДЛЯ ОБРОБКИ ТЕХНІЧНОГО ІНТЕРВ'Ю КАНДИДАТА

3.1 Порівняння і-вектору і d-векторних алгоритмів на експериментальних даних

Проведемо експерименти з усіма комбінаціями як i-vector, так і двекторних моделей, з чотирма алгоритмами кластеризації, обговореними раніше. Обидві моделі навчаються на анонімній колекції голосу пошук, який має близько 36 мільйонів висловлювань та 18 тисяч спікерів. І-векторна модель тренується з використанням 13 коефіцієнтів PLP з дельта-дельта-коефіцієнти. GMM-UBM включає 512 Гауса, а загальна матриця мінливості включає 100 власних векторів. Кінцеві і-вектори зводиться до 50-мірного використання LDA. Модель d-вектора - це 3-шарова мережа LSTM із кінцевим лінійним шаром. Кожен рівень LSTM має 768 вузлів з проекцією [35] 256 вузлів. Наша модель виявлення голосової активності (VAD) дуже мала Модель GMM використовує ті самі функції PLP, що і і-вектор. Це лише має два повні коваріаційні гауссі: один для мови і один для немовлення. Ми виявили, що цей простий VAD узагальнює краще в усьому домени (від запитів до телефону) для діаризації, ніж CLDNN [24] Моделі VAD.

Повідомляємо про частоту помилок діаризації (DER) для трьох стандартних загальнодоступних набори даних: домашніх телефоних розмов американсько-англійською мовою. Перші два набори даних є лише англійською мовою та порівняно меншими. Таким чином, ми використовуємо ці два набори даних для порівняння різних алгоритмів.

Третій набір даних використовується в більшості статей про діаризацію. Це містить 500 висловлювань, розподілених між шістьма мовами: українсько., англійською, німецька, японська, мандаринська та іспанська.

Система оцінки діаризації базується на pyannote.metrics бібліотеці [38]. Набір даних американсько-англійською мовою має 20-розділ висловлювань. Для кожної системи діаризації ми налаштовуємо такі параметри, як:

- Порогове значення детектора голосової активності (VAD)
- розмір вікна / крок вікна LSTM(Рис. 1)
- параметри кластеризації

Для кожного аудіофайлу об'єднується кілька каналів в один канал [15, 18, 32], і деталі не обробляються, які знаходяться перед першою анотацією або після останньої. Крім того, як це прийнято в літературі, виключаємо накладену мову (кілька спікерів говорять одночасно) з нашого оцінювання. Для автономних алгоритмів кластеризації ми обмежуємо систему виробляти принаймні 2 спікери.

Таблиця 3.1: DER(%) для англійської мови для різних алгоритмів та кластеризації:

Тип вектору	Кластеризація	Розгубленість	FA	Промах	Сума
i-vector	Naïve	26.41	2.40	3.55	32.36
	Links	25.40			31.36
	K-means	22.86			28.81
	Spectral	14.59			20.54
d-vector	Naïve	12.41	1.94	4.51	18.87
	Links	11.02			17.47
	K-means	7.29			13.75
	Spectral	6.03			12.48

У експериментальні результатах наведені в таблицях 1 показано загальний DER разом з трьома його компонентами: помилковий сигнал тривоги (FA), промах(Miss) і розгубленість. FA та Miss - здебільшого голосова діяльність помилки виявлення, частково з агрегування на рівні кадру і-вектори або d-вектори на рівні вікна до сегментів. FA та Miss у відмінності між і-вектором та d-вектором обумовлені їх різницею розміри вікон / кроки та логіка агрегування. У таблиці 1 ми

бачимо, що системи діаризації на основі d-вектора значно перевершують і-векторні системи. Для d-векторних систем оптимальним розміром та кроком розсувного вікна є 240 мс та 120 мс, відповідно. Також спостерігається, як очікувалося, офлайн-діаризація виробляє значно кращі результати, ніж онлайн-діаризація. Зокрема, онлайн-діаризація набагато більше передбачає неправильну кількість спікерів частіше, ніж офлайн-діаризація. Ця проблема може потенційно пом'якшити додаванням етапу "вигорання" перед входом в онлайн-режимі. Хоча модель є LSTM і була навчена повністю англійськими даними, все ще може досягти найсучасніших показників на цій багатомовній набору даних. Продуктивність потенційно може бути додатково покращена за допомогою використання внутрішніх навчальних даних та додавання остаточного етапу ресегментації. Кількість доповідачів встановлена на 2 для цього оцінювання.

3.2 Програмування алгоритму розбору тексту

Як було вирішено, для програмування буде обрано мову програмування Python, бо вона має великі переваги задля розробки таких продуктів. Розглянемо, як саме буде працювати данна модель на живому прикладі. Запис кожної співбесіди, навіть якщо це буде відеоконференція, чи просто аудіо, можно розглянути як звичайний звуковий ряд(рис 3.1). Кожна людина має свій тембр голосу, і він виділяється для прикладу окремим кольором. Задача полягає у тому, щоб розділити спікерів на різні потоки(рис 3.2) , та винути початок та кінець їх мови.

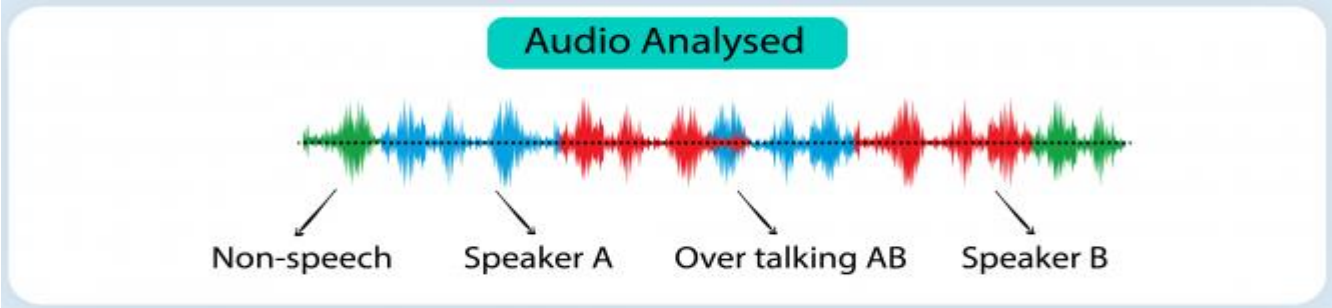


Рис 3.1. Аналіз аудіострічки

Потрібено знати зміни спікерів. Що для співбесіди означає зміна спікеру: це інтерв’юер задає питання, або кандидат починає давати відповідь на запитання. Сценарій для аналізування аудіо повертає данні у вигляді JSON-файлу, як показано на рис. 3.3. Задля біль точного розуміння побудована діаграма розмови спікерів(рис. 3.4) за допомогою бібліотеки matplotlib. Це є стандартна бібліотека Python, яка дозволяє будувати графіки задля аналізу даних.

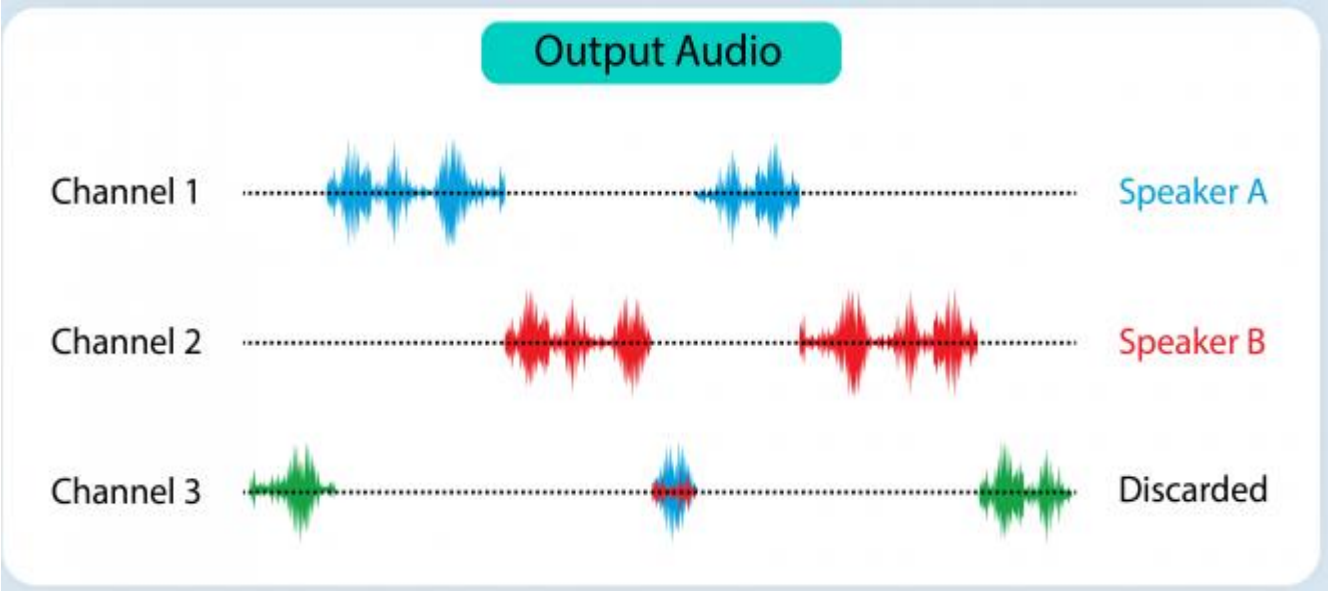


Рис. 3.2 Розділення аудіопотоку на спікерів.

```
{"jobName":"Transcribe","accountId":"780170720354","results":
    {"transcripts":[{"transcript":"Hey, guys, are we doing hey, legs or was it jim ? Jim was really exhausting today.
    "speaker_labels":{"speakers":2,"segments":[{"start_time":"0.83","speaker_label":"spk_1","end_time":"3.75","items":
    [{"start_time":"0.83","speaker_label":"spk_1","end_time":"1.0"},
    {"start_time":"1.0","speaker_label":"spk_1","end_time":"1.36"},
    {"start_time":"1.36","speaker_label":"spk_1","end_time":"1.44"},
    {"start_time":"1.44","speaker_label":"spk_1","end_time":"1.55"},
    {"start_time":"1.56","speaker_label":"spk_1","end_time":"2.05"},
    {"start_time":"2.44","speaker_label":"spk_1","end_time":"2.69"},
    {"start_time":"2.69","speaker_label":"spk_1","end_time":"3.01"},
    {"start_time":"3.02","speaker_label":"spk_1","end_time":"3.15"},
    {"start_time":"3.15","speaker_label":"spk_1","end_time":"3.32"},
    {"start_time":"3.32","speaker_label":"spk_1","end_time":"3.45"},
    {"start_time":"3.45","speaker_label":"spk_1","end_time":"3.75"}]},
    {"start_time":"4.44","speaker_label":"spk_0","end_time":"6.25","items":[
    {"start_time":"4.44","speaker_label":"spk_0","end_time":"4.65"},
    {"start_time":"4.65","speaker_label":"spk_0","end_time":"5.16"},
    {"start_time":"5.17","speaker_label":"spk_0","end_time":"5.35"},
    {"start_time":"5.35","speaker_label":"spk_0","end_time":"5.85"},
    {"start_time":"5.85","speaker_label":"spk_0","end_time":"6.25"}]},
    {"start_time":"6.89","speaker_label":"spk_0","end_time":"8.05","items":[
    {"start_time":"6.89","speaker_label":"spk_0","end_time":"7.07"},
    {"start_time":"7.07","speaker_label":"spk_0","end_time":"7.13"},
    {"start_time":"7.14","speaker_label":"spk_0","end_time":"7.7"},
    {"start_time":"7.7","speaker_label":"spk_0","end_time":"7.84"},
    {"start_time":"7.84","speaker_label":"spk_0","end_time":"8.05"}]},
    {"start_time":"8.84","speaker_label":"spk_1","end_time":"9.35","items":[
    {"start_time":"8.84","speaker_label":"spk_1","end_time":"9.35"}]},
    {"start_time":"10.34","speaker_label":"spk_1","end_time":"12.4","items":[
    {"start_time":"10.34","speaker_label":"spk_1","end_time":"10.56"},
    {"start_time":"10.56","speaker_label":"spk_1","end_time":"10.81"},
    {"start_time":"10.81","speaker_label":"spk_1","end_time":"10.94"},
    {"start_time":"10.94","speaker_label":"spk_1","end_time":"11.49"},
    {"start_time":"11.51","speaker_label":"spk_1","end_time":"11.73"},
```


Рис 3.3. Результат роботи моделі диаризації аудіо

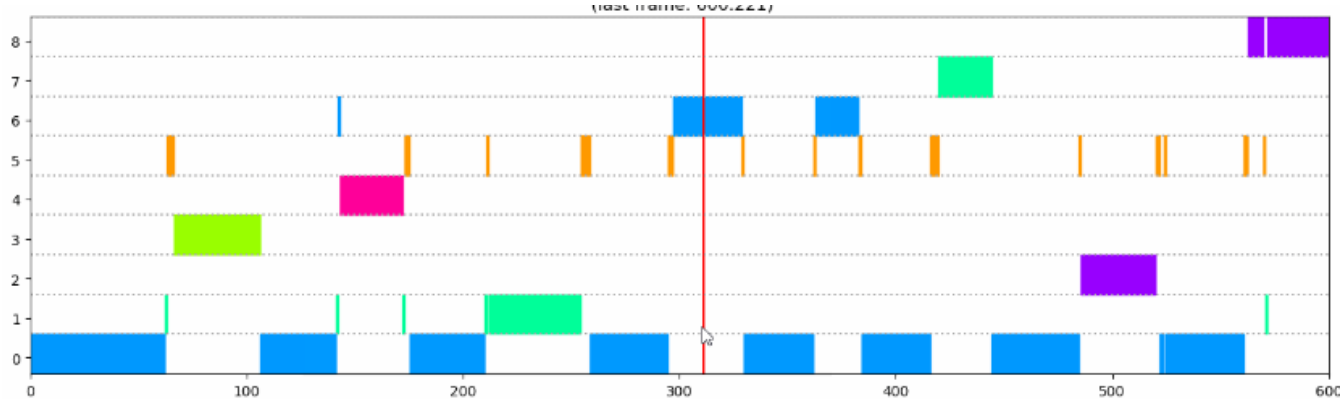


Рис. 3.4 Діаграма розмови спікерів.

Повний код програму буде наведений у Додатку А. Для побудови моделі були задійснені такі бібліотеки як Numpy і Pytorch. Numpy — розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. PyTorch — відкрита бібліотека машинного навчання на основі бібліотеки Torch, що використовують для таких застосувань, як комп'ютерне бачення та обробка природної мови.

3.3. Реалізація моделі у веб-додатку

На рис. 3.5 зображений кінцевий продукт. Це веб-сторінка кандидата з записами інтерв’ю з таймкодами. Назви таймкодів проставляються вручну, але в перспективі цілком можливо замінити це на роботи ще одної нейронної мережі. Записи відео зберігатимуться на Amazon Web Services. AWS - є дочірньою компанією Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки. Технологія дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп'ютерів, який завжди доступний через Інтернет.

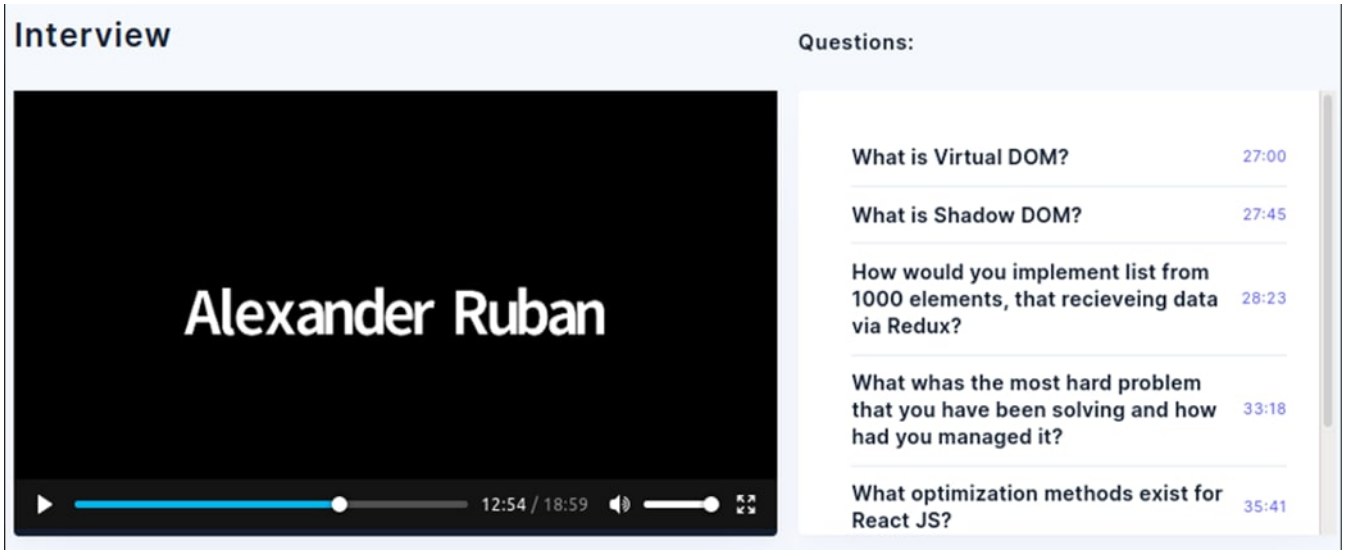


Рис 3.5. Сторінка кандидата з записом технічного інтерв’ю.

Програмна реалізація проекту на основі Django фреймворку використовую Django-моделі – це ORM, технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування. Модель співбесіди виглядає наступним чином:

class Interview(models.Model):

- added_by = models.ForeignKey(CustomUser, null=False, blank=False,related_name='added_interviews', on_delete=models.PROTECT)

- jobseeker = models.ForeignKey(Jobseeker, related_name='interviews', on_delete=models.CASCADE)
- status = models.CharField(choices=INTERVIEW_STATUS_CHOICES, max_length=255, default='waiting_for_decision')
- processing_by = models.ForeignKey(CustomUser, null=True, blank=True, on_delete=models.PROTECT, related_name='processing_interviews')
- active = models.BooleanField(default=False)
- activated_date = models.DateTimeField(null=True, blank=True)

Модель відповідей на запитання:

class AbstractInterviewQuestion(models.Model):

class Meta:

abstract = True

- text = models.CharField(max_length=1000, null=False, blank=False)
- answer_start_time = models.IntegerField(null=True, blank=True)
- answered = models.BooleanField(default=False)
- s3_url = models.FileField(null=True, blank=True)
- third_party_url = models.URLField(null=True, blank=True)

class InterviewMeeting(models.Model):

- interview = models.OneToOneField(Interview, related_name='meeting', null=False, blank=True, on_delete=models.PROTECT)
- start_time = models.DateTimeField(null=True, blank=True)
- communication_method = models.CharField(choices=COMMUNICATION_METHOD_CHOICES, blank=True, max_length=1024)
- communication_url = models.URLField(null=True, blank=True)

- interviewers = models.ManyToManyField(Interviewer, related_name='meetings', blank=True)
- vacancy = models.ForeignKey(Vacancy, related_name='meetings', on_delete=models.PROTECT, null=True, blank=True)
- zoom_meeting_id = models.CharField(max_length=255, blank=True, null=True)

3.4 Прогонозаний результат роботи алгоритму

Метод запису і аналізування технічного інтерв’ю пропонує записувати та зберігати технічні інтерв'ю кандидатів для подальшого їх використання. Це допоможе зберегти час на найм як мінімум тим, що технічний керівник зможе відразу продивитися інтерв'ю кандидата та прийняти рішення: чи треба запрошувати його на більш де-тальну співбесіду. Так як для більш швидкого розбору запису був використаний алгоритм диаризації на основі d-векторів, то це буде виконуватися дуже швидко. Яким чином це буде відбуватися: коли технічний керівник і кандидат ведуть беседу, то вона буде записуватись. Після цього штучний інтелект буде програмно розподіляти звук на цьому записі на два коляри: інтерв’ювера та кандидата. Це дасть звому швидше орієнтуватися у записі. І буде зрозуміло, що якщо помінявся свікер у записі, то це буде відповідть на запитання, чи саме запитання відповідно. Роглянемо, як виглядає процес рекрутингу IT-компаній зараз(рис 3.6) і скільки часу на це витрачається:

Таблиця 3.2 – Час, який потрібний на поточний процес найму

Назва етапу	опис	Час між поточним і попереднім єтапом
Старт комунікації	Початок розмови з ком-панією. Це може бути будь що: лист на пошті, повідомлення у соціаль-них мережах чи будь що інше. Час , коли здобувач почав розмову о вакансії чи коли почали розгляда-ти його як кандидата.	-
Інтерв'ю з рекрутером	На цьому етапі вирішу-ються організаційні мо-менти, можливо рекрутер буде задавати якісь осо-бисті питання, щоб зро-зуміти, чи підходить кан-дидат до команди, чи ні. Також буде розглянута вакансія, і надана інфор-мація про компанію та умови праці.	1-3 дня
Технічне інтерв'ю	Тут вам більш детально сам технічний керівник розповість про проект, технології, і власне про-веде з вами технічне ін-терв'ю.	1-7 дні
Остаточне рішення	Може бути два варіанти: перший, це коли кандида-ту видаюсь оффер, і він виходить на роботу. Дру-гий – це коли компанія відмовляє після усього процессу. Скоріше за все перший варіант буде тяг-нутися довше, бо компанії зазвичай не дають відпо-віді відразу і тримають кандидату поки не прове-дуть співбесіди з іншими кандидатами.	1-14дні

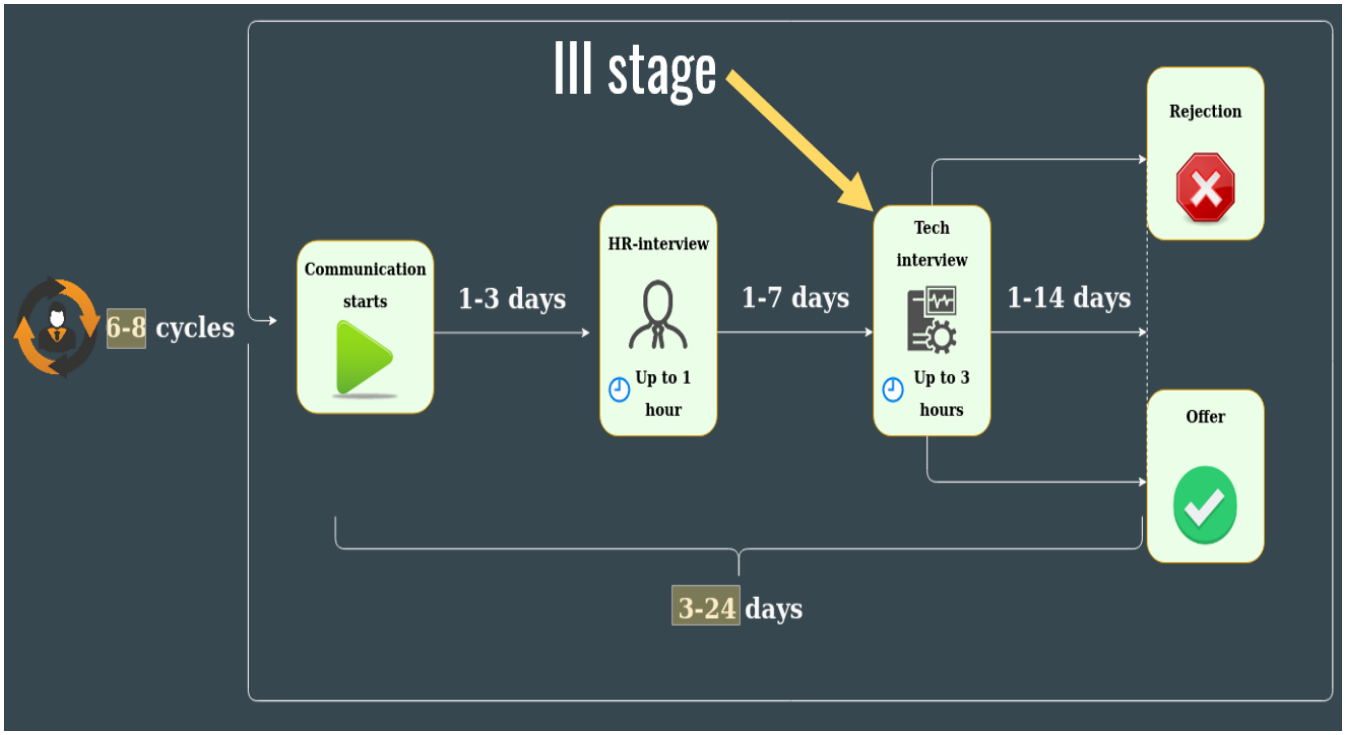


Рис. 3.6. Поточний процес найму

Помітно, що увесь процес достатньо затяжний – до 24х днів Наразі вже більшість компаній викори-стовують метод відео-інтерв'ю для зберігання часу кандидата. Теперь розглянемо, як виглядатиме процес найму(рис. 3.7) у новому випадку і як саме поміняється час на найм у випадку з використанням методом запису технічного інтерв’ю(табл 3.3).

Таблиця 3.3 - Робочий процес найму з переглядом інтерв’ю

Назва етапу	Час між поточним і попереднім етапами
Старт комунікації та пере-гляд технічного інтерв’ю	-
Миттєва відмова	1-3 дня
Інтерв’ю з рекрутером	1-3 дня
Остаточне рішення	1-14 дня

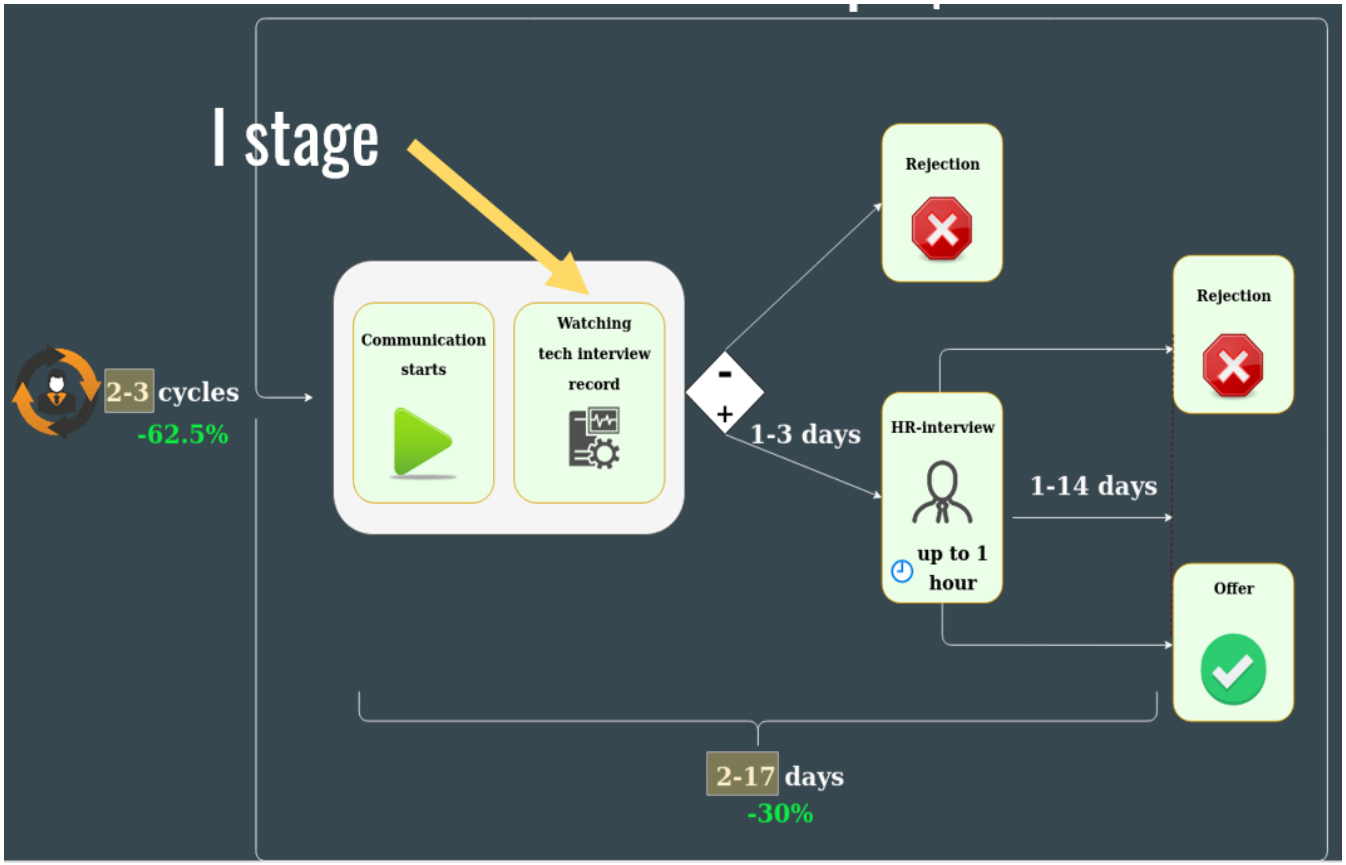


Рис. 3.7. Новий процес найму

Весь процес буде займає від 2х до 17 днів, що є на 30% відсотків менше, ніж поточний алгоритм з поточними методами.

3.5. Вплив рішення на ринок та рекрутинг

Процес найму є важливим для усіх направлень праці, не тільки для IT індустрії. Варто зауважити, що кожна компанія витрачає час і власні кошти на пошук нового співробітника. Щоб зрозуміти більш детально дану проблему, треба підрахувати кількість затрачених коштів з обоїх сторін. Компанія у середньому проводить 10-20 співбесід на закриття однієї позиції. Для цього задійснений: рекрурер, технічний керівник, і можливо не один. Варто зазначити, що одна співбесіда триває від 1 до 3 годин. Враховуючи тільки час висококваліфікованого технічного керівника, який коштує від 20$ за годину(в Україні), стає зрозуміло, що

це є достатньо затратним процесом. Це не враховуючи те, що під час співбесіди ні один з учасників не має змогу вибтрачати свій час та сили на розробку проекту, задля якого вони і були найняті у компанію.

Зарубіжне рекрутингове агенство провело опитування(рис. 3.8), і виявилося, що 56% компаній страдають від затяжного процесу рекрутинга.

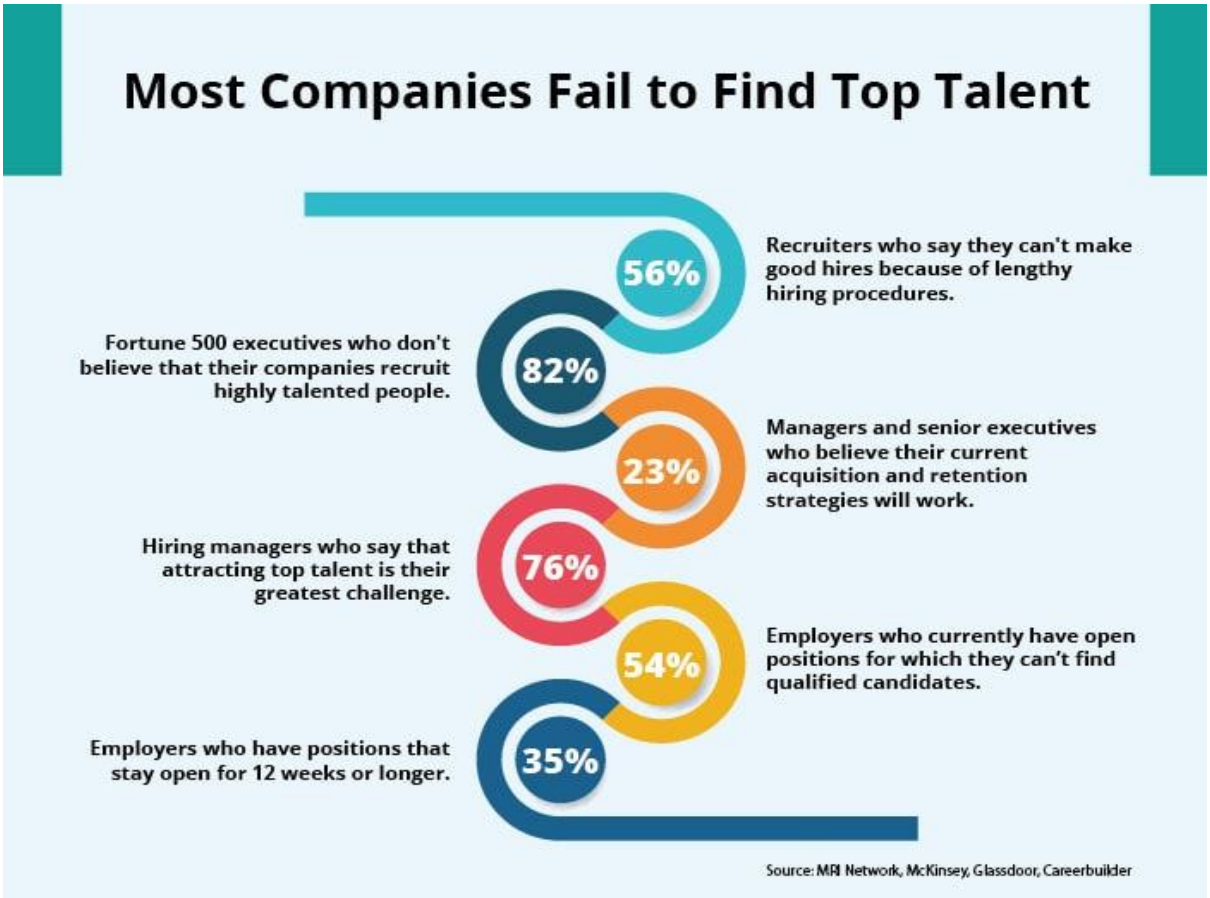


Рис. 3.8 – Найбільші проблеми компаній при пошуку кандидатів

Варто відмітити, що кандидат має свої переваги у використанні методу запису і обробки технічного інтерв’ю у процесі найму. Як для кандидата, який у більшості випадків ще є повноціним працініком, важливим є можливість не відволікатися на безліч техничних інтерв’ю, брати відпустки і відволікатися з роботи. Бо кожна співбесіда це є стрес. У свою чергу, на ринку праці це стане відправною точкою для більш точного пошуку кандидатів. Це дасть змогу більшості розробників проходити співбесіди не звільняючись з роботи.

Висновки до розділу

Розроблена модель для диаризації мовлення повністью вірповідає очікуванням. Задля її розробки була використана мова програмування Python з використанням таких бібліотек як Numpy, matplotlib(для побудування графіків), Pytorch. Для набування презентабильного виду і у подальшому використанні цієї розробки як комерційної, розроблений веб-додаток на основі фреймворку Django, куди безпосередньо інтегрована дана модель. На основі існуючих досліджень з найму і рекрутингу встановлено, що дана пропозиція є актуальною на теперішній час.

Висновок

Анонімний пошук роботи користується великим попитом на ринку праці в Україні і не тільки. Переваги анонімності були викладені у першій частині, але якщо виділити найголовніше – це приборкання дискримінації людей. Так як ми живемо у достатньо складний час, це є дуже важливим.

Методи найму і алгоритми, які задіяні на сервісах пошуку роботу є сталими вже кілька десятків років. Але був найдет варіант покращення цього процессу, а саме – метод запису технічного інтерв'ю. Він базується на основі використання диаризації. Диарізація - процес поділу вхідного аудіосигналу на однорідні сегменти відповідно до приналежністю аудиопотока того чи іншого спікера. Задля цього процесу було проаналізовано різні алгоритма, точніше використання і-вектору і d-вектору у розробці. На експерементальних даних було встановлено, що використання d-вектору дає приблизно на 40% більш точні розброру аудіофайлів.

Щоб інтегрувати дану модель у роботу, був використаний Django фреймфорк, який написаний на мові програмування Python. Задля використання фронтенд частики була обрана мова програмування JavaScript з фреймворком VueJS. Усі розробки по фронтенд частині були проведені по роботам професійних дизайнерів, які спеціально спроектували інтерфейс задля зручного користування(див. Додаток Б для перегляду дизайнерських рішень).

Даний проект має велику перевагу над іншими існуючими анонімними сервісами в Україні, і ці переваги відчуть на собі компанії і кандидати.

ПЕРЕЛІК ПОСИЛАНЬ

1. IT Industry Outlook 2016. Research Report / CompTIA. URL: https://www.comptia.org/ resources/it-industry-outlook-2016-final
2. Савицький Ю. Досвід Польщі для України: чому варто робити ставку на високі технології? URL: https://www.radiosvoboda.org/a/27631224.html
3. Савенко С. Попри всі бурі: чому IT-сектор претендує на роль флагмана української економіки. URL: https://news.finance.ua/ua/news/-/397906/popry-vsi-buri-chomu-it-sektor-pretenduyena-rol-flagmana-ukrayinskoyi-ekonomiky
4. Сич О. Український IT-ринок розвивається, однак відтік кадрів продовжується. URL: http://energolife.info/ua/2016/Business/2208/Український-IT-ринок-розвивається-однак-відтіккадрів-продовжується.МтІ
5. The Global Outsourcing 100, 2016. IAOP. URL: https://www.iaop.org/Content/19/165/4454
6. The State of European Tech. People Flows. Atomico. URL: https://2018.stateofeuropeantech. com/chapter/europes-got-talent/article/people-flows/
7. Терехов Д. С. Економічні проблеми розвитку IT-підприємств України. Ефективна економіка. 2017. № 2. URL: http://www.economy.nayka.com.ua/?op=1&z=6035
8. Публікації. Офіційний веб-сайт Державної служби статистики України. URL: http:// ukrstat.gov.ua/menu/publikac_.htm
9. Українська IT-індустрія: загальні дані та рівень спеціалістів. IT Ukraine Association. URL: https://itukraine.org.ua/infografika.-ukraïnska-it-industriya-zagalni-dani-ta-profesijnij-profilspeczialistiv.html
10. Бабанін О. С. Статистика розвитку IT-ринку в США, Україні та світі // Статистика України. 2013. № 1. С. 22–28. 11. Волошин С. Зарплаты

украинских разработчиков – июнь-июль 2018. URL: https://dou.ua/lenta/articles/salary-report-june-july-2018/

11. Medium Online Publishing Platform [Електронний ресурс] / Angular Introduction to Reactive Extensions (RxJS) – стаття. Режим доступу: https://medium.com/google-developer-experts/angular-introduction-to-reactive-extensions-rxjs-a86a7430a61f
12. The State of Obesity [Електронний ресурс] / The State of Childhood Obesity – стаття. Режим доступу: https://stateofobesity.org/childhood-obesity-trends/
13. Patrick Kenny, Douglas Reynolds, and Fabio Castaldo, “Di-arization of telephone conversations using factor analysis,”IEEE Journal of Selected Topics in Signal Processing, vol. 4,no. 6, pp. 1059–1070, 2010.
14. Fabio Castaldo, Daniele Colibro, Emanuele Dalmasso, PietroLaface, and Claudio Vair, “Stream-based speaker segmen-tation using speaker factors and eigenvoices,” inAcoustics,Speech and Signal Processing, 2008. ICASSP 2008. IEEE In-ternational Conference on. IEEE, 2008, pp. 4133–4136.
15. Stephen H Shum, Najim Dehak, R´eda Dehak, and James RGlass, “Unsupervised methods for speaker diarization: An in-tegrated and iterative approach,”IEEE Transactions on Audio,Speech, and Language Processing, vol. 21, no. 10, pp. 2015–2028, 2013.
16. Mohammed Senoussaoui, Patrick Kenny, Themos Stafylakis,and Pierre Dumouchel,“A study of the cosine distance-based mean shift for telephone speech diarization,”IEEE/ACMTransactions on Audio, Speech and Language Processing(TASLP), vol. 22, no. 1, pp. 217–227, 2014.
17. Gregory Sell and Daniel Garcia-Romero, “Speaker diariza-tion with plda i-vector scoring and unsupervised calibration,”inSpoken Language Technology Workshop (SLT), 2014 IEEE.IEEE, 2014, pp. 413–417.
18. Gregory Sell and Daniel Garcia-Romero,“Diarization re-segmentation in the factor analysis subspace,” inAcoustics,Speech and Signal Processing (ICASSP), 2015 IEEE Interna-tional Conference on. IEEE, 2015, pp. 4794–4798.

19. Ehsan Variani, Xin Lei, Erik McDermott, Ignacio LopezMoreno, and Javier Gonzalez-Dominguez, “Deep neural net-works for small footprint text-dependent speaker verification,”inAcoustics, Speech and Signal Processing (ICASSP), 2014IEEE International Conference on. IEEE, 2014, pp. 4052–4056.
20. Yu-hsin Chen, Ignacio Lopez-Moreno, Tara N Sainath, Mirk´oVisontai, Raziel Alvarez, and Carolina Parada,“Locally-connected and convolutional neural networks for small foot-print speaker recognition,” inSixteenth Annual Conference ofthe International Speech Communication Association, 2015.
21. Georg Heigold, Ignacio Moreno, Samy Bengio, and NoamShazeer,“End-to-end text-dependent speaker verification,”inAcoustics, Speech and Signal Processing (ICASSP), 2016IEEE International Conference on. IEEE, 2016, pp. 5115–5119.
22. F A Rezaur Rahman Chowdhury, Quan Wang, Li Wan,and Ignacio Lopez Moreno,“Attention-based modelsfor text-dependent speaker verification,”arXiv preprintarXiv:1710.10470, 2017.
23. Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno,“Generalized end-to-end loss for speaker verification,”arXivpreprint arXiv:1710.10467, 2017.
24. Guoguo Chen, Carolina Parada, and Georg Heigold, “Small-footprint keyword spotting using deep neural networks,” inAcoustics, Speech and Signal Processing (ICASSP), 2014IEEE International Conference on. IEEE, 2014, pp. 4087–4091.
25. Rohit Prabhavalkar, Raziel Alvarez, Carolina Parada, PreetumNakkiran, and Tara N Sainath, “Automatic gain control andmulti-style training for robust small-footprint keyword spottingwith deep neural networks,” inAcoustics, Speech and SignalProcessing (ICASSP), 2015 IEEE International Conference on.IEEE, 2015, pp. 4704–4708.
26. Daniel Garcia-Romero, David Snyder, Gregory Sell, DanielPovey, and Alan McCree, “Speaker diarization using deep neu-ral network embeddings,” in2017 IEEE International Confer-ence on Acoustics, Speech and Signal Processing (ICASSP).IEEE, 2017, pp. 4930–4934
27. Sepp Hochreiter and J¨urgen Schmidhuber, “Long short-termmemory,”Neural computation, vol. 9, no. 8, pp. 1735–1780,1997.

28. Ulrike Von Luxburg, “A tutorial on spectral clustering,”Statis-tics and computing, vol. 17, no. 4, pp. 395–416, 2007.
29. Huazhong Ning, Ming Liu, Hao Tang, and Thomas S Huang,“A spectral clustering approach to speaker diarization.,” inIN-TERSPEECH, 2006.
30. Philip Andrew Mansfield, Quan Wang, Carlton Downey,Li Wan, and Ignacio Lopez Moreno,“Links: A high-dimensional online clustering method,”arXiv preprintarXiv:1801.10123, 2018.
31. Oshry Ben-Harush, Ortal Ben-Harush, Itshak Lapidot, andHugo Guterman, “Initialization of iterative-based speaker di-arization systems for telephone conversations,”IEEE Transac-tions on Audio, Speech, and Language Processing, vol. 20, no.2, pp. 414–425, 2012.
32. Dimitrios Dimitriadis and Petr Fousek, “Developing on-linespeaker diarization system,” inINTERSPEECH, 2017.
33. David Arthur and Sergei Vassilvitskii, “k-means++: The ad-vantages of careful seeding,” inProceedings of the eighteenthannual ACM-SIAM symposium on Discrete algorithms. Soci-ety for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
34. Ronald R Coifman and St´ephane Lafon, “Diffusion maps,”Applied and computational harmonic analysis, vol. 21, no. 1,pp. 5–30, 2006.
35. Has¸im Sak, Andrew Senior, and Franc¸oise Beaufays, “Longshort-term memory recurrent neural network architectures forlarge scale acoustic modeling,” inFifteenth Annual Conferenceof the International Speech Communication Association, 2014.
36. Rub´en Zazo Candil, Tara N Sainath, Gabor Simko, and Car-olina Parada, “Feature learning with raw-waveform cldnns forvoice activity detection,” 2016.
37. A Canavan, D Graff, and G Zipperlen, “Callhome americanenglish speech ldc97s42,”LDC Catalog. Philadelphia: Lin-guistic Data Consortium, 1997.

38. Herv´e Bredin, “pyannote.metrics: a toolkit for repro-ducible evaluation, diagnostic, and error analysis of speakerdiarization systems,”hypothesis, vol. 100, no. 60, pp. 90, 2017.

ДОДАТОК А

Main.py

```
import numpy as np
import uisrnn
import librosa
import sys
sys.path.append('ghostvlad')
sys.path.append('visualization')
import toolkits
import model as spkModel
import os
from viewer import PlotDiar

# ==========================================
# Parse the argument
# ==========================================
import argparse
parser = argparse.ArgumentParser()
# set up training configuration.
parser.add_argument('--gpu', default='', type=str)
parser.add_argument('--resume', default=r'ghostvlad/pretrained/weights.h5', type=str)
parser.add_argument('--data_path', default='4persons', type=str)
# set up network configuration.
parser.add_argument('--net', default='resnet34s', choices=['resnet34s', 'resnet34l'], type=str)
```

```
parser.add_argument('--ghost_cluster', default=2, type=int)
parser.add_argument('--vlad_cluster', default=8, type=int)
parser.add_argument('--bottleneck_dim', default=512, type=int)
parser.add_argument('--aggregation_mode', default='gvlad', choices=['avg', 'vlad', 'gvlad'], type=str)
# set up learning rate, training loss and optimizer.
parser.add_argument('--loss', default='softmax', choices=['softmax', 'amsoftmax'], type=str)
parser.add_argument('--test_type', default='normal', choices=['normal', 'hard', 'extend'], type=str)

global args
args = parser.parse_args()

SAVED_MODEL_NAME = 'pretrained/saved_model.uisrnn_benchmark'

def append2dict(speakerSlice, spk_period):
key = list(spk_period.keys())[0]
value = list(spk_period.values())[0]
timeDict = {}
timeDict['start'] = int(value[0]+0.5)
timeDict['stop'] = int(value[1]+0.5)
if(key in speakerSlice):
speakerSlice[key].append(timeDict)
else:
speakerSlice[key] = [timeDict]

return speakerSlice

def arrangeResult(labels, time_spec_rate): # {'1': [{'start':10, 'stop':20}, {'start':30, 'stop':40}], '2':
[{'start':90, 'stop':100}]}
lastLabel = labels[0]
```

```
speakerSlice = {}
j = 0
for i,label in enumerate(labels):
if(label==lastLabel):
continue
speakerSlice = append2dict(speakerSlice, {lastLabel: (time_spec_rate*j,time_spec_rate*i)})
j = i
lastLabel = label
speakerSlice = append2dict(speakerSlice, {lastLabel:
(time_spec_rate*j,time_spec_rate*(len(labels)))})
return speakerSlice

def genMap(intervals): # interval slices to maptable
slicelen = [sliced[1]-sliced[0] for sliced in intervals.tolist()]
mapTable = {} # vad erased time to origin time, only split points
idx = 0
for i, sliced in enumerate(intervals.tolist()):
mapTable[idx] = sliced[0]
idx += slicelen[i]
mapTable[sum(slicelen)] = intervals[-1,-1]

keys = [k for k,_ in mapTable.items()]
keys.sort()
return mapTable, keys

def fmtTime(timeInMillisecond):
millisecond = timeInMillisecond%1000
minute = timeInMillisecond//1000//60
```

```
second = (timeInMillisecond-minute*60*1000)//1000

time = '{}:{:02d}.{}'.format(minute, second, millisecond)

return time

def load_wav(vid_path, sr):

wav, _ = librosa.load(vid_path, sr=sr)

intervals = librosa.effects.split(wav, top_db=20)

wav_output = []

for sliced in intervals:

wav_output.extend(wav[sliced[0]:sliced[1]])

return np.array(wav_output), (intervals/sr*1000).astype(int)

def lin_spectogram_from_wav(wav, hop_length, win_length, n_fft=1024):

linear = librosa.stft(wav, n_fft=n_fft, win_length=win_length, hop_length=hop_length) # linear spectrogram

return linear.T


# 0s 1s 2s 4s 6s

# |-------------------|-------------------|-------------------|

# |-------------------|

# |-------------------|

# |-------------------|

# |-------------------|

def load_data(path, win_length=400, sr=16000, hop_length=160, n_fft=512, embedding_per_second=0.5, overlap_rate=0.5):

wav, intervals = load_wav(path, sr=sr)

linear_spect = lin_spectogram_from_wav(wav, hop_length, win_length, n_fft)

mag, _ = librosa.magphase(linear_spect) # magnitude
```

```
mag_T = mag.T

freq, time = mag_T.shape

spec_mag = mag_T

spec_len = sr/hop_length/embedding_per_second

spec_hop_len = spec_len*(1-overlap_rate)

cur_slide = 0.0

utterances_spec = []

while(True): # slide window.

if(cur_slide + spec_len > time):

break

spec_mag = mag_T[:, int(cur_slide+0.5) : int(cur_slide+spec_len+0.5)]

# preprocessing, subtract mean, divided by time-wise var

mu = np.mean(spec_mag, 0, keepdims=True)

std = np.std(spec_mag, 0, keepdims=True)

spec_mag = (spec_mag - mu) / (std + 1e-5)

utterances_spec.append(spec_mag)

cur_slide += spec_hop_len

return utterances_spec, intervals

def main(wav_path, embedding_per_second=1.0, overlap_rate=0.5):

# gpu configuration

toolkits.initialize_GPU(args)

params = {'dim': (257, None, 1),

'nfft': 512,
```

```
'spec_len': 250,

'win_length': 400,

'hop_length': 160,

'n_classes': 5994,

'sampling_rate': 16000,

'normalize': True,

}

network_eval = spkModel.vggvox_resnet2d_icassp(input_dim=params['dim'],

num_class=params['n_classes'],

mode='eval', args=args)

network_eval.load_weights(args.resume, by_name=True)


model_args, _, inference_args = uisrnn.parse_arguments()

model_args.observation_dim = 512

uisrnnModel = uisrnn.UISRNN(model_args)

uisrnnModel.load(SAVED_MODEL_NAME)

specs, intervals = load_data(wav_path, embedding_per_second=embedding_per_second,
overlap_rate=overlap_rate)

mapTable, keys = genMap(intervals)

feats = []

for spec in specs:

spec = np.expand_dims(np.expand_dims(spec, 0), -1)

v = network_eval.predict(spec)

feats += [v]

feats = np.array(feats)[:,0,:].astype(float) # [splits, embedding dim]
```

```
predicted_label = uisrnnModel.predict(feats, inference_args)

time_spec_rate = 1000*(1.0/embedding_per_second)*(1.0-overlap_rate) # speaker embedding every ?ms

center_duration = int(1000*(1.0/embedding_per_second)//2)

speakerSlice = arrangeResult(predicted_label, time_spec_rate)

for spk,timeDicts in speakerSlice.items(): # time map to orgin wav(contains mute)

for tid,timeDict in enumerate(timeDicts):

s = 0

e = 0

for i,key in enumerate(keys):

if(s!=0 and e!=0):

break

if(s==0 and key>timeDict['start']):

offset = timeDict['start'] - keys[i-1]

s = mapTable[keys[i-1]] + offset

if(e==0 and key>timeDict['stop']):

offset = timeDict['stop'] - keys[i-1]

e = mapTable[keys[i-1]] + offset

speakerSlice[spk][tid]['start'] = s

speakerSlice[spk][tid]['stop'] = e

for spk,timeDicts in speakerSlice.items():

print('========= ' + str(spk) + ' =========')

for timeDict in timeDicts:

s = timeDict['start']
```

```
e = timeDict['stop']

s = fmtTime(s) # change point moves to the center of the slice

e = fmtTime(e)

print(s+' ==> '+e)

p = PlotDiar(map=speakerSlice, wav=wav_path, gui=True, size=(25, 6))

p.draw()

p.plot.show()

if __name__ == '__main__':

main(r'wavs/rmdmy.wav', embedding_per_second=1.2, overlap_rate=0.4)
```

model.py

```
from __future__ import print_function

from __future__ import absolute_import

import keras

import tensorflow as tf

import keras.backend as K

import backbone

weight_decay = 1e-4

class ModelMGPU(keras.Model):

def __init__(self, ser_model, gpus):

pmodel = keras.utils.multi_gpu_model(ser_model, gpus)

self.__dict__.update(pmodel.__dict__)

self._smodel = ser_model
```

```
def __getattribute__(self, attrname):
'''Override load and save methods to be used from the serial-model. The
serial-model holds references to the weights in the multi-gpu model.
'''
# return Model.__getattribute__(self, attrname)
if 'load' in attrname or 'save' in attrname:
return getattr(self._smodel, attrname)

return super(ModelMGPU, self).__getattribute__(attrname)


class VladPooling(keras.engine.Layer):
'''
This layer follows the NetVlad, GhostVlad
'''
def __init__(self, mode, k_centers, g_centers=0, **kwargs):
self.k_centers = k_centers
self.g_centers = g_centers
self.mode = mode
super(VladPooling, self).__init__(**kwargs)

def build(self, input_shape):
self.cluster = self.add_weight(shape=[self.k_centers+self.g_centers, input_shape[0][-1]],
name='centers',
initializer='orthogonal')
self.built = True

def compute_output_shape(self, input_shape):
assert input_shape
```

```
return (input_shape[0][0], self.k_centers*input_shape[0][-1])

def call(self, x):
# feat : bz x W x H x D, cluster_score: bz X W x H x clusters.
feat, cluster_score = x
num_features = feat.shape[-1]

# softmax normalization to get soft-assignment.
# A : bz x W x H x clusters
max_cluster_score = K.max(cluster_score, -1, keepdims=True)
exp_cluster_score = K.exp(cluster_score - max_cluster_score)
A = exp_cluster_score / K.sum(exp_cluster_score, axis=-1, keepdims = True)

# Now, need to compute the residual, self.cluster: clusters x D
A = K.expand_dims(A, -1) # A : bz x W x H x clusters x 1
feat_broadcast = K.expand_dims(feat, -2) # feat_broadcast : bz x W x H x 1 x D
feat_res = feat_broadcast - self.cluster # feat_res : bz x W x H x clusters x D
weighted_res = tf.multiply(A, feat_res) # weighted_res : bz x W x H x clusters x D
cluster_res = K.sum(weighted_res, [1, 2])

if self.mode == 'gvlad':
cluster_res = cluster_res[:, :self.k_centers, :]

cluster_l2 = K.l2_normalize(cluster_res, -1)
outputs = K.reshape(cluster_l2, [-1, int(self.k_centers) * int(num_features)])
return outputs

def amsoftmax_loss(y_true, y_pred, scale=30, margin=0.35):
y_pred = y_true * (y_pred - margin) + (1 - y_true) * y_pred
```

```
y_pred *= scale
return K.categorical_crossentropy(y_true, y_pred, from_logits=True)

def vggvox_resnet2d_icassp(input_dim=(257, 250, 1), num_class=8631, mode='train', args=None):
net=args.net
loss=args.loss
vlad_clusters=args.vlad_cluster
ghost_clusters=args.ghost_cluster
bottleneck_dim=args.bottleneck_dim
aggregation = args.aggregation_mode
mgpu = len(keras.backend.tensorflow_backend._get_available_gpus())

if net == 'resnet34s':
inputs, x = backbone.resnet_2D_v1(input_dim=input_dim, mode=mode)
else:
inputs, x = backbone.resnet_2D_v2(input_dim=input_dim, mode=mode)
# ================================================
# Fully Connected Block 1
# ================================================
x_fc = keras.layers.Conv2D(bottleneck_dim, (7, 1),
strides=(1, 1),
activation='relu',
kernel_initializer='orthogonal',
use_bias=True, trainable=True,
kernel_regularizer=keras.regularizers.l2(weight_decay),
bias_regularizer=keras.regularizers.l2(weight_decay),
```

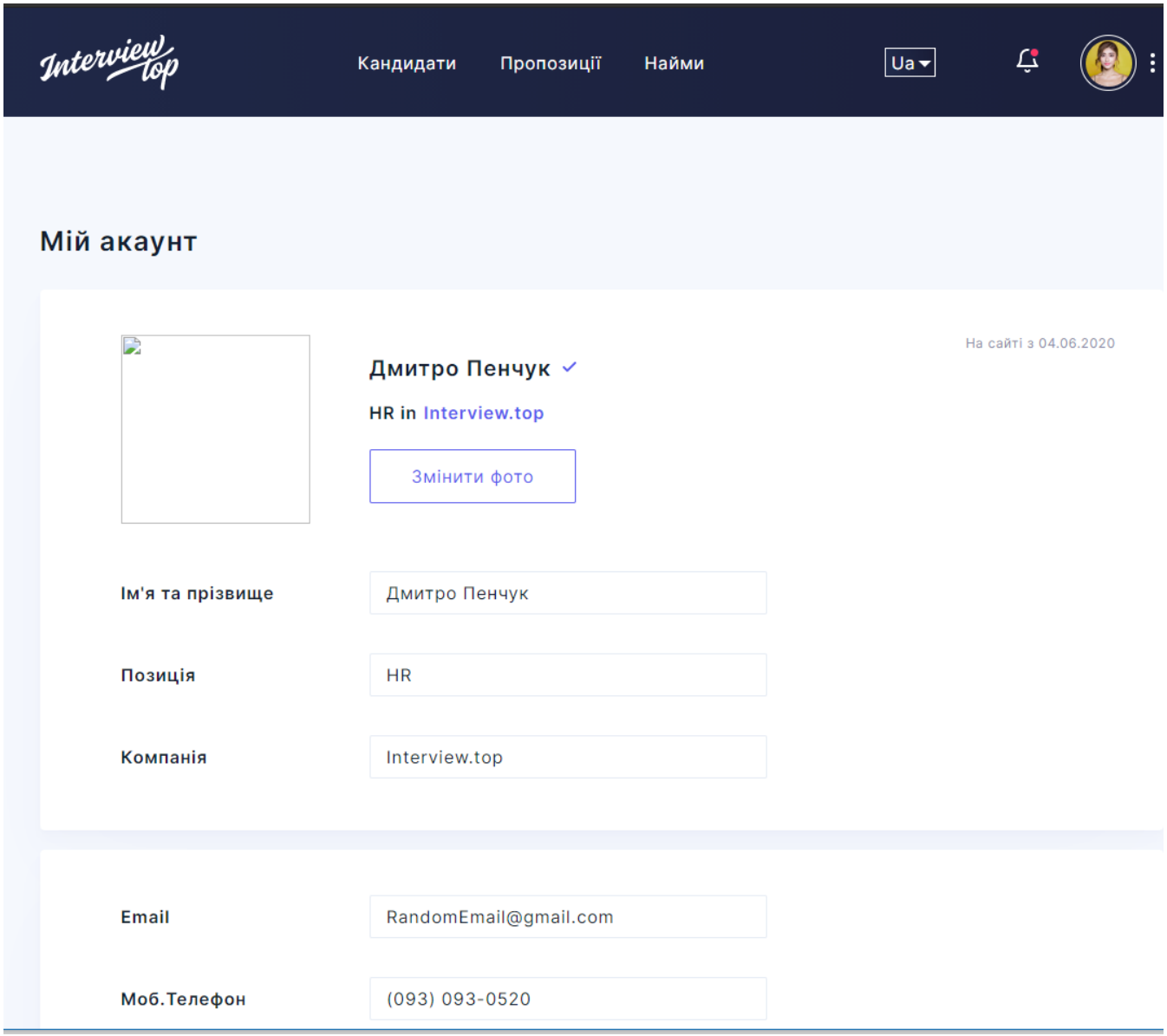
```
name='x_fc')(x)

# ==============================================

# Feature Aggregation

# ==============================================

if aggregation == 'avg':

if mode == 'train':

x = keras.layers.AveragePooling2D((1, 5), strides=(1, 1), name='avg_pool')(x)

x = keras.layers.Reshape((-1, bottleneck_dim))(x)

else:

x = keras.layers.GlobalAveragePooling2D(name='avg_pool')(x)

x = keras.layers.Reshape((1, bottleneck_dim))(x)

elif aggregation == 'vlad':

x_k_center = keras.layers.Conv2D(vlad_clusters, (7, 1),

strides=(1, 1),

kernel_initializer='orthogonal',

use_bias=True, trainable=True,

kernel_regularizer=keras.regularizers.l2(weight_decay),

bias_regularizer=keras.regularizers.l2(weight_decay),

name='vlad_center_assignment')(x)

x = VladPooling(k_centers=vlad_clusters, mode='vlad', name='vlad_pool')([x_fc, x_k_center])

elif aggregation == 'gvlad':

x_k_center = keras.layers.Conv2D(vlad_clusters+ghost_clusters, (7, 1),

strides=(1, 1),

kernel_initializer='orthogonal',
```

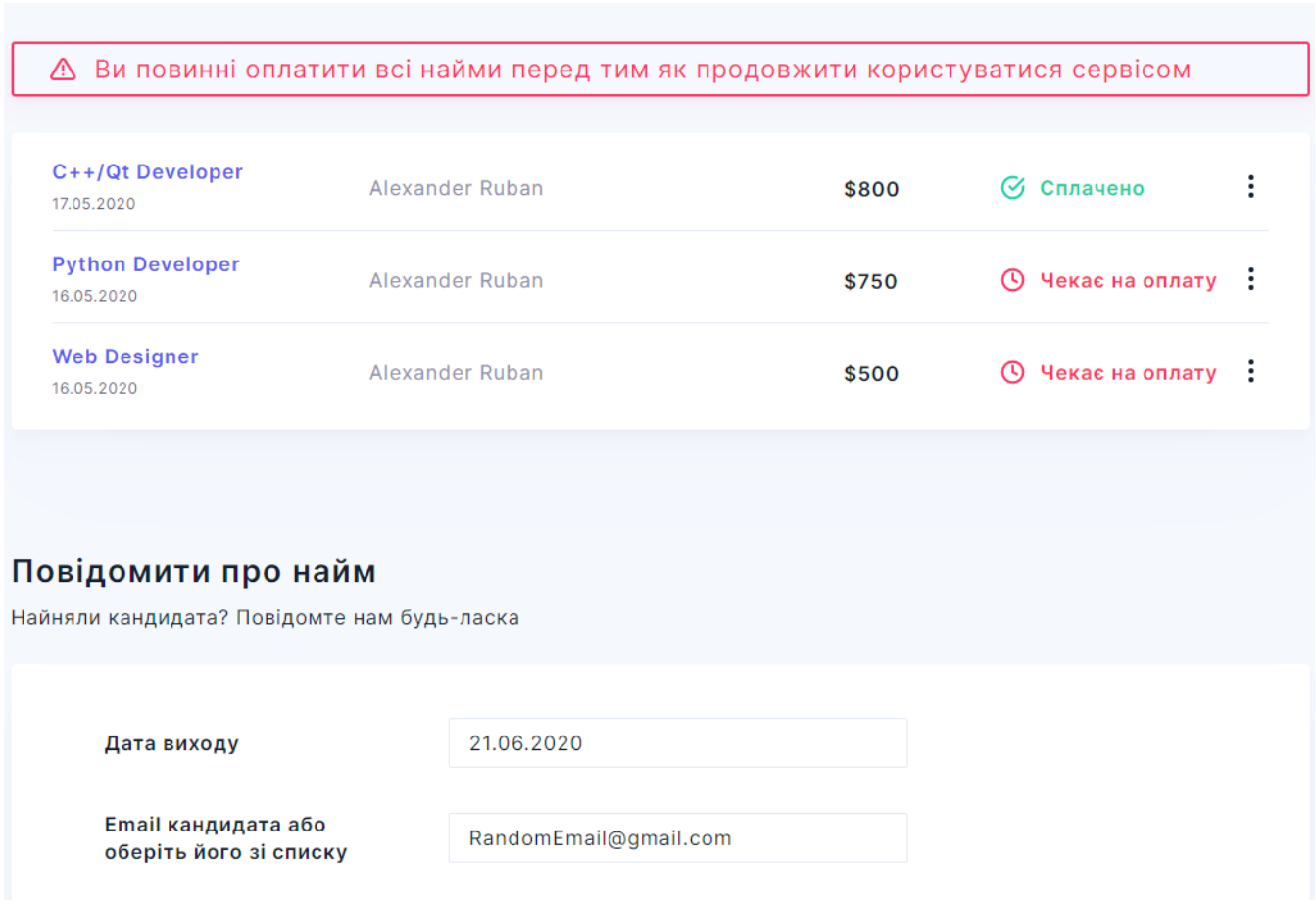
```
use_bias=True, trainable=True,
kernel_regularizer=keras.regularizers.l2(weight_decay),
bias_regularizer=keras.regularizers.l2(weight_decay),
name='gvlad_center_assignment')(x)
x = VladPooling(k_centers=vlad_clusters, g_centers=ghost_clusters, mode='gvlad',
name='gvlad_pool')([x_fc, x_k_center])
else:
raise IOError('==> unknown aggregation mode')
# ================================================
# Fully Connected Block 2
# ================================================
x = keras.layers.Dense(bottleneck_dim, activation='relu',
kernel_initializer='orthogonal',
use_bias=True, trainable=True,
kernel_regularizer=keras.regularizers.l2(weight_decay),
bias_regularizer=keras.regularizers.l2(weight_decay),
name='fc6')(x)
# ================================================
# Softmax Vs AMSoftmax
# ================================================
if loss == 'softmax':
y = keras.layers.Dense(num_class, activation='softmax',
kernel_initializer='orthogonal',
use_bias=False, trainable=True,
kernel_regularizer=keras.regularizers.l2(weight_decay),
```

```
bias_regularizer=keras.regularizers.l2(weight_decay),
name='prediction')(x)
trnloss = 'categorical_crossentropy'

elif loss == 'amsoftmax':
x_l2 = keras.layers.Lambda(lambda x: K.l2_normalize(x, 1))(x)
y = keras.layers.Dense(num_class,
kernel_initializer='orthogonal',
use_bias=False, trainable=True,
kernel_constraint=keras.constraints.unit_norm(),
kernel_regularizer=keras.regularizers.l2(weight_decay),
bias_regularizer=keras.regularizers.l2(weight_decay),
name='prediction')(x_l2)
trnloss = amsoftmax_loss

else:
raise IOError('==> unknown loss.')

if mode == 'eval':
y = keras.layers.Lambda(lambda x: keras.backend.l2_normalize(x, 1))(x)

model = keras.models.Model(inputs, y, name='vggvox_resnet2D_{}_{}'.format(loss, aggregation))

if mode == 'train':
if mgpu > 1:
model = ModelMGPU(model, gpus=mgpu)
# set up optimizer.
if args.optimizer == 'adam': opt = keras.optimizers.Adam(lr=1e-3)
```

```
elif args.optimizer =='sgd': opt = keras.optimizers.SGD(lr=0.1, momentum=0.9, decay=0.0,
nesterov=True)

else: raise IOError('==> unknown optimizer type')

model.compile(optimizer=opt, loss=trnloss, metrics=['acc'])

return model
```

ДОДАТОК Б


Interview top
Кандидати
Пропозиції
Найми
Ua
Мій акаунт
На сайті з 04.06.2020
Дмитро Пенчук
HR in Interview.top
Змінити фото
Ім'я та прізвище
Дмитро Пенчук
Позиція
HR
Компанія
Interview.top
Email
RandomEmail@gmail.com
Моб.Телефон
(093) 093-0520


Ви повинні оплатити всі найми перед тим як продовжити користуватися сервісом
C++/Qt Developer
17.05.2020
Alexander Ruban
$800
Сплачено
Python Developer
16.05.2020
Alexander Ruban
$750
Чекає на оплату
Web Designer
16.05.2020
Alexander Ruban
$500
Чекає на оплату
Повідомити про найм
Найняли кандидата? Повідомте нам будь-ласка
Дата виходу
21.06.2020
Email кандидата або оберіть його зі списку
RandomEmail@gmail.com

ДмитроПенчук

HR in Interview.top

Відкрити контакти

Пропозиція не цікава

Відмовити

Запланувати Інтерв`ю

04.06.2020

Дмитро Пенчук 11:30

Amet minim mollit non deserunt ullamco est sit aliqua dolor do amet sint. Velit officia consequat duis enim velit mollit. Exercitation veniam consequat sunt nostrud amet.

Олег Левченко 11:30

Velit officia consequat duis enim velit mollit. Amet minim mollit non deserunt ullamco est sit aliqua.

Кандидат відкрив вам свої контакти 11:30

Дмитро Пенчук 11:30

Exercitation veniam consequat sunt nostrud ullamco est sit aliqua dolor do amet sint. Velit officia consequat duis enim velit mollit. Exercitation veniam consequat sunt nostrud amet nostrud amet.

Олег Левченко

Ok, velit officia consequat duis enim velit mollit! 11:30

Написати повідомлення Надіслати