НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.912

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра

освітньо-науковою програмою
«Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Алгоритмічно-програмний адаптований метод автоматизованого виявлення трендів в текстових оголошеннях про вакансії»

Виконала:
студентка ІІ курсу, групи КП-81мн
Єрастова Влада Юріївна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПСКС, к.т.н., доцент,
Петрашенко Андрій Васильович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Єрастовій Владі Юріївні

1. Тема дисертації «Алгоритмічно-програмний адаптований метод автоматизованого виявлення трендів в текстових оголошеннях про вакансії», науковий керівник дисертації Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету від «07» квітня 2020 р. № 964-С

2. Термін подання студентом дисертації «15» травня 2020 р.

3. Об'єкт дослідження: процеси виявлення трендів в текстових даних.

4. Предмет дослідження: методи та алгоритми обробки текстових даних оголошень про вакансії для підвищення точності виявлення трендів.

5. Перелік завдань, які потрібно розробити:
- провести аналіз предметної галузі;
- дослідити існуючі методи та системи виявлення трендів;
- проаналізувати специфіку формату та вмісту текстових оголошень про вакансії;
- розробити метод виявлення трендів в текстових даних описів вакансій з урахуванням специфіки останніх;
- здійснити аналіз існуючих засобів для програмної реалізації методу;
- здійснити вибір засобів для реалізації програмного забезпечення;
- розробити програмне забезпечення для виявлення трендів в описах вакансій;
- здійснити тестування реалізації методу та аналіз отриманих результатів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- схема базового алгоритму виявлення трендів;
- схема розробленого адаптованого методу виявлення трендів в описах вакансій;
- компоненти розробленого програмного забезпечення;
- результати роботи розробленого методу.

7. Орієнтовний перелік публікацій:

- Тези доповіді “ Метод автоматизованого виявлення трендів в текстових оголошеннях про вакансії”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «11» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.12.2018	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.03.2019	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	16.05.2019	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення; підготовка матеріалів доповіді на конференції ПМК-2019	14.10.2019	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.12.2019	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	20.02.2020	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; робота над п’ятим розділом	16.04.2020	
8.	Оформлення текстової і графічної частини магістерської дисертації	06.05.2020	

Студент Влада ЄРАСТОВА

Науковий керівник Тетяна ЗАБОЛОТНЯ

РЕФЕРАТ

Актуальність теми. З розвитком різноманітних соціальних майданчиків в мережі Інтернет користувачі кожного дня генерують величезні обсяги природномовних текстових даних. Ці дані часто містять в собі певні закономірності, тож їх аналіз наразі є актуальною задачею. Інформація про майбутні тенденції, яку можна отримати, проаналізувавши ці дані, вважається цінним джерелом знань як для компаній, так і для приватних осіб. Проте обсяги нової інформації та швидкість їх появи є занадто великими для моніторингу їх людьми вручну. Тим не менш, багато аналітиків ринку використовують в своїй роботі саме ручні методи аналізу, що несе ризик пропуску відчутної кількості релевантної інформації. З огляду на це не втрачає своєї актуальності задача щодо розроблення та впровадження нових ефективних методів, способів та алгоритмів автоматизованого виявлення нових трендів в текстових даних.

Об’єктом дослідження є процес виявлення трендів в текстових даних.

Предметом дослідження є методи та алгоритми обробки текстових даних оголошень про вакансії для виявлення трендів.

Мета роботи: підвищення ефективності виявлення трендів в текстових оголошеннях про вакансії за критерієм точності шляхом врахування особливостей вмісту описів цих вакансій.

Методи дослідження. В роботі використовуються статистичні методи, емпіричні методи, методи теорії алгоритмів та програмування.

Наукова новизна роботи полягає в тому, що запропоновано модифікацію методу виявлення трендів, що відрізняється від базового наявністю кроків обробки даних, які допомагають врахувати специфіку формату описів вакансій, чим покращують результати виявлення трендів.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований адаптований метод виявлення трендів в описах вакансій дозволив покращити результати виявлення трендів шляхом врахування специфіки формату текстових даних описів вакансій.

Здійснено програмну реалізацію запропонованого методу, що може бути використана для виявлення трендів на ринку праці в реальному часі.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на XII науково-практичній конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2019 (Київ, 13-15 листопада 2019 р.) та опубліковані у збірнику тез за результатами конференції.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'яти розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень.

У першому розділі проаналізовано існуючі підходи, методи та системи виявлення трендів в текстових даних; описані компоненти систем виявлення трендів, включаючи лінгвістичні та статистичні особливості, алгоритми навчання, створення навчальних і тестових наборів, візуалізацію та оцінку; описані головні недоліки існуючих підходів до виявлення трендів в текстових даних; проаналізовані наявні комерційні рішення вирішення поставленої задачі.

У другому розділі проаналізовано специфіку формату текстових даних описів вакансій, сформовано перелік їх характеристик, які можуть бути враховані під час автоматизованого визначення трендів, обґрунтовано місце та спосіб врахування характеристик цих вакансій; зазначено перелік та детальний опис кроків алгоритму обраного за базовий для задачі виявлення трендів в описах вакансій; запропоновано модифікацію базового методу та засоби реалізації для кожного з етапів модифікації.

У третьому розділі обґрунтовано програмні засоби та технології для реалізації запропонованого методу; описана загальна архітектура системи; наведені особливості реалізації збору даних та особливості реалізації кожного з модулів системи, наведено відповідні графічні матеріали, що ілюструють взаємодію елементів системи.

У четвертому розділі проведено аналіз результатів розробленого програмного забезпечення, що реалізує запропонований метод автоматизованого виявлення трендів в описах вакансій; наведені графічні матеріали кластеризації трендових слів; досліджено розподіл трендових слів в описах та залежність між розташуванням слова та його трендовістю; проведено порівняння результатів з результатами Google Trends, які в певній мірі відрізняються через формат вхідних даних, так як він визначений для конкретної сфери застосування , а саме для оголошень про вакансії.

У п’ятому розділі сформовано та побудовано бізнес-модель кінцевого продукту, що описує ключові моменти в організації діяльності, пов’язаної з поширенням розробленої системи для автоматизованого виявлення трендів в описах вакансій.

У висновках проаналізовано отримані результати роботи.

У додатках наведено фрагменти програмної реалізації запропонованого методу та копії графічних матеріалів.

Робота виконана на 91 аркушах, містить 2 додатки та посилання на список використаних літературних джерел з 37 найменувань. У роботі наведено 18 рисунків та 10 таблиць.

Ключові слова: виявлення трендів, текстові дані, описи вакансій, індекс важливості, лінійна регресія, часові ряди, кластеризація.

ABSTRACT

Actuality. With the development of various social platforms on the Internet, users generate huge amounts of natural text data every day. These data often contain certain patterns, so their analysis is currently an urgent task. Information on future trends that can be obtained by analyzing this data is considered a valuable source of knowledge for both companies and individuals. However, the amount of new information and the speed of its appearance are too large for people to monitor them manually. However, many market analysts use manual analysis methods in their work, which carries the risk of missing a significant amount of relevant information. Given this, the task of developing and implementing new effective methods, techniques and algorithms for automated detection of new trends in text data does not lose its relevance.

Object of research is the process of identifying trends in text data.

Subjects of research are methods and algorithms for processing text data of job advertisements to identify trends.

Goal of the work is to increase the efficiency of trend detection in text job advertisements by the criterion of accuracy by taking into account the peculiarities of the content of the descriptions of these vacancies.

Methods of research. Statistical methods, empirical methods, methods of algorithm theory and programming are used in the work.

Scientific novelty of the work is that a modification of the method of trend detection is proposed, which differs from the basic one by the presence of data processing steps that help take into account the specifics of the format of job descriptions, which improves the results of trend detection.

Practical value of the obtained results is that proposed adapted method of trend detection in job descriptions allowed to improve the results of trend detection by taking into account the specifics of the format of text data of job descriptions.

The software implementation of the proposed method was done, which can be used to identify trends in the labor market in real time.

Approbation. The main provisions and results of the work were presented and discussed at the XII scientific conference of masters and postgraduates "Applied Mathematics and Computer" PMK-2019 and published in the proceedings.

Structure and content of the thesis. Master's thesis consists of an introduction, five chapters, conclusions and appendices.

The introduction provides a general description of the work, evaluated the current state of the problem, substantiated the relevance of the research direction, formulated the purpose and objectives of the study.

In the first chapter the existing approaches, methods and systems for detecting trends in text data were analyzed; components of trend detection systems, including linguistic and statistical features, learning algorithms, creation of training and test kits, visualization and evaluation were described; the main disadvantages of existing approaches to identifying trends in text data were described; the available commercial products for solving task were analyzed.

In the second chapter the specifics of the format of text data of job descriptions were analyzed, a list of their characteristics that can be taken into account during the automated trend determination was formed, the place and method of taking into account the characteristics of these vacancies were substantiated; the list and the detailed description of steps of the algorithm chosen as basic for a task of revealing of trends in job descriptions are specified; a modification of the basic method and means of implementation for each of the stages of modification is proposed.

In the third chapter software tools and technologies for the implementation of the proposed method were substantiated; the general architecture of the system is described; features of realization of data collection and features of

realization of each of modules of system were provided, the corresponding graphic materials illustrating interaction of elements of system were presented.

In the fourth chapter the results of the developed software that implements the proposed method of automated trend detection in job descriptions were analyzed; graphic materials of clustering of trend words are given; the distribution of trend words in descriptions and the relationship between the location of the word and its trend were presented; the results were compared with the results of Google Trends, which differ to some extent in the format of the input data, as it is defined for a specific area of application, namely vacancy announcements.

In the fifth chapter a business model of the final product, which describes the key points in the organization of activities related to the dissemination of the developed system for automated detection of trends in job descriptions was formed and built.

The conclusion contains brief overview of the results obtained in the work. The work is done on 91 pages, contains 2 appendices and reference list of 37 titles. The work contains 18 pictures and 10 tables.

The appendices contain fragments of the software implementation of the proposed method and copies of graphic materials.

Keywords: trend detection, text data, job descriptions, importance index, linear regression, time series, clustering.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

XML – eXtensible Markup Language.

ETD – Emerging Trend Detection.

TDT – Topic Detection and Tracking.

IR – Information Retrieval.

TOA – Technology Opportunities Analysis.

TOAK – Technology Opportunities Analysis Knowbot.

NLP – Natural Language Processing.

SDL – Specification and Description Language.

GSP – Generalized Sequential Patterns.

SQL – Structured Query Language.

HTML – Hypertext Markup Language.

API – Application Program Interface.

POS – Part-of-Speech.

JSON – JavaScript Object Notation

CSV – comma-separated values.

GAM – Generalized Additive Model.

NRMSE – Normalised Root Mean Square Error .

ООП – об’єктно-орієнтоване програмування.

ВСТУП

На сьогоднішній день мережа Інтернет надає широкі можливості її користувачам для поширення і обміну інформацією будь-якого типу. Поява соціальних мереж, блогів, майданчиків для розміщення об’яв посилила комунікацію між значною часткою користувачів Інтернет, які обмінюються величезними обсягами даних. Тому не дивно, що аналізом даних, що генерують люди в мережі щосекунди, цікавляться науковці вже не один рік, адже їх класифікація і аналіз дає змогу отримувати цінні відомості та виявляти певні закономірності. Дані в мережі представлені у різних форматах: тексти, картинки, відео. Для кожного такого формату існує безліч задач пов’язаних з аналізом даних, наприклад: задачі прогнозування, задачі аналізу поведінки користувачів, класифікація зображень, детекція об’єктів, аналіз тональності та ін. Одним із популярних на даний момент напрямом аналізу є виявлення трендів в мережі.

Інформація про майбутні тренди, яку можна отримати проаналізувавши дані в мережі Інтернет, є беззаперечно дуже цінною інформацією для широкого кола людей. Виявлення тенденцій важливе для багатьох людей та організацій. Знання того, що несподівано зростає у популярності, неймовірно важливо для новинних організацій, підприємств, державних структур тощо. На більш конкретному рівні знання тенденційних даних дає поняття про те, що людей приваблює, що люди вважають важливим, до чого люди докладають зусиль тощо. Озброївшись цими знаннями, керівники підприємств чи державних закладів можуть приймати дуже ефективні рішення невідтерміновані у часі.

Виявлення тренду – задача настільки ж популярна, наскільки і важка. Існують технічні, аналітичні та експлуатаційні перешкоди, які можуть зірвати успіх у роботі над розкриттям та розумінням тенденцій таким чином, щоб

відповідати конкретним цілям бізнесу чи організації. З огляду на це не втрачає своєї актуальності задача щодо розроблення та впровадження нових ефективних методів, способів та алгоритмів автоматизованого виявлення нових трендів в текстових даних.

Однією з перспективних сфер для застосування методик автоматизованого виявлення трендів є ринок праці. На сьогоднішній день існує велика кількість сайтів, що допомагають роботодавцям розміщувати об’яви про наявні вакансії, а шукачам роботи – їх знаходити. Такі ресурси містять величезну кількість даних про вакансії, проаналізувавши які, можна отримати інформацію про тенденції на ринку праці, корисну як для працівників, так і для роботодавців. Проте існуючі програмні рішення щодо виявлення трендів в текстових даних є достатньо універсально спрямованими і не дозволяють врахувати особливостей текстів оголошень про вакансії, що негативно впливає на ефективність використання таких сайтів.

Таким чином, з огляду на вищезазначене, розроблення нового методу автоматизованого виявлення трендів в текстових описах вакансій з метою підвищення точності отримуваних результатів є актуальною задачею.

1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА СПОСОБІВ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ТРЕНДІВ

1.1. Поняття тренду. Система виявлення трендів

Новий тренд – це тематична область, корисність та інтерес до якої зростає протягом часу. Наприклад, розширювана мова розмітки (XML) сформувалася як тренд в середині 1990-х. Таблиця 1.1 показує результати пошуку за ключем “XML” в базі даних INSPEC з 1995 року по 1999 рік. Як ми бачимо з таблиці, XML до 1998 вже був добре представлений як тематична область [1].

Таблиця 1.1

Ілюстрація зростання популярності XML в середині 1990-х

Рік	Кількість документів
1995	1
1996	8
1997	10
1998	170
1999	371

Знання нових виникаючих трендів особливо важливо для осіб та компаній, яким доручено контролювати певну сферу чи бізнес. Наприклад, аналітик ринку, який спеціалізується на біотехнологічних компаніях, може захотіти переглянути технічну та пов'язану з новинами літературу щодо останніх тенденцій, які матимуть вплив на компанії, які вона відстежує.

Ручний огляд усіх доступних даних просто неможливий. Людські експерти, які мають завдання визначити нові тенденції, повинні покладатися на автоматизовані системи, оскільки кількість інформації, доступної в цифровій формі, збільшується.

Система для виявлення нових тенденцій (ETD) приймає як вхід колекцію текстових даних та визначає тематичні області, які або є новими, або набувають все більшого значення у корпусі документів. Поточні програми ETD зазвичай поділяються на дві категорії: повністю автоматичні та напівавтоматичні. Повністю автоматичні системи приймають корпус і виокремлюють в ньому список нових тем. Потім рецензент переглядає теми, знайдені системою, щоб визначити, які є справді новими тенденціями. Ці системи часто включають візуальний компонент, що дозволяє користувачеві інтуїтивно відстежувати тему [2, 3]. Напівавтоматичні системи покладаються на введення користувачем початкових даних як перший крок у виявленні тенденцій [4]. Потім ці системи надають користувачеві докази, які вказують на те, чи справді тема є трендом у формі зручних для користувача звітів та екранних форм.

1.2. Аналіз існуючих підходів до виявлення трендів

Почнемо з детального опису декількох напівавтоматичних та повністю автоматичних систем ETD. Розглянемо компоненти системи ETD, включаючи лінгвістичні та статистичні особливості, алгоритми навчання, створення навчальних і тестових наборів, візуалізацію та оцінку.

Найпоширеніше сховище даних для ETD з'явилося в результаті проєкту виявлення та відстеження тем (TDT), який розпочався в 1997 році [5]. В рамках дослідження TDT розроблено алгоритми для виявлення у потоках даних та з’єднання разом матеріалів пов’язаних за тематикою, таких як новини, англійською та мандаринською мовами. Проєкт TDT, хоча і не зосереджений

безпосередньо на виявленні тенденцій, все ж стимулював розробку повністю різних автоматизованих систем, які відстежують зміни актуальності тем у часі. У рамках ініціативи TDT було створено кілька наборів даних. Набори даних TDT – це набори новин та описувачів подій. Кожній парі історія/подія присвоюється відповідне судження. Судження про відповідність – це показник відповідності даної історії до події. У табл. 1.2 зображено декілька прикладів присвоєння судження відповідності пари історії/подія. Таким чином, набори даних TDT можуть використовуватися як навчальні, так і тестові набори для алгоритмів ETD.

Таблиця 1.2

Пари історія/подія

Опис історії	Подія	Судження про відповідність
Story describes survivor's reaction after Oklahoma City Bombing	Oklahoma City Bombing	Yes
Story describes survivor's reaction after Oklahoma City Bombing	US Terrorism Response	No
Story describes FBI's increased use of surveil lance in government buildings as a result of the Oklahoma City Bombing	Oklahoma City Bombing	Yes
Story describes FBI's increased use of surveillance in government buildings as a result of the Oklahoma City Bombing	US Terrorism Response	Yes

Не всі описані нижче системи покладаються на набори даних TDT. Були використані інші підходи до створення тестових даних, наприклад, ручне присвоєння судження відповідності вхідним даним та порівняння результатів системи з результатами, отриманими рецензентами. Цей підхід є виснажливим і обов'язково обмежує розмір набору даних. Вибір атрибутів лежить в основі процесу відстеження, оскільки саме атрибути описують кожен елемент введення та в кінцевому підсумку визначають тенденції. Атрибути, отримані з даних корпусу, вводяться до методів/прийомів, використовуваних кожною системою ETD, які описано нижче.

Деякі дослідницькі групи використовують традиційні методології пошуку інформації (IR) для виявлення тенденцій, що виникають, а інші зосереджуються більше на традиційних підходах до машинного навчання, таких як ті, які використовуються в програмах обміну даними. Робота у сферах візуалізації, що підтримує візуалізацію виявлення тренду, допомогла дослідити декілька методів визначення тем. Коли користувач намагається зрозуміти велику кількість даних, особливо корисною є система, яка дозволяє проводити огляд на різних рівнях деталізації та з різних точок зору. Один з найпростіших підходів - гістограма, де смужки вказують дискретні значення фактичних даних у певний дискретний момент часу. Інформаційна візуалізація призначена для доповнення підходів машинного навчання для виявлення тенденцій.

Накреслення шаблонів тем по часовій шкалі дозволяє побачити швидкість зміни шаблону з часом. Для кожного алгоритму, описаного нижче, обговоримо компонент візуалізації, показуючи, як компонент розширює можливості виявлення трендів системи.

Оцінювання ефективності системи виявлення нових тенденцій може базуватися на формальних показниках, таких як точність (відсоток вибраних елементів, які система отримала правильно) та повнота (питома вага цільових

елементів, які знайшла система), або менш офіційних, суб'єктивних результатах (наприклад, відповіді на питання про зручність використання, такі як: «Чи зрозуміла візуалізація?»). Деталі оцінювання пов'язані з цілями методу і, таким чином, можуть сильно відрізнятися, але для підтвердження ефективності даної системи завжди повинні існувати певні обґрунтування та інтерпретація результатів.

Technology Opportunities Analysis (TOA)

Алан Портер та Майкл Детемпл описують напівавтоматичну систему виявлення тенденцій для аналізу технологічних можливостей [4]. Першим кроком процесу є вилучення документів (наприклад, реферати INSPEC) із галузі знань, що вивчається. Процес вилучення вимагає розроблення списку потенційних ключових слів експертом домену. Ці ключові слова потім об'єднуються у запити, використовуючи відповідні булеві оператори. Цільові бази даних також залучаються на цьому етапі (наприклад, INSPEC, COMPENDEX). Потім запити вводяться в Technology Opportunities Analysis Knowbot (TOAK), спеціальний програмний пакет, який також називають TOAS (Система аналізу можливостей технологій). TOAK витягує відповідні документи (тези) та забезпечує бібліометричний аналіз даних. Бібліометрія використовує таку інформацію, як кількість слів, інформація про дату, інформація про спільну зустрічальність слів, інформація про цитування та інформацію про публікації для відстеження активності в області суб'єкта. TOAK полегшує аналіз даних, наявних у документах. Наприклад, списки ключових слів, що часто зустрічаються, можуть бути швидко сформовані, як і списки приналежності тексту певному автору, країні чи штату.

База даних INSPEC є основним корпусом TOA та його відповідного програмного забезпечення, TOAK. Для вибору атрибутів існують дві можливості.

По-перше (табл. 1.3), список ключових слів (одне слово або кілька слів,

що називаються *n*-грамами) та їх можливі комбінації (за допомогою булевих операторів) надходять до TOAK, який отримує всі відповідні елементи. Кількість появ ключових слів та спільних зустрічей ключових слів – поява двох ключових слів в одному елементі – обчислюється за рік та протягом усіх років. Другий прохід (табл. 1.4) передбачає вибір усіх фраз (одно- та багатомовних) із специфічного поля та обчислення кількості елементів, що містять кожну фразу.

Таблиця 1.3

Перший прохід вибору атрибутів

Атрибут	Опис	Генерація
n-grams	E.g.1 multichip modules1 ball grid array	Manual
Frequency	Count of *n*-gram occurrence	Automatic
Frequency	Count of *n*-gram co-occurrence	Automatic
Date	Given by year	Automatic

Таблиця 1.4

Другий прохід вибору атрибутів

Атрибут	Опис	Генерація
Field	A section of an item (e.g.1 an indexing term or city name)	Manual
Frequency	Count of *n*-gram occurrence in a field	Automatic

Як і більшість систем, що полегшують виявлення трендів у текстових колекціях, TOA покладається на досвід користувача, який досліджує певну область. TOAK забезпечує доступ до багатьох різних джерел даних, включаючи INSPEC, COMPENDEX, патенти США та інші, але обов'язково обмежений, оскільки не всі роботи з науково-дослідної праць запатентовані або опубліковані. Потужність TOAK полягає у візуальному інтерфейсі та простому доступі до різних поглядів даних. У системі відсутні притаманні алгоритми навчання; користувач несе повну відповідальність за виявлення трендів.

Візуалізації в TOA включають таблиці частот, гістограми, зважені коефіцієнти, графіки log-log, криві Фішера-Прая та технологічні карти. Ці засоби графічно представляють інформацію, використовуючи різні зв’язки та кластеризаційні підходи, такі як багатовимірне масштабування. Метою багатовимірного масштабування є зменшення *n*-мірного простору до двох-трьох вимірів.

TOA також може представити карти на основі інших атрибутів, наявних у даних. Зазвичай використовуються такі атрибути, як джерело, країна походження або автор. Подібні методи використовуються для генерування карт ключових слів, які представляють зв'язки між термінами, які часто зустрічаються, та основними картами компонентів, які представляють зв'язки між концептуальними кластерами [4].

TimeMines

Система TimeMines [6] приймає вільні текстові дані з явними тегами дати та розробляє оглядову шкалу статистично важливих тем, охоплених корпусом. Для збору даних TimeMines покладається на методи вилучення інформації та обробки природних мов (NLP). Система використовує методи тестування гіпотез для визначення найбільш релевантних тем у певному часовому інтервалі. Користувачеві надається лише найвагоміша та

найважливіша інформація (визначена програмою).

TimeMines починає обробку за типовою моделлю, яка передбачає, що розподіл функції залежить лише від базової швидкості появи, яка не змінюється в залежності від часу. Кожна ознака в документі порівнюється з типовою моделлю. Статистичний тест використовується для визначення того, чи ознака, що тестується, сильно відрізняється від того, що очікувала б модель. Якщо так, функція зберігається для подальшої обробки, інакше вона ігнорується. Зменшений набір ознак, який розробляється за допомогою першого раунду тестування гіпотез, потім вводиться у другу фазу обробки, яка групує пов'язані ознаки. Групування знову спирається на ймовірнісні прийоми, що поєднують терміни, які, як правило, відображаються в одних і тих же часових рамках в одну тему. Нарешті, використовується поріг для визначення, які теми є найважливішими, і вони відображаються через інтерфейс часової шкали. Поріг встановлюється вручну і визначається емпірично. TimeMines представляє модель даних, не роблячи жодних конкретних висновків про те, чи нова тема популярна чи ні. Вона просто представляє користувачеві найбільш статистично значущі теми та для оцінки тем покладається на знання домену користувача.

У системі TimeMines генерується початковий список атрибутів усіх названих сутностей та певних іменникових словосполучень. Названа сутність визначається як деяка особа, місцеположення чи організація. Іменникові словосполучення відповідають регулярному виразу $(N|J) * N$ менше п’яти слів, де N – іменник, J – прикметник, | позначає союз, а * позначає нуль або більше появ. Таким чином, документи представляються як "мішок атрибутів", де кожен атрибут є істинним або хибним (тобто, чи міститься в документі названа сутність або іменникова фраза). Атрибути наведені у табл. 1.5.

Таблиця 1.5

Атрибути TimeMines

Атрибут	Опис	Генерація
Named Entity	Person, location, or organization	Automatic
n-grams	Follows $(N\|J) * N$ pattern for up to five words e.g. 'textual data mining'	Automatic
Presence	'True' if the named entity or *n*-gram occurs in the document, else 'False'. Each document has a presence attribute for each named entity and *n*-gram.	Automatic
Date	Given by day	Automatic

У додатку TimeMines є два аспекти машинного навчання. По-перше, TimeMines повинен вибрати "найбільш важливу інформацію" для відображення. Для цього TimeMines повинен витягнути "найважливіші" ознаки з вхідних документів.

TimeMines використовує статистичну модель на основі тестування гіпотез для вибору найбільш релевантних ознак. Як зазначалося, система передбачає стаціонарну випадкову модель для всіх ознак (*n*-грам та названих сутностей), витягнутих з корпусу. Стаціонарна випадкова модель передбачає, що всі ознаки є стаціонарними (тобто розподіл їх не змінюється з часом) і випадкові процеси, що генерують будь-яку пару ознак, незалежні. Ознаки, фактичний розподіл яких відповідає цій моделі, вважаються такими, що не містять нової інформації та відкидаються. Ознаки, які сильно відрізняються

від моделі, зберігаються для подальшої обробки. Тестування гіпотези залежить від часу. Іншими словами, протягом певного періоду часу ознака або відповідає моделі (при заданому порозі), або порушує модель. Таким чином, фраза “Бомбардування в Оклахом-Сіті” може бути значущою за один проміжок часу, але не за інший.

Після того, як набір ознак був обрізаний в такий спосіб, TimeMines використовує ще один алгоритм навчання, заснований на тестуванні гіпотез. Використовуючи скорочений набір ознак, TimeMines перевіряє наявність ознак протягом певного періоду часу, які мають подібні розподіли. Ці ознаки згруповані в одну "тему". Таким чином, кожному періоду часу може бути призначена невелика кількість тематичних областей, представлена більшою кількістю ознак.

Існує один з потенційних недоліків ранжирування загальних тем, що випливають із значущих атрибутів. Імовірність появи атрибута вимірюється у порівнянні з усіма іншими його появами в корпусі. В результаті послідовно використовуваний атрибут може бути виявлений неналежним чином. Розслідування Кеннета Старра-президента Клінтона, безперечно, є найбільш висвітленою історією в корпусі TDT-2, проте вона посіла 12 місце, оскільки вона дуже поширена в усьому світі. На відміну від більш тривалого періоду часу, включаючи час після того, як її висвітлення притихло, історія, мабуть, посіла б 1 місце.

Як і всі алгоритми, які тут представлені, остаточне визначення, чи тема є новим трендом чи ні, залишається користувачеві, але на відміну від TOA, користувач не керує системою TimeMines. Ця система повністю автоматизована; даний корпус із міткою часу представляє графічне зображення тем, які домінують над корпусом протягом визначених періодів часу.

PatentMiner

Система PatentMiner була розроблена для виявлення тенденцій патентних даних, використовуючи динамічно генерований SQL-запит, заснований на введених користувачем критеріях вибору [7]. Система підключена до бази даних IBM DB2, що містить усі надані патенти США. У системі є два основні компоненти: модуль ідентифікації фрази за допомогою послідовного майнінгу шаблонів та модуль виявлення трендів за допомогою форми запитів [8, 9].

База даних IBM DB2, що містить усі патенти США послужила основою для корпусу. Кілька процедур готують дані для вилучення атрибутів. Стоп-слова видаляються. Решті слів присвоюються ідентифікатори із зазначенням положення в документі та випадків межі речення, абзацу та розділу. Після того, як підмножина патентів буде визначена за категорією та діапазоном дат, алгоритм узагальнених послідовних патернів (GSP) вибирає атрибути, визначені користувачем, що називаються «фрази» [9]. Розглядаються лише фрази з підтримкою, що перевищує визначений користувачем мінімум. Фразою може бути будь-яка послідовність слів з мінімальним та максимальним «розривом» між будь-яким із слів. Розриви можуть бути описані у словах, реченнях, абзацах чи розділах. Наприклад, якщо мінімальний розрив у реченні дорівнює 1 для фрази "виникаючі тренди", то "виникаючі" та "тренди" повинні з’являтися в окремих реченнях. Або якщо максимальний розрив у абзаці дорівнює одному, то в тому ж абзаці мають виникати "виникаючі" та "тренди". Вікно часу вказує кількість слів, які потрібно згрупувати, перш ніж рахувати довжину розриву. Нарешті, мова специфікації і опису алгоритмів (SDL) [10] визначає, які типи трендів (наприклад, вгору, круговий тощо) відображаються. У табл. 1.5 узагальнено ці ознаки.

Таблиця 1.6

Атрибути PatentMiner

Атрибут	Опис	Генерація
n-grams	Search phrase, e.g., emerging trends	Manual
Size	Minimum gap, with distinct gaps for words, sentences, paragraphs, and sections.	Manual
Size	Maximum gap, with distinct gaps for words, sentences, paragraphs, and sections.	Manual
Size	Time window, groups words in a phrase before determining gaps	Manual
Ratio	Support, number of search phrases returned divided by total number of phrases	Manual
Date	Given by available granularities	Manual
Shape	Graphical trend appearance over time, e.g., spiked or downwards	Manual

Більшість систем використовують традиційні методи IR для отримання ознак з текстового корпусу, який служить вхідним; система PatentMiner використовує інший підхід. PatentMiner адаптує техніку послідовного узгодження зразків, яка часто використовується в системах передачі даних.

Ця техніка трактує кожне слово в корпусі як транзакцію. Система

відповідності шаблонів шукає такі шаблони слів, що часто зустрічаються. Слова можуть бути суміжними або розділеними змінною кількістю інших слів (до деякого максимуму, який встановлює користувач). Ця методика дозволяє системі визначити терміни, що часто зустрічаються, і трактувати їх як єдину тему [7].

Як і у TimeMines, документи у наборі вхідних даних поширюються на різні колекції на основі інформації про їх дату. Вищеописана методика використовується для вилучення фраз з кожного контейнера, і обчислюється частота появи кожної фрази у всіх контейнерах. Форма запиту використовується, щоб визначити, які фрази потрібно витягти, на основі запиту користувача.

Обробка форми запитів є ще одним інструментом навчання, запозиченим з data mining [10]. У системі PatentMiner підрахунок частоти фрази представляє собою сховище даних, яке можна видобути за допомогою інструмента форми запиту. Форма запиту має можливість зіставляти нахили вгору та вниз на основі підрахунку частоти. Наприклад, фраза, яка набирає популярність, може часто виникати протягом двох суміжних часових відрізків, потім вирівнюватися, перш ніж продовжувати тенденцію до зростання. Форма запиту дозволяє користувачеві графічно визначати різні фігури для виявлення трендів (або інших додатків) і отримує фрази з частотними розподілами, які відповідають запиту.

1.3. Аналіз комерційних систем виявлення трендів

Комерційні програмні продукти доступні для допомоги компанії, зацікавленій в ETD. Деякі компанії спеціалізуються на наданні контенту, деякі надають можливості пошуку інформації загального призначення, такі як вилучення функцій, управління документами, пошук, категоризація та кластеризація.

Хоча всі ці продукти можуть бути використані для полегшення зусиль ETD, лише деякі з них мають можливості, спеціально спрямовані на аналіз та виявлення тенденцій. Ці продукти коротко представлені в цьому підрозділі.

Autonomy

Засіб Clusterizer, що надається Autonomy, забезпечує підтримку ETD [11]. Clusterizer приймає yf d[jls корпус і визначає набір кластерів з даних. Ці кластери можна використовувати для ідентифікації нових тем у даних, взявши набір кластерів з попереднього періоду часу та порівнявши їх із набором кластерів у поточному періоді. По суті, Clusterizer показує огляд кластерів у часі. Таким чином, інструмент призначений для відображення тенденцій у кластерах, включаючи появу та зникнення кластерів, а також зміни розміру кластера.

Алгоритми узгодження шаблонів нелінійної адаптивної цифрової обробки сигналів, принципи теорії інформації Клода Шеннона, Байєсові заключення, та нейронні мережі складають ядро технології Autonomy [12]. Оскільки природна мова містить багато дублювання, поняття, які повторюються рідше в документі, вважаються такими, що відповідають суті цього документа. Таким чином, Autonomy описує документ із шаблонами, заснованими на використанні та частоті термінів. Адаптивне імовірнісне моделювання концепції використовується для визначення відповідності між документами та подальшого навчання системи [13].

Модуль Autonomy Clusterizer допомагає експерту у виявленні тенденцій. Панель Breaking News автоматично знаходить нові кластери інформації, порівнюючи кластери різних періодів часу. Панель спектрографа – це інструмент візуалізації, який розміщує кластери як лінії у часі. Колір і ширина ліній вказують на розмір і якість кластера; зміни сигналізують про збільшення чи зменшення значущості.

Як і засоби, описані в попередньому розділі, спектрограф призначений для надання користувачеві уявлення про дані. Потім експерт повинен використовувати ці дані для формування висновків щодо обґрунтованості даної тенденції. З точки зору оцінки, не було проведено офіційного оцінювання ефективності цих інструментів при використанні для ETD.

SPSS LexiQuest

Продукти LexiQuest використовують передову технологію обробки природних мов для доступу, керування та отримання текстової інформації. LexiQuest Lex-iMine – це інструмент для видобутку тексту, розроблений, щоб допомогти користувачеві отримати нові відомості шляхом визначення ключових понять та взаємозв'язків між ними. Він використовує поєднання лінгвістичного аналізу на основі словника та відповідності статистичної близькості для виявлення ключових понять, а також ступеня взаємозв'язку між поняттями.

Ідентифікація поняття досягається за допомогою використання невпорядкованих словників, які містять (багатослівні) терміни, які використовуються для узгодження термінів у вхідному тексті LexiMine. Поява термінів та спільна поява термінів підраховуються або по абзацам, або по документу і використовуються для побудови реляційних карт понять. Хоча жодні алгоритми машинного навчання не використовуються як такі, формули подібності терміна LexiQuest подібні до тих, що використовуються в аналізі даних [14].

Поняття взаємозв'язку зображені на графічній карті, що відображає сукупне виникнення понять. Карту можна використовувати для дослідження тенденцій. Подальший аналіз можна отримати, імпортуючи дані LexiMine у відповідний інструмент Clementine [15]. Як і у багатьох дослідницьких та комерційних інструментах, згаданих у дослідженні, обґрунтованість тенденцій в остаточному підсумку залишається експертом, а ефективність

інструментів не вимірюється і не оцінюється формально.

ClearForest

ClearForest надає платформу та продукти для вилучення відповідної інформації з великої кількості тексту та подання зведеної інформації користувачеві [16]. Два додатки, ClearResearch та ClearSight, корисні для програм ETD. ClearResearch призначений для представлення на одному екрані складних взаємозв'язків, що дозволяє користувачам переглядати новини та контент дослідження в контексті. ClearSight забезпечує прості графічні візуалізації стосунків між компаніями, людьми та подіями у діловому світі. Він також надає в реальному часі оновлення нових запусків продуктів, зміни в управлінні, нові технології, тощо, в будь-якому заданому контексті. Користувачі можуть детальніше ознайомитися з кожною темою, щоб переглянути більше інформації або прочитати пов’язані статті.

ClearForest використовує підхід, заснований на правилах, для визначення «сутності» та «фактів». Сутність – це послідовність одного або декількох слів, що відповідають одному поняттю. Факт – це відносини між сутностями. Правила можуть містити part-of-speech чи stem ідентифікатори для слів, посилання на словники та лексикони, та структурні характеристики. Двигун вилучення застосовує правила до вхідних документів та виводить інформацію з тегами. Точне розташування всієї вилученої інформації записується. Виникнення та спільне виникнення понять використовуються аналітичним механізмом для узагальнення інформації. Інструменти візуалізації відображають цю інформацію на різних рівнях деталізації.

ClearForest ділить процес виявлення на чотири етапи. Перший – збір інформації, виконується пошуковими системами. Другий і третій – вилучення і поєднання інформації, керує ClearForest. Останній етап, що ідентифікує дійсний тренд, обробляє експерт з предметної області за допомогою відображення графіку тенденцій ClearForest. З точки зору оцінки, як і

LexiQuest, ClearForest не проводив офіційної оцінки компонента ETD [16].

1.4. Висновки до розділу 1

У цьому розділі було розглянуто і проаналізовано існуючі методи для виявлення трендів та існуючі системи. Були досліджені і проаналізовані напівавтоматичні, автоматичні та комерційні підходи та системи для задачі виявлення трендів. Були розглянуті компоненти систем, включаючи лінгвістичні та статистичні особливості, алгоритми навчання, створення навчальних і тестових наборів, візуалізацію та оцінювання результатів.

Очевидно, що недоліком напівавтоматичних систем є залучення спеціалістів саме до процесу виявлення трендів.

TimeMines використовує статистичну модель на основі тестування гіпотез для вибору найбільш релевантних ознак. TimeMines представляє модель даних, не роблячи жодних конкретних висновків про те, чи нова тема популярна чи ні. Вона просто представляє користувачеві найбільш статистично значущі теми та для оцінки тем покладається на знання домену користувача.

Система PatentMiner розроблена для виявлення тенденцій патентних даних, використовуючи динамічно генерований SQL-запит, заснований на введених користувачем критеріях вибору.

Ці системи використовують набори даних і бази даних, які були створені давно і не факт, що вони оновлюються і залишаються актуальними, , що робить і самі результати неактуальними.

Також існують підходи з використанням методів машинного навчання, проте застосування такого підходу передбачає наявність великої розміченої колекції даних, а у випадку з поняттям тренду це може бути складною і доволі ресурснозатратною задачею.

Описані комерційні системи є достатньо повними у наявних

функціональних можливостях, проте такі системи зазвичай коштують великих грошей і є недоступними для простого користувача мережі.

Таким чином, з огляду на вищезазначене, розроблення нового методу автоматизованого виявлення трендів в текстових описах вакансій з метою підвищення точності отримуваних результатів є актуальною задачею.

2. АДАПТОВАНИЙ МЕТОД АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ТРЕНДІВ В ТЕКСТОВИХ ОГОЛОШЕННЯХ ПРО ВАКАНСІЇ

2.1. Аналіз специфіки вмісту текстових даних оголошень про вакансії

На сьогоднішній день існує велика кількість сайтів, що допомагають роботодавцям розміщувати об’яви про наявні вакансії, а шукачам роботи – їх знаходити. Такі ресурси відносяться до окремого типу соціальних мереж, що мають на меті встановити зв'язок між роботодавцем та працівником, який шукає роботу.

В більшості випадків користувачами таких соціальних мереж виступають компанії або підрозділи компаній, які задіяні у пошуках нових працівників. Їх головна мета – це розмістити текст вакансії на сайті та обмінюватись повідомленнями з можливими кандидатами. Такі ресурси містять величезну кількість даних про вакансії, проаналізувавши які, можна отримати інформацію про тенденції на ринку праці, корисну як для працівників, так і для роботодавців.

Підходи до виявлення трендів, які наразі існують, та методи, які для цього застосовуються, не є достатньо ефективними, так як вони є достатньо універсально спрямованими та мають низку недоліків.

Перш за все на сьогодні дуже часто використовують ручне виявлення трендів. Цей підхід має свої недоліки. Це насамперед швидкість обробки даних, яка є доволі низькою, вартість такого процесу, яка є високою, та якість результатів не належного рівня, так як різні спеціалісти трактують тренди по-різному.

Також наразі існує багато методів та систем автоматизованого виявлення трендів в текстових даних, проте наявні методи мають свої недоліки, головний з яких це те, що вони дають загальну оцінку текстів та не враховують специфіку текстових даних (в нашому випадку – оголошень про

вакансії). Якщо знехтувати врахуванням даної специфіки, виявлені тренди можуть бути помилковими, загальновживані поняття пов’язані з роботою , можуть хибно бути розціненими за тренди, що знизить ефективність всього процесу виявлення трендів і знецінить результати його роботи. Тому можна зробити висновок, що автоматизоване визначення трендів в описах вакансій є особливо актуальним.

Провівши дослідження існуючих методів та систем автоматизованого виявлення трендів в текстових даних, можна зробити висновок, що необхідним є детальне вивчення специфіки оголошень про вакансії, що допоможе підвищити ефективність процесу виявлення трендів в текстових даних такого роду шляхом врахування їх характеристик.

Текстові оголошення про вакансії на соціальних платформах для пошуку роботи мають свої особливості, саме тому вони відрізняються від інших типів природномовних текстових даних та можуть ускладнити задачу виявлення тренду. Розглянемо їх докладніше.

Зазвичай описи вакансій пишуться діловою мовою без використання художніх зворотів та проявів емоцій. В залежності від сфери діяльності, в якій представлена вакансія, використовується різна термінологія для позначення навичок та вимог здобувача роботи. Уся інформація подається стисло у вигляді понять в називному відмінку. Описи вакансій зазвичай мають схожу структуру, яка містить таку інформацію: посада, компанія, вимоги до претендентів, технічні та особисті навички, обов’язки, умови праці. Проте в описах вакансій часто міститься багато шуму, а ключові слова, які позначають навички чи інші поняття, що цікавлять здобувачів, важко знайти. Вони іноді містять фрази, пов'язані не з роботою, а з описом компанії, або фрази, що використовують для заохочення кандидатів. Це призводить до погіршення точності виявлення трендів та помилкового визначення загальних понять, пов’язаних з роботою, як трендів.

Отже, врахування вищезгаданих особливостей текстових оголошень про вакансії може підвищити якість виявлення трендів в такого роду текстових даних.

2.2. Базовий метод визначення трендів в текстових даних, представлених в мережі

Сьогодні популярним джерелом даних для вивчення тих чи інших трендів, що існують в суспільстві, є текстові дані соціальних медіа-джерел. У своїй роботі Hennig та ін. [17], спираючись на працю Abe та ін. [18], запропонували метод виявлення трендів в блог-постах, який фокусується на різних аспектах тексту цих постів. Цей метод дозволяє визначити найважливіші слова в корпусі документів та прослідкувати зміни показників вагомості цих слів з часом, на основі чого можна зробити висновки, чи є слово трендом на даний момент.

2.2.1. Передумови виявлення трендів

Через те, що виявлення трендів повинно здійснюватися над текстовими даними, згенерованими в мережі, та виявляти останні тенденції, необхідно використовувати певний часовий проміжок. Наприклад, останніх три місяці або навіть тижнів повинно бути достатньо.

Необхідно отримати найважливіші слова чи фрази із вмісту публікацій в блозі. В інтелектуальному аналізі даних існує декілька відомих методів для обчислення значення важливого слова. Якщо припустити, що вміст усіх описів вакансій є корпусом документів, то для нього можна використовувати індекс TFIDF – один з найбільш відомих показників важливості, який співвідносить частоту певного терміну з оберненою частотою документа. Ці терміни витягуються заздалегідь методом термінного вилучення. Для виявлення тенденцій було змінено цей показник. Для того, щоб виявити

тенденції, обов'язково потрібно знати значення терміну в певний момент часу для моніторингу змін. Тому визначення дефініції TFIDF було дещо змінено. Отже, його було визначено так:

$$t = t_j - t_{j-1} \tag{1}$$

$$\text{TFIDF}(term_i, p_{tj}) = \frac{TF(term_i, P_{tj})}{TP(term_i, P_{tj})} * \log \frac{|P|}{PF(term_i, P)} \tag{2}$$

Рівняння (1) і (2) описують визначення важливості терміну для певних часових рамок. Визначений часовий інтервал можна регулювати від днів до мілісекунд. TF означає частоту термінів у наборі повідомлень P_{tj} всередині тимчасового періоду, TP повертає кількість термінів у наборі повідомлень P_{tj}, P – кількість повідомлень, PF – кількість вакансій, що містять цей термін у загальному корпусі. Для подальшого використання необхідно нормалізувати показник TFIDF.

Для виявлення тенденцій необхідно стежити за змінами в часі; для цієї роботи ідеально підходить лінійна регресія. Koegh та ін. [19] описали різні зразки часових рядів, які можуть бути використані в даному випадку. Тому для моніторингу змін у часі використовується проста лінійна регресія, описана у книзі «Вступ до аналізу лінійної регресії Монтгомері» [20]. За допомогою лінійної регресії можна обчислити лінію тренду. Для цієї лінії тренду нахил та перехоплення обчислюються за допомогою наступного загального визначення:

$$Slope = \frac{\sum_{j=1}^{n}(y_{tj} - \bar{y})(x_j - \bar{x})}{\sum_{j=1}^{n}(x_j - \bar{x})^2} \quad (3)$$

$$Intercept = \bar{y} - Slope * \bar{x} \quad (4)$$

У рівняннях (3) та (4) y – залежна пояснювана змінна, x – незалежна, а $\bar{y}$ та $\bar{x}$ – їх середні значення відповідно.

Визначений нахил дає інформацію про те, збільшується чи зменшується тенденція з часом. Крім того, значення перехоплення надає інформацію про базову лінію значень. Якщо перехоплення позитивне, лінійна регресія буде недостатньо зростаючою. Тому для нової тенденції лінія тренду повинна мати негативне перехоплення. Крім того, якщо лінійна регресія має позитивне перехоплення, то тенденція може бути популярною або знижуватися залежно від того, чи нахил позитивний, чи негативний.

2.2.2. Алгоритм виявлення трендів

Усі необхідні етапи, включаючи підготовку, наведені на рис. 2.1 і виглядають так:

I. Збір даних блог-постів.

II. Розрахунок лінії тренду.

1. Розрахунок індексу важливості для знаходження найбільш важливих слів.
2. Розрахунок нахилу та вільного коефіцієнта для індексу важливості.
3. Розрахунок нахилу та вільного коефіцієнта для структури посилань.
4. Розрахунок нахилу та вільного коефіцієнта для тегів.

III. Кластеризація.

IV. Присвоєння значень [17].

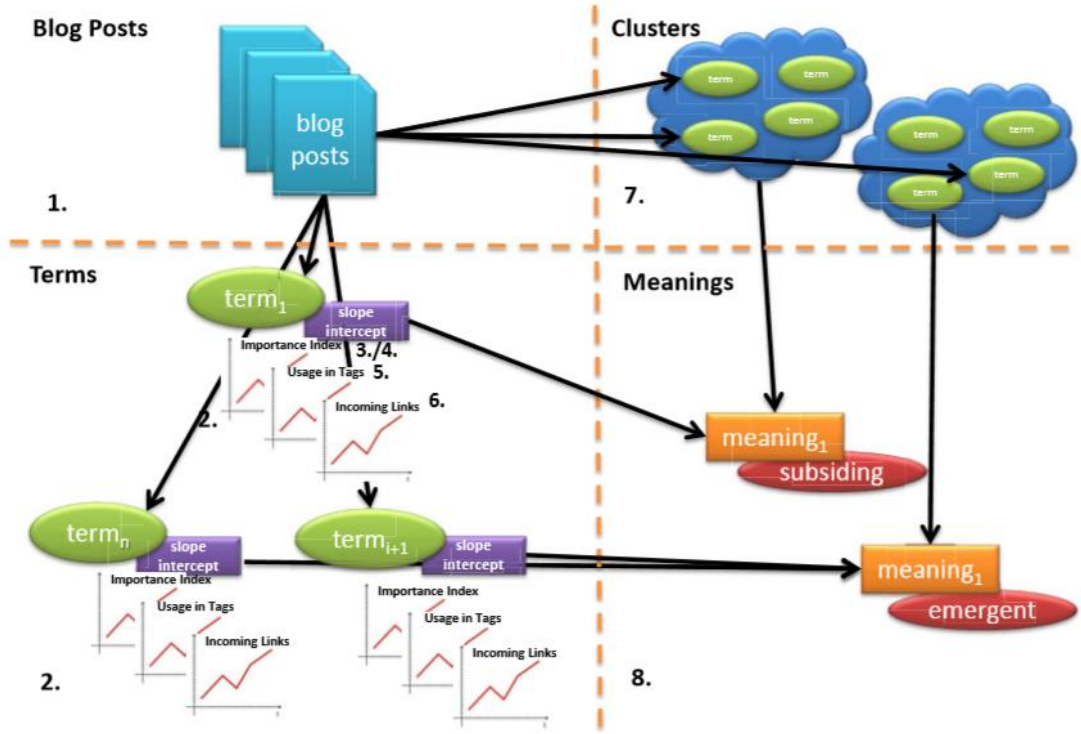


Рис. 2.1. Схема базового алгоритму виявлення трендів

Розглянемо більш докладно деякі з етапів алгоритму.

А. Аналіз посилань. Однією з найпотужніших структур всередині блогосфери є взаємозв'язок між блогами або публікаціями. Між блогами існує багато різних типів посилань. Посилання, розміщені на домашній сторінці блогів, дуже потужні. Ці посилання часто представляють інші блоги, що містять посилання на цей блог. Оскільки вони іноді відображається на всіх підсторінках, посилання можуть бути потужним механізмом зв’язку. На додаток до цих посилань, посилання всередині вмісту публікації є принаймні настільки ж потужними, як і посилання на домашній сторінці блогів. Ці посилання вказують на публікації навіть від інших блогерів, які

пишуть про подібну або споріднену тему. Для пошукових систем ці посилання часто є дуже цінними щодо актуальності даної теми. Це особливо важливо, оскільки може викликати інші дискусії. Крім того, існує ще одна категорія посилань на блоги, які використовуються в коментарях. Оскільки коментування пов'язане з великою кількістю зловживань та спаму, ці посилання майже не мають значення та мають менший вплив, ніж інші посилання в публікаціях та блогах. Усі ці посилання часто містяться як традиційні гіперпосилання у змісті публікації. Крім того, часто можливо, що у стрічці блогу публікуються деякі дуже важливі посилання. Як вже було зазначено із вмісту посту вилучаються терміни і структура посилань, тому терміни можуть бути пов’язані з посиланнями. Для подальшого поєднання показників необхідно нормалізувати цей набір даних.

B. Аналіз вмісту посту. Другий аспект, на який звертається увага – це вміст кожної публікації. Крім реального змісту публікації, у корпусі наявні також заголовки та короткий опис. Показники тренду ґрунтуються на зміненому індексі важливості. Як було зазначено раніше, це визначення описує важливість одного слова у певний момент часу. Використання значень TFIDF для виявлення тенденцій використовували Abe et el. [18] вперше із заголовками доповідей для двох конференцій. Показники TFIDF відстежують зміну значень індексу важливості з часом. Якщо є значна зміна, ці показники є одним із трьох аспектів, які можуть класифікувати, з’являється тенденція чи ні.

C. Аналіз тегів. Оскільки блогосфера складається з даних напівструктурованого формату, то недостатньо орієнтуватися на різні текстові джерела всередині блогосфери. Крім того, якість

результатів виявлення тенденцій може бути покращена шляхом включення додаткової метаінформації. Найвідоміші частини структури блогосфери – це теги та категорії. Теги – це ключові слова, що описують зміст публікації у стислому вигляді. Вони повинно зробити більш зручним пошук всередині блогосфери. Основна проблема тегів полягає в тому, що їх використання сильно залежить від автора публікації. Деякі автори описують вміст публікацій, використовуючи занадто багато різних тегів, а інші повністю забувають анотувати свій матеріал тегами. Якщо припустити, що всі автори використовують теги, існує проблема, що автори можуть описати подібний вміст абсолютно різними словами. Оскільки в кожній публікації є також позначка часу, можна проаналізувати використання тегів у часі.

D. Виявлення трендів. Нарешті, відповідно до показників для кожного аспекту, слід визначити фінальні показники тренду, а також значення для цих показників. Проаналізовано три основні аспекти. Для одержання двох кінцевих показників нахилу та перехоплення для кожного виразу показники різних аспектів публікації повинні поєднуватися з кластерною інформацією про терміни.

    1) Показники терміну: на першому кроці необхідно визначити загальний показник для схилу та перехоплення для кожного терміну.
    2) Показники кластеру: оскільки тренди, чутливі до тематики також мають бути враховані, недостатньо обчислити нахил та значення перехоплення для одного терміну. Оскільки подібні терміни групуються разом у кластери за попередньо виконаною кластеризацією, необхідно класифікувати, чи кластер представляє нову тенденцію чи ні. Це робиться

шляхом обчислення середнього нахилу та перехоплення всередині кластера.

3) Значення: Після обчислення показників для кожного кластеру, необхідно визначити значення для цих показників. За допомогою цих значень стає можливим визначити, чи містить кластер тренд чи ні. Як вже було описано, нахил та перехоплення є достатніми для класифікації кластеру як такого, що містить виникаючий тренд, популярний або затухаючий [17].

2.3. Модифікований метод виявлення трендів для текстових даних описів вакансій

Як вже було зазначено, існують певні особливості, що характеризують оголошення про вакансії, що розміщенні на соціальних платформах для пошуку роботи. Дуже важливим кроком є врахування таких характеристик вакансій:

1. Наявність цифр і символів, які часто заплутують розуміння слів і виразів і негативно впливають на коректність виявлення трендів.
2. Переважання слів в називному відмінку – для покращення словника пропонується привести словоформи до основи слова шляхом відкидання допоміжних частин, таких як закінчення чи суфікс.
3. Багатослівні вирази можуть позначати навички чи інші фрази, що стосуються роботи, і виступати трендами, тому їх також треба враховувати.
4. Наявність власних назв, таких як місцезнаходження чи імена людей, не повинні відображатись як знайдені тенденції, так як вони також можуть негативно вплинути на виявлення трендів: виявити, що

Лондон є трендом для описів вакансій у Великобританії, не є корисним.

5. Дієслова та прислівники – частини мови, які не мають сенсу з точки зору виявлення тенденцій. Тому пропонується зберігати лише слова, позначені як іменники чи прикметники, або частину мови яких неможливо визначити.

Наступним треба визначити де саме доцільно врахувати описані вище характеристики оголошень про вакансії.

Процес виявлення трендів складається з таких етапів: етап передоброблення тексту, етап виявлення тренду та етап аналіз результату виявлення тренду. Автор вважає, що найкраще модифікувати етап передоброблення текстових даних, щоб врахувати багатослівні вирази, різні частини мови та власні назви, що покращить результати роботи виявлення трендів.

Перш за все, вхідний текст повинен пройти процес лематизації, тобто приведення слів до їх основи. Це має бути виконано для зменшення шуму в описах вакансій.

По-друге, пропонується використати алгоритм GSP для збору всіх n-грам з $n > 1$ з мінімальною підтримкою 0.2. Застосуємо алгоритм для всіх описів вакансій, тож вони можуть поділяти однакові n-грами для кращого виявлення тенденцій. GSP - один з перших алгоритмів виявлення послідовних шаблонів у базах даних послідовностей, запропонований Srikant [21]. Він використовує априорі-подібний підхід для виявлення послідовних зразків.

У загальному вигляді робота алгоритму полягає в декількох проходах по початковому набору даних. На першому проході алгоритм підраховує підтримку для кожного елемента (число послідовностей, в які входить елемент). Під час даного кроку алгоритм виділяє з вихідного набору елементів елементи, що часто зустрічаються і задовольняють значенням мінімальної

підтримки. Кожен подібний елемент на першому кроці є одноелементною послідовністю. На початку кожного наступного проходу є певна кількість послідовностей, що часто зустрічаються, виявлених на попередньому кроці алгоритму. З них будуть формуватися більш довгі послідовності, звані «кандидатами». Кожен кандидат – це послідовність, яка на один елемент довше ніж ті з яких кандидат був сформований. Таким чином, число елементів всіх кандидатів однакове. Після цього розраховується підтримка для кожного кандидата. В кінці кроку стане відомо, які з послідовностей кандидатів насправді є частими. Знайдені часті послідовності слугуватимуть вихідними даними для наступного кроку алгоритму. Робота алгоритму завершується тоді, коли не знайдено жодної нової частої послідовності в кінці чергового кроку, або коли неможливо сформувати нових кандидатів.

Таким чином, в роботі алгоритму можна виділити дві основні операції:

1) генерація кандидатів послідовності;
2) підрахунок підтримки кандидатів.

По-третє, автор пропонує провести фільтрування власних назв. Для виконання розпізнавання іменованих осіб (Named-Entity Recognition) в оголошеннях про роботу на відповідних мовах пропонується використати Polyglot-NER [22]. Завдання вилучення іменованих об'єктів має на меті вилучити фрази з простого тексту, які відповідають сутностям. Polyglot-NER розпізнає 3 категорії сутностей:

1) Місцеположення (тег: I-LOC): міста, країни, регіони, континенти, мікрорайони, адміністративні підрозділи і т.д.
2) Організації (тег: I-ORG): спортивні команди, газети, банки, університети, школи, некомерційні організації, компанії і т.д.
3) Особи (тег: I-PER): політики, вчені, художники, спортсмени і т.д.

Моделі навчалися на наборах даних, витягнутих автоматично з Вікіпедії. Наразі Polyglot підтримує 40 основних мов, зображених на рис. 2.2.

```
1. Polish                   2. Turkish                  3. Russian
4. Indonesian               5. Czech                    6. Arabic
7. Korean                   8. Catalan; Valencian       9. Italian
10. Thai                    11. Romanian, Moldavian,    12. Tagalog
13. Danish                  14. Finnish                 15. German
16. Persian                 17. Dutch                   18. Chinese
19. French                  20. Portuguese              21. Slovak
22. Hebrew (modern)         23. Malay                   24. Slovene
25. Bulgarian               26. Hindi                   27. Japanese
28. Hungarian               29. Croatian                30. Ukrainian
31. Serbian                 32. Lithuanian              33. Norwegian
34. Latvian                 35. Swedish                 36. English
37. Greek, Modern           38. Spanish; Castilian      39. Vietnamese
40. Estonian
```

Рис. 2.2. Підтримувані мови Polyglot

По-четверте, пропонується провести фільтрацію частин мов в описах вакансій. Для цього пропонується використати Treetagger [23] – тегер частин мов, що не залежить від мови. TreeTagger – це інструмент для анотування тексту з частиною мови та інформацією про лему. Він був розроблений Гельмутом Шмідом у проєкті TC в Інституті обчислювальної лінгвістики університету Штутгарта. TreeTagger успішно використовується для тегування різних мов, включаючи німецьку, англійську, французьку, італійську, голландську, іспанську, болгарську, російську, грецьку, португальську, китайську, суахілі, латинську, естонську та старі французькі тексти, і адаптується до інших мов, якщо доступні лексикон та навчальний корпус із тегами, що вказується вручну [23].

Визначення трендів в описах вакансій після того, як вони пройшли етап передоброблення, пропонується проводити за допомогою базового методу виявлення трендів, що запропонований для текстових даних блог-постів. Проте, з огляду на те, що текстові дані описів вакансій дещо відрізняються від вмісту постів, базовий алгоритм потребує деяких змін. Зокрема, описи вакансій не містять структури посилань та тегів, тому кроки, що стосуються аналізу посилань і тегів пропонується вилучити.

Отже, у методі для виявлення трендів, який був запропонований, можна виокремити наступні етапи (рис. 2.3):

I. Збір даних оголошень про вакансії.
II. Попереднє оброблення:
    1. Видалення стоп-слів, пунктуації, приведення до нижнього регістру.
    2. Лематизація.
    3. Формування *n*-грам.
    4. Фільтрування іменованих об’єктів.
    5. Фільтрування частин мов.
III. Розрахунок лінії тренду:
    1. Розрахунок індексу важливості для знаходження найбільш важливих слів.
    2. Розрахунок нахилу та вільного коефіцієнта для індексу важливості.
IV. Кластеризація.
V. Присвоєння значень на основі нахилу та вільного коефіцієнту (новий; популярний; такий, що знижується).

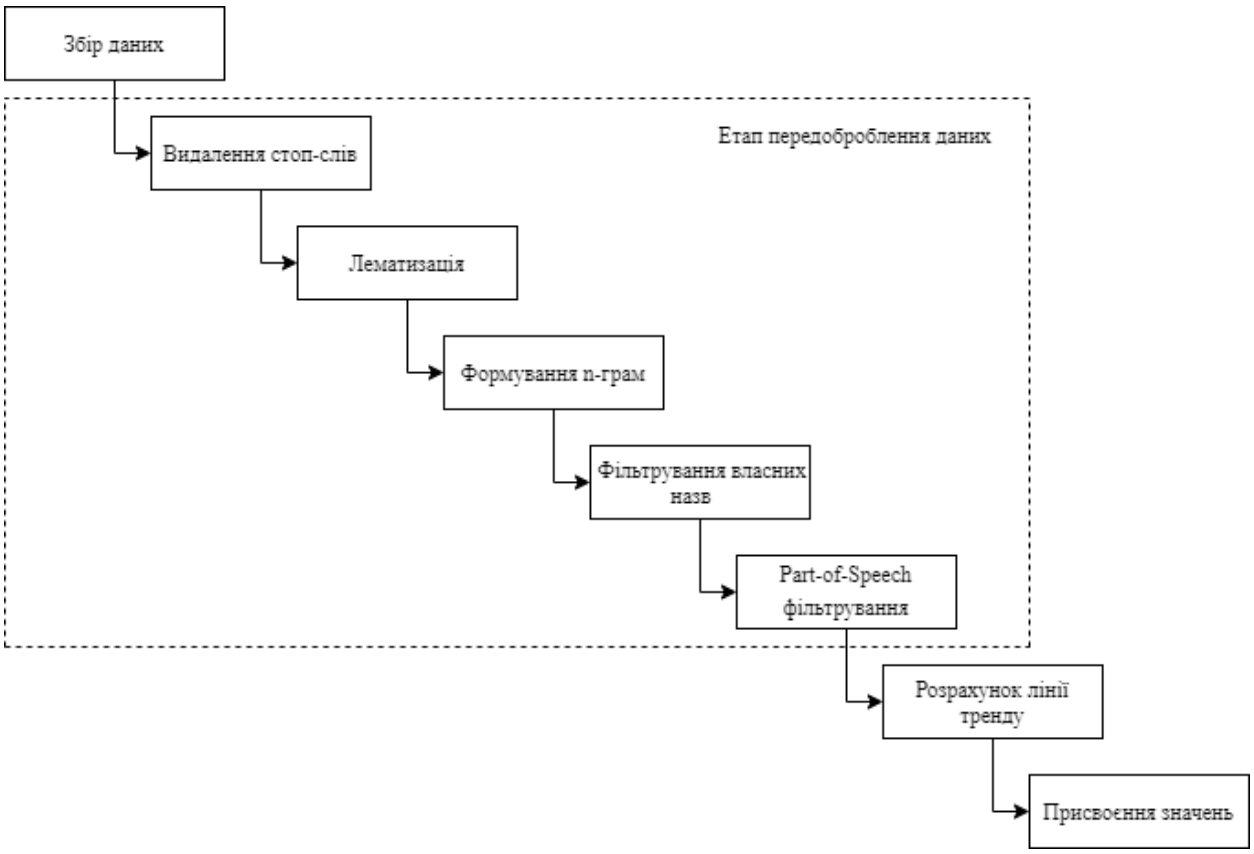


Рис. 2.3. Послідовність виконання кроків алгоритму

2.4. Висновки до розділу 2

В результаті дослідження, що було зроблене в цьому розділі, специфіка описів вакансій, представлених в мережі, була детально вивчена і описана. Розглянуто існуючі механізми виявлення трендів в текстових даних, визначено їх головні недоліки, такі як неможливість врахування специфіки тестових даних описів вакансій, вразливість до шуму в описах, відсутність обробки багатослівних виразів. На основі проаналізованих характеристик існуючих рішень та наявних в них недоліків було сформовано перелік характеристик описів вакансій, що представлені в мережі, які потрібно врахувати під час виявлення трендів, а саме: наявність цифр та символів, які часто погіршують сприйняття опису вакансії; переважання слів в називному відмінку, що покращить словник; наявність власних назв, таких як

місцезнаходження чи імена людей, що може погіршити виявлення трендів; наявність частин мов, які не мають сенсу з точки зору виявлення тенденцій. Проаналізовано процес виявлення трендів з можливістю врахування в ньому специфіки описів вакансій і в результаті запропоновано модифікацію етапу передоброблення вхідних даних запропонованого базового методу виявлення трендів. Розроблено адаптований метод виявлення трендів в описах вакансій, що представлені в мережі, з можливістю врахування особливостей описів тестових даних цих вакансій.

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ТРЕНДІВ В ТЕКСТОВИХ ОГОЛОШЕННЯХ ПРО ВАКАНСІЇ

3.1. Аналіз засобів розроблення програмного забезпечення

В якості мови програмування був обраний Python, оскільки поставлена задача не пред’являє високих вимог до продуктивності. Python є високорівневою крос-платформною мовою програмування, яка підтримує декілька парадигм програмування, в тому числі об’єктно-орієнтоване програмування. Для багатьох людей мова програмування Python є дуже привабливою. З моменту своєї першої появи в 1991 році Python став однією з найпопулярніших інтерпретованих мов програмування, поряд з Perl, Ruby та іншими. Такі мови часто називають скриптовими мовами, оскільки їх можна використовувати для швидкого написання невеликих програм або скриптів для автоматизації інших завдань. Серед інтерпретованих мов Python створив велике та активне наукове співтовариство з обчислювальними технологіями та аналізом даних. Протягом останніх 10 років Python перейшов від найрізноманітнішої наукової обчислювальної мови "на свій страх і ризик" до однієї з найважливіших мов для вивчення даних, машинного навчання та загальної розробки програмного забезпечення в наукових колах та промисловості. В останні роки покращена підтримка Python для бібліотек (таких як pandas та scikit-learn) зробила його популярним вибором для завдань аналізу даних. У поєднанні із загальною силою Python для інженерії програмного забезпечення загального призначення, це відмінний варіант як основна мова для побудови додатків даних.

Перевагою Python у наукових обчисленнях є простота інтеграції коду C, C++ та FORTRAN. Більшість сучасних обчислювальних середовищ поділяють

подібний набір застарілих бібліотек FORTRAN і C для лінійної алгебри, оптимізації, інтеграції, швидких перетворень Фур'є та інших таких алгоритмів.

У багатьох організаціях прийнято досліджувати, прототипувати і перевіряти нові ідеї, використовуючи більш спеціалізовану обчислювальну мову, наприклад SAS або R, а потім пізніше перенести ці ідеї на частину більшої виробничої системи, написаної, скажімо, на Java, C# або C++. Люди все частіше виявляють, що Python є підходящою мовою не тільки для дослідження та складання прототипів, але й для побудови виробничих систем.

Хоча Python є прекрасним середовищем для створення багатьох видів аналітичних додатків та систем загального призначення, існує ряд застосувань, для яких Python може бути менш придатним.

Оскільки Python – інтерпретована мова програмування, загалом більшість кодів Python буде працювати значно повільніше, ніж код, написаний на мові компіляції, як Java або C++. Оскільки час програміста часто є більш цінним, ніж час процесора, багато хто із задоволенням робить цей компроміс. Однак у програмі з дуже низькою затримкою або вимогливими вимогами до використання ресурсів (наприклад, високочастотна система торгівлі) час програмування на мові нижчого (але також низької продуктивності) типу C++ для досягнення максимально можливої продуктивності може бути добре витрачений час [24].

Важливим критерієм при обранні мови програмування була наявність сторонніх бібліотек для роботи з даними та алгоритмами машинного навчання.

NumPy, скорочено від Numerical Python, вже давно є наріжним каменем чисельних обчислень у Python. Він надає структури даних, алгоритми та бібліотечний цукор, необхідний для більшості наукових застосувань, що включають числові дані в Python [25]. NumPy, серед іншого, містить:

1. Швидкий та ефективний багатовимірний об'єкт ndarray.

2. Функції для виконання елементарних обчислень з масивами або математичних операцій між масивами.
3. Інструменти для читання та запису наборів даних на основі масивів на диск.
4. Операції лінійної алгебри, перетворення Фур'є та генерація випадкових чисел.
5. Зрілий C API, що дозволяє розширенням Python та рідному коду C або C++ отримати доступ до структур даних NumPy та обчислювальних засобів.

Крім можливостей швидкої обробки масивів, які NumPy додає Python, одне з основних напрямків його використання в аналізі даних є контейнером для передачі даних між алгоритмами та бібліотеками. Для числових даних масиви NumPy ефективніші для зберігання та маніпулювання даними, ніж інші вбудовані структури даних Python. Також бібліотеки, написані мовою нижчого рівня, наприклад C або Fortran, можуть працювати над даними, що зберігаються в масиві NumPy, не копіюючи дані в якесь інше представлення пам'яті. Таким чином, багато інструментів обчислювальної техніки для Python або приймають масиви NumPy як первинну структуру даних, або ж спрямовані на безперебійну сумісність з NumPy.

Pandas надають структури даних високого рівня та функції, призначені для швидкої, легкої роботи зі структурованими або табличними даними [26]. З моменту появи в 2010 році це допомогло Python стати потужним і продуктивним середовищем для аналізу даних. Основними об’єктами в pandas є DataFrame, таблична структура, орієнтована на стовпці, структура даних з мітками рядків і стовпців, а також Series, одновимірний мічений масив. Pandas поєднує високопродуктивні, обчислювальні масиви ідеї NumPy з гнучкими можливостями маніпулювання даними електронних таблиць та реляційних баз даних (наприклад, SQL). Він надає складну функціональність індексації, щоб

полегшити її перетворення, виконувати агрегацію та вибирати підмножини даних. Оскільки маніпулювання, підготовка та очищення даних є такою важливою майстерністю в аналізі даних, pandas є важливою перевагою у виборі Python як мови для роботи з даними саме через наявність:

1. Структури даних з міченими осями, що підтримують автоматичне або явне вирівнювання даних – це запобігає поширеним помилкам, що виникають внаслідок несогласованних даних та роботи з різними індексованими даними, що надходять з різних джерел.
2. Інтегрована функціональність часових рядів.
3. Одні і ті ж структури даних обробляють як дані часових рядів, так і дані не часових рядів.
4. Арифметичні операції та скорочення, що зберігають метадані.
5. Гнучка робота з відсутніми даними.
6. Об'єднання та інші реляційні операції, знайдені в популярних базах даних (наприклад, на основі SQL).

Matplotlib – найпопулярніша бібліотека Python для створення графіків та інших візуалізацій двовимірних даних [27]. Він призначений для створення графіків, придатних для публікації. Хоча існують інші бібліотеки візуалізації, доступні програмістам Python, matplotlib є найбільш широко використовуваним і як такий має загалом хорошу інтеграцію з рештою екосистеми.

IPython – це вдосконалена оболонка Python, призначена для прискорення написання, тестування та налагодження коду Python. Систему IPython можна використовувати через Jupyter Notebook, інтерактивний веб-орієнтований «записник» для коду, який пропонує підтримку десятків мов програмування. Оболонка IPython та Jupyter Notebook особливо корисні для дослідження та візуалізації даних. Система Jupyter notebook також дозволяє створювати вміст у Markdown та HTML, надаючи засоби для створення

багатих документів із кодом та текстом. Інші мови програмування також реалізували ядра для Jupyter, щоб дозволити використовувати інші мови, ніж Python в Jupyter [28].

SciPy – це сукупність пакетів, що розглядають ряд різних стандартних проблемних областей наукових обчислень. Приклад пакетів, що включені:

- scipy.integrate – числові інтеграційні підпрограми та розв'язки диференційних рівнянь;
- scipy.linalg – підпрограми лінійної алгебри та матричні декомпозиції, що виходять за рамки передбачених в numpy.linalg;
- scipy.optimize – оптимізатори функцій (мінімізатори) та алгоритми знаходження кореня;
- scipy.signal – інструменти обробки сигналів;
- scipy.sparse – розріджені матриці та розв’язувачі розріджених лінійних систем ;
- scipy.stats – стандартні безперервні та дискретні розподіли ймовірностей (функції щільності, вибірки, функції безперервного розподілу), різні статистичні тести та більше описової статистики.

Разом NumPy та SciPy утворюють достатньо повний та зрілий обчислювальний фундамент для багатьох традиційних наукових обчислень.

Scikit-learn, який став головним інструментом загального цільового машинного навчання для програмістів Python. Він включає підмодулі для таких моделей, як:

1. Класифікація: SVM, найближчі сусіди, випадковий ліс, логістична регресія тощо.
2. Регресія: Лассо, регресія хребта тощо.
3. Кластеризація: k-means, спектральна кластеризація тощо.

4. Зменшення розмірності: PCA, вибір ознак, матрична факторизація тощо.
5. Вибір моделі: пошук сітки, перехресне підтвердження, метрики.
6. Попередня обробка: вилучення ознак, нормалізація.

Разом з pandas, statsmodels та IPython, scikit-learn має вирішальне значення для того, щоб Python був продуктивною мовою програмування наукових даних [29].

Statsmodels – це пакет статистичного аналізу. У порівнянні з scikit-learn, statsmodels містять алгоритми класичної (насамперед частістської) статистики та економетрики. Сюди входять такі підмодулі, як:

1. Регресійні моделі: лінійна регресія, узагальнені лінійні моделі, стійкі лінійні моделі, лінійні моделі змішаних ефектів тощо.
2. Аналіз дисперсії (ANOVA).
3. Аналіз часових рядів: AR, ARMA, ARIMA, VAR та інші моделі.
4. Непараметричні методи: оцінка щільності ядра, регресія ядра.
5. Візуалізація результатів статистичних моделей statsmodels більш орієнтована на статистичні умовиводи, забезпечуючи оцінки невизначеності та *p*-значень параметрів. Тоді як scikit-learn є більш орієнтованим на прогнози.

3.2. Архітектура розробленого програмного забезпечення

Розроблену систему можна розділити на вісім модулів, зображених на рис. 3.1:

1. Модуль вилучення оголошень про вакансії – надає можливість автоматично програмними засобами мови Python вилучити з мережі Інтернет відкриті дані про наявні вакансії. В Інтернеті є мільйони API, які надають доступ до даних. API або інтерфейс програмування прикладних програм – це сервер, який можна

використовувати для отримання та відправлення даних за допомогою коду. Щоб використовувати API, необхідно зробити запит на віддалений веб-сервер і отримати потрібні дані. Використання API надає більше переваг аніж використання статичного набору даних CSV, адже дані швидко змінюються, немає сенсу регенерувати набір даних та завантажувати його щохвилини – це займе багато пропускної здатності та буде досить повільним процесом. Протокол передачі гіпертексту (HTTP) визначається як протокол програми, який використовується розподіленими інформаційними системами для зв'язку у всесвітній павутині. Python безперебійно обробляє всі HTTP запити та інтеграцію з веб-сервісами. Коли користувач використовує мову Python, немає необхідності вручну додавати рядки запитів до URL-адрес або формувати кодування POST-даних. JSON – мова API. JSON – це спосіб кодування структур даних, що забезпечує їх легкість для читання на машинах. JSON – це основний формат, в якому дані передаються в API, і більшість серверів API надсилатиме свої відповіді у форматі JSON, що є дуже зручно для їх подальшої обробки.

2. Модуль вилучення описів вакансій – надає можливість вилучити лише потрібні дані описів вакансій для їх подальшої обробки. Структура відповіді API серверів відрізняється в залежності від використовуваного API і зазвичай містить дуже багато різноманітної інформації починаючи від статусу запиту і закінчуючи списком результатів. На вхід даного модуля подається JSON-відповідь від API сервера, і результатом роботи модуля є перелік описів вакансії з міткою дати створення у форматі CSV.

3. Модуль загальної передобробки даних – забезпечує перші кроки передобробки даних, які включають:
   - Визначення мови вхідного тексту.
   - Токенізація.
   - Лематизація.
   - Видалення стоп-слів в залежності від визначеної мови.
   - Видалення рядків за такими шаблонами: hex-кольори, числа, рядки, що починаються з цифри.

   На вхід даного модуля подається повний опис вакансії, результат роботи – набір токенів для відповідної вакансії.
4. Модуль формування *n*-грам – забезпечує наповнення словника можливих трендових слів виразами довжиною більше ніж одне слово. Модуль реалізовує алгоритм GSP для збору всіх *n*-грам із заданим порогом для кожного опису. В результаті отримуємо розширений словник токенів для кожного опису вакансії.
5. Модуль фільтрації даних – забезпечує фільтрування даних за трьома аспектами:
   - Перехресне доменне фільтрування – забезпечує видалення слів і виразів, які не є специфічними для конкретної категорії вакансій, тобто таких, які з’являються в декількох категоріях з подібною частотою.
   - Фільтрування частин мов – забезпечує видалення частин мов, які не несуть важливого смислового навантаження в рамках задачі виявлення трендів в описах вакансій, а саме всіх, крім іменників, прикметників та виразів, для яких неможливо визначити частину мову.
   - Фільтрування власних назв – забезпечує видалення слів на

позначення місцезнаходження, осіб та організацій.

6. Модуль формування часових рядів – надає можливість сформувати в рамках кожного набору описів вакансій для кожного можливого трендового слова або виразу лінію тренду, беручи до уваги вагу показника TFIDF у певний проміжок часу для цього слова або виразу. Тобто робота модуля полягає у розбитті набору описів на часові проміжки згідно даті створення вакансії та підрахунку показника TFIDF для кожного трендового слова в рамках часового проміжку. В результаті роботи даного модуля отримаємо набір часових рядів зміни показника вагомості можливих трендових слів.
7. Модуль виявлення трендів – надає можливість аналізу отриманих часових рядів. Модуль реалізує обчислення показників нахилу та перехоплення простої лінійної регресії для кожного терміну з описів вакансій.
8. Модуль прийняття рішень – надає можливість зробити висновки чи є слово новим трендом, чи таким, що набирає популярність, чи таким, що її втрачає. Визначений нахил дає інформацію про те, збільшується чи зменшується тенденція з часом, а визначене перехоплення – про те, наскільки швидко вона це робить. Значення наведені у табл. 3.1.

Таблиця 3.1

Значення нахилу та перехоплення

	Нахил	Перехоплення
Новий тренд	Позитивний	Негативний
Набирає популярність	Позитивний	Позитивний
Втрачає популярність	Негативний	Позитивний

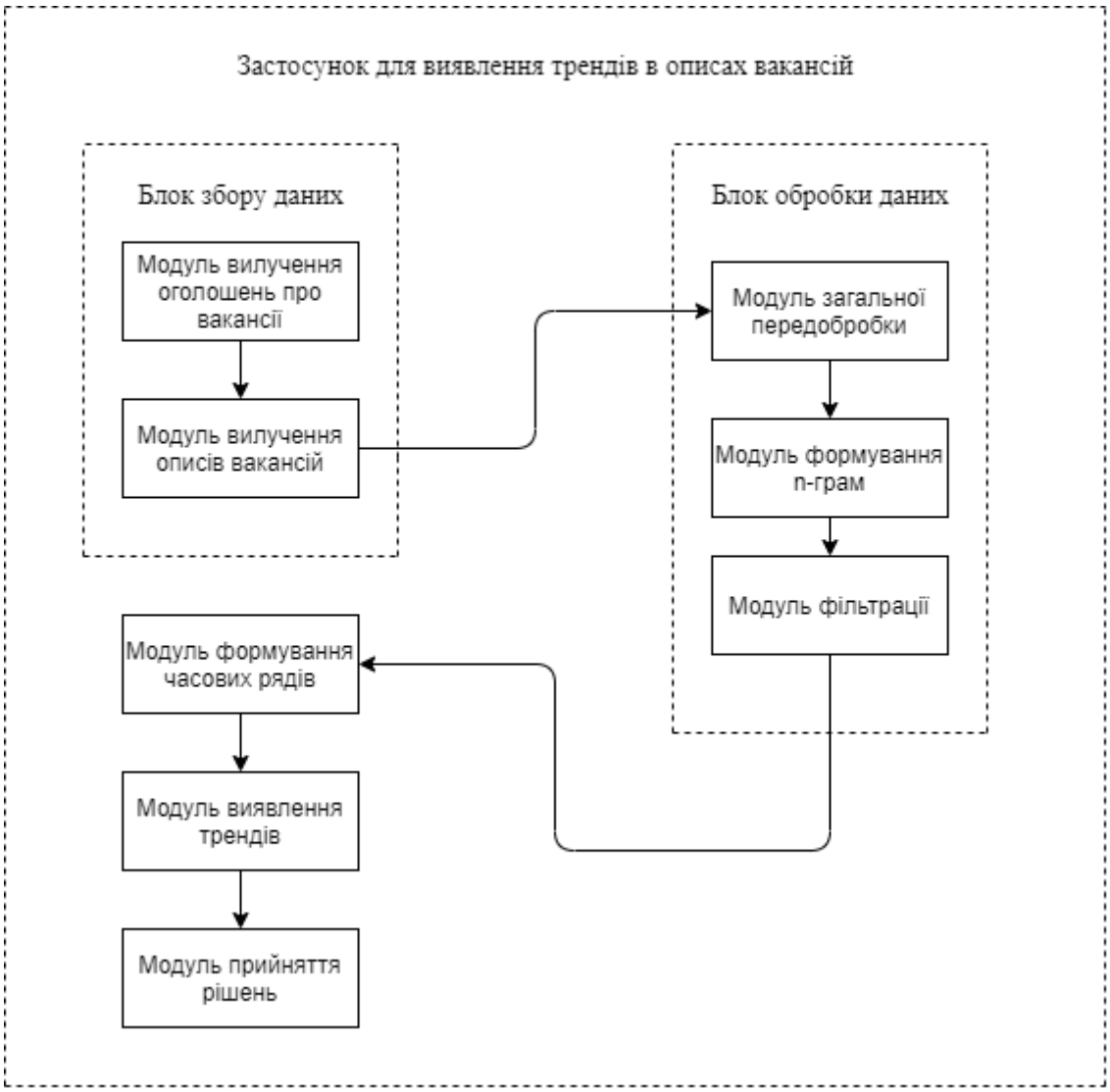


Рис. 3.1. Схема модулів програми

3.3. Особливості реалізації збору даних

Для дослідження та аналізу даних, в першу чергу, необхідно вибрати відповідний до поставленої задачі набір даних (DataSet). DataSet зазвичай являє собою файл з таблицею у форматі JSON або CSV. Різноманітні готові набори даних можна знайти на просторах мережі Інтернет, але нерідко вони не можуть задовільнити потребам поставленої задачі до аналізу даних. Іншим варіантом є створення власного набору даних, використовуючи програмні засоби. Для цього використовується веб-скрейпінг – технологія, яка

використовує скрипти для заходу на сайт під виглядом звичайного користувача і збирає інформацію по заздалегідь встановленим параметрам, таким чином, можна отримувати, обробляти, систематизувати і зберігати в звичайному текстовому форматі дані тисяч веб-сторінок за лічені хвилини – або більш простий спосіб – API, що надає доступ до даних. В якості джерела відкритих даних мережі Інтернет про вакансії було обрано API, яке надає Великобританський сайт adzuna.com [30]. Adzuna – це пошукова система оголошень про роботу, в якій зібрані оголошення про роботу з багатьох куточків світу. Сервіс здійснює пошук на тисячах веб-сайтів, щоб користувачам не довелося збирати мільйони оголошень, і вони могли знайти будь-яку роботу, скрізь, через Adzuna. Сервіс надає розумніші варіанти пошуку та потужні дані про ринок роботи.

Adzuna надає доступ до RESTful API. Наразі API складається з дев'яти кінцевих точок, які можна розділити на такі групи:

1) Отримати оголошення про вакансії. Кінцеві точки пошуку дозволяють направляти запити до бази оголошень про роботу.
2) Отримати дані про зайнятість. Чотири кінцеві точки розкривають останні тенденції щодо зарплат та вакансій. Історичні дані про зарплату показують, як із часом змінюються зарплати. Гістограма повертає розподіл поточної зарплати. А регіональні дані та кінцеві точки кращих компаній показують поточну кількість вакансій за регіонами чи компаніями.
3) Категорії. Надає інформацію про категорії, які Adzuna застосовує до вакансій про роботу.
4) Версія. Надає інформацію про версію API, до якої здійснюється запит.

Коренева URL-адреса API знаходиться за адресою:

```
https://api.adzuna.com/v1/api
```

Далі необхідно додати кінцеву точку, до якої треба здійснити запит, наприклад, щоб отримати списки вакансій у Великобританії:

```
https://api.adzuna.com/v1/api/jobs/gb/search/1
```

Також до запиту завжди потрібно додавати два обов'язкових параметра. app_id та app_key, завдяки чому кінцевий запит виглядає так:

```
https://api.adzuna.com/v1/api/jobs/gb/search/1?app_id={YOUR_APP_ID}&app_key={YOUR_APP_KEY}
```

Приклад відповіді наведений на рис. 3.2.

Рис. 3.2. Приклад відповіді від API Adzuna

Кожен раз, коли робиться запит, відповідь повертається як об'єкт, серіалізований за допомогою кодування JSON, JSONP або XML. Формат, що повертається, визначається заголовком Accept HTTP, але також може бути встановленим параметром рядка запиту `content-type`, який замінює заголовок Accept.

Якщо запит не може бути оброблений, буде повернуто код відповіді HTTP, відмінний від 200, і помилка буде серіалізована у запитуваному форматі серіалізації.

API надає широкі можливості для формування запитів про пошук вакансії. Усі параметри наведені у табл. 3.2.

Таблиця 3.2

Параметри запиту про пошук оголошень

Параметр	Значення	Опис
country	Gb, at, au, br, ca, de, fr, in, it, nl, nz, pl, ru, zg, us, za	ISO 8601 код країни, що цікавить
app_id	рядок	Ідентифікатор програми, наданий Adzuna
app_key	рядок	Ключ програми, наданий Adzuna
page	число	Номер сторінки
results_per_page	число	Кількість результатів для включення на сторінку результатів пошуку.
what	рядок	Ключові слова для пошуку. Кілька термінів можуть бути розділені пробілом.
what_and	рядок	Ключові слова для пошуку, усі ключові слова повинні бути знайдені.

Продовження табл. 3.2

what_phrase	рядок	Ціла фраза, яку потрібно знайти в описі чи назві.
what_or	рядок	Ключові слова для пошуку, будь-які ключові слова можуть бути знайдені. Кілька термінів можуть бути розділені пробілом.
what_exclude	рядок	Ключові слова, які потрібно виключити з пошуку. Кілька термінів можуть бути розділені пробілом.
title_only	рядок	Ключові слова, але лише для заголовку. Кілька термінів можуть бути розділені пробілом.
where	рядок	Географічний центр пошуку. Можуть використовуватись назви топонімів, поштові індекси тощо.
distance	число	Відстань у кілометрах від центру місця, описаного параметром "where". За замовчуванням – 5 км.
location0-7	рядок	Поля місцеположення можуть бути використані для опису місцеположення
max_days_old	число	Вік найдавнішого оголошення, що буде повернуте.
category	рядок	Тег категорії, що повертається кінцевою точкою "category".
sort_dir	up, down	Напрямок результатів пошуку.
sort_by	default, hybrid, date, salary, relevance	Впорядкування результатів пошуку

Продовження табл. 3.2

salary_min	число	Мінімальна зарплата
salary_max	число	Максимальна зарплата
salary_include_unknown	0, 1	Якщо "1", вакансії без відомої зарплати повертаються
full_time	0, 1	Якщо "1", будуть повернуті лише вакансії повного робочого дня.
part_time	0, 1	Якщо "1", будуть повернуті лише вакансії неповного робочого дня.
contract	0, 1	Якщо "1", будуть повернуті лише вакансії за контрактом.
permanent	0, 1	Якщо "1", будуть повернуті лише вакансії перманентні
company	рядок	Канонічна назва компанії

В якості мови вхідних текстів обрано 8 мов, на яких представлені вакансії для обраного API:

1) Данська (Нідерланди).
2) Англійська (Австралія, Канада, Індія, Нова Зеландія, Сінгапур, Великобританія, США).
3) Французька (Франція)
4) Німецька (Австрія, Німеччина).
5) Італійська (Італія).
6) Польська (Польща).
7) Португальська (Бразилія, Португалія).
8) Російська (Росія).

Також було обрано 6 категорій вакансій, для яких здійснюється пошук:

1) Консалтинг.

2) Інформаційні технології.

3) Бухгалтерія і фінанси.

4) Інженерно-технічна.

5) Зв'язки з громадськістю, реклама, маркетинг.

6) Наукові.

3.4. Особливості реалізації програмного застосунку

На основі наведених теоретичних засад та розробленого методу виявлення трендів в описах вакансій було розроблено програмний засіб автоматизованого виявлення трендів в описах вакансій. Програмне забезпечення реалізоване з використанням принципів ООП і має об’єктно-орієнтовану структуру. Об'єктно-орієнтована структура характерна тим, що в ній, на відміну від колишніх підходів, поняття процедур і даних замінені таким поняттям, як об'єкт. Об'єктно-орієнтовані системи можна розглядати як сукупність автономних і незалежних об'єктів. Зміна реалізації якого-небудь об'єкта або додавання йому нових функцій не впливає на інші об'єкти системи. Об'єкти можуть бути повторно використовуваними компонентами, вони незалежно інкапсулюють дані про стан та операції. Це знижує вартість проєктування, програмування і тестування ПЗ.

Для реалізації кожного модуля системи було розроблено відповідний клас або декілька класів за необхідності.

Блок збору даних було реалізовано за допомогою двох класів JobAdRetriever та JobDescriptionFetcher.

Структура класів JobAdRetriever та JobDescriptionFetcher зображена на рис. 3.3. Клас JobAdRetriever реалізує два методи:

1) Request_adzuna() – цикл над вибраними країнами та категоріями, запитуючи всі можливі сторінки на Adzuna. Усі можливі параметри визначені нижче:

param countries: список рядків, країни, над якими петлі пошуку

param categories: список рядків, категорії, над якими петлі пошуку

return: Немає, результати друкуються у файлах json та зберігаються у папку “Raw data”, а журнали прогресу в консолі.

2) Request_page() – запитує сторінку від Adzuna, використовуючи додані Adzuna APP_ID та APP_KEY. Усі можливі параметри визначені нижче:

param country: string, країна, за якою шукає Adzuna

param category: рядок, категорія завдань

param page: int, номер сторінки

return: 0, якщо результатів не знайдено, 1, якщо вони знайдені та файл друкується для збереження результатів пошуку

Клас JobDescriptionFetcher реалізує метод fetch_description() – витягує описи, якщо вони доступні і зберігає їх у папку "Raw text".

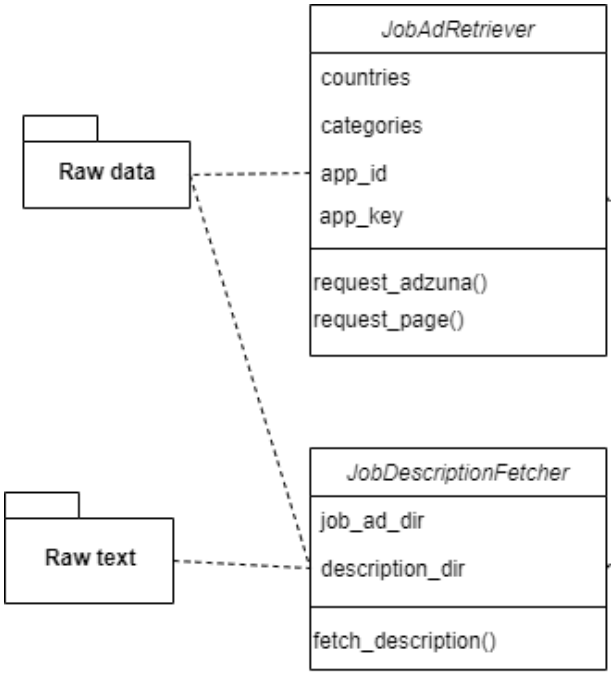


Рис. 3.3. Класи JobAdRetriever та JobDescriptionFetcher

Блок обробки даних реалізований за допомогою чотирьох класів Preprocessor, TreeTagger, SequenceMining, TrendDetectionPipeline. Їх структура зображена на рис. 3.4.

Клас Preprocessor реалізує методи для попередньої обробки даних описів вакансій.

1) Get_entries() – отримує описи з файлу.
2) Which_language() – повертає назву мови.
3) Tokenize() – токенізотор.
4) Matches() – перевіряє, чи знайдено шаблон в рядку.
5) Lematize() – лематизатор.
6) Process_tokens() – попередньо обробляє токени для даної мови.

Клас TreeTagger виконує тегування частин мов. Він наслідується від інтерфейсу TreeTaggerI, що доступний у пакеті nltk.tag.api.

Клас SequenceMining реалізує кроки GSP-алгоритму для формування n-грам.

Клас TrendDetectionPipeline використовуючи попередньо описані класи реалізує методи для повної обробки описів, після чого зберігає їх у папку "Preprocessed Text".

Сама детекція трендів відбувається за допомогою класів TimeSeries та TrendDetection. Їх структура наведена на рис. 3.5.

Метод make_timeseries() класу TimeSeries розбиває оброблені дані на основі встановленого часового проміжку та обчислює показник tf-idf в рамках кожного проміжку за допомогою методу get_tf_idf(). Потім зберігає отримані часові ряди у папку “Time Series”.

Клас TrendDetection має два основних методи calculate_slope() та calculate_intercent() для обчислення нахилу та перехоплення часового ряду.

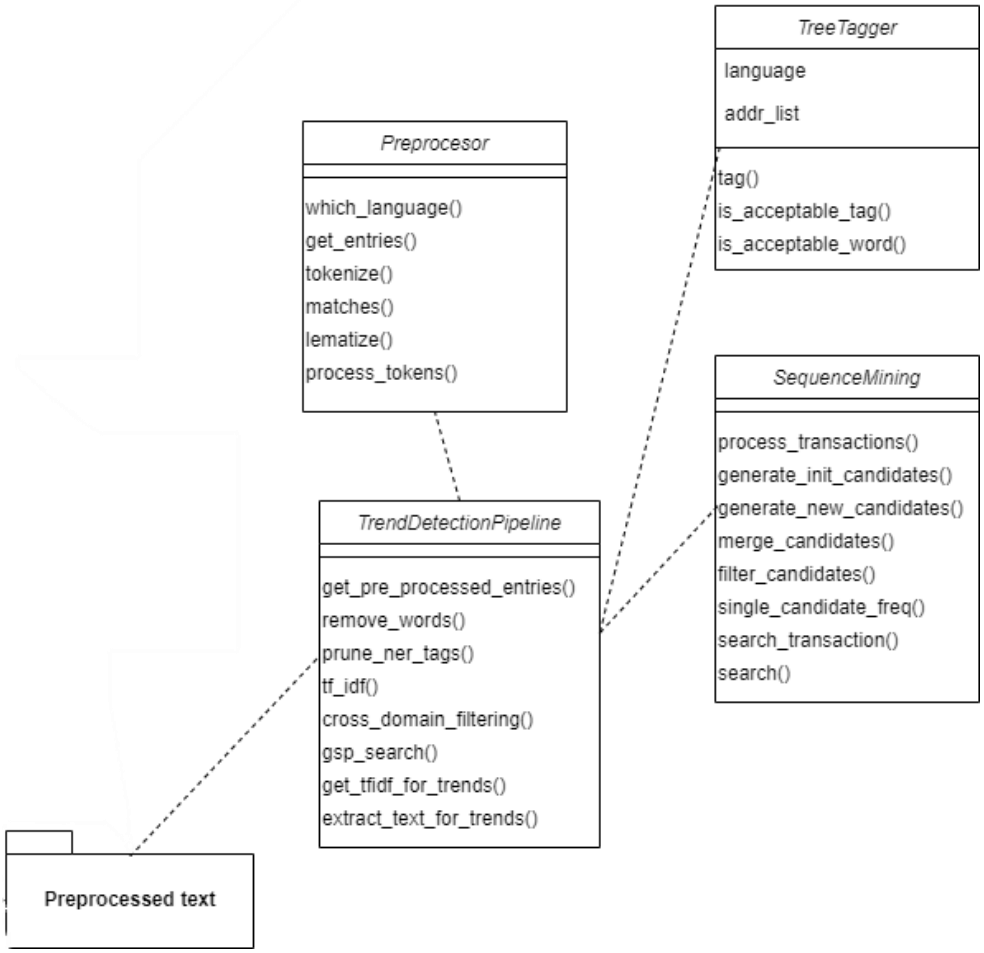


Рис. 3.4. Діаграма класів Preprocessor, TreeTagger, SequenceMining, TrendDetectionPipeline

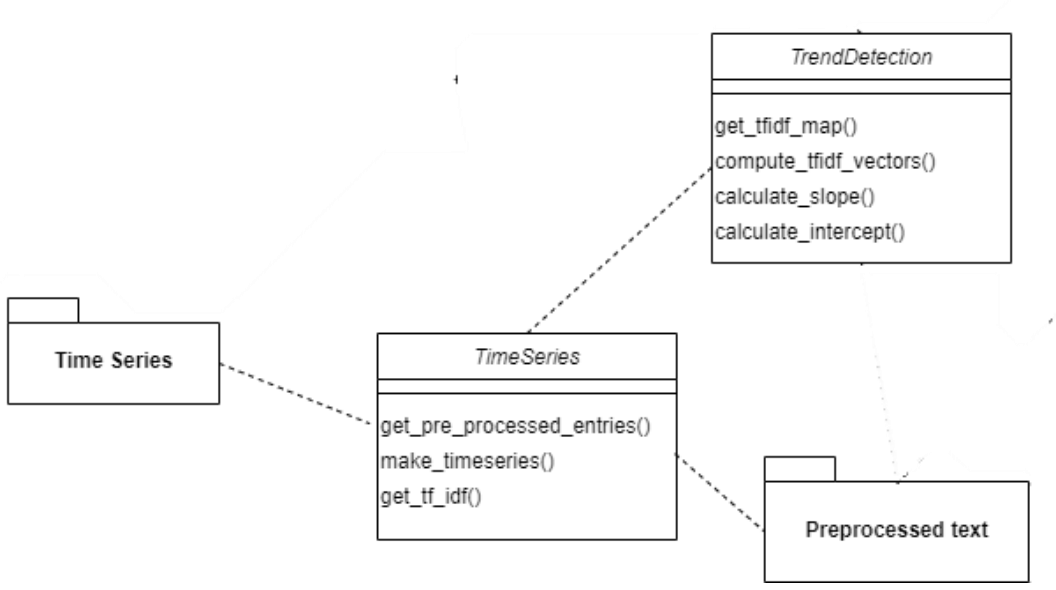


Рис. 3.5. Класи TimeSeries та TrendDetection

Загальна діаграма класів зображена на рис. 3.6.

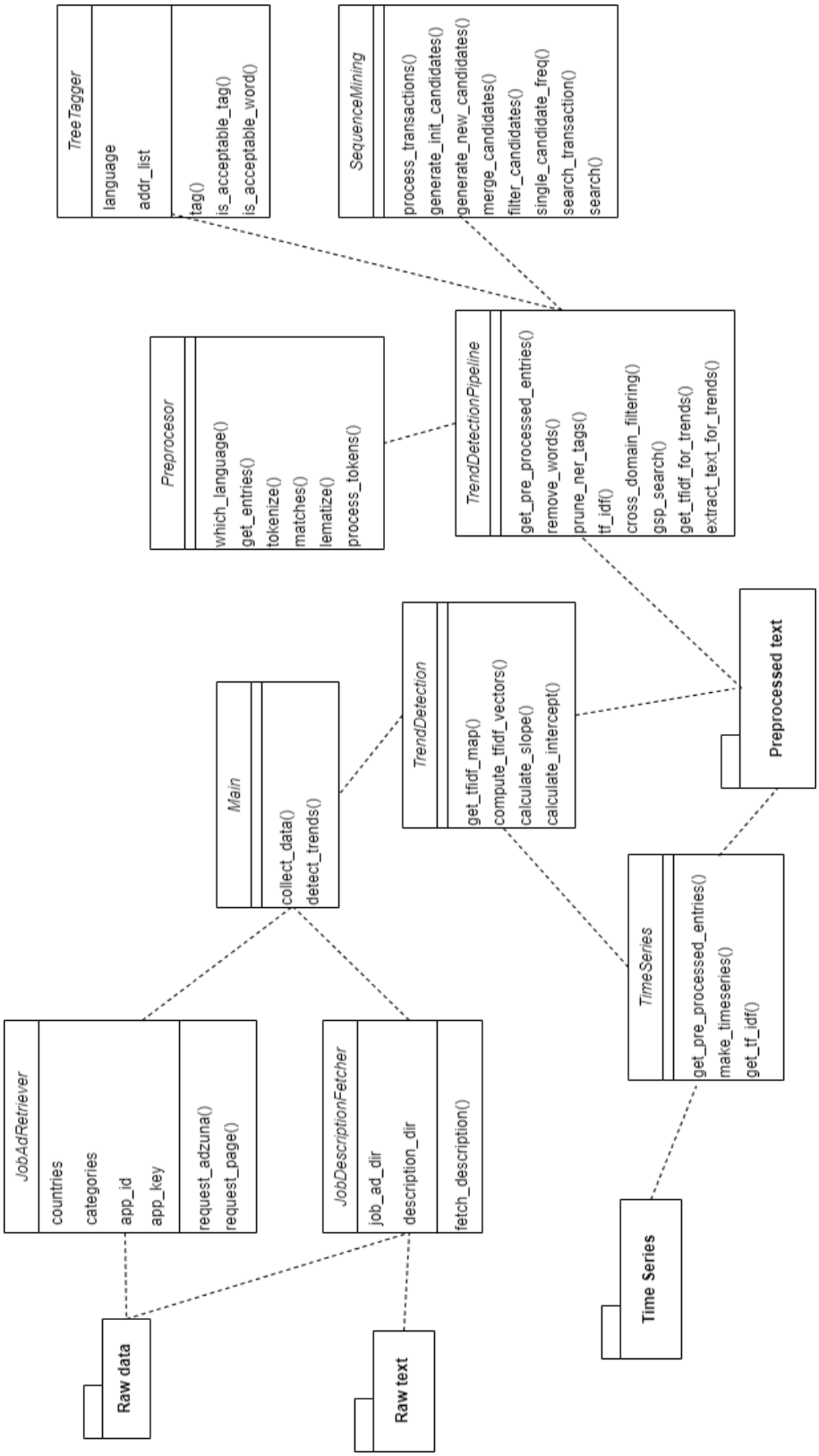


Рис. 3.6. Загальна діаграма класів

3.5. Висновки до розділу 3

У розділі наведено особливості процесу збору даних для поставленої задачі, особливості реалізації програмних блоків та модулів.

Розроблене програмне забезпечення призначене для автоматизованого виявлення трендів в описах вакансій на основі запропонованого в роботі адаптованого методу автоматизованого виявлення трендів в текстових оголошеннях про вакансії.

Розроблена система надає можливості автоматично отримати актуальні оголошення про вакансії та аналізувати їх для виявлення трендових виразів та слів.

Завдяки об’єктно-орієнтованому підходу до створення архітектури програмного забезпечення можна з легкістю додавати реалізацію інших методів обробки даних та виявлення трендів, а також змінювати, видаляти чи додавати певні кроки до вже реалізованого методу.

Реалізована програмна система може бути також з легкістю інтегрована в інші системи або працювати з будь-яким іншим програмним забезпеченням, як його частина.

4. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Сфера застосування

Для тестування розробленого методу були зібрані дані вакансій за останній 2019 рік з веб-сервісу Adzuna, який надає доступ через своє API до даних про вакансії з багатьох куточків світу. Усі отримані дані були розбиті часовим проміжком у 3 місяці, тобто було отримано 4 набори даних для кожної категорії вакансій з усіх країн за такі періоди:

1. Січень-Березень.
2. Квітень-Червень.
3. Липень-Вересень.
4. Жовтень-Грудень.

У зібраних даних вакансій не всі категорії у всіх країнах мали дані ще з самого початку 2019 року. Тому на рис. 4.1 та рис. 4.2 представлені лише ті випадки, для яких дані були доступні.

Загалом було зібрано дані для 8 країн: Австралія, Франція, Німеччина, Нідерланди, Польща, Росія, Південна Африка та Велика Британія.

Для більш давнього періоду було доступно досить мало вакансій, тоді як для недавнього достатньо.

Загалом зібрано понад 500 тисяч вакансій різних сфер діяльності. Найбільше наявних вакансій із категорії “Інформаційні технології” майже для кожної країни.

Загалом розмір словника для певної країни і для певної категорії вакансій коливається від 100 до 1000 слів. Безумовно цей розмір залежить напряму від кількості доступних вакансій для цієї категорії.

Країна	Категорія	Січень-Березень		Квітень-Червень	
		Розмір словника	К-сть вакансій	Розмір словника	К-сть вакансій
Австралія	Бухгалтерія і фінанси	39	16	56	20
	Інженерно-технічна	47	11	64	24
Франція	Інженерно-технічна	195	156	229	266
Німеччина	Консалтинг	250	53	420	77
	Інженерно-технічна	397	342	811	685
	Інформаційні технології	502	623	782	1277
Нідерланди	Бухгалтерія і фінанси	256	112	680	431
	Інженерно-технічна	316	98	582	347
Польща	Бухгалтерія і фінанси	15	539	62	1009
	Інженерно-технічна	93	57	154	68
	Інформаційні технології	86	1039	136	1078
Росія	Бухгалтерія і фінанси	254	154	481	343
	Інженерно-технічна	59	28	107	49
	Інформаційні технології	121	99	176	126
Південна Африка	Бухгалтерія і фінанси	63	113	87	196
	Інженерно-технічна	65	298	179	533
	Інформаційні технології	205	234	347	438
Велика Британія	Консалтинг	68	13	149	18
	Інженерно-технічна	251	372	389	574
	Інформаційні технології	91	58	210	105

Рис. 4.1. Статистика зібраних вакансій за період Січень-Червень

Країна	Категорія	Липень-Вересень		Жовтень-Грудень		Загальний розмір словника
		Розмір словника	К-сть вакансій	Розмір словника	К-сть вакансій	
Австралія	Бухгалтерія і	2154	2655	4143	6612	69
	Інженерно-	1997	1071	3102	1990	228
Франція	Інженерно-	3762	5129	6152	8091	459
Німеччина	Консалтинг	2810	10991	7294	17630	101
	Інженерно-	6129	7363	12819	12901	804
	Інформаційні	10871	13758	13267	21722	1003
Нідерланди	Бухгалтерія і	3709	2798	5182	4711	690
	Інженерно-	9162	12981	15825	20712	716
Польща	Бухгалтерія і	9733	4892	13821	8702	33
	Інженерно-	7956	5119	10628	9093	59
	Інформаційні	5876	19734	10365	28710	151
Росія	Бухгалтерія і	3549	11983	5213	13270	548
	Інженерно-	7650	32812	10054	51262	68
	Інформаційні	11794	28117	17439	42356	254
Південна Африка	Бухгалтерія і	1859	3878	1995	4611	97
	Інженерно-	687	1892	1165	2061	145
	Інформаційні	2431	3532	5490	7650	425
Велика Британія	Консалтинг	2996	9675	3008	15081	138
	Інженерно-	8850	27579	10922	43591	402
	Інформаційні	5127	32123	10865	53118	217

Рис. 4.2. Статистика зібраних вакансій за період Липень-Грудень

4.2. Трендові слова

За допомогою описаної в попередньому розділі програмної реалізації розроблюваного методу виявлення трендів в описах вакансій отримано трендові слова для кожного з випадків, наведених у табл. 4.1. Проведемо аналіз результатів, зосередившись на описах вакансій у Великої Британії в категорії «Інформаційні технології».

Із 217 спільних слів для всіх чотирьох періодів часу 2019 року 7.8% (17) мали позитивний показник нахилу та негативний показник перехоплення для часового ряду показника TFIDF, і тому саме ці слова можна вважати трендовими для цієї категорії. Вони показані на рис. 4.3.

Трендові слова містять інформацію про фахові знання та навички, такі як мови програмування(Python, PHP, Java), абревіатури(AMP, що позначає Apache, MySQL і PHP; TDD, Test-Driven Development) або інші поняття в Computer Science (Cloud, analytics, troubleshoot).

На рис. 4.4 зображений результат кластеризації трендових слів в дендограмі.

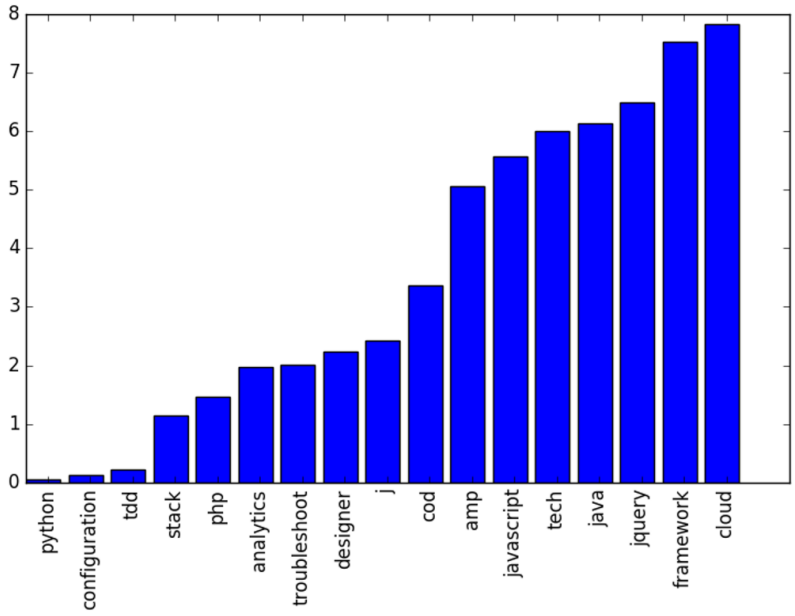


Рис. 4.3. Трендові слова в текстах оголошень у Великій Британії в категорії «Інформаційні технології за 2019 рік»

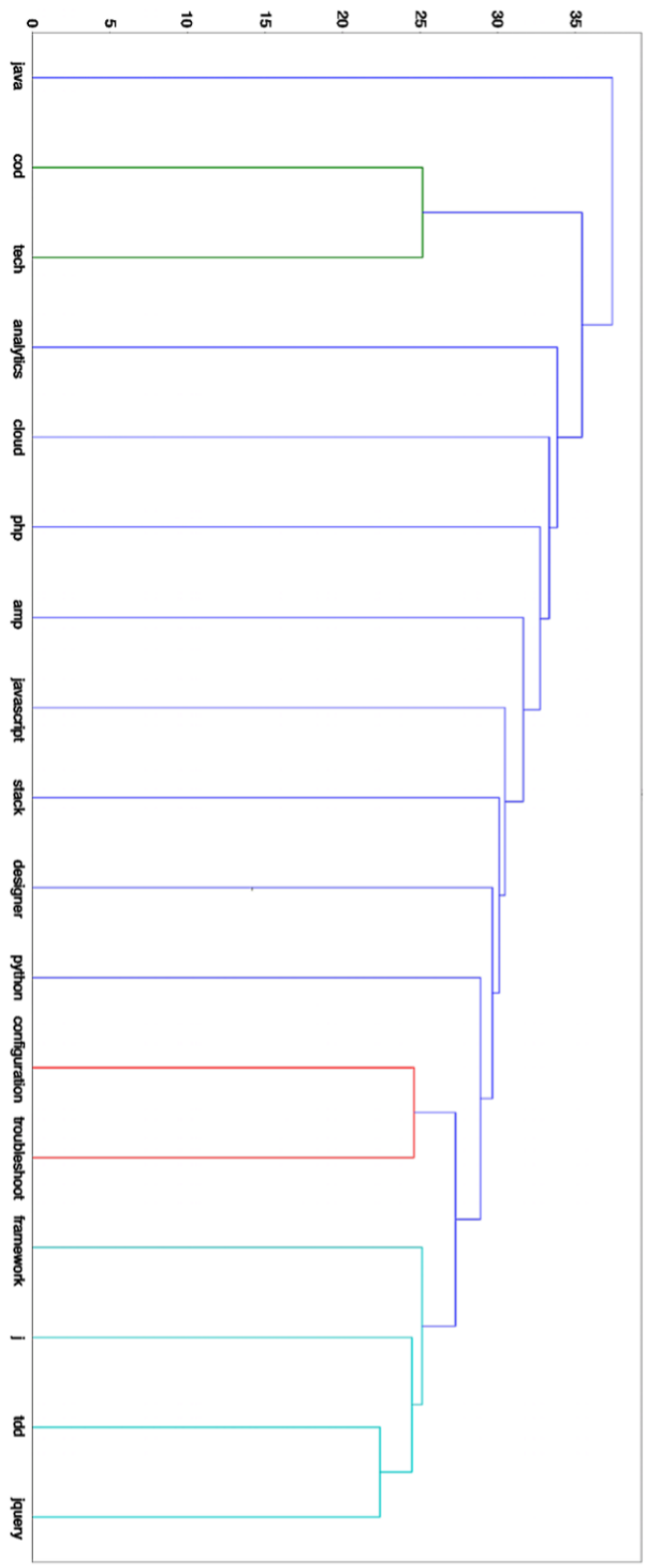


Рис. 4.4. Кластеризація трендових слів у текстах оголошень у Великій Британії в категорії «Інформаційні технології за 2019 рік»

Ця кластеризація на основі TFIDF охопила кілька пар слів, які мають подібні частотні схеми, як configuration і troubleshoot, або TDD та jQuery.

Існує значна частина зібраних слів, які не є трендовими, хоча за формулою TFIDF нормалізовано частоту термінів для кожного тексту, а також для довжини корпусу. Ймовірно, це викликано дисбалансом у кількості оголошень про роботу.

4.3. Розподіл трендових слів в описах вакансій

Маргуліс [31] досліджував статистичні дані щодо розташування слів у корпусах документів і ним було виявивлено в експериментах над великими колекціями документів, що припущення про те, що твердження "частота появи текстових токенів у повнотекстових документах у колекції документів може бути описана сумою розподілів Пуассона", справедливе для більше ніж 70% термінів, які часто зустрічаються в корпусі документів. Лі та Байк [32] визначають ключове слово у колекції документів як слово з високим балом TFIDF та високим стандартним відхиленням у розподілі його розташувань у текстах.

У нашому випадку позиція слова в його тексті не впливала на можливість того, що воно стане трендовим словом. Було обчислено кількість зустрічей трендових слів у описах вакансій в категорії «Інформаційні технології” Великої Британії 2019 року» та виявлено розподіл, показаний на рис. 4.5. На рис. 4.6 показана згладжена крива цього розподілу. У описах вакансій у Великої Британії за 2019 рік середня довжина тексту становила 112,6, тоді як стандартне відхилення 129,2. Тому для середнього оголошення про роботу, розподіл перекошений вліво, це означає, що тенденційне слово, ймовірно, з’явиться на початку оголошення про роботу.

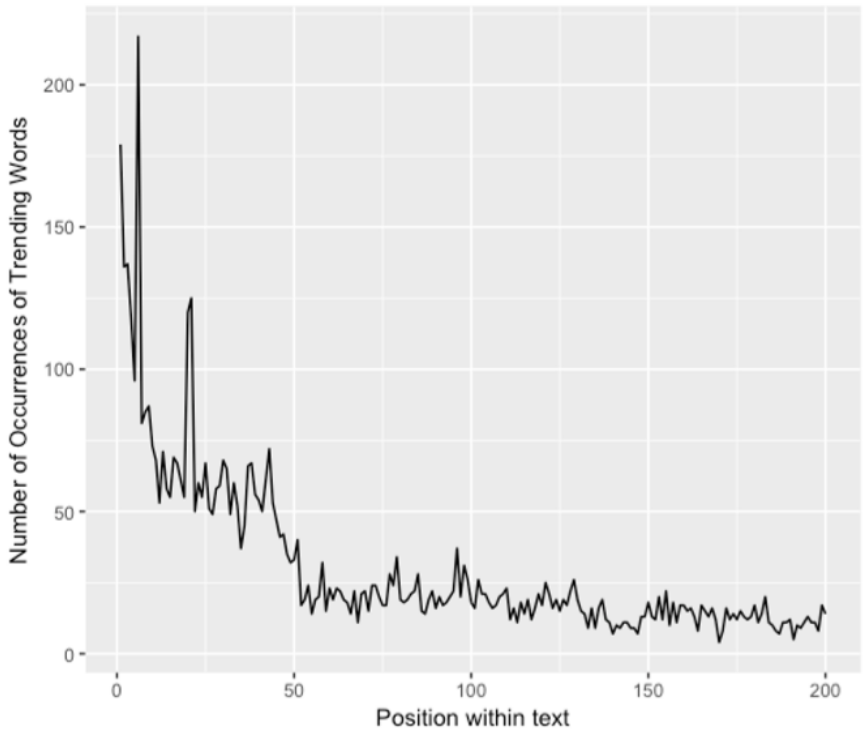


Рис. 4.5. Число появ трендових слів в текстах оголошень у Великої Британії в категорії «Інформаційні технології за 2019 рік»

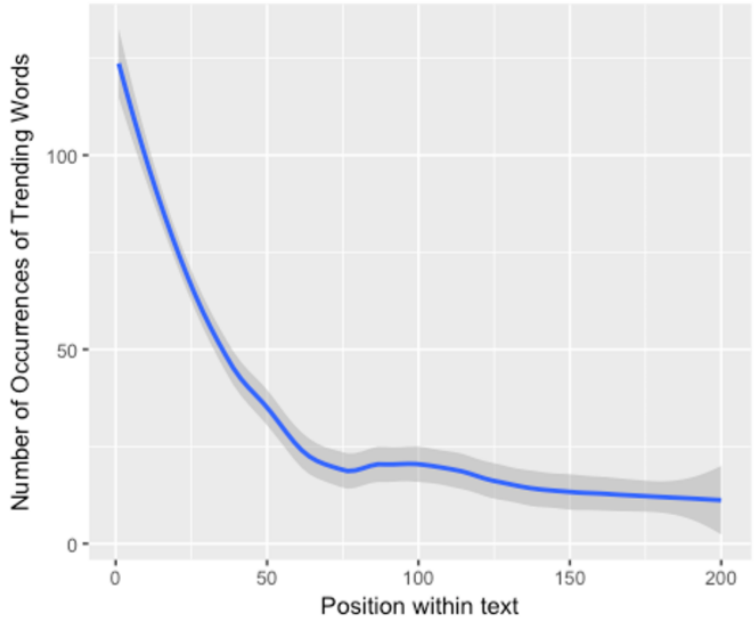


Рис. 4.6. Згладжена крива числа появ трендових слів в текстах оголошень у Великої Британії в категорії «Інформаційні технології за 2019 рік»

Такі ж результати є характерними і для інших 19 випадків, де доступні трендові слова. Отже, автором було перевірино, чи можна передбачити розподіл позиції трендових слів у оголошеннях про роботу певного випадку, знаючи розподіли в інших випадках. Спочатку було натреновано Generalized Additive Model (GAM) [33, 34] для кожного випадку, а потім обчислено Normalised Root Mean Square Error (NRMSE). Отримані результати наведені на рис. 4.7.

Прогноз дуже точний, оскільки середній показник NRMSE становить лише 3,3%, з медіаною 2,0% та стандартним відхиленням 3,5%. Він показує тісний взаємозв'язок, незалежно від країни, мови чи сфери роботи, між розташуванням слова в оголошенні про роботу та його трендовістю.

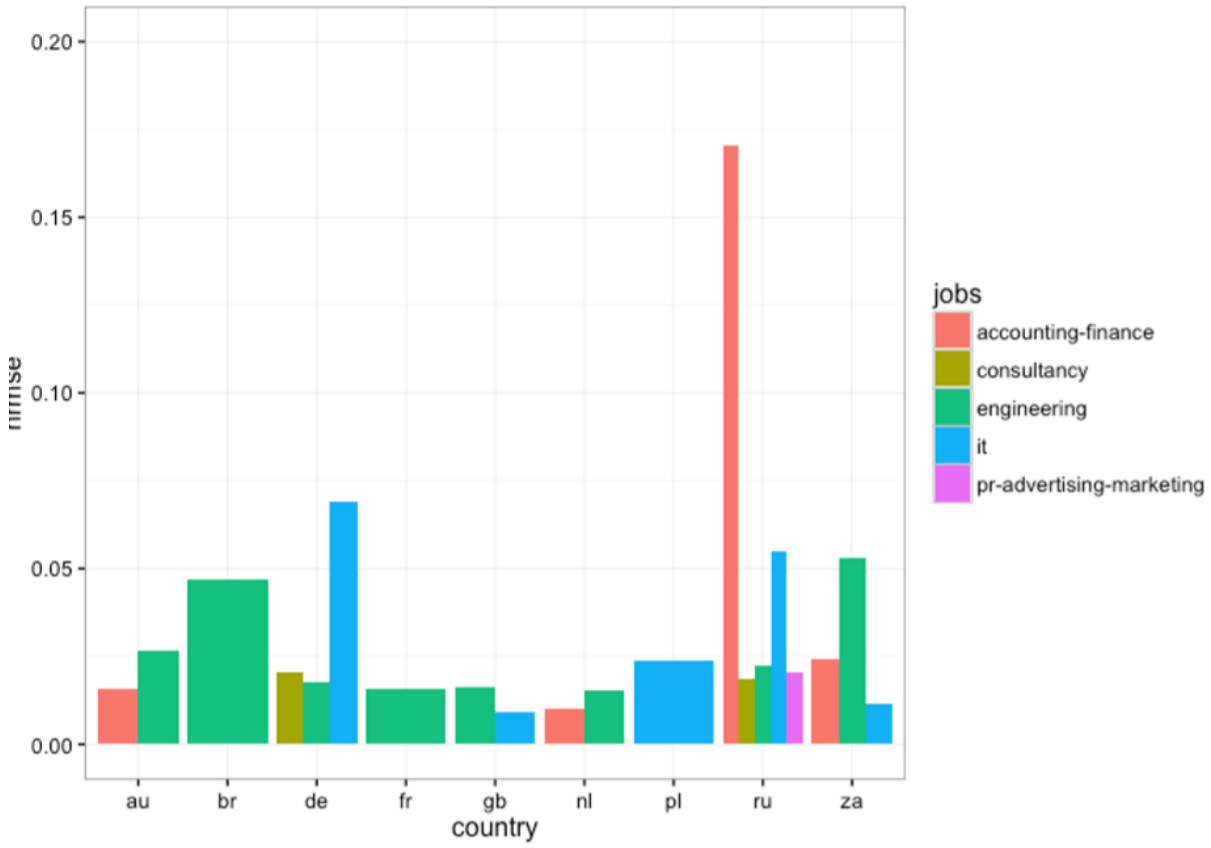


Рис. 4.7. GAM передбачення розподілу позицій трендових слів у випадках оголошень про роботу

4.4. Порівняння з Google Trends

Google Trends використовується для прогнозування тенденцій в інших сферах. В економічному відношенні, Carrière-Swallow та Labbé [35] з'ясували, що дані Google Trends покращують прогнозування поточного стану продажів автомобілів в Чилі до 14%. Чой та Варіан [36] створюють прості сезонні автоматичні регресивні моделі з відповідними даними Google Trends для випадків, включаючи планування впевненості споживачів та місця призначення подорожей, і вони випереджають подібні моделі без даних Google Trends на 5–20 відсотків. У галузі охорони здоров'я GoogleFluTrends може скористатися тим, що мільйони користувачів у всьому світі шукатимуть інформацію про хворобу, яка їх хвилює, і тому вона може виявити регіональні спалахи грипу до 10 днів швидше ніж загальноосвітні системи спостереження для контролю та профілактики захворювань [37].

Для кожного випадку з трендовими словами було зроблено запит Google Trends для заданого трендового слова *w* для відповідної країни протягом періоду з січня 2019 року по грудень 2019 року. Google Trends не дає абсолютних даних про кількість пошукових запитів, але дає відносні дані, максимум яких нормалізовано до 100, а 0 означає, що пошукового запиту не було. Ми обчислюємо тенденцію енергії слова w в країні c і за набір місяців M за формулою (5). Від'ємне значення вказує на зменшення популярності, а позитивне значення – на її збільшення.

$$energy(v) = \sum_{m=1}^{|M|-1} (v_{|M|}{}^2 - v_m{}^2) * \frac{1}{|M| - i} \tag{5}$$

Результати трендових слів у IT-роботах Великої Британії зображені на рис. 4.8. З рисунку видно, що результати отримані за допомогою Google

Trends суперечать тим результатам, які були отримані в цьому дослідженні.

Варто відзначити, що отримані в досліджені результати визначені для конкретного домену, тоді як розрізнення сенсу слова не може бути виконано в пошуковому запиті google для полісемічних трендових слів, таких як python, stack, amp та framework. Більше того, хоча енергії тенденції, отримані за допомогою Google Trends, є репрезентативними запитами широкої аудиторії, наші тенденції безпосередньо випливають з оголошень на ринку праці і, таким чином, дають уявлення про попит на ринку саме для IT та в більш широкому розумінні, що стосуються конкретних доменних потреб на ринку праці.

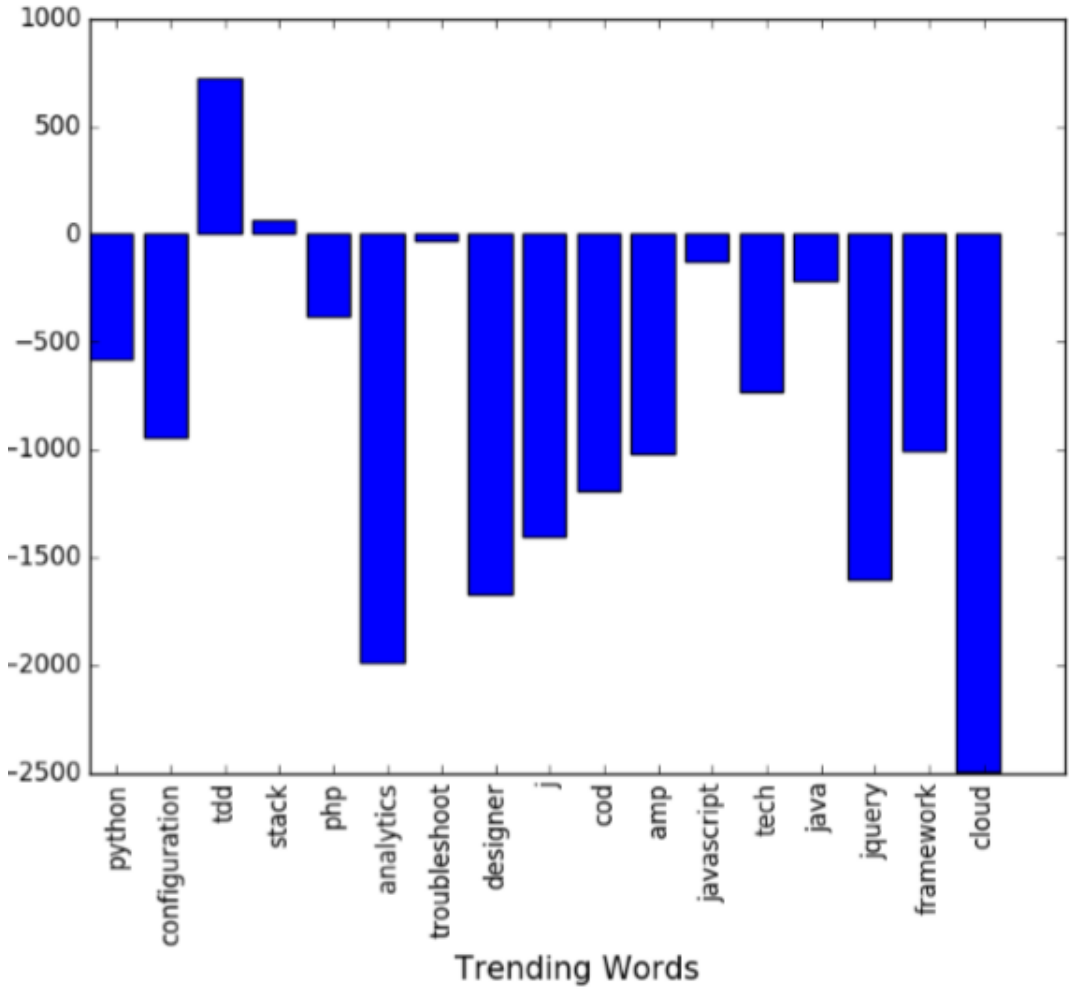


Рис. 4.8. Трендові слова в оголошеннях про роботу в категорії «Інформаційні технології у Великій Британії» з їх енергією, обчисленою Google Trends за 2019 рік

4.5. Висновки до розділу 4

У цьому розділі було проаналізовано результати роботи запропонованої програмної реалізації адаптованого метода виявлення тенденцій в описах вакансій. По-перше, було зібрано оголошення про вакансії з 16 країн та у 6 сферах роботи. Ці оголошення про роботу були 8 мовами. Було застосовано запропонований метод у випадках, коли були доступні дані та обчислено еволюцію ваг TFIDF за 2019 рік. Тенденційне слово визначається як слово, що з’являється у кожному визначеному періоді 2019 року та має позитивний нахил і негативне перехоплення показників лінійної регресії. Аналіз результатів був зосереджений на вакансіях у Великої Британії в сфері Інформаційні технології. Отримані трендові слова містили такі навички, як мови програмування та засоби і концепти Computer Science.

Кластеризацією трендових слів вдалось зафіксувати декілька змістовних зв’язків. Потім проаналізувавши розподіл позицій тенденційних слів у текстах оголошень про роботу було з’ясовано, що більшість розподілів зміщені вліво, що означає, що трендове слово, ймовірно, з’явиться на початку текстів оголошень про роботу.

Було також виявлено, що передбачення розподілу за допомогою GAM інших розподілів дає велику точність, що становить в середньому 3,3% NRMSE. Це вказує на тісний взаємозв'язок положення слова в його тексті та його трендовістю, незалежно від сфери роботи, мови чи країни.

Знаючи ефективність Google Trends у прогнозах в економіці та профілактиці захворювань, було досліджено, чи дають пошукові запити натовпу аналогічні результати запропонованому методу. Google Trends суперечить результатам, отриманим запропонованим методом, швидше за все, через те, що об'яви про роботу випливають із конкретних доменних вимог ринку праці, а не від загальної аудиторії.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1. Опис проблеми

На сьогоднішній день в мережі Інтернет генеруються величезні потоки даних, які збільшують зусилля як вчених так і розробників по забезпеченню неупереджених та переконливих свідоцтв думок та інтересів користувачів. Застосування соціальних медіа стало потужним засобом комунікації для людей, які прагнуть поділитися інформацією та обмінятися інформацією з широкого кола питань. Ці теми варіюються від популярних, широко відомих (наприклад, концерту популярного музичного гурту) до менших масштабів, місцевих (наприклад, місцевих громадських зборів, протестів або нещасних випадків), а їх популярність коливається з часом. Опубліковані користувачем повідомлення, розміщені на сайтах соціальних медіа, зазвичай відображають ці теми в їх фактичному вимірі, і тому вміст сайтів соціальних медіа особливо корисний для ідентифікації тенденцій у реальному часі. Виявлення тематичних тенденцій, безумовно, викликає значний інтерес насамперед через те, що тенденції:

a. Можуть використовуватися для виявлення поведінки підозрілої або тої, що виникла раптово в мережі.
b. Можуть розглядатися як відображення соціальних проблем або навіть як консенсус у прийнятті колективних рішень.

Одним із видом соціальних платформ є сайти для пошуку роботи. На сьогоднішній день існує велика кількість сайтів, що допомагають роботодавцям розміщувати об’яви про наявні вакансії, а шукачам роботи – їх знаходити. Це є особливим типом соціальних мереж, де контактуючими є безпосередньо роботодавці, які додають інформацію про вакансії, та претенденти, що додають свої резюме або ж відповідають на наявні вакансії. Такі платформи стали дуже популярними завдяки своїм можливостям

розповсюдження інформації в режимі реального часу. Такі ресурси містять величезну кількість даних про вакансії, проаналізувавши які, можна отримати інформацію про тенденції на ринку праці, корисну як для працівників, так і для роботодавців. Інформація про майбутні тенденції вважається цінним джерелом знань як для компаній, так і для окремих осіб.

Існує велика кількість маркетингових аналітиків, які працюють над моніторингом конкретної сфери бізнесу, і багато з них використовують ручні методи для цього. Великих грошових витрат вимагає утримання таких спеціалістів, адже в першу чергу їм потрібно виплачувати заробітні плати, можливо соціальні виплати і медичні страховки, забезпечити робочим місцем та інше. Інший важливий мінус такого методу – це людський фактор тому, що поняття тренду досить суб’єктивне і різні спеціалісти можуть розуміти однаковий вміст вакансій по різному, оцінювати рівень важливості інформації по різному та взагалі проводити процес виявлення тенденцій інакше, що може впливати на результати. І останнім, але не менш важливим недоліком є те, що обсяги доступних даних в Інтернеті є занадто великі для моніторингу людьми, це становить великий ризик пропуску значної кількості інформації. Саме через цю низку вразливостей виникла необхідність автоматичного виявлення нових тенденцій на ринку праці. Усі недоліки описані вище узагальнені на рис. 5.1 «Дерево проблем».

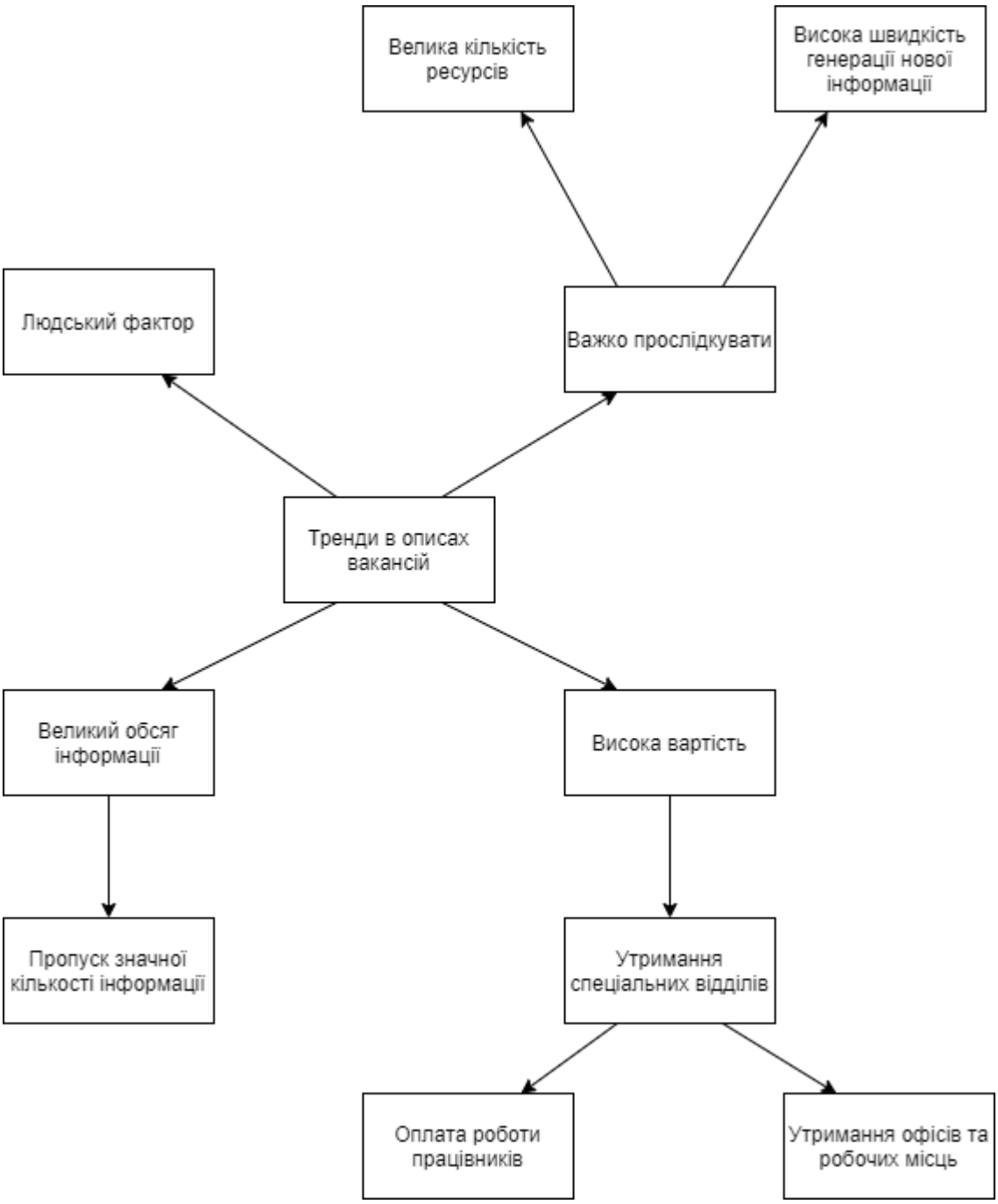


Рис. 5.1. Дерево проблем

5.2. Зацікавлені сторони

У вирішенні описаних у попередньому підрозділі проблем існує багато зацікавлених сторін. В першу чергу найбільш зацікавленою стороною є

відділи кадрів різного роду компаній, які займаються пошуком нових працівників. Саме їм надзвичайно важливо в час стрімкого розвитку інформаційних технологій прослідковувати тенденції на ринку праці для того, щоб мати можливість запропонувати вигідну пропозицію та залучити найкращих кандидатів. Іншою зацікавленою стороною є компанії, які пропонують послуги безпосереднього пошуку нових працівників для інших компаній. Причини зацікавленості є такими ж, які були описані вище.

Також не менш зацікавленою стороною є прості користувачі мережі Інтернет, які є в пошуках роботи і використовують для цього різні онлайн платформи. Це є надзвичайно корисно для людини, яка знаходиться в пошуках роботи, знати, що є найбільш популярним в реаліях сьогодення на ринку праці. Це може стосуватися і досвідчених спеціалістів, адже з розвитком нових технологій змінюється кожна сфера людської діяльності і вимоги для спеціалістів стають все ширшими і вищими. Але особливо це стосується людей, які знаходяться в пошуках роботи на ранніх життєвих етапах і інформацію про тренди на ринку праці допоможе зрозуміти, чого найбільше прагне роботодавець на даний момент та в яку сторону треба рухатись та розвиватись, щоб отримати бажану вакансію.

Також не менш зацікавленою стороною є соціальні платформи для пошуку роботи, де і відбувається діалог роботодавця і кандидата, так як інструменти пропонування користувачам нових публікацій активно використовують аналіз даних самих користувачів, проте тематики, які тільки набирають популярності відслідкувати важко, аналізуючи тільки дії користувача в мережі, тому задля того, щоб пропонувати користувачам “свіжі” вакансії, які набирають популярності варто виявляти тенденції, які існують в тематичних вакансіях.

Також зацікавленою стороною можуть виступати організатори та компанії, що організовують різні професійні навчальні платформи та курси.

Адже їх головна мета – це надавати актуальні знання та інформацію, які необхідні реальним працівникам та яких вимагають реальні роботодавці на даний момент. Знання про тренди в вакансіях допоможе сформувати навчальні плани відповідно до актуальних вимог в тій чи іншій професійній сфері, що в свою чергу залучить більшу кількість охочих відвідати ці курси і принесе компанії більші заробітки.

Іншою зацікавленою стороною можна вважати редакторів професійних журналів та власників професійних блогів. Беззаперечно для них є дуже важливим висвітлювати теми, що пов’язані з актуальними на даний момент вимогами на ринку праці, і тим самим заохочувати більшу кількість читачів.

Усі зацікавлені сторони, їх вплив та інтереси наведені у табл.5.1.

Таблиця 5.1

Зацікавлені сторони

Зацікавлена сторона	Інтерес зацікавленої сторони	Вплив зацікавленої сторони	Стратегія приваблення зацікавлених сторін
HR-відділи	Можливість запропонувати вигідну пропозицію та залучити найкращих кандидатів	Високий	Проведення презентації для представників зацікавлених осіб. Участь у спеціалізованих конференціях і форумах
Компанії по рекрутингу	Можливість запропонувати вигідну пропозицію та залучити найкращих кандидатів	Високий	

Продовження табл. 5.1

Користувачі платформ по пошуку роботи	Знання, що потрібно роботодавцям найбільше	Високий	Проведення презентації для представників зацікавлених осіб. Участь у спеціалізованих конференціях і форумах
Платформи по пошуку роботи	Пропонувати користувачам нові вакансії	Середній	
Організатори навчальних курсів	Надання актуальних знань, залучення більшої кількості охочих	Низький	
Редактори професійних журналів та блогів	Зацікавлення більшої кількості читачів	Низький	

5.3. Комерційне рішення. Основні характеристики

Зважаючи на всі проблеми описані вище, кінцевий програмний продукт має реалізовувати автоматизований метод виявлення тенденцій в описах вакансій, який був описаний у попередніх розділах. Потоки Web 2.0, такі як публікації на відповідних сайтах для пошуку роботи пропонують велику кількість вакансій за різними категоріями та сферами людської діяльності. Наприклад, для України існує низка популярних сайтів, які дають змогу роботодавцям додавати вакансії, а користувачам цих сайтів проводити пошук за різними категоріями та фільтрами, такі як: www.work.ua, www.rabota.ua, www.dou.ua та інші.

З іншого боку, різноманіття наявних вакансій може легко перевантажити користувачів. Для того, щоб впоратися з результуючим

перевантаженням інформації, веб-сторінки з вакансіями часто організовані в дрібномасштабні таксономії, і користувачі керуються темами та категоріями для пошуку та дослідження. Користувачів особливо цікавлять виникаючі тематики, які з’являються з огляду на актуальний стан ринку праці, але які не можуть бути легко знайдені за допомогою існуючих категорій. Звичайно, користувачі можуть здійснювати пошук за ключовими словами, але справа в тому, що вони хочуть автоматично отримувати сповіщення про нещодавно виниклу тему, яка збирається стати популярною, - і вони хочуть знати це якомога швидше.

Треба також звернути увагу, що виявлення таких тенденцій дуже відрізняється від визначення популярних тем; нас цікавлять раптові зміни кореляції між тегами, пов'язаними з даними. Той факт, що ці тенденції складаються з пар або, загалом, наборів тегів, дає можливість повного вивчення соціальних медіа з урахуванням виявленого тегу, встановленого як вхідний, наприклад, у вигляді традиційного ключового слова.

5.4. Конкурентні переваги рішення

Було зазначено, що на сьогоднішній день існує дуже багато проєктів, які аналізують потоки в соціальних медіа ресурсів та визначають популярні тематики, проте наразі тема виникаючих тенденцій на ринку праці є недостатньо вивченою і не існує розробленого програмного продукту, який би вирішував проблеми виявлення нових тенденцій, які тільки починають набирати популярності на ринку праці. Для вирішення цього завдання компанії наймають відділи аналітиків, які аналізують нові тематики та виокремлюють тенденції, які тільки розвиваються. В цьому підході є дуже багато недоліків, які були зазначені в підрозділі 5.1. Майже всі проблеми пов’язані з даним підходом можна вирішити за допомогою запропонованого програмного метода автоматизованого виявлення тенденцій в описах

вакансій. Програмний продукт, який був запропонований, має такі конкурентні переваги:

- зменшення витрат, так як не буде необхідності утримувати додаткових спеціалістів;
- підвищення швидкості обробки потоків інформації з соціальних платформ для пошуку роботи;
- покращення якості виявлення тенденцій, так як буде відсутній суб'єктивізм;
- модифікований метод виявлення нових тенденцій в описах вакансій, що враховує специфіку останніх.

5.5. Клієнти. Сегменти ринку споживання

Головні клієнти такого програмного забезпечення - це великі компанії, які знаходяться постійно у пошуках нових співробітників та активно використовують онлайн-ресурси для розміщення вакансій та пошуку резюме. Також невід'ємними клієнтами є користувачі цих онлайн-ресурсів, які знаходяться у пошуках роботи. Як вже було зазначено, іншими клієнтами можуть виступати редактори професійних видань та блогів і організатори навчальних курсів. Говорячи про компанії, які активно використовують онлайн-ресурси для розміщення вакансій, особливу увагу треба приділити саме тим компаніям, які у своїй роботі використовують мережу Інтернет, такі як IT-компанії, маркетингові, ті, що займаються продажами в мережі. Саме цей сегмент в найбільшій мірі залучений до соціальних платформ.

Сьогодні майже в кожній країні є великі соціальні проєкти для пошуку роботи, проте в мережі найбільш вживаною мовою є саме англійська. Англомовні країни, такі як Сполучені Штати Америки, Велика Британія, Канада, є найбільш розвиненими, проте і в інших країнах часто роботодавці розміщують оголошення саме на англійській мові.

Мовна ознака спричинила сегментацію ринку не тільки через те, що англомовний сегмент є найбільш розвиненим, але також через те, що часто методи передобробки природномовних текстів наявні саме для анлійської мови і для інших мов просто неможливо здійснити якісну передобробку текстів через відсутність наборів даних для інших мов.

Проте в тестуванні даного методу використано і інші мови, для яких доступні інструменти передоброблення тексту, але аналіз результатів проводити досить важко без наявних знань інших використовуваних мов.

Сегментацію ринку можна також провести за формою контенту, що представлений на такого роду соціальних платформах. Так як нас цікавлять саме текстові дані, тому доцільно розглядати соціальні платформи, в яких інформація подається саме в текстовій формі.

5.6. Унікальна ціннісна пропозиція

Ціннісна пропозиція – це усі переваги, які отримує споживач, вибираючи послуги або товар. Низка проблем була виділена у дереві проблем і інтереси зацікавлених сторін були проаналізовані. Компанії хочуть отримати менш витратний спосіб в плані людських ресурсів, часових ресурсів та грошових для виявлення нових трендів для формування найкращих пропозицій та пошуку найкращих кандидатів, користувачі платформ для пошуку роботи -знання які потрібні роботодавцям найбільше, платформи для пошуку роботи – актуальну рекламу для користувачів, а організатори навчальних курсів та редактори журналів і блогів – залучення більшої кількості аудиторії.

Спираючись на все вищесказане можна виокремити унікальну ціннісну пропозицію, якою є розроблений адаптований метод автоматизованого виявлення трендів в описах вакансій, що враховує специфіку текстових даних оголошень, що є основною перевагою.

5.7. Доходи та витрати

Користувач буде мати доступ до програмного продукту після придбання ліцензії на продукт.

Виділяють такі типи ліцензій програмного забезпечення:

1. Freeware. Вільно та безкоштовно розповсюджувані повнофункціональні програми. Купувати подібні програми не потрібно. Вони, як правило, поширюються через середу Інтернет або в якості доповнення з платними комерційними продуктами.
2. Shareware. Умовно-безкоштовне програмне забезпечення – користувачеві безкоштовно надається програмне забезпечення неповного функціоналу, тобто з деякими обмеженнями, що діють до тих пір, поки не буде проведена оплата за повнофункціональний продукт. Обмеження можуть бути:
   - функціональними, тобто користувачеві доступні не всі функціональні можливості – це так звані демо-версії (demo);
   - тимчасовими, тобто без оплати продукт в повному функціоналі працює якийсь календарний час або певну кількість запусків – пробні версії (trial).
3. Public domain software. Даний тип ліцензій схожий на freeware – програмні продукти цього типу також поширюються безкоштовно. Однак, на відміну від freeware, де за автором програми зберігаються всі права на неї, у випадку з public domain у автора ці права відсутні. Програмне забезпечення поширюється разом з вихідним кодом, і автор, викладаючи свій продукт в загальний доступ, повністю відмовляється від своїх прав. Поширення такого роду ліцензій було характерно для початку масового доступу в Інтернет. У даний час продукти з цим типом ліцензії практично не випускаються

4. Open Source. Програмне забезпечення поширюється на безкоштовній основі разом з вихідним кодом. Однак автор уже не відмовляється від своїх прав. Існує міжнародна система вимог до ліцензії на програмний продукт, який називається The Open Source Definition (OSD). Модифіковане будь-яким співавтором забезпечення повинно поширюватися на тих же умовах, що і вихідний продукт – тобто модифіковане ПО не можна перевести в платне і комерційне.
5. Commercial. Комерційне програмне забезпечення, тобто програмне забезпечення, завжди розповсюджується тільки за плату. Оплата повинна бути здійснена авансом або відразу після отримання копії на ліцензійному диску або дискеті у фірмовій упаковці.

До витрат можна віднести:

1. Виплати заробітних плат, соціальних виплат та страховок співробітникам.
2. Оренда невеликого приміщення.
3. Покупка обладнання.
4. Послуги юриста.
5. Послуги бухгалтера.
6. Податки.

5.8. Бізнес-модель

Складемо тепер бізнес-модель, узагальнивши усе вище написане і перенесемо на канву, яка зображена на рис. 5.2.

Споживачі:

- HR-відділи;
- компанії по рекрутингу;

– користувачі платформ по пошуку роботи;
– платформи по пошуку роботи;
– організатори навчальних курсів;
– редактори професійних журналів та блогів.

Проблема:

– наявні автоматизовані рішення не задовільнять потребам завдання виявлення трендів в даних оголошень про вакансії;
– людські послуги занадто дорогі для вирішення такого роду завдання;
– обсяги інформації занадто великі;
– людський фактор.

Рішення: програмний застосунок, що реалізує метод автоматизованого виявлення трендів в описах вакансій на основі аналізу природомовних людський текстів.

Унікальна ціннісна пропозиція:

– програмний застосунок, що реалізує адаптований метод автоматичного виявлення трендів в описах вакансій;
– зменшення витрат компаній на аналіз потоків платформ для пошуку роботи.

Потоки доходів:

– продаж ліцензій;

Структура витрат:

– заробітна плата працівникам;
– оренда невеликого приміщення;
– покупка обладнання;
– послуги юриста;
– послуги бухгалтера;

– податки.

Взаємовідносини з клієнтами:

– проведення презентацій, участь у спеціалізованих конференціях і форумах;
– технічна підтримка.

Канали: відділи співпраці великих компаній по пошуку роботи, HR-відділи.

Ключові метрики: кількість ліцензій, які були продані.

Таблиця 5.2

Канва бізнес-моделі

<table>
<tr>
<td rowspan="2">Проблема
1. Наявні рішення не задовільнять потребам.
2. Людські послуги занадто дорогі.
3. Обсяги інформації занадто великі.
4. Людський фактор.</td>
<td>Рішення
1.Програмне забезпечення, що реалізує метод автоматизованого виявлення трендів в описах вакансій на основі аналізу природомовних людський текстів</td>
<td rowspan="2">Унікальна ціннісна пропозиція
1.Програмний застосунок, що реалізує адаптований метод;
2.Зменшення витрат компаній на аналіз потоків платформ для пошуку роботи.</td>
<td>Взаємодія з клієнтами
1. Проведення презентацій, участь у спеціалізованих конференціях і форумах
2. Технічна підтримка</td>
<td rowspan="2">Споживачі
1.HR-відділи
2.Компанії по рекрутингу
3.Користувачі платформ по пошуку роботи
4.Платформи для пошуку роботи
5.Організатори навчальних курсів
6.Редактори професійних журналів та блогів</td>
</tr>
<tr>
<td>Ключові метрики
Кількість ліцензій, що були продані</td>
<td>Канали
Відділи співпраці компаній по пошуку роботи,HR-відділи</td>
</tr>
<tr>
<td colspan="3">Структура витрат
1. Заробітна плата.
2. Оренда приміщення
3. Податкові витрати.
4. Послуги юриста, бухгалтера.</td>
<td colspan="2">Потоки доходів
Доходи від продажу ліцензій</td>
</tr>
</table>

5.8. Висновки до розділу 5

У даному розділі проаналізовано поточну ситуацію у сфері виявлення трендів в описах вакансій.

Створено дерево проблем з усіма наявними перешкодами у цій сфері. Також були описані всі зацікавлені сторони, їх вплив та зацікавленість. В результаті цього була сформульована унікальна ціннісна пропозиція програмного продукту. Були проаналізовані сегменти ринку споживання та основні клієнти.

Крім цього були визначені потоки доходів та витрат.

Була побудована бізнес-модель, яка описує, як продукт створює цінність і може заробити на цьому.

ВИСНОВКИ

Такі завдання були виконані у роботі:

1. Проаналізовані існуючі підходи та системи автоматизованого виявлення трендів в текстових даних, визначено їх переваги та недоліки.
2. Була вивчена специфіка оголошень про вакансії в мережі Інтернет на відповідних платформах для пошуку роботи та враховані характеристики цих оголошень під час виявлення. Ці характеристики включають: наявність цифр і символів, які часто заплутують розуміння слів і виразів; наявність власних назв, таких як місцезнаходження чи імена людей, що може погіршити результат виявлення трендів ; наявність малозначущих частин мов та наявність багатослівних виразів, які можуть позначати навички і мають бути враховані.
3. Проаналізовано процес виявлення трендів в текстових даних. Для врахування специфіки оголошень про вакансії запропоновано модифікувати етапи передоброблення тексту.
4. Розроблено метод автоматизованого виявлення трендів в описах вакансій, доступних в мережі Інтернет на основі комплексного урахування специфіки таких текстових даних.
6. Програмне забезпечення для виявлення трендів в описах вакансій, що реалізує запропонований у роботі метод було реалізоване.
7. Проведено аналіз результатів роботи розробленого програмного забезпечення на зібраних даних з платформи для розміщення вакансій Adzuna.

Наразі не існує офіційних методик оцінки якості таких систем для виявлення трендів і остаточні висновки по їх роботі покладаються все ж на

людського експерта. Тому одним із перспективних напрямків для наступних досліджень є розробка метрик та методик для ефективного оцінювання якості методів для виявлення трендів в текстових даних.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Gray, Comparison of Trend Detection Methods [Text] / Katharine Lynn. – Dissertations, 2007. – 228p.
2. Davidson, Knowledge mining with VxInsight. Discovery through interaction [Text] / G.S. Davidson, B. Hendrickson, D.K. Johnson, C.E. Meyers, B.N. Wylie. – Journal of Intelligent Information Systems, 1998. – 259-285p.
3. Swan, Automatic generation of overview timelines. [Text] / R. Swan, J. Allan. – In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, 2000.
4. Porter, Technology opportunities analysis. Technological Forecasting and Social Change [Text] / A.L. Porter, M.J. Detampel, 1995. – 237-255p.
5. TDT [Електронний ресурс] – Режим доступу: www.nist.gov/speech/tests/tdt/index.html – Дата доступу: грудень 2019. – Назва з екрану.
6. Swan, Timemines: Constructing timelines with statistical models of word usage [Text] / R. Swan, D. Jensen. – In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.
7. Lent, Discovering trends in text databases [Text] / B. Lent, R. Agrawal, R. Srikant. – In Proceedings of the Third International Conference On Knowledge Discovery and Data Mining, California, 1997.
8. Agrawal, Mining sequential patterns [Text] / R. Agrawal, R. Srikant. – In Proceedings of the International Conference on Data Engineering (ICDE), Taipei, Taiwan, 1995.
9. Srikant, Mining sequential patterns: Generalizations and performance improvements [Text] / R. Srikant, R. Agrawal. – In Proceedings of the Fifth International Conference on Extending Database Technology (EDBT), Avignon, France, 1996.

10. Agrawal, Querying shapes of histories [Text] / R. Agrawal, G. Psaila, E.L. Wimmers, M. Zait. – In Proceedings of the 21st International Conference on Very Large Databases, Zurich, Switzerland, 1995.
11. ClusterizerTM [Електронний ресурс] – Режим доступу: www.autonomy.com/Extranet/Technical/Modules/TBAutonomyClusterizer.pdf – Дата доступу: грудень 2019. – Назва з екрану.
12. S. Clarke. Knowlege Suite (Review) [Електронний ресурс] – Режим доступу: www.autonomy.com/Extranet/Marketing/AnalystWhitePapers/ButlerReportonAutonomySuite200299.pdf – Дата доступу: грудень 2019. – Назва з екрану.
13. Autonomy [Електронний ресурс] – Режим доступу: www.autonomy.com/Content/Technology/Background/IntellectualFoundations – Дата доступу: грудень 2019. – Назва з екрану.
14. SPSS LexiQuest.SPSS LexiQuest [Електронний ресурс] – Режим доступу: www.spss.com/spssbi/lexiquest – Дата доступу: грудень 2019. – Назва з екрану.
15. SPSS Clementine.SPSS Clementine [Електронний ресурс] – Режим доступу: www.spss.com/spssbi/clementine – Дата доступу: грудень 2019. – Назва з екрану.
16. ClearForest. [Електронний ресурс] – Режим доступу: www.clearforest.com – Дата доступу: грудень 2019. – Назва з екрану.
17. Abe. Evaluating a temporal pattern detection method for finding research keys in bibliographical data [Text] / H. Abe, S. Tsumoto // Transactions on Rough Sets XIV. Lecture Notes in Computer Science. – 2011. – P. 1 – 17.
18. P. Hennig. Identify Emergent Trends Based on the Blogosphere / P. Hennig, P. Berger, C. Meinel // IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT) – 2013.

19. Keogh, Segmenting time series: A survey and novel approach. [Text] / E. Keogh, S. Chu, D. Hart, M. Pazzani. – In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific. Publishing Company, 1993. – 1-22p.
20. Montgomery, Introduction to linear regression analysis, [Text] / D. Montgomery, E. Peck, G. Vining. – Wiley series in probability and statistics. – 3rd edition. – New York: NY Wiley, 2001.
21. Srikant, Mining Sequential Patterns: Generalizations and Performance Improvements [Text] / R. Srikant, R. Agrawal. – In Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '96), 1996. – 3-17p.
22. POLYGLOT-NER [Електронний ресурс] – Режим доступу: https://polyglot.readthedocs.io/en/latest/NamedEntityRecognition.html – Дата доступу: січень 2020. – Назва з екрану.
23. TreeTagger [Електронний ресурс] – Режим доступу: https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ – Дата доступу: січень 2020. – Назва з екрану.
24. McKinney , Python for Data Analysis [Text] / Wes McKinney. – O'Reilly, 2017. – 10-26p.
25. NUMPY [Електронний ресурс] – Режим доступу: https://numpy.org/ – Дата доступу: березень 2020. – Назва з екрану.
26. Pandas [Електронний ресурс] – Режим доступу: pandas.pydata.org – Дата доступу: березень 2020. – Назва з екрану.
27. Matplotlib [Електронний ресурс] – Режим доступу: https://matplotlib.org/ – Дата доступу: березень 2020. – Назва з екрану.
28. Jupyter Notebook [Електронний ресурс] – Режим доступу: https://jupyter.org/ – Дата доступу: березень 2020. – Назва з екрану.

29. Scikit-learn. Machine learning for Python [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/ – Дата доступу: березень 2020. – Назва з екрану.
30. Adzuna API [Електронний ресурс] – Режим доступу: https://developer.adzuna.com/ – Дата доступу: березень 2020. – Назва з екрану.
31. Margulis, N-poisson document modelling revisited [Text] / E. L. Margulis, 1991.
32. Lee, A model for extracting keywords of document using term frequency and distribution [Text] / J. W. Lee, D. K. Baik. – Computational Linguistics and Intelligent Text Processing, 2004. – 437-440p.
33. Hastieand, Generalized additive models [Text] / T. Hastieand, R. Tibshirani. – Wiley Online Library, 1990.
34. Wood, Generalized additive models: an introduction with R [Text] / S.N.Wood. – CRC press, 2017.
35. Carrière-Swallow, Nowcasting with google trends in an emerging market [Text] / Y. Carrière-Swallow, F. Labbé. – Journal of Forecasting. – vol. 32. – no. 4. – 2013. – 289–298p.
36. Choi, Predicting the present with google trends [Text] / H. Choi, H. Varian. – Economic Record. – vol. 88. – no. s1, 2012. – 2–9p.
37. Carneiro, Google trends: a web-based tool for real-time surveillance of disease outbreaks [Text] / H. A. Carneiro, E. Mylonakis. – Clinical infectious diseases, – vol. 49, –no. 10, 2009. – 1557–1564p.

ДОДАТКИ

Додаток 1

Фрагменти тексту програми

Лістинг 1. JobAdRetriever.py

```
import urllib.request

def request_page(country, category, page):
    r =
urllib.request.urlopen("http://api.adzuna.com:80/v1/api/jobs/{}/search/{}?a
pp_id={}&app_key={}&results_per_page=50&max_days_old=20000&category={}&sort
_direction=up&sort_by=date".format(country, page, APP_ID, APP_KEY,
category)).read().decode('utf-8')
    if (NO_RESULTS in r):
        return 0
    else:
        file = open("Raw Data/{}_{}_{}.json".format(country, category,
page), "w")
        print(r, file=file)
        file.close()
        return 1

def request_adzuna(countries, categories):
    for country in countries:
        for category in categories:
            continue_requests = 1
            page_index = 1
            while (continue_requests == 1):
                print("Printing page {} of {} in {}.".format(page_index,
category, country))
                continue_requests = request_page(country, category,
page_index)
                page_index += continue_requests

if __name__ == "__main__":

    # 2-letter codes of countries that we loop over, these are all the ones
available in Adzuna
    COUNTRIES = "gb au at br ca de fr in it nl nz pl ru sg us za".split(" ")

    # Categories of jobs we will look for, not exhaustive list
    CATEGORIES = "scientific-qa-jobs consultancy-jobs pr-advertising-
marketing-jobs engineering-jobs it-jobs accounting-finance-jobs".split(" ")

    # String found if search yields no results
    NO_RESULTS = "\"results\":[]"

    # Adzuna-provided app_id and app_key, no longer valid
    APP_ID = "92cd0f19"
    APP_KEY = "8a9c57473fdeded0f90386b69569e1ba"

    # Requests Adzuna to save jobs as json files for all categories in all
countries
    request_adzuna(COUNTRIES, CATEGORIES)
```

Лістинг 2. JobDescriptionFetcher.py

```
import os
import json
import urllib.request as url
from bs4 import *

import csv

if __name__ == "__main__":

    # Dictionary collecting all data
    data_dict = dict()

    old_category = None

    # Folder where the json files containing Adzuna search results are
found
    JOB_AD_DIR = 'Raw Data'

    # Folder where the job ad descriptions will be saved
    DESCRIPTION_DIR = 'Raw Text'

    # If the Job Ad Description Fetcher was interrupted, this enables it to
restart where it left off
    printed_out_categories = set(filename.split('.')[0] for filename in
os.listdir(DESCRIPTION_DIR))

    # File Extensions
    JOB_AD_EXT = '.json'
    DESCRIPTION_EXT = '.csv'

    # CSV Header
    ID = 'id'
    TITLE = 'title'
    DATE = 'created'
    DESC_STATE = 'description_from_url'
    DESC = 'description'
    FIELDNAMES = [ID, TITLE, DATE, DESC_STATE, DESC]

    # Filenames to loop over
    job_ad_file_names = [filename for filename in os.listdir(JOB_AD_DIR) if
filename.endswith(JOB_AD_EXT)]

    for filename in job_ad_file_names:

        # Log
        print(filename)

        # Category of file
        category = '_'.join(filename.split("_")[:2])

        # Proceeding if category was not already finished
        if category not in printed_out_categories:
```

```
            # Initialising old_category if None
            if old_category == None:
                old_category = category

            # If we change category, corresponding CSV file is printed
            if not old_category == category:
                print("Printing the CSV file of {}".format(old_category))
                with open('{}/{}{}'.format(DESCRIPTION_DIR, old_category,
DESCRIPTION_EXT), 'w') as csvfile:
                    writer = csv.DictWriter(csvfile, fieldnames=FIELDNAMES)
                    writer.writeheader()
                    for job in data_dict[old_category]:
                        writer.writerow(job)
                old_category = category

            # Initialises empty array in the data dictionary
            if category not in data_dict:
                data_dict[category] = []

            # Loading json data as dictionary
            with open(os.path.join(JOB_AD_DIR, filename)) as f:
                json_data = json.load(f)["results"]

            # Looping over json data
            for key in range(len(json_data)):

                # Log
                print("{} {}".format(filename, key))

                # Job Data
                job = dict()
                job[ID] = json_data[key][ID]
                job[TITLE] = json_data[key][TITLE]
                job[DATE] = json_data[key][DATE]
                description = json_data[key][DESC]

                # Checking state of description
                desc_from_url = -1

                # Trying to get description from Redirect Source URL
                try:
                    redirect_url =
BeautifulSoup(url.urlopen(json_data[key]['redirect_url']).read())

                    # Determining if Redirect URL is in Adzuna or not
                    if len(redirect_url('a')) == 1:
                        job_url =
str(redirect_url('a')[0]).split("href=\"")[1].split("\">")[0]
                        soup = BeautifulSoup(url.urlopen(job_url).read())
                        print("This is a redirect URL")
                    else:
                        soup = redirect_url
                        print("The Job Ad Description will be retrieved
from Adzuna")
```

```
                    # Trying to get description from Redirect Target URL
                    try:
                        [s.extract() for s in soup('script')]
                        [s.extract() for s in soup('a')]
                        text = soup.get_text()
                        if description[:10] in text:
                            description =
text[text.index(description[:10]):]
                            desc_from_url = 1
                            print("Description from URL")
                        else:
                            desc_from_url = 0
                            print("Redirect Target URL doesn't have
Description")
                    except:
                        print("Could not get Redirect Target URL")
                        pass
                except:
                    print("Could not get Redirect Source URL")
                    pass
                job[DESC_STATE] = desc_from_url
                job[DESC] = description
                data_dict[category].append(job)

    # Printing last category
    try:
        print("Printing the CSV file of {}".format(old_category))
        with open('{}/{}{}'.format(DESCRIPTION_DIR, old_category,
DESCRIPTION_EXT), 'w') as csvfile:
            writer = csv.DictWriter(csvfile, fieldnames=FIELDNAMES)
            writer.writeheader()
            for job in data_dict[category]:
                writer.writerow(job)
    except:
         print("Last category was empty")
         pass
```

Лістинг 3. GoogleTrends.py

```
from pytrends.request import TrendReq
import math
import os
import pandas as pd
import requests
import json
from TrendPositions import get_trending_words

### JSON ###
```

```
USER_AGENT = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"
HEADERS = {'User-Agent' : USER_AGENT}

def get_interest_over_time_json(country, keyword):
    url = "https://trends.google.com/trends/fetchComponent?hl=en-US&q={}&geo={}&cid=TIMESERIES_GRAPH_0&export=3&w=500&h=300".format(keyword, country.upper())
    r = requests.get(url, headers=HEADERS).text
    r = r.split("google.visualization.Query.setResponse(")[1][:-2].replace("new Date", "\"").replace("),", ")\",")
    data = json.loads(r)
    if data["status"] == "error":
        return "NaN"
    numbers = []
    for row in data["table"]["rows"]:
        label = row["c"][0]["f"]
        if "2016" in label or "2017" in label:
            numbers.append(row["c"][1]["v"])
    return get_energy_json(numbers)

def get_energy_json(energy_list):
    energy = 0
    last_element = energy_list[-1]
    interval_count = len(energy_list) - 1
    for index in range(interval_count):
        energy += (math.pow(last_element, 2) - math.pow(energy_list[index], 2)) * 1 / (interval_count - index)
    return energy

### HTML ###

def get_interest_over_time(pytrend, country, keyword, recursion):
    try:
        pytrend.build_payload([keyword], timeframe="all", geo=country.upper())
        try:
            return pytrend.interest_over_time()
        except:
            return EMPTY_DF
    except:
        if recursion <= 100:
```

```
            pt = TrendReq('', '')
            return get_interest_over_time(pt, country, keyword, recursion +
1)
        else:
            return EMPTY_DF

def get_energy(df):
    if df.empty:
        return "NaN"
    energy = 0
    rows = list(df.iterrows())[-20:-2]
    last_element = rows[-1][1][0]
    interval_count = len(rows) - 1
    for index in range(interval_count):
        energy += (math.pow(last_element, 2) - math.pow(rows[index][1][0],
2)) * 1 / (interval_count - index)
    return energy

### CHOOSE METHOD ###

def energy_method(method, pytrend, country, keyword, recursion):
    if method == "json":
        return get_interest_over_time_json(country, keyword)
    elif method == "html":
        interest = get_interest_over_time(pytrend, country, new_keyword,
recursion)
        return get_energy(interest)
    return "NaN"

### MAIN ###

# Destination Directory
DEST_DIR = "Google Trends"

# Processed Files, to pick up where the program left off if interrupted
PROCESSED_FILES = os.listdir(DEST_DIR)

# Valid file directory
valid_dir = "Trend Positions"

# Files that were not processed
valid_files = [filename for filename in os.listdir(valid_dir)
```

```
                if filename.endswith(".txt")
                and not filename.endswith("_no_nlp.txt")
                and filename not in PROCESSED_FILES]

# Initialising empty PyTrend request
pt = TrendReq('', '')

# DataFrame returned if request yields no results
EMPTY_DF = pd.DataFrame({'A' : []})

# Looping over the files that were pre-processed
for filename in valid_files:

    # Log
    print(filename)

    # Reads Keywords
    keywords = get_trending_words(filename)

    # Printing corresponding energy for each keyword
    file = open("{}/{}".format(DEST_DIR, filename), "w")
    for keyword in keywords:
        new_keyword = keyword.replace("_", " ")
        country = filename[:2]
        energy = energy_method("html", pt, country, new_keyword, 1)
        print(keyword, keywords[keyword], energy)
        print("{}\t{}\t{}".format(keyword,    keywords[keyword],    energy),
file=file)
    file.close()
```

Лістинг 4. SequenceMining.py

```
class GspSearch(object):
    def __init__(self, raw_transactions):
        self.freq_patterns = []
        self.process_transactions(raw_transactions)

    def process_transactions(self, raw_transactions):
        self.transactions = []
        alpha = {}
        for r in raw_transactions:
            for c in r:
                alpha[c] = True
            self.transactions.append(r)
```

```
        self.alpha = alpha.keys()

    def generate_init_candidates(self):
        return list(self.alpha)

    def generate_new_candidates(self, freq_pat)
        old_cnt = len(freq_pat)
        old_len = freq_pat[0].count(' ')
        print("Generating new candidates from %s %s-mers ..." % (old_cnt,
old_len))

        new_candidates = []
        for c in freq_pat:
            for d in freq_pat:
                merged_candidate = self.merge_candidates(c, d)
                if merged_candidate and (merged_candidate not in
new_candidates):
                    new_candidates.append(merged_candidate)

        ## Postconditions & return:
        return new_candidates

    def merge_candidates(self, a, b):
        c1 = a.split(' ')
        c2 = b.split(' ')
        if (len(c1) == 1 and len(c2) == 1) or c1[1:] == c2[:-1]:
            c1.append(c2[len(c2) - 1])
            return ' '.join(c1)
        else:
            return None

    def filter_candidates(self, trans_min):
        filtered_candidates = []
        index = 0
        for c in self.candidates:
            curr_cand_hits = self.single_candidate_freq(c)
            if trans_min <= curr_cand_hits:
                filtered_candidates.append((c, curr_cand_hits))
            index+=1
        return filtered_candidates

    def single_candidate_freq(self, c):
        hits = 0
        for t in self.transactions:
            if self.search_transaction(t, c):
                hits += 1
        return hits

    def search_transaction(self, t, c):
        if c in t:
            return True
        t_len = len(t)
        found = False
        tokens = c.split(' ')
        found_total = len(tokens)
```

```
        for i in range(t_len):
            if tokens[0] == t[i]:
                found = True
                break
        if not found or i + found_total > t_len:
            return False
        for j in range(1, found_total):
            if tokens[j] != t[j + i]:
                return False
        return True

    def search(self, threshold):
        ## Preparation:
        assert (0.0 < threshold) and (threshold <= 1.0)
        trans_cnt = len(self.transactions)
        trans_min = trans_cnt * threshold

        print("The number of transactions is: %s" % trans_cnt)
        print("The minimal support is: %s" % threshold)
        print("The minimal transaction support is: %s" % trans_min)

        ## Main:
        # generate initial candidates & do initial filter
        self.candidates = list(self.generate_init_candidates())
        print("There are %s initial candidates." % len(self.candidates))
        freq_patterns = []
        new_freq_patterns = self.filter_candidates(trans_min)
        print("The initial candidates have been filtered down to %s." %
len(new_freq_patterns))

        while True:
            # is there anything left?
            if new_freq_patterns:
                print(freq_patterns)
                freq_patterns = new_freq_patterns
                freq_terms = [x[0] for x in freq_patterns]
            else:
                self.freq_patterns = [x[0] for x in self.freq_patterns if '
' in x[0]]
                print("The results are: ", self.freq_patterns)
                return self.freq_patterns

            # if any left, generate new candidates & filter
            self.candidates = self.generate_new_candidates(freq_terms)
            print("There are %s new candidates." % len(self.candidates))
            new_freq_patterns = self.filter_candidates(trans_min)
            new_freq_terms = [x[0] for x in new_freq_patterns]
            for term, freq in self.freq_patterns:
                found = False
                new_freq_term_index = 0
                while not found and new_freq_term_index <
len(new_freq_terms):
                    if term in new_freq_terms[new_freq_term_index]:
                        found = True
                    else:
```

```
                        new_freq_term_index += 1
                if found:
                    self.freq_patterns.remove((term, freq))
            self.freq_patterns.extend(new_freq_patterns)
            print("The candidates have been filtered down to %s." %
len(new_freq_patterns))
```

Лістинг 5. TimeSeries.py

```
import os
import json

    # Where files are fetched
    DIRECTORY = 'Raw Data'

    # Saves relevant data here
    data_dict = {}

    # Loops over json files only
    for filename in os.listdir(DIRECTORY):
        if filename.endswith(".json"):

            # Job Category
            category = '_'.join(filename.split("_")[:2])

            # Initialises dict if not already done
            if category not in data_dict:
                data_dict[category] = {}

            # Gets json data from file
            with open(os.path.join(DIRECTORY, filename)) as f:
                json_data = json.load(f)["results"]

            # Fills counts in data dictionary, per category, year, month
and day
            for key in range(len(json_data)):
                date = json_data[key]['created'][:10]
                year = int(date[3])
                month = int(date[5:7])
                day = int(date[8:])
                if year not in data_dict[category]:
                    data_dict[category][year] = {}
                    data_dict[category][year][0] = 0
```

```
                if month not in data_dict[category][year]:
                    data_dict[category][year][month] = {}
                    data_dict[category][year][month][0] = 0
                if day not in data_dict[category][year][month]:
                    data_dict[category][year][month][day] = 0
                data_dict[category][year][month][day] += 1
                data_dict[category][year][month][0] +=1
                data_dict[category][year][0] +=1
        sorted(data_dict[category])
        for year in data_dict[category]:
            print("201{}\t{}".format(year, data_dict[category][year][0]),
file=file)
            for month in range(1,13):
                if month in data_dict[category][year]:
                    for day in range(0,32):
                        if day in data_dict[category][year][month]:
                            print("201{}-{}-{}\t{}".format(year, month,
day, data_dict[category][year][month][day]), file=file)
        file.close()
```

Лістинг 6. TreeTagger.py

```
import os
from subprocess import Popen, PIPE

from nltk.internals import find_binary, find_file
from nltk.tag.api import TaggerI
from sys import platform as _platform

_treetagger_url = 'http://www.cis.uni-
muenchen.de/~schmid/tools/TreeTagger/'

_treetagger_languages = ['bulgarian', 'dutch', 'english', 'estonian',
'finnish', 'french', 'galician', 'german',
                         'italian', 'polish', 'russian', 'slovak',
'slovak2', 'spanish', 'portuguese']

# Acceptable parts of speech tags by language: only nouns, abbreviations
and unknown tags
polish = "subst xxx ign brev burk adj".split(" ")
dutch = "noun adj".split(" ")
french = "abr nom nam adj".split(" ")
italian = "abr fw nom npr adj".split(" ")
english = "nn fw np jj".split(" ")

class TreeTagger(TaggerI):
```

```
    def __init__(self, path_to_home=None, language='german',
                 verbose=False, abbreviation_list=None):
        """
        Initialize the TreeTagger.
        :param path_to_home: The TreeTagger binary.
        :param language: Default language is german.
        The encoding used by the model. Unicode tokens
        passed to the tag() and batch_tag() methods are converted to
        this charset when they are sent to TreeTagger.
        The default is utf-8.
        This parameter is ignored for str tokens, which are sent as-is.
        The caller must ensure that tokens are encoded in the right
charset.
        """
        treetagger_paths = ['.', '/usr/bin', '/usr/local/bin',
'/opt/local/bin',
                            '/Applications/bin', '~/bin',
'~/Applications/bin',
                            '~/work/tmp/treetagger/cmd',
'~/TreeTagger/cmd']
        treetagger_paths = list(map(os.path.expanduser, treetagger_paths))
        self._abbr_list = abbreviation_list
        self.language = language

        if language in _treetagger_languages:
            if _platform == "win32":
                treetagger_bin_name = 'tag-' + language
            else:
                treetagger_bin_name = 'tree-tagger-' + language
        else:
            raise LookupError('Language not in language list!')

        try:
            self._treetagger_bin = find_binary(
                treetagger_bin_name, path_to_home,
                env_vars=('TREETAGGER', 'TREETAGGER_HOME'),
                searchpath=treetagger_paths,
                url=_treetagger_url,
                verbose=verbose)
        except LookupError:
            print('NLTK was unable to find the TreeTagger bin!')

    def tag(self, sentences):
        """Tags a single sentence: a list of words.
        The tokens should not contain any newline characters.
        """

        # Write the actual sentences to the temporary input file
        if isinstance(sentences, list):
            _input = '\n'.join((x for x in sentences))
        else:
            _input = sentences

        # Run the tagger and get the output
        if (self._abbr_list is None):
```

```
            p = Popen([self._treetagger_bin],
                      shell=False, stdin=PIPE, stdout=PIPE, stderr=PIPE)
        elif (self._abbr_list is not None):
            p = Popen([self._treetagger_bin, "-a", self._abbr_list],
                      shell=False, stdin=PIPE, stdout=PIPE, stderr=PIPE)

        # (stdout, stderr) = p.communicate(bytes(_input, 'UTF-8'))
        (stdout, stderr) = p.communicate(str(_input).encode('utf-8'))

        # Check the return code.
        if p.returncode != 0:
            print(stderr)
            raise OSError('TreeTagger command failed!')

        treetagger_output = stdout.decode('UTF-8')

        # Output the tagged sentences
        tagged_sentences = []
        for tagged_word in treetagger_output.strip().split('\n'):
            tagged_word_split = tagged_word.split('\t')
            tagged_sentences.append(tagged_word_split)

        return tagged_sentences

    def is_acceptable_portuguese_tag(self, tag):
        return (tag.startswith('n') and not tag.endswith('g0')) or
tag.startswith('a')

    def is_acceptable_russian_tag(self, tag):
        return tag.startswith('n') or tag.startswith('a')

    def is_acceptable_german_tag(self, tag):
        return tag.startswith('n') or tag.startswith('adj')

    def is_acceptable_tag(self, tag, acceptable_tags):
        for acceptable_tag in acceptable_tags:
            if acceptable_tag in tag:
                return True
        return False

    def is_acceptable_word(self, word):
        try:
            word_tag = self.tag(word)[0][1].lower()
            return {
                'english': self.is_acceptable_tag(word_tag, english),
                'french': self.is_acceptable_tag(word_tag, french),
                'german': self.is_acceptable_german_tag(word_tag),
                'portuguese': self.is_acceptable_portuguese_tag(word_tag),
                'italian': self.is_acceptable_tag(word_tag, italian),
                'polish': self.is_acceptable_tag(word_tag, polish),
                'russian': self.is_acceptable_russian_tag(word_tag),
                'dutch': self.is_acceptable_tag(word_tag, dutch)
            }[self.language]
        except:
            return True
```

Лістинг 7. TrendDetectionPipeLine.py

```
import csv
import sys
from nltk.tokenize import RegexpTokenizer
import os
from nltk.corpus import stopwords
from collections import Counter
import math
import operator
import re
from pprint import pprint
import numpy as np
from SequenceMining import GspSearch
import matplotlib
matplotlib.use('Agg')
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

### LANGUAGE RECOGNITION ###

if __name__ == '__main__':
    # These are the available languages with stopwords from NLTK
    languages = stopwords.fileids()

    # Fill the dictionary of languages, to avoid  unnecessary function
calls
    print("Loading stop words...", end='\r')
    try:
        dict_list = np.load('stopwords.npy').item()
    except:
        dict_list = {}
        for lang in languages:
            dict_list[lang] = {}
            for stop_word in stopwords.words(lang):
                dict_list[lang][stop_word] = 0
        np.save('stopwords.npy', dict_list)
    print("Loaded stop words.      ")

def test_for_language(descriptions):
```

```
    tokens = [item for d in descriptions for item in d]
    lang = which_language(tokens)
    return lang

def score(tokens):
    # Evaluating scores for each language
    scorelist = {}
    for lang in dict_list:
        scorelist[lang] = 0
        for word in tokens:
            if lang in dict_list:
                if word in dict_list[lang]:
                    scorelist[lang] += 1
    return scorelist

def which_language(text):
    scorelist = score(text)
    sorted_scorelist = sorted(scorelist.items(),
key=operator.itemgetter(1))
    maximum = sorted_scorelist[-1][1]
    # Default Language is English
    if maximum == 0:
        return "english"
    return sorted_scorelist[-1][0]

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet as wn
from pattern3.fr import parse as frparse
from pattern3.nl import parse as nlparse
from pattern3.de import parse as deparse
from pattern3.it import parse as itparse
from pydic import PyDic
from pymystem3 import Mystem

def pl_lemmatise(word):
    word_forms = pl_dict.word_base(word)
    if word_forms:
        return word_forms[0]
    return word

def ru_lemmatise(word):
    return ru_lemmatiser.lemmatize(word)[0]
```

```
def en_lemmatise(word):
    new_word = en_lemmatiser.lemmatize(word, wn.NOUN)
    if new_word == word:
        new_word = en_lemmatiser.lemmatize(word, wn.VERB)
        if new_word == word:
            new_word = en_lemmatiser.lemmatize(word, wn.ADJ)
            if new_word == word:
                new_word = en_lemmatiser.lemmatize(word, wn.ADV)
    return new_word

def lemmatise(word, language):
    try:
        return {
            'english': en_lemmatise(word),
            'french': frparse(word, lemmata=True).split('/')[-1],
            'dutch': nlparse(word, lemmata=True).split('/')[-1],
            'german': deparse(word, lemmata=True).split('/')[-1],
            'italian': itparse(word, lemmata=True).split('/')[-1],
            'polish': pl_lemmatise(word),
            'russian': ru_lemmatise(word)
        }[language]
    except:
        return word

### TEXT PRE-PROCESSING FOR LEMMATISATION ###

# Tokenizing, lemmatising, removing stop words, and patterns of format (hex
color, digit-only or digit-started strings)

def get_entries(directory, filename):
    lemmatised_files = os.listdir(DESC_DIR)
    if filename in lemmatised_files:
        with open('{}/{}'.format(DESC_DIR, filename), 'r') as csvfile:
            reader = list(csv.reader(csvfile))
            del reader[0]
            descriptions = [x[4].split(' ') for x in reader]
            return descriptions
    else:
        with open(os.path.join(directory, filename), 'r') as csvfile:
            # Reading file, deleting header, getting descriptions
            reader = list(csv.reader(csvfile))
```

```
            with open('{}/{}'.format(DESC_DIR, filename), 'w') as
new_csvfile:
                header = reader[0]
                writer = csv.DictWriter(new_csvfile, fieldnames=header)
                writer.writeheader()
                del reader[0]
                descriptions = [tokenize(x[4]) for x in reader]
                # Language recognition and selection of corresponding stop
word corpus
                lang = test_for_language(descriptions)
                # Pre-processing
                new_descriptions = []
                index = 0
                length = len(descriptions)
                for description in descriptions:
                    new_description = pre_process(description, lang)
                    new_descriptions.append(new_description)
                    writer.writerow({header[0]: reader[index][0],
header[1]: reader[index][1], header[2]: reader[index][2], header[3]:
reader[index][3], header[4]: ' '.join(new_description)})
                    index += 1
                    print("--> Lemmatising: {}/{}     ".format(index,
length),end='\r')
                print("--> Done pre-processing for {} job ads.
".format(length))
                return new_descriptions

def tokenize(description):
    tokenizer = RegexpTokenizer(r'\w+')
    return tokenizer.tokenize(description.lower())

def matches(pattern, string):
    re_match = pattern.match(string)
    if bool(re_match):
        re_span = re_match.span()
        return re_span[0] == 0 and re_span[1] == len(string)
    return False

def pre_process(tokens, lang):
    pattern_1 = re.compile(r"\d+([a-z]+)?")
    pattern_2 = re.compile(r"([a-f]|(\d)){6}")
    # Filtering Stop Words and details of words
```

```
    new_tokens = []
    for token in tokens:
        if lang not in dict_list or token not in dict_list[lang]:
            if not matches(pattern_1, token) and not matches(pattern_2,
token) and '_' not in token and len(token) <= 20:
                new_token = lemmatise(token, lang)
                new_tokens.append(new_token)
    return new_tokens

from TreeTagger import TreeTagger

def get_pre_processed_entries(filename, to_delete, min_month, max_month,
year):
    with open(filename, 'r') as csvfile:
        # Reading file, deleting header, getting descriptions
        reader = list(csv.reader(csvfile))
        del reader[0]
        # Language Detection
        lang = get_language(reader)
        try:
            tt = TreeTagger(language=lang)
        except:
            print("THE REJECTED LANGUAGE", lang)
            tt = TreeTagger(language='english')
        # Descriptions
        descriptions = [tokenize(x[4]) for x in reader if check_time(x[2],
min_month, max_month, year)]
        descriptions = gsp_search(descriptions, 0.2)
        descriptions = [remove_words(prune_ner_tags(description),
to_delete, tt) for description in descriptions]
        return descriptions

def check_time(string, min_month, max_month, year):
    return int(string[:4]) == year and int(string[5:7]) <= max_month and
int(string[5:7]) >= min_month

### POS TAG AND CROSS DOMAIN FILTERING

from polyglot.detect import Detector

def get_language(reader):
    desc = [x[4] for x in reader]
```

```
    text = ' '.join(desc)
    try:
        lang = Detector(text).language.name
        if lang == "un":
            return test_for_language(desc)
        else:
            return lang.lower()
    except:
        return test_for_language(desc)

def remove_words(description, to_delete, tt):
    new_description = []
    for word in description:
        if word not in to_delete:
            if tt.is_acceptable_word(word):
                new_description.append(word)
    return new_description

### NER TAG PRUNING ###

from polyglot.text import Text

def prune_ner_tags(tokens):
    try:
        entities = set([location for entity in Text('
'.join(tokens)).entities for location in list(entity) if entity.tag in ["I-
LOC", "I-PER"]])
        new_description = [token for token in tokens if token not in
entities]
        return new_description
    except:
        return tokens

### SELECTION OF FREQUENT TERMS ###

def tf_idf(corpus):
    df = {}
    tf = {}
    doc_len = len(corpus)
    for text in corpus:
        word_set = set(text)
        # Computing df
```

```
        for word in word_set:
            try:
                df[word] += 1
            except:
                df[word] = 1
        # Computing tf
        counter = Counter(text)
        try:
            tf_max = counter.most_common(1)[0][1]
        except:
            tf_max = len(text)
        for word in word_set:
            try:
                tf[word] += counter[word]/tf_max
            except:
                tf[word] = counter[word]/tf_max
    # Computing TF-IDF
    tfidf = {}
    for word in df:
        tfidf[word] =
(math.log(tf[word]/doc_len)+1)*math.log(doc_len/df[word])
    return tfidf

def get_keywords_per_document(tfidf):
    sorted_tfidf = sorted(tfidf.items(), key=operator.itemgetter(1))
    ranking = {}
    word_count = len(sorted_tfidf)
    for x in range(word_count):
        ranking[sorted_tfidf[x][0]] = word_count - x
    return ranking, sorted_tfidf

def select_keywords(corpus):
    tfidf = tf_idf(corpus)
    keywords, sorted_tfidf = get_keywords_per_document(tfidf)
    return keywords, sorted_tfidf

### CROSS-DOMAIN FILTERING ###

def cross_domain_filtering(keywords_per_country):
    keywords = set([keyword for category in keywords_per_country for
keyword in category])
    threshold = min(len(keywords)*0.1, 1000)
```

```
    to_delete = {}
    for keyword in keywords:
        ranking = []
        for category in keywords_per_country:
            try:
                ranking.append(category[keyword])
            except:
                continue
        if len(ranking) >= 4 and np.std(ranking) <= threshold:
            to_delete[keyword] = 0
    return to_delete

### GSP SEARCH ###

def introduce_n_grams(text, n_grams):
    new_text = ' '.join(text)
    for n_gram in n_grams:
        if n_gram in new_text:
            new_text = new_text.replace(n_gram, n_gram.replace(' ', '_'))
    return new_text.split(' ')

def gsp_search(descriptions, threshold):
    gsp = GspSearch(descriptions)
    n_grams = gsp.search(threshold)
    new_descriptions = [introduce_n_grams(description, n_grams) for
description in descriptions]
    return new_descriptions

def extract_keywords(directory, filename):
    descriptions = get_entries(directory, filename)
    keywords, sorted_tfidf = select_keywords(descriptions)
    return keywords

def get_tfidf_map(tfidf):
    tfidf_map = {}
    for entry in tfidf:
        tfidf_map[entry[0]] = entry[1]
    return tfidf_map

def compare_tfidfs(tfidf_2016, tfidf_2017):
    tfidf_map_2016 = get_tfidf_map(tfidf_2016)
    tfidf_map_2017 = get_tfidf_map(tfidf_2017)
```

```
    tfidf_delta = {}
    for keyword in tfidf_map_2016:
        try:
            score_2017 = tfidf_map_2017[keyword]
            score_2016 = tfidf_map_2016[keyword]
            delta = score_2017 - score_2016
            tfidf_delta[keyword] = delta
        except:
            continue
    sorted_tfidf_delta = sorted(tfidf_delta.items(),
key=operator.itemgetter(1))
    return sorted_tfidf_delta

### DENDROGRAM OF TRENDING TERMS ###

def compute_tfidf_vectors(corpus, trending_words):
    text_count = len(corpus)
    word_count = len(trending_words)
    # Initialising DF
    df = np.zeros(word_count)
    tfidf = np.zeros((word_count, text_count))
    pprint(trending_words)
    for text_index in range(text_count):
        text = corpus[text_index]
        counter = Counter(text)
        try:
            tf_max = counter.most_common(1)[0][1]
        except:
            tf_max = len(text)
        for word_index in range(word_count):
            count = counter[trending_words[word_index]]
            if count > 0:
                # Computing df
                df[word_index] += 1
                # Computing tf
                tfidf[word_index][text_index] = count/tf_max
    # Computing TF-IDF
    for word_index in range(word_count):
        tfidf[word_index] =
np.multiply(np.log(np.add(tfidf[word_index],1)),
math.log(text_count/df[word_index]))
    return tfidf
```

```
def get_countries(directory):
    countries = set()
    for filename in os.listdir(directory):
        if not filename.startswith("._") and filename.endswith(".csv"):
            countries.add(filename[:2])
    return countries

def get_keywords_per_country(directory, dest_dir):
    countries = get_countries(directory)
    # Keyword Extraction
    try:
        keywords_per_country = np.load('keys.npy').item()
    except:
        keywords_per_country = {}
        for country in countries:
            keywords_per_country[country] = []
        processed_files = os.listdir(dest_dir)
        for filename in os.listdir(directory):
            if not filename.startswith("._") and filename.endswith(".csv")
and "{}.txt".format(filename[:-4]) not in processed_files:
                print("File: {}".format(filename))
                keywords = extract_keywords(directory, filename)
                country = filename[:2]
                keywords_per_country[country].append(keywords)
        np.save("keys.npy", keywords_per_country)
    for country in countries:
        print(country)
        categories = sorted([filename for filename in os.listdir(directory)
if filename.startswith(country) and filename.endswith(".csv")])
        if len([category for category in categories if
"{}.txt".format(category[:-4]) not in os.listdir(dest_dir)]) > 0:
            # Cross-Domain Filtering
            to_delete =
cross_domain_filtering(keywords_per_country[country])
            # Trend Detection
            for index in range(len(categories)):
                filename = categories[index]
                if "{}.txt".format(filename[:-4]) not in
os.listdir(dest_dir):
                    desc_2016, desc_2017 =
extract_text_for_trends(filename, to_delete)
```

```
                    print(desc_2016)
                    print(desc_2017)
                    tfidf_2016, tfidf_2017 =
get_tfidf_for_trends(desc_2016, desc_2017)
                    sorted_tfidf_delta = compare_tfidfs(tfidf_2016,
tfidf_2017)
                    trending_words = []
                    file = open("{}/{}.txt".format(dest_dir, filename[:-
4]), "w")
                    print("Words in 2016: {}".format(len(tfidf_2016)),
file=file)
                    print("Words in 2017: {}".format(len(tfidf_2017)),
file=file)
                    print("Common words:
{}".format(len(sorted_tfidf_delta)), file=file)
                    for keyword in sorted_tfidf_delta:
                        print("{}\t{}".format(keyword[0], keyword[1]),
file=file)
                        if keyword[1] > 0:
                            trending_words.append(keyword[0])
                    file.close()
                    # Building dendrogram
                    if trending_words:
                        try:
                            trending_tfidf_vectors =
compute_tfidf_vectors(desc_2019, trending_words)
                            plt.figure(figsize=(100, 40))
                            plt.title(filename[:-4], fontsize=13)
                            plt.xlabel('Trending Words', fontsize=10)
                            plt.ylabel('Cosine Distance of TF-IDF vectors',
fontsize=10)
                            distances = linkage(trending_tfidf_vectors,
'average')
                            dendrogram(distances, labels=trending_words,
leaf_font_size=8)
                            plt.savefig('{}/{}.png'.format(dest_dir,
filename[:-4]))
                        except:
                            continue
```

Додаток 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

АЛГОРИТМІЧНО-ПРОГРАМНИЙ АДАПТОВАНИЙ МЕТОД АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ТРЕНДІВ В ТЕКСТОВИХ ОГОЛОШЕННЯХ ПРО ВАКАНСІЇ

Виконала: Єрастова Влада Юріївна

Науковий керівник: доцент, к.т.н. Заболотня Тетяна Миколаївна

Київ – 2020

АКТУАЛЬНІСТЬ

- Інформація про майбутні тренди вважається цінним джерелом знань як для компаній, так і для приватних осіб.
- Доступні обсяги даних, які є занадто великими для обробки людьми вручну.
- Ринок праці – перспективна сфера для застосування методик виявлення трендів.
- Існуючі програмні рішення є достатньо універсально спрямованими.

ОБЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об’єкт дослідження: процес виявлення трендів в текстових даних.

Предмет дослідження: методи та алгоритми обробки текстових даних оголошень про вакансії для виявлення трендів.

Наукове завдання: розробити метод виявлення трендів в текстових оголошеннях про вакансії.

Мета дослідження: підвищення ефективності виявлення трендів в текстових оголошеннях про вакансії шляхом врахування особливостей вмісту описів цих вакансій.

Окремі завдання

1. Дослідити існуючі методи та системи виявлення трендів.
2. Проаналізувати специфіку формату та вмісту текстових оголошень про вакансії.
3. Розробити метод виявлення трендів в текстових даних описів вакансій з урахуванням специфіки останніх.
4. Здійснити аналіз та вибір існуючих засобів для програмної реалізації методу.
5. Розробити програмне забезпечення для виявлення трендів в описах вакансій.
6. Здійснити тестування реалізації методу та аналіз отриманих результатів.
7. Побудувати бізнес-модель для представлення структурних, операційних та фінансових механізмів роботи для створення програмного продукту.

ІСНУЮЧІ ПІДХОДИ ДО ЗАДАЧІ ВИЯВЛЕННЯ ТРЕНДІВ

1. Напівавтоматичні та автоматичні системи
   - Technology Opportunities Analysis (TOA)
   - TimeMines
   - PatentMiner
2. Комерційні
   - Autonomy
   - SPSS LexiQuest
   - ClearForest

БАЗОВИЙ МЕТОД ВИЯВЛЕННЯ ТРЕНДІВ

Метод запропонований дослідниками P. Hennig, P. Berger, C. Meinel (стаття “Identify Emergent Trends based on the Blogosphere”), що дозволяє визначити найважливіші слова в корпусі документів блог-постів та прослідкувати зміни показників вагомості цих слів з часом, на основі чого можна зробити висновки, чи є слово трендом на даний момент.

БАЗОВИЙ МЕТОД ВИЯВЛЕННЯ ТРЕНДІВ. КЛЮЧОВІ ФОРМУЛИ

Отримання найвагоміших слів чи фраз із вмісту тексту в певний період часу:

$$TFIDF(term_i, p_{tj}) = \frac{TF(term_i, P_{tj})}{TP(term_i, P_{tj})} \times \log \frac{|P|}{PF(term_i, P)} \quad (1)$$

TF обчислює частоту термінів у наборі повідомлень P_{tj} всередині часового проміжку, TP повертає кількість термінів у наборі повідомлень P_{tj}, P— кількість повідомлень, PF - кількість повідомлень, що містять цей термін у загальному корпусі.

БАЗОВИЙ МЕТОД ВИЯВЛЕННЯ ТРЕНДІВ. КЛЮЧОВІ ФОРМУЛИ

Для моніторингу змін показників важливості у часі використовуються характеристики лінійної регресії: нахил та вільний коефіцієнт:

$$Slope = \frac{\sum_{j=1}^{n}(y_{tj}-\overline{y})(x_j-\overline{x})}{\sum_{j=1}^{n}(x_j-\overline{x})^2} \quad (2)$$

$$Intercept = \overline{y} - Slope \times \overline{x} \quad (3)$$

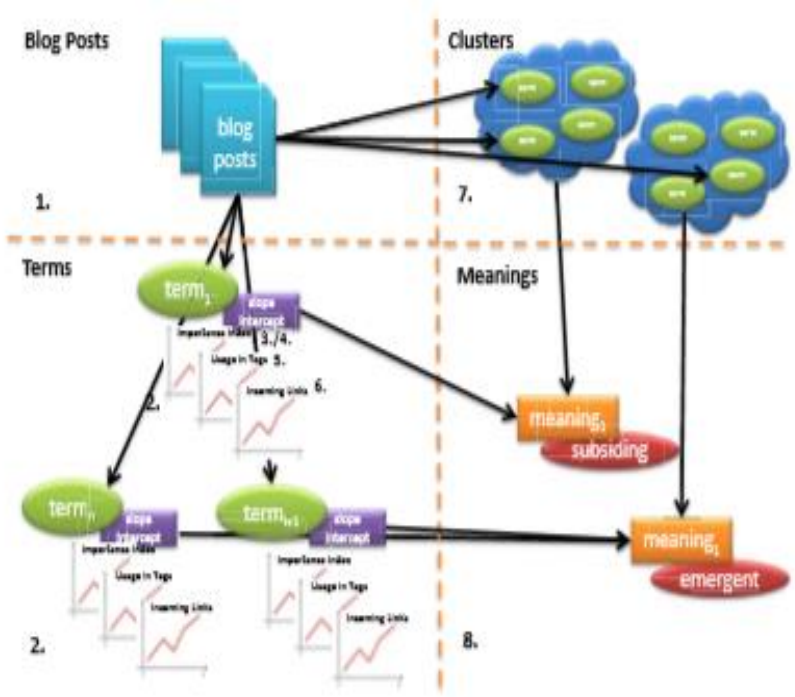
БАЗОВИЙ МЕТОД ВИЯВЛЕННЯ ТРЕНДІВ. КЛЮЧОВІ ЕТАПИ

I. Збір даних блог-постів

II. Розрахунок лінії тренду
   - Розрахунок індексу важливості для знаходження найбільш важливих слів
   - Розрахунок нахилу та вільного коефіцієнта для індексу важливості
   - Розрахунок нахилу та вільного коефіцієнта для структури посилань
   - Розрахунок нахилу та вільного коефіцієнта для тегів

III. Кластеризація

IV. Присвоєння значень



ОСОБЛИВОСТІ ВМІСТУ ОПИСІВ ВАКАНСІЙ

- Написані діловою мовою
- Не використовуються художні звороти та прояви емоцій
- Використовується різна термінологія, в залежності від сфери діяльності, в якій представлена вакансія
- Інформація подається стисло у вигляді понять в називному відмінку
- Мають визначену структуру

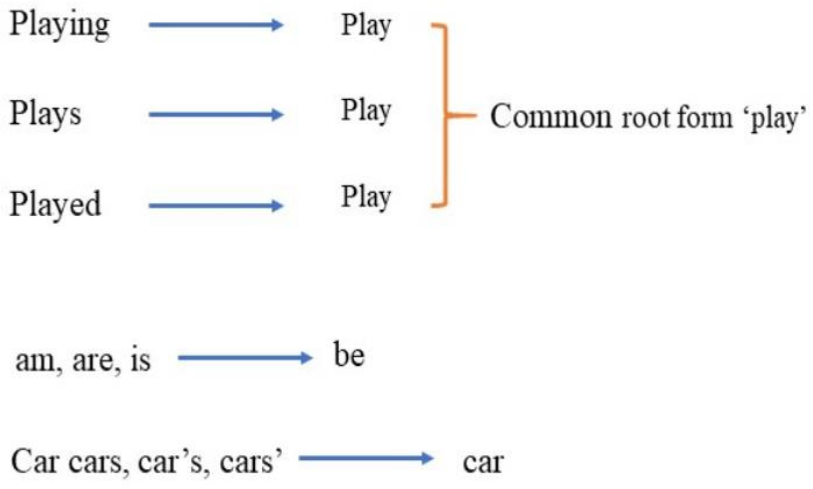
ЕТАПИ ПЕРЕДОБРОБЛЕННЯ ВХІДНИХ ТЕКСТОВИХ ДАНИХ. ВИДАЛЕННЯ СТОП-СЛІВ

До типових загальних шумових слів належать:

- цифри
- окремо розташовані знаки пунктуації: . , = + /! "; :%? * ()
- окремо розташовані букви алфавіту
- займенники, дієприкметники, прийменники, вигуки, суфікси і поєднання букв: без, більш, б, був, була, були, було, бути , вам, вас, адже, весь, вздовж, замість, поза, вниз, внизу, всередині, під, навколо, от, все, завжди, все, всіх, ви, де, да, давай, давати, навіть, для, до і т. д.
- нецензурна мова

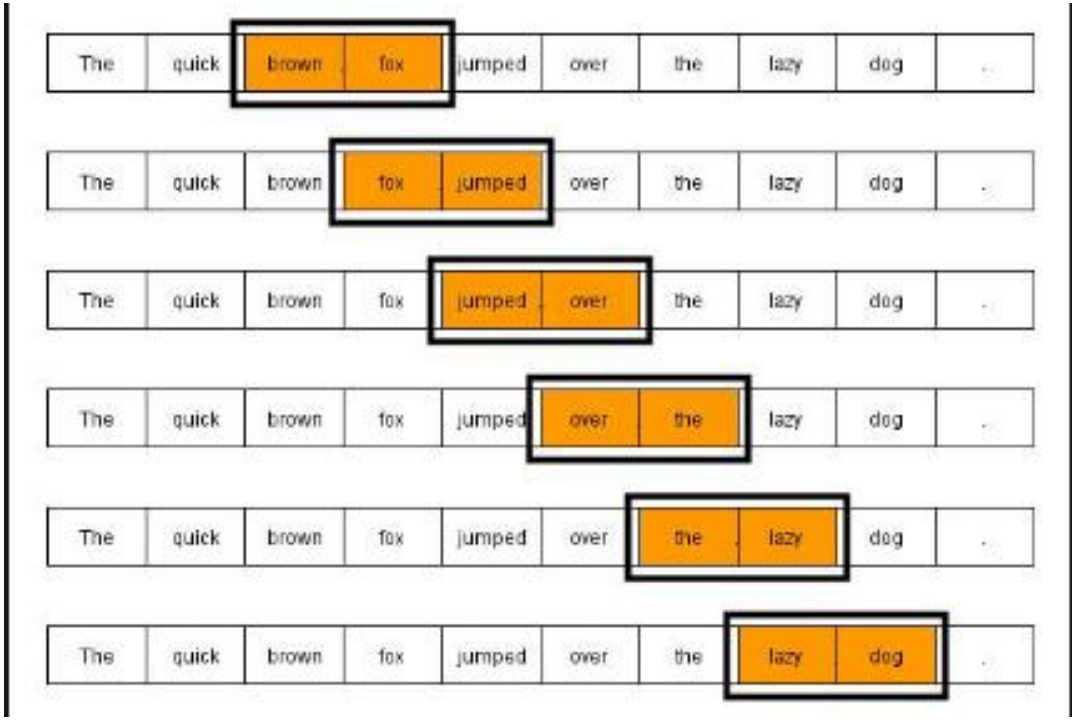
ЕТАПИ ПЕРЕДОБРОБЛЕННЯ ВХІДНИХ ТЕКСТОВИХ ДАНИХ. ЛЕМАТИЗАЦІЯ

Автоматизована лематизація здійснюється за допомогою спеціального програмного забезпечення – лематизаторів. Багато з них представлені у мережі Інтернет у відкритому доступі і можуть використовуватися безкоштовно.



ЕТАПИ ПЕРЕДОБРОБЛЕННЯ ВХІДНИХ ТЕКСТОВИХ ДАНИХ. ФОРМУВАННЯ N-ГРАМ

Для покращення наповнюваності отриманого словника наступним кроком пропонується провести формування n-грам.



ЕТАПИ ПЕРЕДОБРОБЛЕННЯ ВХІДНИХ ТЕКСТОВИХ ДАНИХ. ФІЛЬТРАЦІЯ ІМЕНОВАНИХ ОБ'ЄКТІВ ТА ЧАСТИН МОВ

- Власні назви не повинні відображатись як тренди.
- Пропонується зберігати лише слова, позначені як іменники чи прикметники, або частину мови яких неможливо визначити (наприклад, для n-грам).

МЕТОД АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ТРЕНДІВ В ТЕКСТОВИХ ОПИСАХ ВАКАНСІЙ

I. Збір даних оголошень про вакансії

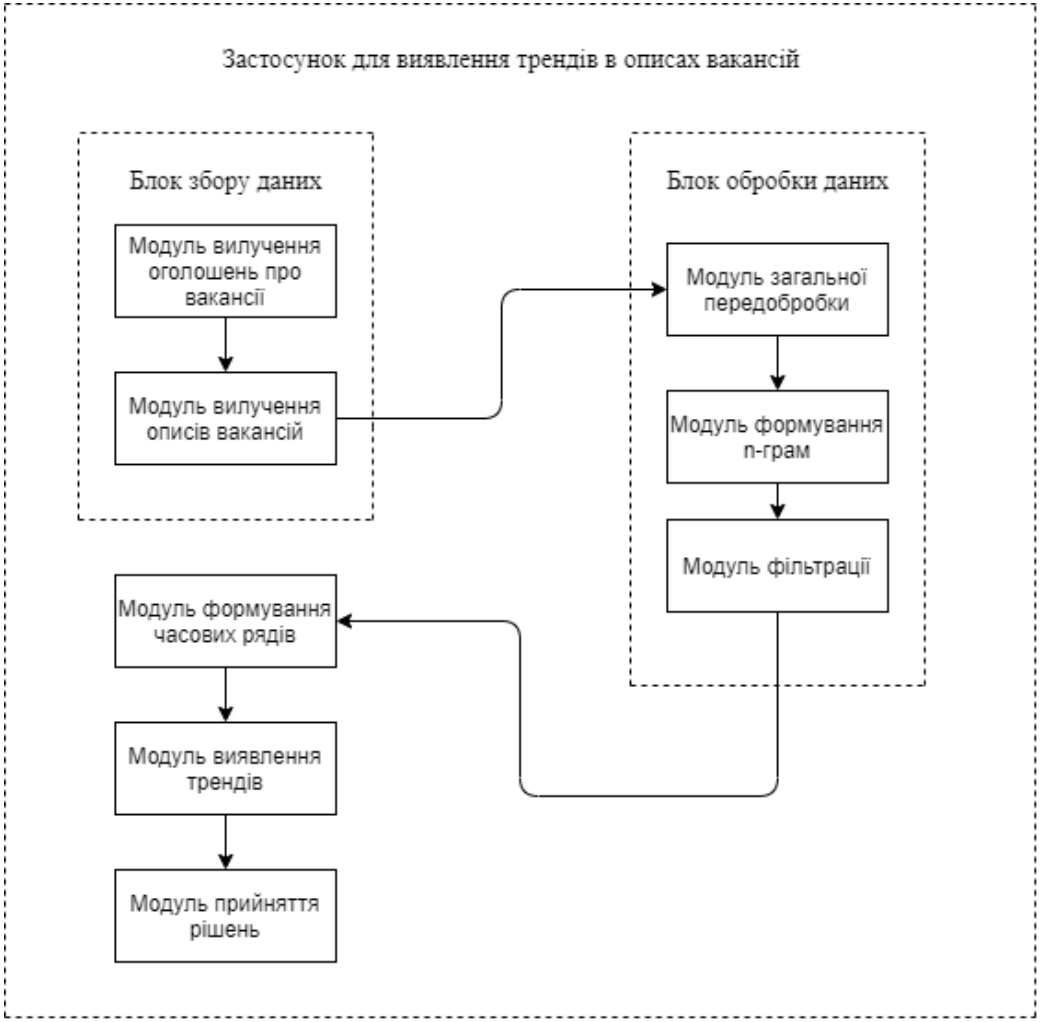
II. Попереднє оброблення
- видалення стоп-слів, пунктуації, приведення до нижнього регістру;
- лематизація;
- формування n-грам;
- фільтрування іменованих об’єктів;
- фільтрування частин мов.

III. Розрахунок лінії тренду
- Розрахунок індексу важливості для знаходження найбільш важливих слів
- Розрахунок нахилу та вільного коефіцієнта для індексу важливості

IV. Кластеризація

V. Присвоєння значень

АРХІТЕКТУРА РОЗРОБЛЕНОГО ПЗ



ЗБІР ДАНИХ

```
https://api.adzuna.com/v1/api/jobs/gb/search/1?app_id={YOUR_APP_ID}&app_key={YOUR_APP_KEY}
```

__CLASS__
Adzuna::API::Response::JobSearchResults
count
634916
mean
34735.59
results

0. **__CLASS__**
Adzuna::API::Response::Job
adref
eyJhbGciOiJIUzI1NiJ9.eyJpIjoiMTUwOTA5MDI2OSIsInMiOiJHS0E0WEdGMDZoR1NTNX
category

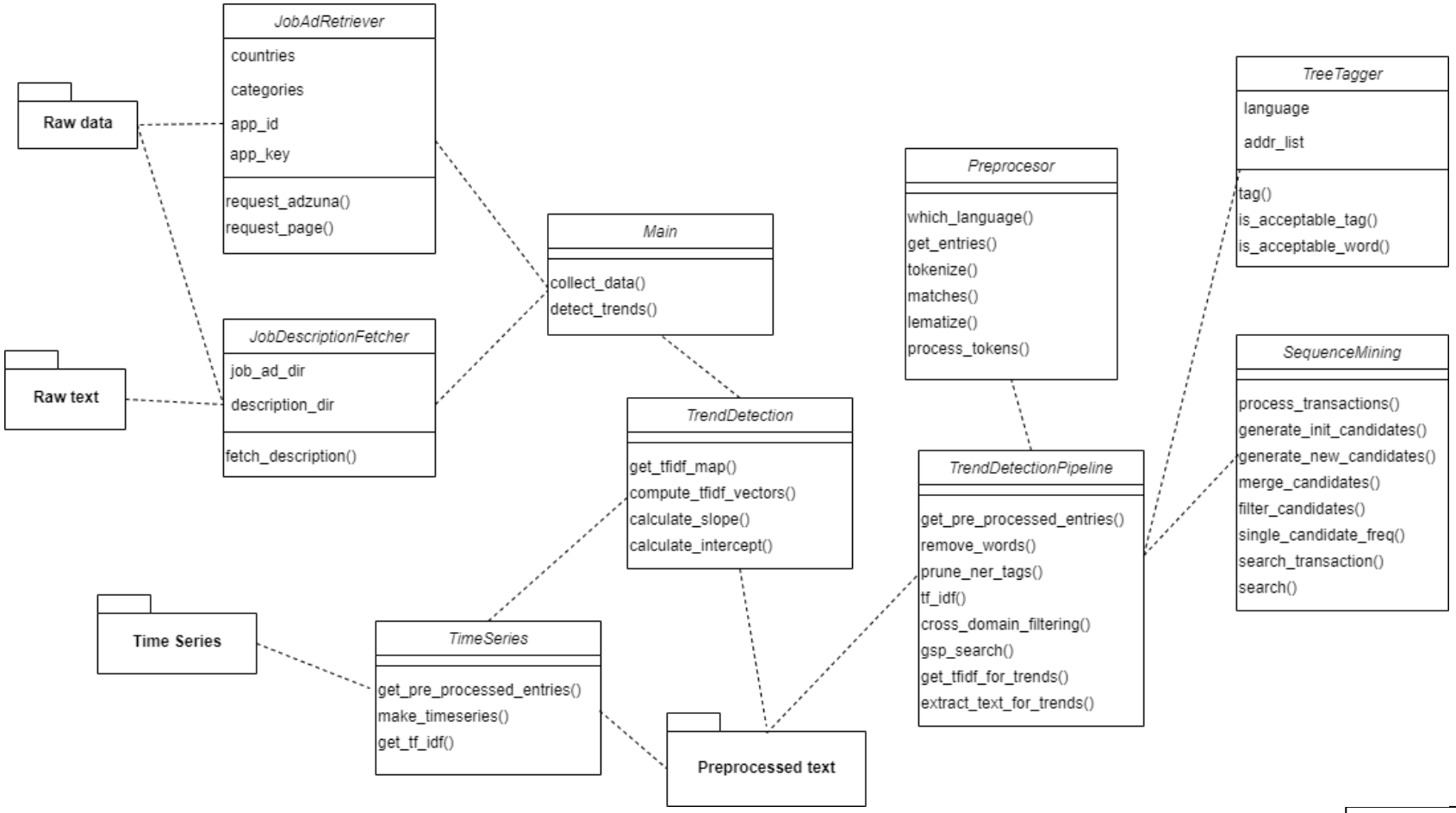
__CLASS__
Adzuna::API::Response::Category
label
PR, Advertising & Marketing Jobs
tag
pr-advertising-marketing-jobs

company

__CLASS__
Adzuna::API::Response::Company
display_name
TikTok

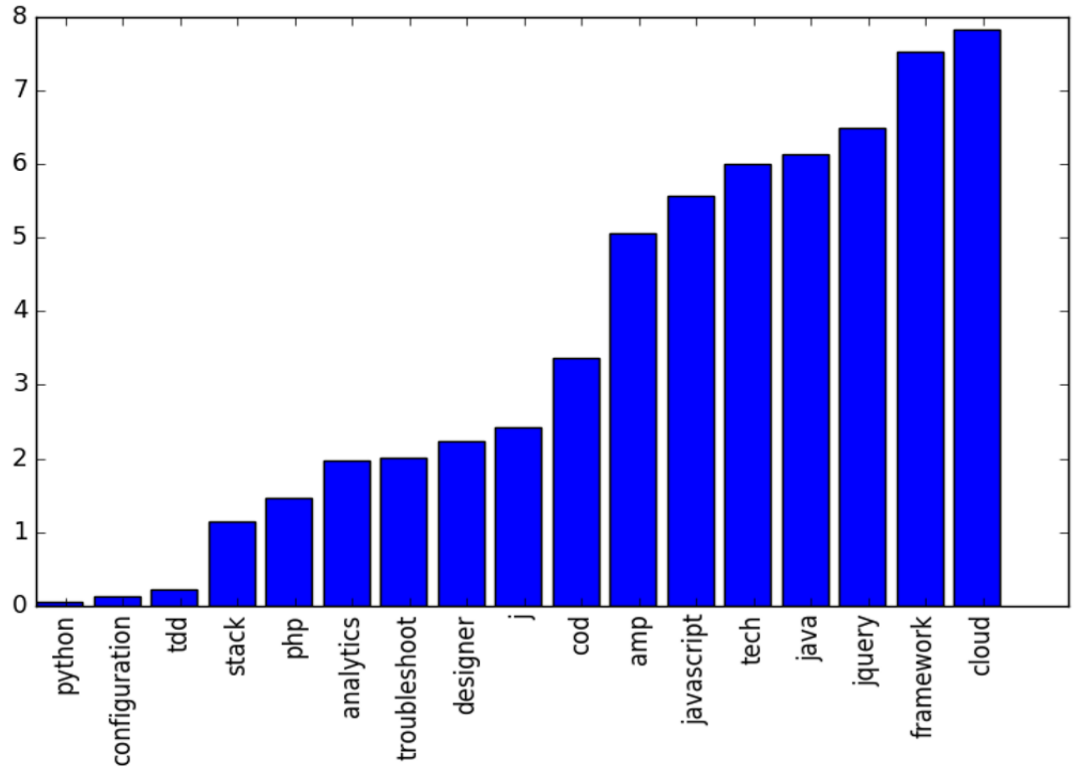
contract_time
full_time
created
2020-03-31T13:03:27Z
description
Client Solutions Executive Ready to grow with the biggest name in short-form mobile video? Start exploring the possibilities in campaign management at TikTok. You will be located in London City Centre with regular client travel. Create. Spread Joy. Build Brands.

ЗАГАЛЬНА ДІАГРАМА КЛАСІВ

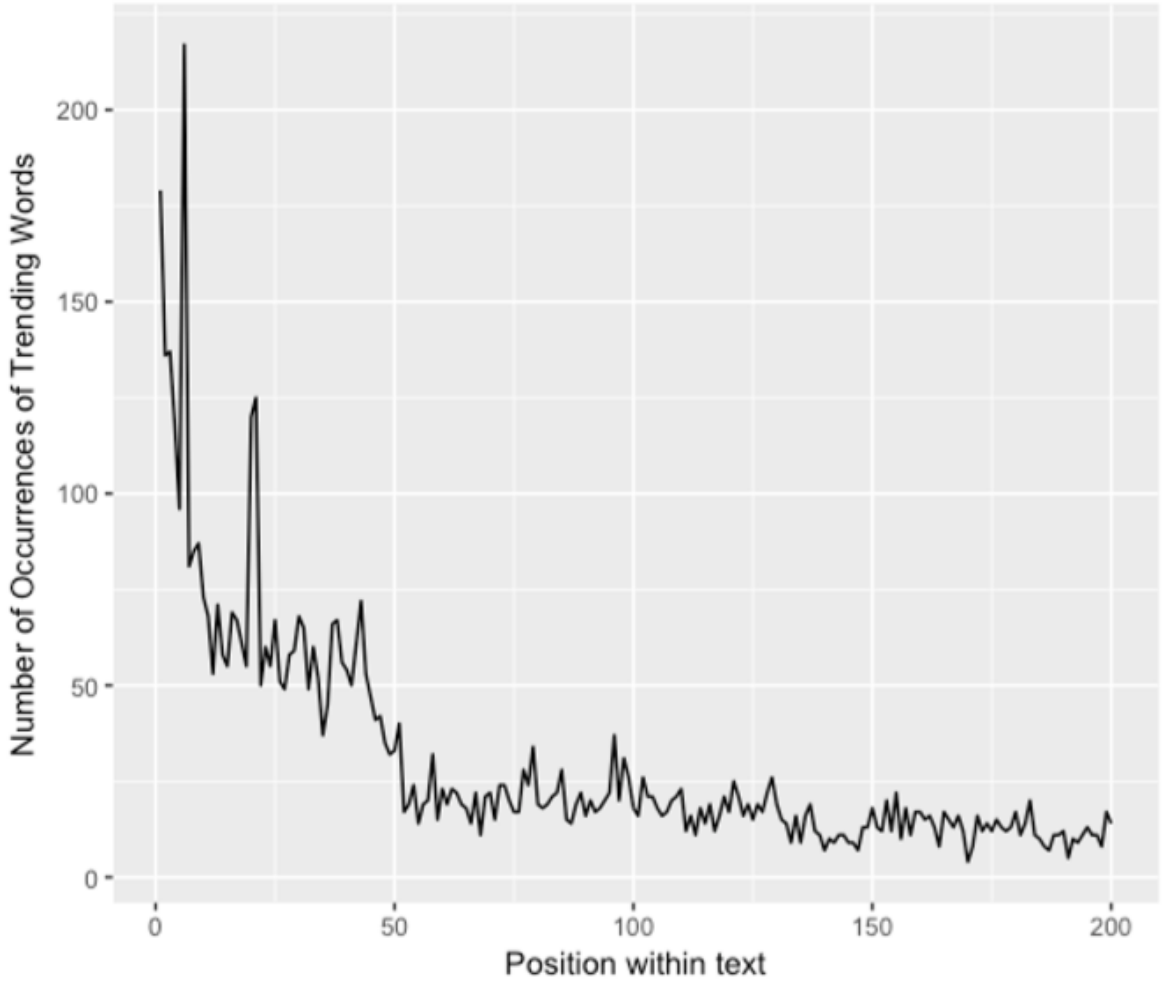


АНАЛІЗ РЕЗУЛЬТАТІВ. ТРЕНДОВІ СЛОВА

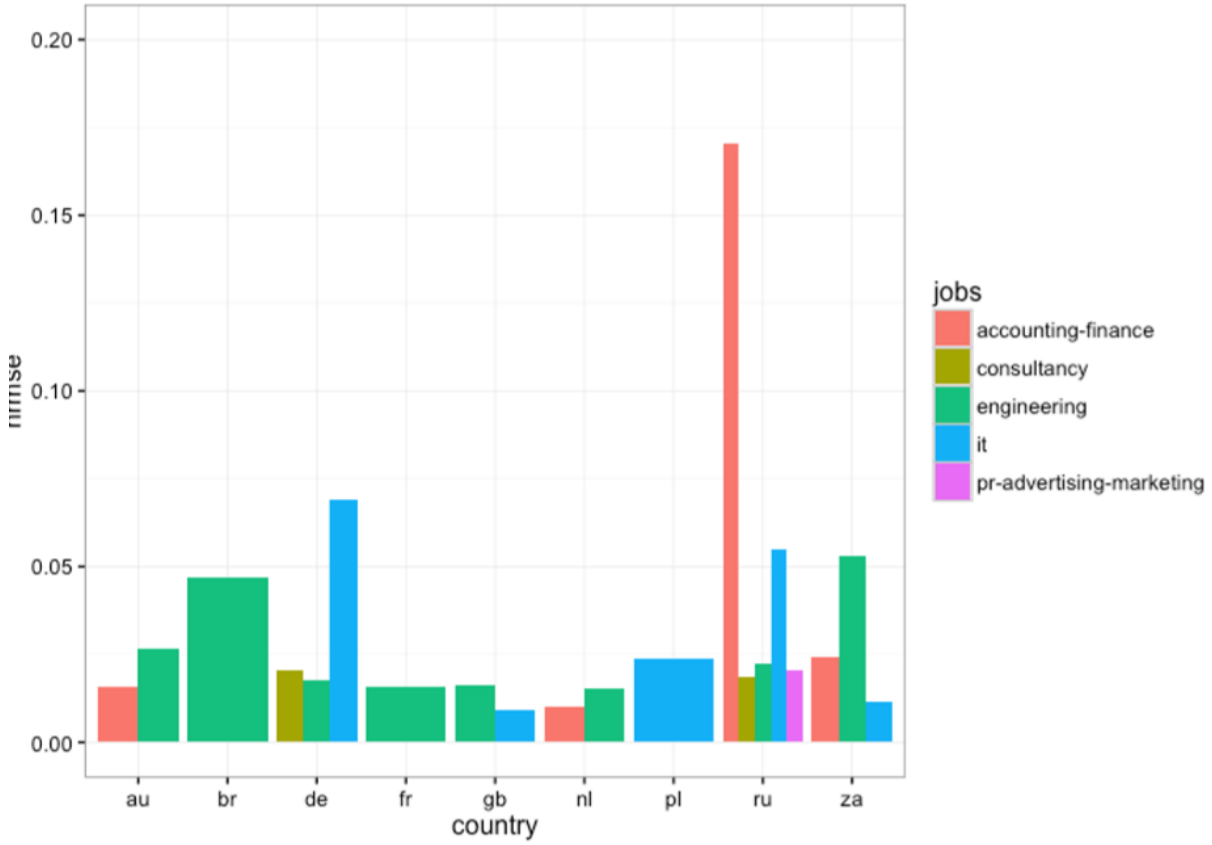
Аналіз результатів зосереджений на описах вакансій у Великобританії в категорії “Інформаційні технології”.



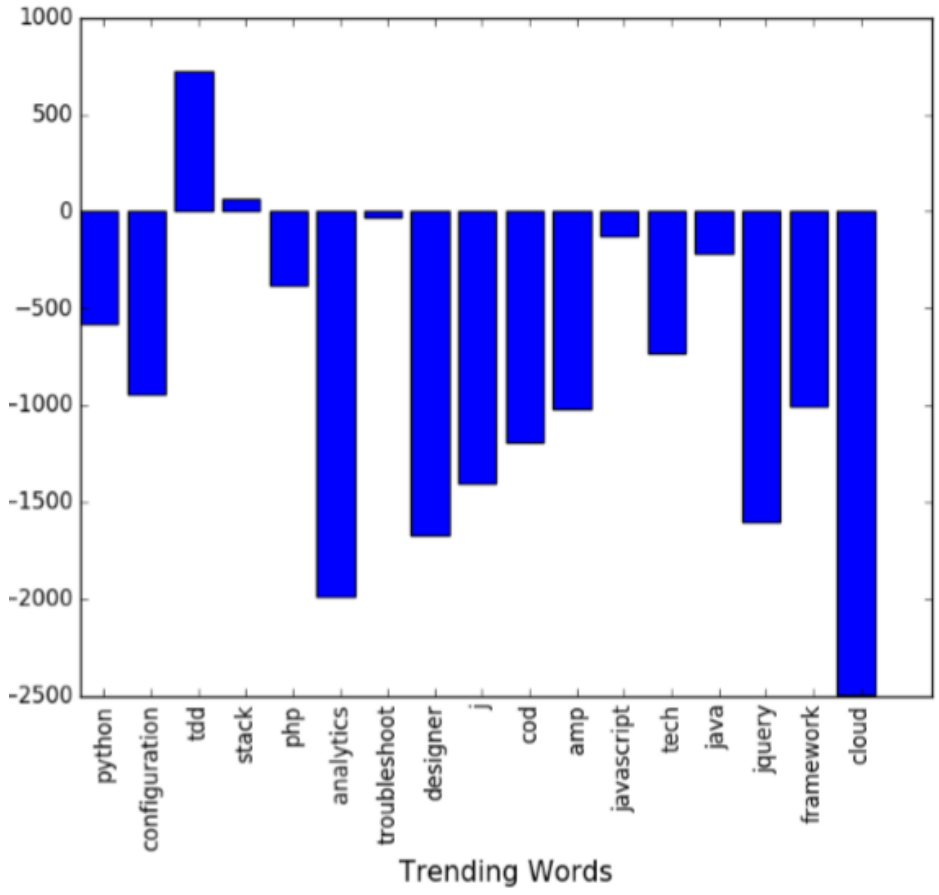
АНАЛІЗ РЕЗУЛЬТАТІВ. РОЗПОДІЛ ТРЕНДОВИХ СЛІВ



АНАЛІЗ РЕЗУЛЬТАТІВ. ПЕРЕДБАЧЕННЯ РОЗПОДІЛУ ПОЗИЦІЙ ТРЕНДОВИХ СЛІВ



АНАЛІЗ РЕЗУЛЬТАТІВ. ПОРІВНЯННЯ З GOOGLE TRENDS



БІЗНЕС-МОДЕЛЬ

<table>
<tr><th>Проблема</th><th>Рішення</th><th>Ціннісна пропозиція</th><th>Відношення з клієнтами</th><th>Сегменти користувачів</th></tr>
<tr><td rowspan="2">1. Відсутність автоматизованих рішень
2. Висока вартість людських послуг
3. Великі обсяги інформації
4. Людський фактор</td><td>1. Програмний застосунок, що реалізує метод автоматизованого виявлення трендів в описах вакансій на основі аналізу природомовних людський текстів.</td><td rowspan="2">1. Програмне забезпечення, що реалізує новий метод автоматичного виявлення трендів в описах вакансій.
2. Зменшення витрат компаній на аналіз потоків платформ для пошуку роботи.</td><td>1. Проведення презентацій, участь у спеціалізованих конференціях і форумах
2. Технічна підтримка</td><td rowspan="2">1. HR-відділи
2. Компанії по рекрутингу
3. Користувачі платформ по пошуку роботи
4.Платформи по пошуку роботи.
5. Організатори навчальних курсів.
6. Редактори професійних журналів та блогів.</td></tr>
<tr><td>Ключові метрики
1. Кількість проданих ліцензій</td><td>Канали
Дідділи співпраці великих компаній по пошуку роботи, HR-відділи.</td></tr>
<tr><th colspan="3">Структура витрат</th><th colspan="2">Джерела доходу</th></tr>
<tr><td colspan="3">1. Оренда приміщення, з/п персоналу, податки.</td><td colspan="2">1. Доходи від продажу ліцензій</td></tr>
</table>

НАУКОВА НОВИЗНА

- Запропоновано модифікацію методу виявлення трендів, що відрізняється від базового наявністю кроків обробки даних, які допомагають врахувати специфіку формату описів вакансій, чим покращують результати виявлення трендів.

ВИСНОВКИ

1. Проаналізовано існуючі підходи до виявлення трендів у текстових даних
2. Запропоновано застосувати методи виявлення трендів для нового типу вхідних текстових даних
3. Проведено вивчення специфіки оголошень про вакансії в мережі Інтернет на відповідних платформах для пошуку роботи
4. Сформовано перелік їх характеристик, які можуть бути враховані під час виявлення трендів

ВИСНОВКИ
(продовження)

5. Проведено аналіз процесу виявлення трендів в текстових даних з точки зору можливості урахування в ньому специфіки оголошень про вакансії, а саме запропоновано модифікацію етапів передоброблення тексту.
6. Розроблено метод автоматизованого виявлення трендів в описах вакансій, доступних в мережі Інтернет на основі комплексного урахування специфіки таких текстових даних.
7. Реалізоване програмне забезпечення для виявлення трендів в описах вакансій, яке реалізує запропонований метод.
8. Проведено аналіз результатів роботи розробленого програмного забезпечення на зібраних даних з платформи для розміщення вакансій Adzuna.

АПРОБУВАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

1. Основні положення і результати роботи були представлені та обговорювались на XII науково-практичній конференції магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК-2019 (Київ, 13-15 листопада 2019 р.) та опубліковані у збірнику тез за результатами конференції.

Дякую за увагу!