МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о.завідувача кафедри
_____ Сафонова С.О.
«____»_____ 2020 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Технологія аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 122 – "Комп'ютерні науки"

Науковий керівник роботи: _____ (підпис) Білобородова Т.О.. (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Критська Я.О. (ініціали, прізвище)

Студент: _____ (підпис) Давіденко М.О. (ініціали, прізвище)

Група: КН-18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень Магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 122 – "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри
_____С.О. Сафонова
«_____» _____2020 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Давіденку Микиті Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Технологія аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації

керівник проекту (роботи) к.т.н. Білобородова Тетяна Олександрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 11" 10 2019 р. № 35/15.15
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Матеріали науково-дослідної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз галузі застосування та технології аналізу текстового потоку в режимі реального часу та визначення етапів технології аналізу текстового потоку
2. Розробка програмного комплексу для аналізу текстового потоку в режимі реального часу;
3. Тестування запропонованої технології аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці;
4. Охорона праці в галузі;
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст. викл. Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд та аналіз вимог до роботи.	02.09.19-19.09.19	
2	Аналіз досліджень технології аналізу потоку текстової інформації в режимі реального часу в галузі виявлення фальсифікованих новин	22.09.19-10.10.19	
3	Розробка програмного комплексу для аналізу текстового потоку в режимі реального часу	11.10.19-15.11.19	
4	Тестування запропонованої технології аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці	16.11.18-09.12.19	
5	Дослідження та аналіз отриманих результатів	10.12.19-15.12.19	
6	Розробка заходів з охороні праці.	16.12.19-20.12.19	
7	Оформлення пояснювальної записки.	21.12.19-31.12.19	
8	Підготовка та подання магістерської роботи до захисту.	02.01.20-08.01.20	

Студент _____ Давіденко М.О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Білобородова Т. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Давіденко М.О. Технологія аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації.

В роботі представлено вирішення задачі підвищення точності аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації в умовах нерозмічених даних за рахунок удосконалення сукупності методів та инструментів технології аналізу текстового потоку. Розглянуто та проаналізовано галузь застосування та проблеми аналізу текстового потоку. Досліджено технологію аналізу текстового потоку в режимі реального часу. Виконано аналіз методів, технологій вилучення, передобробки та аналізу вилученної інформації текстового потоку. Досліджено процес аналізу текстового потоку з використанням технологій передобробки текстового потоку TF-IDF, word2vec, doc2vec та методу кластеризації для аналізу вилученної інформації. Представлена практична реалізація наступних етапів: розроблення програмного комплексу для аналізу текстового потоку в режимі реального часу, тестування запропонованої технології аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці. Якість виявлення фальсифікованих новин в текстовому потоці визначена з використанням критеріїв точності кластеризації.

Ключові слова: аналіз тексту, TF-IDF, word2vec, doc2vec, кластеризація, DBSCAN, fake news recognition

АННОТАЦИЯ

Давиденко Н.А. Технология анализа потока текстовой информации в режиме реального времени с использованием методов кластеризации.

В работе представлено решение задачи повышения точности анализа потока текстовой информации в режиме реального времени с использованием методов кластеризации в условиях неразмеченных данных за счет совершенствования совокупности методов и инструментов технологии анализа текстового потока. Рассмотрены и проанализированы область применения и проблемы анализа текстового потока. Исследована технология анализа текстового потока в режиме реального времени. Выполнен анализ методов, технологий извлечения, предобработки и анализа извлеченной информации текстового потока. Исследован процесс анализа текстового потока с использованием технологий предобработки текстового потока TF-IDF, word2vec, doc2vec и метода кластеризации для анализа извлеченной информации. Представлена

практическая реализация следующих этапов: разработка программного комплекса для анализа текстового потока в режиме реального времени, тестирование предложенной технологии анализа текстового потока путем проведения эксперимента по выявлению фальсифицированных новостей в текстовом потоке. Качество обнаружения фальсифицированных новостей в текстовом потоке определено с использованием критериев точности кластеризации.

Ключевые слова: анализ текста, TF-IDF, word2vec, doc2vec, кластеризация, DBSCAN, fake news recognition.

ABSTRACT

Davidenko Mykyta. Technology of real-time text mining using clustering.

The paper presents solutions to the problem of improving the accuracy of real-time analysis of text flow using clustering methods in unlabeled data by improving the set of methods and the tool of text flow analysis technology. The scope and problems of text flow analysis are considered and analyzed. The technology of text flow analysis in real time is investigated. The analysis of methods, technologies of extraction, pre-processing and analysis of extraction of information of a text stream is carried out. The process of text flow analysis using TF-IDF, word2vec, doc2vec, and clustering techniques for analyzing information retrieval was investigated. The practical implementation of the following steps is presented: development of a software complex for the analysis of text flow in real time, testing of the proposed technology of text flow analysis by conducting the experiment of detection of fake news in the text stream. The quality of detection of fake news in a text stream is determined using clustering accuracy criteria.

Keywords: text mining, TF-IDF, word2vec, doc2vec, clustering, DBSCAN, fake news recognition.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК І СКОРОЧЕНЬ

ЗМІ	Засоби масової інформації
СУБД	Система управління базами даних
CBOW	Continuous Bag-of-Words
CSI	Capture, Score, Integrate
DBSCAN	Density-based spatial clustering of applications with noise
HDFS	Hadoop Distributed File System
KNN	k-nearest neighbor
NLP	Natural language processing
PV-DM	Distributed Memory version of Paragraph Vector
PV-DBOW	Distributed Bag of Words version of Paragraph Vector
RDD	Resilient Distributed Dataset
REST	Representational State Transfer
SQL	Structured Query Language
SVM	Support Vector Machine
TF-IDF	Term frequency - Inverse document frequency

ВСТУП

Останні дослідження показують, що 80% всіх даних є текстовими даними. Використання технології аналізу текстових даних створило справжню індустрію, що займається зберіганням, аналізом і витяганням текстових даних. Сучасний розвиток цієї технології направлений на створення великої бази даних текстових даних з сайтів соціальних мереж та вилучення з цих даних корисної інформації за різноманітними напрямами.

Аналіз текстової інформації передбачає три основні завдання: пошук інформації (збір відповідного тексту), вилучення інформації (виявлення інформації, що цікавить в цих даних), і аналіз вилученої нформації (виявлення нових зв’язків серед видобутої інформації).

Складності, пов'язані зі збором, підготовкою та аналізом неструктурованих текстових даних, роблять аналітику тексту унікальною сферою дослідження та застосування.

Актуальним напрямком аналізу текстової інформації в сучасному часі є застосування аналізу текстової інформації для виявлення фальсифікованих новин. Сучасний розвиток інформаційних технологій відкрив широкі можливості для маніпулювання настроєм та вподобаннями користувачами за допомогою засобів масової медійної інформації через надання фальшивої або недостовірної інформації. Таким чином, текстова інформації соціальних медіа відіграє дуже важливу роль впливаючи на соціальні, економічні, політичні області щоденного прийняття рішень.

Для збереження права людини на отримання правдивої інформації дуже важливо надати можливість диференціювати правдиву інформацію від фальсифікованої. В останні роки для виконання цієї мети набуло актуальності застосування аналізу текстової інформації в режимі реального часу з метою виявлення фальсифікованих новин, тексту.

Аналіз текстової інформації може проводитись з використанням розмічених даних, коли для навчання моделі аналізу тексту використовуються дані, що складаються з текстових даних і кожнен фрагмент текста помічений мітками правда або неправда. Але, розмітка текстових даних вимагає багато часу, ресурсів, спеціальних знань. Ці перепони потенціально впливають на зменшення розміру даних, що використовуються для навчання моделі. Для аналізу текстової інформації в умовах нерозмічених даних все більшої популярності набуває застосування методів кластеризації. Незважаючи на розвиток технології аналізу текстової інформації, результати аналізу нерозмічених текстових даних все ще потребують удосконалення.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне завдання** удосконалення технології аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації.

Об'єкт дослідження: процес вилучення, обробки, аналізу потоку текстової інформації.

Предмет дослідження: сукупність методів та інструментів для аналізу потоку текстової інформації в режимі реального часу.

Мета і завдання дослідження. Метою дослідження є підвищення точності аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації в умовах нерозмічених даних, дослідження методів, технологій вилучення, передобробки та аналізу текстового потоку.

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

– Аналіз галузі застосування та технології аналізу текстового потоку в режимі реального часу.

– Визначення етапів, сукупності методів та інструментів технології аналізу текстового потоку.

– Розробка програмного комплексу для аналізу текстового потоку в режимі реального часу.

– Тестування запропонованої технології аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці.

– Оцінка якості виявлення фальсифікованих новин в текстовому потоці.

Методи дослідження. Проведені в роботі дослідження основані на модель-орієнтованому підході аналізу текстової інформації, технологіях передобробки текстового потоку, методів кластеризації, що використовувався для аналізу виявленої інформації текстового потоку. Перевірка результатів дослідження ґрунтувалась на методах експерименту та порівняння, які використовувались при розробленні практичної частини дипломного проекту.

Особистий внесок здобувача полягає у розробці технології аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації, розробці програмного комплексу аналізу потоку текстової інформації в режимі реального часу, що дозволяє вирішити поставлені задачі. Усі основні результати отримані автором особисто.

Апробація матеріалів магістерської роботи. Основні положення, ідеї, та висновки магістерської роботи доповідалися та обговорювалися на IV регіональному форумі IT-Ідея (м. Сєвєродонецьк, 2018), III міжнародній конференції «Theoretical and Applied Computer Science and Information Technology – 2019», V регіональному форумі IT-Ідея (м. Сєвєродонецьк, 2019).

Практичне значення отриманих результатів полягає в тому, що основні наукові положення реалізовані у вигляді програмного комплексу, що складає систематизовану сукупність методів та інструментів для аналізу потоку текстової інформації, що утворюють

технологію аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації.

Публікації. За темою магістерської роботи з викладенням її результатів опубліковані дві наукових статті у наукових фаховових виданнях України; три тези доповідей всеукраїнських конференцій.

Структура та обсяг магістерської роботи. Кваліфікаційна магістерська робота складається із вступу, чотирьох розділів, висновків, переліку посилань. Загальний обсяг складає 144 сторінки, з яких основний текст на 84 сторінках, список використаних джерел із 33 найменувань на 3 сторінках та 2 додатків на 58 сторінок. Робота містить 4 таблиці, 24 рисунка.

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ АНАЛІЗУ ПОТОКУ ТЕКСТОВОЇ ІНФОРМАЦІЇ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Останні дослідження показують, що 80% всіх даних є текстовими даними [1]. Використання технології аналізу текстових даних створила справжню індустрію, що займається зберіганням, аналізом і витяганням текстових даних. Сучасний розвиток цієї технології направлений на створення великої бази даних текстових даних з сайтів соціальних мереж та вилучення з цих даних корисної інформації за різноманітними напрямами.

Аналіз тексту, іноді званий аналізом текстових даних, визначають [2] як автоматичне виявлення нової, раніше невідомої інформації з неструктурованих текстових даних. Аналіз тексту також може бути описаний як процес отримання якісної інформація з тексту. Цей процес передбачає три основні завдання: пошук інформації (збір відповідного тексту), вилучення інформації (виявлення інформації, що цікавить в цьому тексті), і аналіз вилученої нформації (виявлення нових зв’язків серед видобутої інформації).

Текстова аналітика зазнала впливу багатьох сфер та зробила вагомий внесок у багато дисциплін. Сучасні програми для аналізу тексту охоплюють багато дисциплін та завдань. Окрім того, що має мультидисциплінарне походження, текстова аналітика продовжує мати додатки та просуватися у багатьох галузях дослідження. Він перетинає багато галузей дослідження, включаючи:

- Бібліотекознавство та інформатику;
- Соціальні науки;
- Комп'ютерні науки;
- Бази даних;
- Вилучення даних;
- Статистичний аналіз;
- Штучний інтелект;
- Обчислювальну лінгвістику.

Більшість з цих напрямків сприяють сучасному розвитку аналізу текстової інформації. Деякі дослідники вважають аналіз тектової інформації галуззю інтелектуального аналізу даних [3].

Складності, пов'язані зі збором, підготовкою та аналізом неструктурованих текстових даних, роблять аналітику тексту унікальною сферою дослідження та застосування.

Актуальним напрямком аналізу текстової інформації в сучасному часі є застосування аналізу текстової інформації для виявлення фальсифікованих новин. Сучасний розвиток інформаційних технологій відкрив широкі можливості для маніпулювання настроєм та вподобаннями користувачами за допомогою засобів масової інформації (ЗМІ) через надання фальшивої або недостовірної інформації. Таким чином, текстова інформації соціальних медіа відіграє дуже важливу роль впливаючи на соціальні, економічні, політичні області щоденного прийняття рішень.

Для збереження права людини на отримання правдивої інформації дуже важливо надати можливість диференціювати правдиву інформацію від фальсифікованої. В останні роки для виконання цієї мети набуло актуальності застосування аналізу текстової інформації в режимі реального часу з метою виявлення фальсифікованих новин, тексту.

1.1 Передумови необхідності виявлення фальсифікованих новин

На сьогоднішній день значно зросла потреба в виявленні фальсифікованих новин. Це пов'язано із зростанням кількості даних, внаслідок чого проаналізувати їх вручну не представляється можливим. З кожним роком зростає число користувачів Інтернету, які поповнюють всесвітню павутину різними даними. Соціальні мережі дозволяють людям виказувати свою думку в новинах та передавати інформацію із засобів масової інформації своїй аудиторії. Тому соціальні медіа грають дуже важливу роль впливаючи на соціальні, економічні, політичні області щоденного прийняття рішень. Наприклад, під час президентських виборів в Україні 2019 року, кандидати ефективно використовували Facebook та Twitter для відправки повідомлень та вислову своїх думок для своїх прихильників. Хибна інформація, навмисно вироблена для маніпулювання читачами, називається дезінформацією для різних цілей, наприклад, для вказівки упередженого переконання або спокушання аудиторії перейти за посиланням. З іншого боку, дезінформація - це ненавмисна передача невірних фактів аудиторії, яка може сприймати надану інформацію як правдиву. Недавні дослідження показують, що фальшиві новини швидко поширюються через соціальні мережі (наприклад, ураган «Сенді» в 2012 році [4] і вибухи на Бостонському марафоні в 2013 році [5]), і це може негативно посилити громадський неспокій. Наприклад, в 2013 році 130 мільярдів доларів «вартості акцій» були раптово знищені після того, як пройшла новина, що Барак Обама був поранений в результаті вибуху в Білому домі [6]. Так, наприклад, “фальсифікованою новиною” є інформація про, нібито, заклик Дмитра Яроша до лідера чеченських бойовиків Доку Умарова, яку 1–2 березня 2014 року розповсюдили провідні російські ЗМІ. В подальшому ця новина була використана для легітимізації застосування Російською Федерацією своєї армії на території України. Це

свідчить про спланований та підготовлений завчасно характер цієї інформаційної операції. Нерідко поширення фальсифікованих новин пришвидшується завдяки реплікуванню журналістами та редакторами онлайн-платформ гучних та скандальних новинних матеріалів (що є типовим для фальсифікованих новин). Тобто, фальсифікована новина поширюється без належної перевірки фактів, з метою підвищення рейтингів чи відвідуваності інформаційних ресурсів. В цілому, гонитва окремих ЗМІ за увагою споживачів інформації істотно сприяє здійсненню впливу на широку аудиторію. Також соціологічне опитування та тест USAID-Internews "Ставлення населення до ЗМІ та споживання різних типів медіа у 2019 році" [7] показало що в Україні лише 11% громадян можуть правильно відрізнити правдиві новини від дезінформації. Основна мета дослідження - вивчити звички українців, які стосуються споживання медіа контенту, а також їхню довіру до ЗМІ, оцінити рівень медіа грамотності та обізнаність населення щодо впровадження реформ в Україні. Протягом червня-липня 2019 року представники InMind опитали 4056 респондентів із 12-ти областей. Похибка вибірки не перевищує 2,5% [8]. У 2018 році Україна фігурувала в 461 з тисячі випадків дезінформації в російських ЗМІ. Про це повідомляють аналітики проекту "EU vs Disinformation"[9]. Аналізуючи новини можливо виявляти чи вони фальсифіковані, та несуть ризики негативного впливу. Останнє виявляється корисним для уникання ситуацій, які можуть нанести шкоду людям, групам людей, державам, компаніям тощо. Нам необхідно завчасно виявити підроблені новини, перш ніж вони потраплять в онлайн-сховища з перевіркою фактів, потенційно використовуючи виключно їх контент.

1.2 Класифікація видів фальсифікованих новин

Класифікувати фальсифіковану інформацію можна за різними ознаками. Проаналізувавши найтиповіші приклади фальсифікованих новин, виділено такі критерії, які можуть бути основою класифікації: за формою подання (текст, фото, відео, голосовий запис); за змістом (агітація, пропаганда, маніпуляція тощо); за тематикою (політичні, соціальні, світські, тощо); за призначенням для певної вікової категорії (для молоді, для зрілих людей, для пенсіонерів тощо); за джерелом інформації (від першого джерела, без джерела, невідоме джерело і т. д.).

Зазвичай, фальсифіковану інформацію, пишуть з певною метою. Можна виділити декілька тиіви фальсифікацій у соціальних медіа в залежності від їх мети. Фальсифікації можуть бути такими, що:

- сіють паніку серед людей;

- розпалюють міжнаціональну (расову, релігійну тощо) ворожнечу;
- поширюють хибні думки для того, щоб заплутати нас, відвернути від правди;
- маніпулюють свідомістю;
- рекламують когось або щось;
- приносять прибуток засобу масової інформації, що його поширює («жовта преса»);
- плямують чиюсь репутацію (найчастіше це фото, добре опрацьовані у Photoshop);
- мають розважальний характер.

1.3 Аналіз методів, технологій попередньої обробки текстової інформації

Більшість описаних алгоритмів не працюють з словами у чистому виді, для коректної роботи потрібно приводити слова або документи до векторного представлення, щоб алгоритми основуючись на порівнянні відстаней між векторами, видавали коректний результат, далі розглянуто найпопулярніші з цих алгоритмів.

1.3.1 Технологія обробки текстової інформації TF-IDF

В основі технології TF-IDF полягає показник, який відображає, наскільки важливим є слово для документа в наборі документів. Він має багато застосувань, головне в автоматизованому аналізі тексту, і дуже корисний для зарахування слів в алгоритмах машинного навчання для обробки природних мов (NLP). TF-IDF був винайдений для пошуку документів та пошуку інформації. Він працює, збільшуючись пропорційно до кількості появи слова в документі, але компенсується кількістю документів, які містять слово. Наприклад використовується в аналітичному дослідженні щодо новинних ЗМІ в контексті політичної перевірки фактів та фальшивого виявлення новин[10]. В ньому порівнюється мова реальних новин з мовою сатири, миролюбства та пропаганди, щоб знайти мовну характеристику надійного тексту. Щоб перевірити доцільність автоматичної перевірки політичних фактів, представляється тематичне дослідження, засноване на PolitiFact.com, використовуючи їх судження про фактичність за 6-бальною шкалою. Експерименти показують, що хоча перевірка фактів засобів масової інформації залишається відкритим дослідницьким питанням, стилістичні підказки можуть допомогти визначити правдивість тексту.

1.3.2 Технологія Word2Vec

Word2Vec - це набір моделей для аналізу семантики природних мов, що представляє собою технологію, яка заснована на дистрибутивній семантиці і векторному поданні слів.

Робота цієї технології здійснюється наступним чином: word2vec приймає великий текстовий корпус в якості вхідних даних і зіставляє кожному слову вектор, видаючи координати слів на виході. Спочатку він створює словник, «навчаючись» на вхідних текстових даних, а потім обчислює векторне подання слів. Векторне подання ґрунтується на контекстній близькості: слова, що зустрічаються в тексті поруч з однаковими словами (а отже, мають схожий зміст), у векторному поданні матимуть близькі координати векторів-слів, як це представлено на рисунку 1.1. Отримані вектори-слова можуть бути використані для обробки природної мови та машинного навчання.

Рисунок 1.1 - Принцип векторного подання текстової інформації

Представлення word2vec створюється за допомогою 2 алгоритмів: Модель Continuous Bag-of-Words (CBOW) та модель Skip-Gram [11].

CBOW створює розсувне вікно навколо поточного слова, щоб передбачити його з "контексту" - оточуючих слів. Кожне слово представлено у вигляді вектора ознак. Після

тренування ці вектори перетворюються на слова вектори. Принцип роботи алгоритму CBOW представлено на рисунку 1.2.

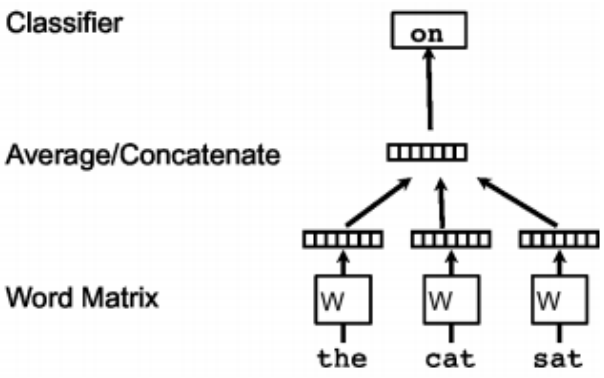


Рисунок 1.2 - Ескіз алгоритму CBOW: слова "the" "cat" "sat" використовуються для прогнозування слова "on"

Другий алгоритм Skip-Gram абсолютно протилежний CBOW: замість того, щоб передбачати одне слово кожного разу, ми використовуємо 1 слово для передбачення всіх навколишніх слів («контекст»). Він набагато повільніший, ніж CBOW, але вважається більш точним з нечастими словами.

Технологія word2vec досить успішно використовується в виявлення фальсифікованих новин за допомогою ансамблю класифікаторів [12].

1.3.3 Технологія Doc2Vec

Метою doc2vec є створення векторного представлення документу незалежно від його довжини. Але на відміну від слів, документи не представлені такими логічними структурами, як слова, тому векторне представлення документів потребує дещо відмінних методів.

Концепція, яку запропоновали дослідники [13], є простою, але розумною: вони використали модель word2vec та додали ще один вектор (ідентифікатор абзацу), як це представлено на рисунку 1.3.

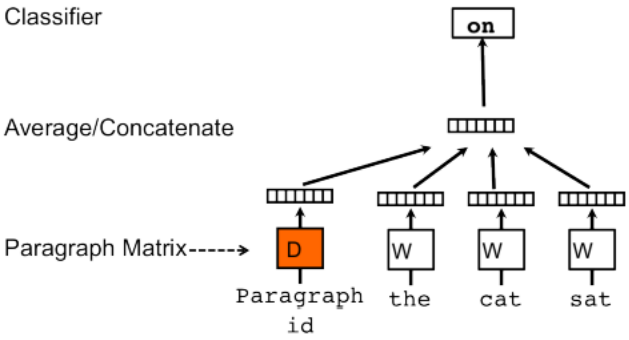


Рисунок 1.3 - PV-DM модель

Ця концепція є розширенням моделі CBOW. Але замість того, щоб використовувати просто слова для передбачення наступного слова, також додано ще один функціональний вектор, який є унікальним для документа.

Наведена на рисунку 1.4 модель називається Distributed Memory version of Paragraph Vector (PV-DM). Вона виступає як пам'ять, яка запам’ятовує те, чого не вистачає в поточному контексті, або як тема абзацу. Хоча вектори слова представляють поняття слова, векторний документ має намір представляти поняття документа.

Як і у word2vec, може бути використаний інший алгоритм, схожий на Skip-Gram, це є алгоритм Distributed Bag of Words version of Paragraph Vector (PV-DBOW) Рисунок. Тут цей алгоритм насправді швидший (на відміну від word2vec) і витрачає менше пам'яті, оскільки не потрібно зберігати слова вектори.

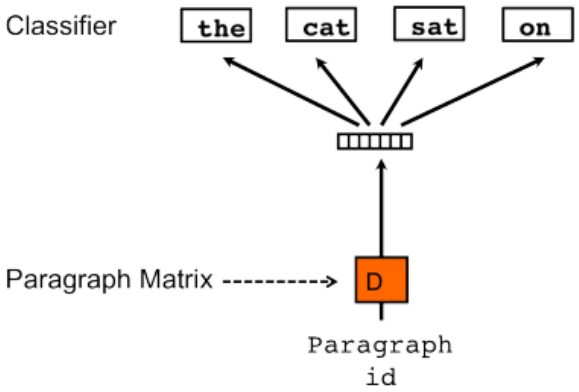


Рисунок 1.4 – PV-DBOW модель

Автори [13] рекомендують використовувати комбінацію обох алгоритмів, хоча модель PV-DM є найкращою і зазвичай сама по собі досягає кращих результатів.

Моделі doc2vec можна використовувати наступним чином: для навчання потрібен набір документів. Для кожного слова генерується вектор слова W, а для кожного документа

генерується вектор D. Модель також підготовлює ваги для прихованого шару SoftMax. На етапі логічного висновку, новий документ може бути представлений, а всі ваги фіксовані для обчислення вектора документа.

Doc2Vec використовується у дослідженні в якому пропонується модель CSI (Capture, Score, Integrate) [14]. Перший модуль заснований на відповіді та тексті; він використовує рекурентну нейронну мережу для зйомки тимчасового шаблону діяльності користувачів щодо наданої статті. Другий модуль вивчає характеристику ресурсів, що базується на поведінці користувачів, та обидва інтегруються з третім модулем для класифікації статті як фальсифікованої чи ні.

1.4 Аналіз підходів до виявлення фальсифікованих новин

Дослідження підходів до виявлення фальсифікованих новин можна визначити наступним чином: орієнтовані на дані, орієнтовані на особливості, модель-орієнтовані, орієнтовані на додатки (рисунок 1.5).

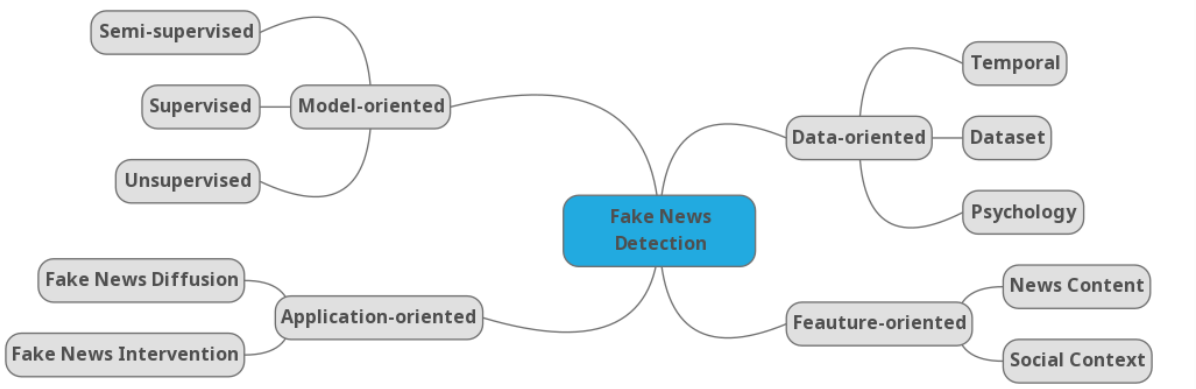


Рисунок 1.5 - Категорії підходів для виявлення фальсифікованих новин

1.4.1 Підходи орієнтовані на дані

Дослідження фальсифікованих новин, орієнтованих на дані, дають різні типи даних, такі як: набори даних, часові та психологічні. З точки зору набору даних було продемонстровано [15], що не існує жодного набору даних, що включає ресурси для вилучення всіх відповідних функцій. Перспективним напрямком є створення всебічного та великомасштабного набору даних з фальсифікованими новинами, який може бути використаний для полегшення подальших досліджень у цій галузі. Із часової точки зору фальшиве розповсюдження новин на

соціальних засобах масової інформації демонструє унікальні часові закономірності, що відрізняються від справжніх новин. У зв'язку з цим існує одна цікава проблема щодо виявлення фальшивих новин, що має на меті сповіщення про підроблені новини під час процесу поширення. Наприклад, такий підхід міг би розглядати лише соціальні медіа пости протягом певного часу затримки оригіналу публікації як джерела для перевірки новин.

1.4.2 Підходи, орієнтовані на особливості

Орієнтовані на особливості дослідження фальшивих новин спрямовані на визначення ефективних функцій для виявлення фальшивих новин з кількох джерел даних. Авторами [15] зазначено, що існує два основні джерела даних: зміст новин та соціальний контекст. З точки зору змісту новин було запроваджено лінгвістичні та візуальні методи вилучення функцій з текстової інформації. Лінгвістичні особливості були широко вивчені для загальних завдань обробки природнього мовлення, таких як класифікація тексту та кластеризація тексту, а також конкретні програми, такі як ідентифікація авторів та виявлення обману, але основні характеристики фальшивих новин не були глибоко зрозуміли. Крім того, методи вбудовування, такі як вбудовування слів і глибокі нейронні мережі, привертають велику увагу для вилучення текстурних функцій і мають потенціал для вивчення кращих уявлень. Окрім реклами, візуальні особливості, вилучені із зображень, також є важливими показниками фальшивих новин. Наприклад було проведене дуже обмежене дослідження з метою використання ефективних візуальних особливостей, включаючи традиційні локальні та глобальні особливості та нові глибокі мережеві особливості, для проблеми виявлення підроблених новин.

1.4.3 Підходи, орієнтовані на додатки

Дослідження фальсифікованих новин, орієнтовані на додатки, охоплюють дослідження, що переходять у інші сфери, що не стосуються виявлення фальсифікованих новин. Наприклад, запропоновано два основні напрямки [15]: Поширення фальсифікованих новин характеризує шляхи розповсюдження та зразки підроблених новин на сайтах соціальних медіа. Деякі ранні дослідження показали, що правдива інформація та неправдива інформація дотримуються різних моделей під час поширення в соціальних мережах. Аналогічно, розповсюдження фальшивих новин в соціальних засобах масової інформації демонструє власні характеристики, які потребують подальшого дослідження, такі як асоціальні виміри, життєвий цикл, ідентифікація

розповсюджувачів тощо. Соціальні виміри стосуються неоднорідності та слабкої залежності соціальних зв'язків у різних соціальних спільнотах. На сприйняття користувачами фальшивих новин сильно впливають їхні однодумці в соціальних мережах, в той час як ступінь споріднення відрізняється в різних соціальних аспектах. Таким чином, варто вивчити, чому і як різні соціальні виміри відіграють певну роль у поширенні фальшивих новин на різні теми, такі як політика, освіта, спорт тощо. Процес розповсюдження підроблених новин також має різні етапи з точки зору уваги людей і реакції з плином часу, що призводять до унікального життєвого циклу.

1.4.4 Модель-орієнтовані підходи

Модель-орієнтовані підходи виявлення фальсифікованих новин відкривають нові можливості до створення більш ефективних та практичних моделей для виявлення фальшивих новин. Більшість згаданих раніше підходів зосереджуються на витягуванні різних особливостей в керовані класифікаційні моделі, такі як Наївний Баєсів классифікатор, дерево рішень, логістична регресія, k найближчий сусід (KNN) та метод опорних векторів, а потім вибір класифікатора, який працює краще. Більше досліджень може бути проведено для створення більш складних та ефективних моделей та для кращого використання отриманих особливостей, таких як методи агрегації, імовірнісні методи, ансамблеві методи чи методи проекції. Більше того, більшість існуючих підходів є контрольованими, що вимагає попередньо анотованих фальсифікованих та правдивих новин та даних для встановлення моделі. Однак отримання надійного набору даних про фальсифіковані новини є дуже трудомістким, оскільки цей процес часто вимагає від експертів анотаторів ретельного аналізу заяв, додаткових доказів, контексту та звітів з авторитетних джерел.

У цих категоріях зазвичай використовуються наступні методи інтелектуального аналізу даних: контрольоване навчання – методи класифікації, неконтрольоване навчання – методи кластеризації та часткове навчання - методи ансамблю. Далі представлено результати проведеного аналізу методів модель-орієнтованого підходу та розглянуто основні алгоритми кожного методу.

1.5 Застосування методу класифікації для виявлення фальсифікованих новин

Як це зазначено вище, на даний час більшість досліджень з виявлення фальсифікованих новин спираються на методи класифікації модель-орієнтованого підходу. У дослідженні [16] розглядаються попередні та сучасні методи виявлення фальсифікованих новин у текстових

форматах, також описано, як і чому існують фальшиві новини. В статті розглянуто підходи Linguistic Cue і Network Analysis та запропановано трискладовий метод, що включає Наївний Баєсів алгоритм, алгоритм опорних векторів та семантичний аналіз як точний спосіб виявлення фальсифікованих новин у соціальних медіа.

Наївний Баєсів [17] алгоритм є простим імовірнісним класифікатором, заснованим на застосуванні теореми Байеса зі строгими (наївними) припущеннями про незалежність. Залежно від точної природи ймовірнісної моделі, наївні байєсівські класифікатори можуть навчатися дуже ефективно. У багатьох практичних додатках для оцінки параметрів для наївних байєсівих моделей використовують метод максимальної правдоподібності; іншими словами, можна працювати з наївною байєсівської моделлю, не вірячи в байєсіву ймовірність і не використовуючи байєсівські методи. Незважаючи на наївний вигляд і, безсумнівно, дуже спрощені умови, наївні байєсовські класифікатори часто працюють набагато краще в багатьох складних життєвих ситуаціях. Перевагою наївного байєсівського класифікатора є мала кількість даних необхідних для навчання, оцінки параметрів і класифікації.

Алгоритм опорних векторів (Support Vector Machine – SVM) вважається алгоритмом навчання, що контролюється. SVM працює, навчаючись із конкретними даними, вже організованими у дві різні категорії. Отже, модель будується після того, як вона вже пройшла навчання. Крім того, мета SVM полягає в тому, щоб розрізнити, до якої категорії підпадають нові дані, крім того, він також повинен максимально збільшити запас між двома класами. Оптимальна мета SVM знайти гіперплощину, яка розділяє набір даних на дві групи. Для подальшого розвитку, вектори підтримки є «точками даних, найближчими до гіперплощини», і якщо їх видалити, зміниться розташування розділювальної гіперплощини. Таким чином, вектори підтримки є важливими елементами набору даних. Крім того, гіперплощину можна розглядати як "лінію, яка лінійно відокремлює і класифікує набір даних" і "чим далі від гіперплощини лежать точки даних", тим більше шанс на точність класифікації точок даних.

Переваги використання методу SVM полягають у тому, що він, як правило, дуже точний та працює надзвичайно добре на наборах даних, які є меншими та стислими. Крім того, цей метод дуже гнучкий, оскільки його можна використовувати для класифікації або навіть визначення чисел. Також SVM має можливість обробляти великі об’єми і, як правило, є ефективними по використанню пам'яті.

Недоліками використання підходу SVM є те, що він має труднощі з великими наборами даних, оскільки «час навчання з SVM може бути високим», і він «менш ефективний для шумних (безглуздих) наборах даних з класами, що перетинаються».

Нейронні мережі не програмуються в звичному сенсі цього слова, вони навчаються. Можливість навчання - одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних і/або зашумлених, частково спотворених даних. Що дуже підходить для виявлення фальсифікованих новин, як приклад в одній з аналізуємих робіт [18] була представлена нова автоматична модель підрахунку фальсифікованих новин, з назвою FAKEDETECTOR. Спираючись на набір явних та прихованих особливостей, витягнутих з текстової інформації, FAKEDETECTOR будує глибоку дифузну мережеву модель, щоб одночасно вивчати представлення новинних статей, творців та предметів. Також у роботі проведено широкі експерименти на реальних даних з фальсифікованими новинами для порівняння цієї моделі з декількома сучасними, а експериментальні результати показали ефективність запропонованої моделі.

Загалом застосування алгоритмів класифікації для виявлення фальсифікованих новин показує якісні результати, але розмітка текстових даних, що використовуються при застосуванні методів класифікації, вимагає дуже багато часу, ресурсів та спеціальних знань. Ці перепони потенціально впливають на зменшення розміру даних, що використовуються для навчання моделі, що, в свою чергу, негативно впливає на якість класифікації.

Для аналізу текстової інформації в умовах нерозмічених даних все більшої популярності набуває застосування методів кластеризації.

1.6 Аналіз методів кластеризації

Велика кількість джерел інформації, невпинне збільшення обсягу текстової інформації новинних джерел, необхідність їх швидкої обробки викликали потребу в створенні систем автоматизованого аналізу новинного потоку. Завдання, пов'язані з обробкою новинного потоку, мають специфічні особливості, які визначаються природою новинних даних. Новинні повідомлення є відгуками на події реального світу і тому крім текстового наповнення сюжет об'єднують причинно-наслідкові, часові та інші фактори. Ця особливість використовується деякими системами, які враховують інформацію про час публікації повідомлень і основних дійових осіб при вирішенні задач обробки новин. Той факт, що новинні повідомлення стають доступні користувачеві в вигляді потоку даних, унеможливлює розбиття вже сформованих

сюжетів. Відзначимо так само, що число сюжетів заздалегідь невідомо і потенційно необмежена. Одним з найважливіших завдань автоматизованої обробки новинного потоку є розбиття новинних повідомлень на сюжети (кластеризація новинного потоку). Внаслідок потокової природи новинних даних в значній частині досліджень використовується алгоритм інкрементальної кластеризації, що має наступну послідквність дій:

- обирається міра близькості нового повідомлення і кластера;
- для кожного нового повідомлення обирається кластер, найбільш близький до повідомлення;
- в разі, якщо значення міри близькості перевищує деяке порогове значення, повідомлення додається в уже існуючий кластер;
- в разі якщо значення міри близькості не перевищило порогове значення, створюється новий кластер на основі нового повідомлення.

В таблиці 1.1 представлений порівняльний аналіз методів кластеризації за їх параметрами, можливістю масштабування, особливості використання та мірою відстані, що використовується для кластеризації.

Таблиця 1.1 - Порівняльний аналіз методів кластеризації за їх параметрами

Назва методу	Параметри	Масштабованість	Особливості використання	Міра відстані
K-Means	Кількість кластерів	Дуже велика кількість зразків, середня кількість кластерів	Рівномірний розмір кластера, плоска геометрія, не надто багато кластерів	Відстані між точками даних
Спорідненість поширення	Демпфування, перевага зразка	Не масштабується за кількістю зразків	Багато кластерів, нерівномірний розмір кластера, неплоска геометрія	Діаграма відстані (наприклад, діаграма найближчого сусіда)
Середній зсув	Пропускна здатність	Не масштабується за кількістю зразків	Багато кластерів, нерівномірний розмір кластера, неплоска геометрія	Відстані між точками даних
Спектральна кластеризація	Кількість кластерів	Середня кількість зразків, мала кількість кластерів	Малі кількість кластерів та навіть розмір кластера, неплоска геометрія	Діаграма відстані (наприклад, діаграма найближчого сусіда)

Продовження таблиці 1.1

Ієрархічна Кластеризація Ward	Кількість кластерів або відстань	Велика кількість зразків і кластерів	Багато кластерів, можливо обмеження підключення	Відстані між точками даних
Агломеративна кластеризація	Кількість кластерів або відстань, тип зв'язку, відстань	Велика кількість зразків і кластерів	Багато кластерів, можливо обмеження зв'язку, не евклідова відстань	Будь-яка попарна відстань
DBSCAN	Околиці розмір	Дуже велика кількість зразків і середня кількість кластерів	Неплоска геометрія, нерівномірні розміри скупчень	Відстані між найближчим и точками даних
OPTICS	Мінімальне членство в кластері	Дуже велика кількість зразків і велика кількість кластерів	Неплоска геометрія, нерівномірний розмір кластера, щільність кластера	Відстані між точками даних
Гауссівські суміші	Багато	Не масштабується	Плоска геометрія, що підходить для оцінки щільності	Відстань Махаланобіс а між центрами кластерів
Birch	Коефіцієнт розгалуженн я, поріг, необов'язков ий глобальний к ластер .	Велика кількість зразків і кластерів	Великий набір даних, видалення сторонніх даних, зменшення даних.	Евклідова відстань між точками даних

Для оптимізації якості кластеризації можуть використовуватися різні варіації наведеного алгоритму. Ефективність роботи системи кластеризації в першу чергу залежить від обраної міри визначення відстані між центроїдами кластерів та правильного визначення їх граничного значення.

Існує певна кількість мір визначення відстані і базованих на них алгоритмів кластеризації новинного потоку, основні з яких розглянуті далі.

1.6.1 Міра Джаккарда

Найбільш простою мірою визначення відстані двох документів видається міра збігу множин термів документів, відома так само як міра Джаккарда (Jaccard). Ця міра визначається наступним чином.

$$Sim = \frac{|A \cap B|}{|A \cup B|}, \quad (1.1)$$

де A і B - множини термів, що складають текстові документи. З формули видно, що міра приймає значення від 0 до 1 і досягає максимуму при повному збігу двох множин. Дана міра надзвичайно проста, проте містить ряд недоліків. По-перше, міра не враховує різницю в розмірі порівнюваних документів, а по-друге, при її обчисленні не використовується інформація про частоту вживання термів, що складають документи. Існує декілька алгоритмів кластеризації, що використовують цю міру відстані: Story Similarity, Story Minimal Similarity, Subject Similarity, Named Entities Similarity.

При застосуванні кластеризації для виявлення фальсифікованих новин необхідно брати до уваги той факт, що в ході розвитку подій більш ранні повідомлення збираються більш пізніми. Для перевірки цієї гіпотези введена міра Sub Similarity, що визначається як відношення потужності перетину множин термів до потужності найменшої множини, що обчислюється наступним чином (1.2).

$$SubSim = \frac{|A \cap B|}{min\ (|A|,|B|)}. \quad (1.2)$$

Міра Sub Similarity показує, наскільки одне новинне повідомлення включається в інше.

1.6.2 SVM Similarity

Аналіз методів класифікації текстових документів показав, що алгоритм опорних векторів (SVM) є одним з найбільш ефективних. У дослідженні методів кластеризації текстів [19] застосовано SVM класифікатор для завдання кластеризації новинного потоку. Як вектор простору ознак використовувався вектор, елементами якого є значення параметрів, отриманих при обчисленні по алгоритмам Story Similarity, Subject Similarity, Named Entities Similarity, Substory Similarity. Для навчання класифікатора використовується розмічений новинний набір даних текстової інформації. Навчання виконується наступним чином: для кожного нового документу з існуючих кластерів послідовно обираються повідомлення, щодо яких будується вектор ознак. У разі якщо нове новинне повідомлення і існуюче повідомлення повинні, згідно з розміткою, належати різним сюжетам, вектор співставляється до безлічі негативних прикладів. У разі, коли нове повідомлення і існуючий документ належать одному кластеру, вектор позначається, як позитивний приклад, з потоку обирається наступний документ і процес

починається спочатку. Виконане таким чином навчання дозволяє звести задачу кластеризації до класифікації на дві множини. Описаний алгоритм ґрунтується на гіпотезі, що вектор ознак містить достатньо інформації для прийняття рішення про належність, чи неналежність текстової інформації кластеру.

1.6.3 Алгоритм DBSCAN

DBSCAN є алгоритмом кластеризації, що заснований на щільності. Якщо дано набір точок в деякому просторі, алгоритм групує разом точки, які тісно розташовані (точки з багатьма близькими сусідами), позначаючи як викиди точки, які знаходяться окремо в областях з малою щільністю (найближчі сусіди яких лежать далеко). DBSCAN є одним з найбільш часто використовуваних алгоритмів кластеризації [20], і найбільш часто згадується в науковій літературі. Тут екземпляр даних вважається стороннім, якщо він не належить до жодного кластеру. Цей алгоритм досить добре використовується для виявлення сторонніх (outlier) елементів текстової інформації [21]. Цей алгоритм буде корисним у виявленні аномалій серед текстової інформації новин, які аналізуються.

Алгоритм DBSCAN вимагає двох параметрів:

ε – параметр визначає місцевину навколо точки даних, тобто, якщо відстань між двома точками менше або дорівнює ε, то вони вважаються сусідами. Якщо значення ε вибрано дуже маленьким, велика частина даних буде вважатися викидами або аномадіями. Якщо воно вибрано дуже великим, кластери будуть об'єднані, і більшість точок даних будуть знаходитися в одних і тих же кластерах. Один із способів знайти значення ε заснований на графіку відстані.

minPts – параметр визначає мінімальну кількість сусідів (точок даних) в радіусі ε. Чим більший набір даних, тим більше значення *minPts* має бути вибрано. Як правило, мінімальні *minPts* можуть бути отримані з числа вимірювань D в наборі даних, як, *minPts* > = D+1. Мінімальне значення *minPts* має бути обрано не менше 3.

У цьому алгоритмі використовується 3 типи точок даних.

Базова точка: точка є ключовою точкою, якщо вона має більш ніж *minPts* точок в межах ε.

Гранична точка: точка, у якій в ε менше, ніж *minPts*, але вона знаходиться поблизу центральної точки.

Шум або викид: точка, яка не є базовою або граничною точкою.

Алгоритм DBSCAN може бути визначений в наступних кроках.

Крок 1. Знаходження всіх сусідніх точок в межах ε і визначення основних точкиточок які відвідували більше ніж сусіди *minPts*.

Крок 2. Створення нового кластеру для кожної базової точки, якщо для неї ще не призначений кластер.

Крок 3. Рекурсивне знаходження всіх точок, пов'язаних з щільністю цього кластеру і призначення їх до того ж кластеру, що і точка ядра. Вважається, що точки *a* та *b* щільно пов'язані, якщо існує точку *c*, яка має достатню кількість сусідніх точок і обидві точки *a* і *b* знаходяться в межах відстані ε. Це ланцюгової процес. Таким чином, якщо *b* є сусідом *c*, *c* є сусідом *d*, *d* є сусідом *e*, що, в свою чергу, є сусідом *a*, вважається, що *b* є сусідом *a*.

Крок 5. Перевірка всіх, ще не задіяних точок в наборі даних. Ті точки, які не належать ні одному кластеру, є шумом.

Перевагами алгоритму можна назвати наступні властивості:

- DBSCAN не вимагає вказувати кількість кластерів у даних, на відміну від алгоритму k-mean.
- DBSCAN може знайти кластери довільної форми. Він може знайти кластер, повністю оточений (але не пов'язаний з) іншим кластером. Завдяки параметру *minPts* зменшується так званий ефект одного зв'язку, коли різні кластери поєднані низкою точок.
- DBSCAN може знаходити викиди, шуми, аномалії.
- DBSCAN вимагає лише двох параметрів і в основному нечутливий до впорядкування точок у базі даних. Однак точки, що містяться на краю двох різних кластерів, можуть змінити принадлежність кластеру, якщо впорядкованість точок зміниться.
- Параметри *minPts* і ε є настроюваними.

1.7 Постановка наукової задачі та обґрунтування методики досліджень

Результати проведеного аналізу показали, що актуальною галуззю застосування технології аналізу потоку текстової інформації в режимі реального часу є виявлення фальсифікованих новин в засобах масової інформації та соціальних медіа.

Процес аналізу потоку текстової інформації передбачає три основні завдання: пошук інформації, що включає збір відповідного тексту; вилучення інформації; аналіз вилученої інформації для виявлення нових зв’язків серед видобутої інформації.

Завдання вилучення текстової інформації залежить від поставленої задачі і типу текстового блоку. Якщо мова йде про вилучення текстової інформації з цілого документу доцільно використовувати технологію обробки тексту word2vec.

Підходи аналізу текстової інформації представлені наступним чином: орієнтовані на дані, орієнтовані на особливості, модель-орієнтовані, орієнтовані на додатки.

Модель-орієнтований підход дозволяє провести аналіз вилученої інформації та виявити корисні зв'язки в текстових даних. Цей підхід включає застосування методів класифікації або кластеризації. Більшість підходів аналізу обробленої текстової інформації включає вилучення різноманітних ознак текстової інформації із застосування методів контрольованого навчання, такого як класифікація. Однак використання методів класифікації має на увазі використання розмічених наборів даних для навчання моделі, що потребує значної кількості часу, ресурсів та спеціалізованих знань. Процес отримання надійного набору даних для навчання моделі при застосуванні методів класифікації фальсифікованих новини є складним з наступних причин:

- онлайновий набір текстової інформації у реальному світі, як правило, великий, неповний, неструктурований, нерозмічений та зашумлений;

- щодня в соціальних мережах генерується велика кількість текстової інформації з різноманітними намірами та різними мовними характеристиками, що унеможливлює урахування варіацій фальсифікованих новин.

Хоча моделі, створені з використанням методів класифікації, можуть бути більш точними з огляду на якісний набір даних для навчання, але кластеризація є більш практичною, тому що нерозмічених даних набагато більше.

В умовах відсутності розмічених даних доцільним є застосування методів кластеризації аналізу потоку текстової інформації, але існуючі технології з використанням методів кластеризація є недостатньо ефективними при використанні в режимі реального часу. Виявлення фальсифікованих новин все ще залишається недостатньо точним через динамічний характер соціальних медіа, складність та різноманітність текстових даних в Інтернеті.

В даному контексті слід визначити наступні задачі магістерської роботи:

1) Аналіз галузі застосування та технології аналізу текстового потоку в режимі реального часу.

2) Визначення етапів технології аналізу текстового потоку.

3) Розробка програмного комплексу для аналізу текстового потоку в режимі реального часу.

4) Тестування запропонованої технології аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці.

5) Оцінка якості виявлення фальсифікованих новин в текстовому потоці.

Для проведення досліджень доцільно використовувати технологію обробки тексту doc2vec та модель-орієнтований підхід аналізу потоку текстової інформації, а саме метод кластеризації.

1.8 Висновки до першого розділу

Аналіз досліджень технології аналізу потоку текстової інформації в режимі реального часу дозволив сформулювати наступні висновки.

1. Актуальною галуззю застосування технології аналізу потоку текстової інформації в режимі реального часу є виявлення фальсифікованих новин в засобах масової інформації та соціальних медіа.

2. Для вилучення текстової інформації доцільно використовувати технологію обробки тексту word2vec.

3. В умовах відсутності розмічених даних доцільним є застосування методів кластеризації аналізу потоку текстової інформації, але існуючі технології з використанням методів кластеризація є недостатньо ефективними при використанні в режимі реального часу.

4. Сформульована загальна науково-технічна задача дослідження, як задача підвищення точності технології аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації на підставі нерозмічених даних.

РОЗДІЛ 2 ТЕХНОЛОГІЯ АНАЛІЗУ ПОТОКУ ТЕКСТОВОЇ ІНФОРМАЦІЇ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Запропонована технологія аналізу потоку текстової інформації в режимі реального часу складається с трьох етапів, що базуються на сновних завданнях аналізу тексту:

Eman 1: збір текстової інформації новин.

Eman 2: передобробка текстової інформації з використанням технології word2vec, що дозволяє вилучити інформацію.

Eman 3: аналіз вилученої нформації з використанням методу кластеризації алгоритму DBSCAN.

Далі розглянуто загальна структура та сукупність методів, інструментів, що реалізують запропоновану технологію в програмному комплексі аналізу потоку текстової інформації в режимі реального часу.

2.1 Архітектура програмного комплексу аналізу потоку текстової інформації в режимі реального часу

Технологія аналізу потоку текстової інформації в режимі реального часу має на увазі розробку програмного комплексу, що дозволяє у реальному часі отримувати текстовий потік новин, проводити передобробку, проводити аналіз вилученної за допомогою передобробки інформації з використанням модель-орієнтованого підходу заснованого на методу кластеризації, оцінювати результат аналізу текстового потоку та зберігати у базу для використання на наступних ітераціях.

В ході реалізації технології була розроблена архітектура програмного комплексу, представлена на рисунку 2.1, яка включає наступні модулі: модулі збору новин, модуль прийому новин у вигляді REST сервісу, модуль черги, модуль передобробки, модуль аналізу, модуль оцінки, база даних.

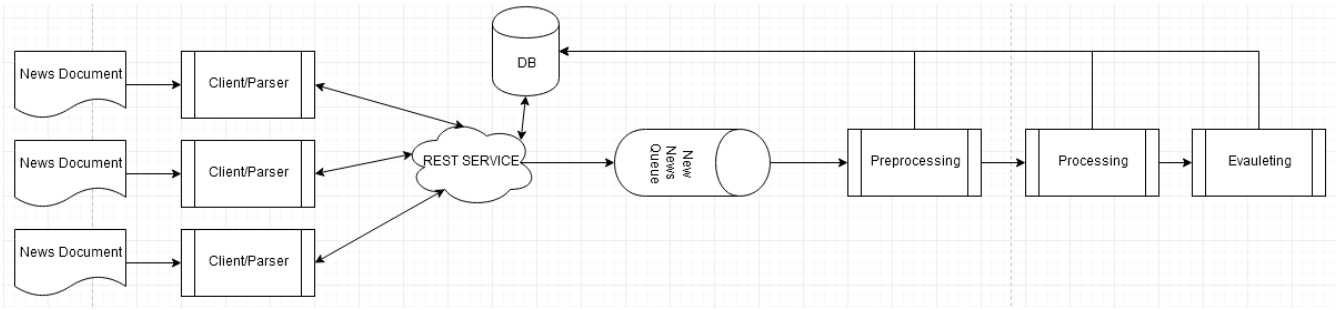


Рисунок 2.1 – Архітектура системи аналізу потоку текстової інформації в режимі реального часу

Модуль прийому новин або клієнт займається збором новин та передає їх до REST сервісу. Може бути наприклад додатками до браузера, веб додатком, настільним або мобільним в цьому досліженні використовується настільний додаток та додаток до браузера. До його обов'язків входить передача новини в аналіз, та опціонально забирати результати аналізу.

REST Service надає клієнту методи, які дозволяють завантажити нову новину для аналізу або переглянути вже проаналізовану новину, якщо вона є в базі даних. Якщо новини немає в черзі або базі даних, додає її у чергу нових новин.

Модуль передобробки забирає новини з черги та пре обробляє їх використовуючи в алгоритмах дані з бази та поточного пакету новин. Після передобробки зберігає результати у базі даних та складає їх у чергу для аналізу алгоритмами.

Модуль аналізу забирає передоброблені новини з черги, та застосовує алгоритми які виявляють аномалії серед корпусу, після отримання результатів вони зберігаються в базу даних, дані старих новин також оновлюються за потреби. Нові данні додаються у чергу оцінювання результатів.

Модуль оцінки якості виявлення аномалій, забирає данні з черги, застосовуючи алгоритми оцінювання, та зберігає результати до бази даних.

Всі модулі працюють на движку Spark. Спроектовані компоненти і класи можуть бути визначені наступним чином.

Компонент Configuration необхідний для зберігання загальних налаштувань програми, необхідних для роботи зі Spark, він складається з наступних елементів:

- Клас SparkConfiguration зберігає настройки Spark.
- Клас FSConfiguration зберігає настройки поточної файлової системи, тому що при запуску в локальному режимі файлова систему буде інша.
- Перерахування FileSystemTypeEnum містить варіанти поточної файлової системи.

Компонент SparkIO необхідний для роботи з файлами і складається з наступних елементів які представлені в Додатку Б:

- Абстрактний клас FileManager визначає базові команди для роботи з файлами.
- Клас WindowsFileManager реалізує клас FileManager для роботи з файлами в системі Windows.
- Клас HadoopFileManager реалізує клас FileManager для роботи з файлами в системі Hadoop.

Компонент Singletons реалізує шаблон «Одинак», він необхідний для зберігання екземплярів об'єктів в єдиному варіанті і складається з наступних елементів:

- Статичний клас SparkSingleton дає доступ до об'єктів, необхідним для роботи зі Spark.
- Статичний клас FileManagerSingleton дає доступ до поточного файлового менеджера і необхідний для єдиного формату роботи з файлами.

2.2 Технологія передобробки текстового потоку

Передобробка грає важливу роль у роботі алгоритмів, від якості пре обробки залежить якість результатів. Так як передобробка дозволяє видалити мусорні слова, виявити міру близькості слів, або документів. Розроблений програмний комплекс дозволяє вибір трьох технологій передобробки в процесі аналізу потоку текстової інформації в режимі реального часу. Це технології передобробки текстової інформації TF-IDF, word2vec та doc2vec. Далі представлені математичні моделі вказаних технологій преобробки.

2.2.1 Математична модель технології TF-IDF

Це статистична міра, яка використовується для оцінки важливості слова в контексті документа, що є частиною колекції документів або корпусу. Вага деякого слова пропорційний частоті вживання цього слова в документі і обернено пропорційна частоті вживання слова в усіх документах колекції.

Міра TF-IDF часто використовується в задачах аналізу текстів та інформаційного пошуку, наприклад, як один із критеріїв релевантності документа пошуковому запиту, при розрахунку міри близькості документів при кластеризації.

TF (term frequency - частота слова) - відношення числа входжень деякого слова до загальної кількості слів документа. Таким чином, оцінюється важливість слова t_i в межах окремого документа.

$$tf(t,d) = \frac{n_t}{\sum_k n_k}, \tag{2.1}$$

де n_t є число входжень слова t в документ, а в знаменнику - загальне число слів в даному документі.

IDF (inverse document frequency - зворотна частота документа) - інверсія частоти, з якою деяке слово зустрічається в документах колекції. Облік IDF зменшує вагу широковживаних слів. Для кожного унікального слова в межах конкретної колекції документів існує тільки одне значення IDF.

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}, \tag{2.2}$$

де | D | - число документів у колекції, $|\{d_i \in D \mid t \in d_i\}|$ - число документів з колекції D, в яких зустрічається t (коли $n_t \neq 0$). Вибір підстави логарифма в формулі не має значення, оскільки зміна підстави призводить до зміни ваги кожного слова на постійний множник, що не впливає на співвідношення ваг. Таким чином, міра TF-IDF є утворенням двох співмножників:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)\,. \tag{2.3}$$

Велику вагу в TF-IDF отримають слова з високою частотою в межах конкретного документа і з низькою частотою вживань в інших документах.

2.2.2 Математична модель технології Word2Vec

Як показано у розділі 1, представлення word2vec формується за допомогою CBOW та Skip-Gram. Модель CBOW з налаштуванням багатослівного контексту. Під час обчислення виводу прихованого шару замість прямого копіювання вхідного вектору вхідного контекстного

слова бере середнє значення векторів вхідних контекстних слів і використовує добуток вхідної → прихованої матриці ваги та середній вектор як вихід.

$$h = \frac{1}{C} W^T (x_1 + x_1 + \cdots + x_C) = \frac{1}{C} (V_{w_1} + V_{w_2} + \cdots + V_{w_C})^T, \quad (2.4)$$

де C - кількість слів у контексті, w_1, ⋯, w_C - слова, що є в контексті, а v_w – вхідний вектор слова w.

Функція втрат може бути представлена наступним чином (2.5).

$$E = -\log p(w_O | w_{I,1}, \cdots, w_{I,C}) = -u_{j*} + \log \sum_{j'}^{V} \exp(u_{j'}) = \quad (2.5)$$

$$= -{v'_{w_O}}^T \cdot h + \log \sum_{j=1}^{V} \exp\left({v'}_{w_j}^T \cdot h\right).$$

Цю функцію втрат можна зрозуміти як особливий випадок крос-ентропійного вимірювання між двома ймовірнісними розподілами.

Тепер можна отримати рівняння оновлення ваг між прихованим та вихідним шарами. Потрібно взяти похідну *E* щодо чистого входу *j*-ої одиниці, u_j.

$$\frac{dE}{du_j} = y_j - t_j := e_j, \quad (2.6)$$

де $t_j = 1$ $(j = j*)$, тобто, буде лише 1, коли *j*-та одиниця є фактичним вихідним словом, інакше $t_j = 0$. Ця похідна є помилкою прогнозування e_j вихідного шару.

Також потрібно взяти похідну на w'_{ij}, щоб отримати градієнт на прихованих → вихідних вагах наступним чином.

$$\frac{dE}{dw'_{ij}} = \frac{dE}{du_j} \cdot \frac{du_j}{dw'_{ij}} = e_i \cdot h_i \quad (2.7)$$

Тепер, використовуючи стохастичний градієнтний спуск, можна отримати рівняння оновлення ваги для прихованих → вихідних ваг:

$$w'_{ij}{}^{(new)} = w'_{ij}{}^{(old)} - \eta \cdot e_j \cdot h_i, \quad (2.8)$$

де $\eta > 0$ - швидкість навчання, $e_j = y_j - t_j$, а h_i - і-та одиниця в прихованому шарі; v'_{w_j} - вихідний вектор w_j. Зауважимо, що це рівняння оновлення означає, що потрібно пройти через кожне можливе слово у словнику, перевірити його вихідну ймовірність y_j і порівняти y_j з очікуваним виходом, тобто (або 0, або 1). Якщо $y_j > t_j$ («завищення»), то відняти частку прихованого вектора h (тобто v_{w_I}) від v'_{w_j}, тим самим роблячи v'_{w_j} далі від v_{w_I}, якщо $y_j < t_j$ («недооцінка», що вірно лише у тому випадку, якщо $t_j = 1$, тобто $w_j = w_O$), то треба додати деякий h до v'_{w_O}, тим самим зробивши v'_{w_O} ближчим до v'_{w_I}. Якщо y_j дуже близько до t_j, то відповідно до рівняння поновлення, будуть зроблені дуже незначні зміни до ваг. Слід пам'ятати, що v_w (вхідний вектор) і v'_w (вихідний вектор) є двома різними представленнями вектора слова w.

Модель Skip-Gram є протилежністю моделі CBOW. Цільове слово зараз знаходиться на вхідному шарі, а контекстні слова - на вихідному шарі. У цьому випадку також використовується v_{w_I} для позначення вхідного вектору єдиного слова на вхідному шарі, і таким чином, можна отримати те саме визначення вихідних даних прихованого шару, що означає h – це просто копіювання (і транспонування) рядка вхід → матриця прихованої ваги, W, пов'язана з вхідним словом w_I. Таким чином копіюємо визначення h нижче:

$$h = W^T_{(k,\cdot)} \coloneqq v^T_{w_I} \tag{2.9}$$

На вихідному шарі, замість того, щоб виводити один багаточленний розподіл, виводиться C багаточленних розподілів. Кожен вихід обчислюється за допомогою тієї ж прихованої → вихідної матриці:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})}, \tag{2.10}$$

де $w_{c,j}$ - j-е слово на c-й панелі вихідного шару; $w_{O,c}$ - власне c-е слово у вихідних контекстних словах; w_I - єдине вхідне слово; $y_{c,j}$ - вихід j-ої одиниці на c-й панелі вихідного шару; $u_{c,j}$ - чистий вхід j-го блоку на c-й панелі вихідного шару. Тому що панелі вихідного шару мають однакові ваги, таким чином (2.11).

$$u_{c,j} = u_j = v'^T_{w_j} \cdot h, \ for\ c = 1,2,\cdots,C, \tag{2.11}$$

де v'_{w_j} - вихідний вектор j-го слова в словнику, w_j, а також v'_{w_j} взято зі стовпця прихованої → вихідної матриці ваги, W'.

Виведення рівнянь оновлення параметрів не так відрізняється від односкладної контекстної моделі. Функція втрат змінюється на:

$$\begin{aligned} E = & -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\ & = -log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_j'\right)} \\ & = -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right), \end{aligned} \tag{2.12}$$

де j_c^* - індекс фактичного с-го вихідного контекстного слова у словнику. Тому береться похідна E з урахуванням чистого вводу кожної одиниці на кожній панелі вихідного шару, $u_{c,\ j}$ та виходить:

$$\frac{dE}{du_{c,j}} = y_{c,j} - t_{c,j} \coloneqq e_{c,j} \tag{2.13}$$

Для простоти позначення потрібно визначити *V*-мірний вектор *EI = {EI1, ···, EIV}*, як суму помилок прогнозування для всіх контекстних слів:

$$EI_j = \sum_{c=1}^{C} e_{c,j} \tag{2.14}$$

Далі береться похідна E відносно прихованої → вихідної матриці W′ та виводиться наступне рівняння (2.15).

$$\frac{dE}{dw_{ij}'} = \sum_{c=1}^{C} \frac{dE}{du_{c,j}} \cdot \frac{du_{c,j}}{dw_{ij}'} = EI_j \cdot h_i \tag{2.15}$$

Таким чином, отримуємо рівняння оновлення для прихованої → вихідної матриці W' (2.16).

$$w_{ij}'^{(new)} = w_{ij}'^{(old)} - \eta \cdot EI_j \cdot h_i \tag{2.16}$$

Помилка передбачення підсумовується у всіх словах контексту у вихідному шарі, також потрібно застосовувати це рівняння оновлення для кожного елемента прихованої → вихідної матриці для кожного навчального екземпляру.

2.2.3 Математична модель технології Doc2Vec

Doc2vec - це розширення до word2vec для вбудовування документа. У межах doc2vec є два підходи: dbow та dmpv. dbow працює так само, як і Skip-Gram, за винятком того, що вхід замінюється спеціальним маркером, що представляє документ (тобто v_{w_I} - це вектор, який повторно рецензує документ). У цій архітектурі порядок слів у документі ігнорується. dmpv працює аналогічно cbow. Для введення dmpv вводить додатковий маркер документа на додаток до кількох цільових слів. Однак, на відміну від CBOW, ці вектори не підсумовуються, а єднаються (тобто v_{w_I} - це об'єднаний вектор, що містить маркер документа та кілька цільових слів). Мета алгоритму передбачити контекстне слово з урахуванням об'єднаного вектору документа та слова.

2.3 Аналіз вилученої інформації із застосуванням алгоритму DBSCAN

Аналіз аномалій та кластеризація є двома дуже пов'язаними завданнями. Кластеризація знаходить більшість шаблонів у наборі даних та упорядковує їх відповідно, тоді як виявлення аномалій намагається зафіксувати ті виняткові випадки, які суттєво відхиляються від більшості шаблонів. Шум слід видаляти під час застосування виявлення аномалій. Це може спотворювати нормальні об'єкти і розмивати відмінність між нормальними об'єктами і потоками. Це може допомогти приховати викиди і знизити ефективність виявлення аномального значення. В цілому, викиди можуть бути розділені на три категорії, а саме глобальні викиди, контекстні (або умовні) останці і колективні викиди.

Формально, набір вилученої текстової інформації новин можна представити наступним чином. Відповідно до модельно-орієнтованого підходу для виявлення фальсифікованих новин може використовуватися техніка машинного навчання без вчителя та с вчителем. Ми вивчаємо класифікацію як контрольований метод навчання та виконуємо кластеризацію як метод без вчителя.

Кластеризацію в контексті виявлення підроблених новин можна визначити наступним чином. Дано корпус фальсифікованих новин $J = \{j_1, j_2, j_3, ..., j_n\}$ з розміром n, де кожен документ $\overrightarrow{n_i}$ це вектор термів(terms це строк чи термінів?) у словнику, $\sum = \{t_1, t_2, t_3, \dots, t_T\}$ з розміром $T = |\sum|$. Проблема полягає в кластеризації документів на основі їх термів в однорідні класи щодо фальсифікованих категорій новин. З цією метою ми спочатку кластеризуємо документи на основі позицій зовнішнього вигляду кожного терма в статті та їх співвідношень з іншими термами (вилучення просторових відношень) наступним чином, розробляючи

автоматичний ансамблевий спільний кластер для кластеризації документів відповідно до їх позицій у різних факторах серед різних факторів декомпозиції низького рангу.

Модель, запроваджена DBSCAN, використовує просту оцінку рівня мінімальної щільності, засновану на спортивному рівні для кількості сусідів, minPts, в радіусі ε (з довільною мірою відстані). Об'єкти з більш ніж minPts сусідами в цьому радіусі (включаючи точку запиту) вважаються основною точкою. Інтуїція DBSCAN полягає в пошуку тих областей, які задовольняють цю мінімальну щільність і які розділені областями меншої щільності. З міркувань ефективності DBSCAN не проводить оцінку щільності між проміжками. Натомість, всі сусіди в радіусі ε основної точки вважаються частиною того ж кластера, що і основна точка (називається доступною прямою щільністю). Якщо будь-який з цих сусідів знову є основною точкою, їх мікрорайони включаються транзитивно (щільність доступна). Непрофільні точки в цьому наборі називаються граничними точками, а всі точки в межах одного набору пов'язані щільністю. Точки, які не мають густини, доступні у будь-якій точці ядра, вважаються шумом і не належать до жодного кластеру.

Алгоритм для обчислення кластерів відповідно до вищевказаної моделі (за винятком того, що граничні точки, що належать до декількох кластерів, призначені лише одному з них). У цьому алгоритмі база даних лінійно сканується на об'єкти, які ще не оброблені. Неядерні точки прирівнюються шуму, і коли виявлена основна точка, її сусіди ітеративно розширюються та додаються до кластера. Об'єкти, які були призначені кластеру, будуть пропущені пізніше при лінійному скануванні. Цей базовий алгоритм є стандартним підходом для обчислення транзитивного закриття відношення з мінімальною модифікацією, що розширюється лише основними точками.

В свою чергу на основі дистанційного методу виявлення аномалій визначається сусідство об'єкта, яке визначається заданим радіусом. Потім об'єкт вважається чужим, якщо в його околицях недостатньо інших точок. Відстань - поріг, який можна визначити як розумне сусідство об'єкта. Для кожного об'єкта *o* можна знайти розумну кількість сусідів об'єкта.

Формально, нехай *r (r> 0)* - поріг відстані, а π *(0 <* π *<1)* - поріг дробу. Тоді виразити *DB (r,* π*)* для об'єкта *o* можна наступним чином (2.17).

$$\frac{|o\prime|dist(o,o\prime)\leq r||}{||D||} \leq \pi, \tag{2.17}$$

де *dist* - міра відстані;

r – заданий радіус для виявлення сусідства;

DB – вираз дистанційного методу виявлення аномалій;

D – набір даних.

Прямий підхід займає час O (n^2).

Метод виявлення аномалій на основі щільності даних досліджує щільність об'єкта та щільність його сусідів. Тут об'єкт ідентифікується як зовнішній, якщо його щільність порівняно значно нижча, ніж у сусідів. Методи засновані на використанні відстані між об'єктами даних можуть бути визначені наступним чином.

$$N_k(o) = [o' | o \in D, dist(o, o') \leq dist_k(o)], \tag{2.18}$$

де *dist_k (o)* -відстань між об'єктом *o* та *k*-найближчими сусідами. Район k-відстані *o* містить усі об'єкти, де відстань до *o* не більше *dist_k (o) k*-ої відстані *o.*

Середня відстань між об'єктами у N_k *(o)* до *o* може бути використана як міра локальної щільності *o*. У випадках, коли у *o* дуже близькі сусіди *o'* такі, що *dist (o, o')* є дуже малою, статистичні коливання вимірювання відстані можуть бути небажано великими. Для подолання цієї проблеми пропонується перейти до наступної міри відстані досяжності, додавши ефект згладжування.

$$reachdist_k(o, o') = max[dist_k(o), dist(o, o')], \tag{2.19}$$

де k - заданий користувачем параметр, який контролює ефект згладжування. По суті, k задає мінімальну околицю, яку слід досліджувати, щоб визначити локальну щільність об'єкта. Відстань досяжності не є симетричною. Локальною щільністю доступності об'єкта *o* є:

$$lrd_k(\frac{||N_k(o)||}{\sum_{o' \in N_k(o)} reachdist_k(o, o')}) \tag{2.20}$$

Обчислити локальну щільність досяжності для об'єкта і порівняти його з щільністю сусідів для кількісного визначення ступеня, до якої об'єкт вважається чужим.

$$LOF_k(o) = \frac{\sum_{o' \in N_k(o)} \frac{lrd_k(o')}{lrd_k(o)}}{||N_k(o)||} = \sum_{o' \in N_k(o)} lrd_k(o') \cdot \sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o) \tag{2.21}$$

Місцевий коефіцієнт випередження (local outlier factor скор. LOF) - це середнє співвідношення локальної щільності доступності *o* та коефіцієнта *k*-найближчих сусідів. Чим менше локальна щільність досяжності *o* і чим вище локальна щільність досяжності *k*-

найближчих сусідів *o*, тим вище значення LOF. Це точно фіксує місцеву аномалію, локальна щільність якої низька порівняно з локальною щільністю її *k*-найближчих сусідів.

2.4 Критерії оцінки якості кластеризації

Для оцінки якості доступні наступні критерії оцінки якості кластеризації: Silhouette Index, Davies–Bouldin index, Dunn index, Cohen's kappa coefficient

2.4.1 Критерій Silhouette Index

Silhouette відноситься до методу інтерпретації та перевірки послідовності в кластерах даних. Методика дає стисле графічне зображення того, наскільки добре класифікований кожен об'єкт.

Значення Silhouette - це міра того, наскільки об’єкт схожий на власний кластер (згуртованість) порівняно з іншими кластерами (розділення). Silhouette коливається від -1 до +1, де високе значення вказує на те, що об'єкт добре підходить до власного кластеру і погано відповідає сусіднім кластерам. Якщо більшість об'єктів мають високе значення, то конфігурація кластеризації підходить. Якщо багато точок мають низьке або негативне значення, то в конфігурації кластеризації може бути занадто багато або занадто мало кластерів.

Припустимо, дані були кластеризовані за допомогою k-means, у кластери k. Для точки даних $i \in C_i$, нехай

$$a(i) = \frac{1}{|C_i|-1}\sum_{j\in C_i, i\neq j} d(i,j), \tag{2.22}$$

де *a(i)* середня відстань між *i* та всіма іншими точками даних у тому ж кластері, а *d (i, j)* - відстань між даними точки *i* та *j* у кластері C_i (ділимо на | C_i | - 1 , оскільки не треба включати відстань *d (i, i)* у сумі). Можна інтерпретувати *a(i)* як міру того, наскільки добре *i* присвоєно його кластеру (чим менше значення, тим краще призначення).

Потім визначається середня різниця між точкою *i* деяким кластером *C* як середнє відстані від *i* до всіх точок *C* (де $C \neq C_i$). Для кожної точки даних $i \in C$, далі визначається:

$$b(i) = \min_{k\neq i} \frac{1}{|C_k|}\sum_{j\in C_k} d(i,j)\,, \tag{2.23}$$

де оператор min у формулі означає середню відстань і до всіх точок будь-якого іншого кластеру, учасник якого і не є членом. Кластер з цією найменшою середньою різницею називається «сусіднім кластером» і, оскільки це наступний кластер, що найкраще підходить для точки і. Далі потрібно визначити значення однієї точки даних *i*:

$$s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}, if\ |C_i| > 1 \tag{2.24}$$

та

$$s(i) = 0, if\ |C_i| = 1 \tag{2.25}$$

Що також можна записати наступним чином (2.26).

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, if\ a(i) < b(i) \\ 0, \qquad if\ a(i) = b(i). \\ \frac{b(i)}{a(i)} - 1, if\ a(i) > b(i) \end{cases} \tag{2.26}$$

З наведеного визначення видно, що:

$$-1 \le s(i) \le 1. \tag{2.27}$$

Щоб *s(i)* був близьким до 1 потрібно $a(i) \ll b(i)$. Оскільки *a(i)* є мірою того, наскільки *i* відрізняється від власного кластера, невелике значення означає, що воно добре відповідає. Крім того, з великого *b(i)* випливає, що *i* погано узгоджується із сусіднім кластером. Таким чином, якщо *s(i)*, близький до одного, означає, що дані належним чином кластеризовані. Якщо *s(i)* близький до негативного, то за тією ж логікою можна побачити, що *i* було б доцільніше, якби він був кластеризований у сусідньому кластері. Близький до нуля *s(i)* означає, що значення знаходиться на межі двох натуральних кластерів. Середнє значення *s(i)* для всіх точок кластера - це міра того, наскільки щільно згруповані всі точки кластера. Таким чином, середнє значення *s(i)* для всіх даних усього набору даних є мірою того, наскільки належним чином класифіковані дані. Якщо є занадто багато або занадто мало кластерів, як це може статися, коли в алгоритмі кластеризації використовується неправильний вибір k (наприклад, k-means), деякі кластери, як правило, відображатимуть набагато вужчі значення Silhouette, ніж решта. Таким чином, засоби

Silhouette можуть використовуватися для визначення натуральної кількості кластерів у наборі даних. Можна також збільшити ймовірність того, що значення Silhouette буде максимізовано на правильну кількість кластерів шляхом повторного масштабування даних за допомогою вагових характеристик, які є специфічними для кластера.

2.4.2 Критерій Davies–Bouldin index

Це внутрішня схема оцінювання, де перевірка того, наскільки добре проведено кластеризацію, здійснюється за допомогою кількості та особливостей, властивих набору даних.

З огляду на n розмірних точок, нехай C_i - це кластер точок даних. Нехай X_j - n-мірний вектор ознак, призначений кластеру C_i.

$$S_i = \left(\frac{1}{T_i}\sum_{j=1}^{T_i}\left|X_j - A_i\right|^p\right)^{\frac{1}{p}}. \tag{2.28}$$

В цьому виразі A_i - центроїд C_i і T_i - розмір кластера і. S_i - міра розсіювання всередині кластера. Зазвичай значення p дорівнює 2, що робить цю евклідову функцію відстані між центроїдом кластера та окремими векторами ознак. Багато інших показників відстані можуть бути використані, в разі колекторів і більш високих просторових даних, де евклідова відстань не може бути кращим показником для визначення кластерів. Важливо зазначити, що ця метрика відстані повинна відповідати метриці, що використовується в самій схемі кластеризації для значущих результатів.

$$M_{i,j} = \left\|A_i - A_j\right\|_p = \left(\sum_{k=1}^{n}\left|a_{k,i} - a_{k,j}\right|^p\right)^{\frac{1}{p}}, \tag{2.29}$$

де $M_{i,j}$ - міра поділу між кластером C_i і кластером C_j. В свою чергу $a_{k,i}$ - це k-й елемент A_i, і існує n таких елементів у A, оскільки це n розмірний центроїд.

Нехай $R_{i,j}$ - міра того, наскільки якісна схема кластеризації. Цей показник, за визначенням, повинен враховувати $M_{i,j}$ розділення між i-м та j-м кластером, яке в ідеалі повинно бути якомога більшим, і S_i, всередині кластерного розсіювання для кластера i, яке повинно бути якомога найменшим. Отже, Davies–Bouldin index визначається як таке відношення S_i та $M_{i,j}$, що ці властивості зберігаються:

$R_{i,j} \geq 0$

$R_{i,j} = R_{j,i}$

Де $S_j \geq S_k$ та $M_{i,j} = M_{i,k}$, тоді $R_{i,j} = R_{i,k}$

Та де $S_j = S_k$ та $M_{i,j} \leq M_{i,k}$, тоді $R_{i,j} > R_{i,k}$

При цій формулі, чим нижче значення, тим краще розділення кластерів та «герметичність» всередині кластерів. Рішенням, яке задовольняє ці властивості, є:

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}}. \tag{2.30}$$

Це використовується для визначення D_i:

$$D_i \equiv \max_{j \neq i} R_{i,j} \tag{2.31}$$

Якщо N - кількість кластерів, то вираз може бути представлений наступним чином (2.32).

$$DB \equiv \frac{1}{N} \sum_{i=1}^{N} D_i \tag{2.32}$$

де *DB* це Davies–Bouldin index, який залежність як від даних, так і від алгоритму. D_i вибирає найгірший сценарій, і це значення дорівнює $R_{i,j}$ для найбільш подібного кластера до кластеру *i*. У цій формулі може бути багато варіантів вибір середньої схожості кластера, середньозваженого рівня тощо.

2.4.3 Критерій Dunn index

Це показник для оцінки алгоритмів кластеризації. Також частина групи індексів валідності, включаючи Davies–Bouldin index або Silhouette index, оскільки це внутрішня схема оцінки, де результат ґрунтується на самих кластерних даних. Як і всі інші подібні індекси, мета полягає у визначенні компактних наборів кластерів з невеликою дисперсією між членами

кластера та добре розділеними, де значення різних кластерів досить далеко один від одного, порівняно з кластером всередині дисперсії. Існує багато способів визначити розмір або діаметр кластера. Це може бути відстань між самими далекими двома точками всередині кластера, або середнім значенням всіх відстаней між попарними точками даних усередині кластера, а також може бути відстанню кожної точки даних до центроїда кластеру. Кожне з цих рішень математично розглянуто нижче.

Нехай C_i - це скупчення векторів. Нехай x і y - будь-які дві n-мірні особливості векторів, присвоєні одному кластеру C_i.

$$\Delta_i = \max_{x,y \in C_i} d(x,y), \tag{2.33}$$

де Δ_i це максимальна відстань.

$$\Delta_i = \frac{1}{|C_i|(|C_i|-1)} \sum_{x,y \in C_i, x \neq y} d(x,y), \tag{2.34}$$

де Δ_i це середня відстань між усіма парами.

$$\Delta_i = \frac{\sum_{x \in C_i} d(x,\mu)}{|C_i|}, \mu = \frac{\sum_{x \in C_i} x}{|C_i|}, \tag{2.35}$$

де Δ_i це відстань усіх точок від центроїда.

Також це можна сказати про відстань між кластерами, де можна скласти подібні формули, використовуючи найближчі дві точки даних по одній у кожному кластері, найдальші дві, або відстань між центроїдами тощо. Визначення індексу включає будь-яке таке формулювання, а сімейство індексів, утворених таким чином, називають Dunn-подібними індексами. Можна сказати що метрикою відстані між кластерами C_i та C_j є $\delta(C_i, C_j)$.

З вищенаведеними позначеннями, якщо є m кластери, то Dunn index для набору визначається наступним чином (2.36).

$$DI_m = \frac{\min_{1 \leq k < j \leq m} \delta(C_i, C_j)}{\max_{1 \leq k \leq m} \Delta_k} \tag{2.36}$$

2.4.4 Критерій Cohen's kappa coefficient

Cohen's kappa coefficient - вимірює узгодженість між двома рейтингами, які класифікують N предметів у категоріях C, що взаємовиключні. Визначенням *κ* є:

$$\text{к} \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}, \tag{2.37}$$

де p_o - відносна спостережувана узгодженість між переможцями (ідентична точності), а p_e - гіпотетична ймовірність випадкової згоди, використовуючи спостережувані дані для обчислення ймовірностей кожного спостерігача, який випадково бачить кожну категорію. Якщо рейтинги повністю узгоджуються, то $\kappa = 1$. Якщо серед рейтингів немає узгодженості, окрім того, що можна було б очікувати випадково (за даними p_e), $\kappa = 0$. Якщо статистика є негативною, це означає, що між двома рейтингами не існує ефективної узгодженості або вона є гіршою, ніж випадкова.

Для k категорій, N спостережень для категоризації та n_{ki} кількість разів, коли рейтинг *i* прогнозує категорію k:

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2} \tag{2.38}$$

Це випливає з наступної формули:

$$p_e = \sum_k \widehat{p_{k12}} = \sum_k \widehat{p_{k1}} \widehat{p_{k2}} = \sum_k \frac{n_{k1}}{N} \frac{n_{k2}}{N} = \frac{1}{N^2} \sum_k n_{k1} n_{k2}, \tag{2.39}$$

де $\widehat{p_{k12}}$ це оціночна ймовірність того, що і оцінщик 1, і оцінщик 2 класифікують один і той же елемент як *k*, тоді $\widehat{p_{k1}}$ оціночна ймовірність того, що оцінщик 1 класифікує елемент як k (і аналогічно для оцінщика 2). Співвідношення $\widehat{p_k} = \sum_k \widehat{p_{k1}} \widehat{p_{k2}}$ засноване на використанні припущення про незалежність рейтингу двох оцінщиків. Терм $\widehat{p_{k1}}$ оцінюється, використовуючи кількість елементів, класифікованих як *k* за оцінщиком 1, поділену на загальну кількість елементів для класифікації *N* (і аналогічно для оцінщика 2).

2.5 Візуалізація результатів кластеризації

Запропонована технологія аналізу текстового потоку включає візуалізацію результатів кластеризації фальсифікованих новин.

Візуалізація кластерів - це спосіб полегшити розуміння оцінки, дослідження або інтерпретації результатів кластерного аналізу. Як вже було зазначено вище, кластеризація - це неконтрольований метод навчання, що групує набір з n об’єктів даних $D=\{x_1, \ldots, x_n\}$ у кластери, так що об’єкти в одному кластері схожі, а об'єкти з різних кластерів відрізняються один від одного. Дані можуть представлені наступним чином.

1. У вигляді матриці подібностей (або відмінностей) (n x n).

2. У вигляді матриці даних (n x d), яка описує кожен об'єкт даних за допомогою d-розмірного вектора. Це представлення повинно супроводжуватися відповідною мірою подібності чи несхожості, яка обчислює для пари двовимірних векторів показник схожості або несхожості.

Результати кластеризації можуть мати різні форми:

1. У вигляді частини D.
2. У вигляді моделі, яка узагальнює властивості D.
3. У вигляді ієрархічного набору вкладених частин D.

Візуалізація цих результатів фокусується на одній або декількох властивостях, таких як відносини подібності між кластерами та / або базовими об'єктами даних, компонентами моделі кластеризації або представлення об'єктів даних. Візуалізація кластерів служить для визначення семантики кластерів, що мають відношення до даного додатка шляхом перевірки, для перевірки того, чи відповідає кластерна модель та її параметризація (наприклад, кількість кластерів) значущим чином, або досліджувати ієрархічні зв’язки між кластерами.

2.6 Висновок до другого розділу

В розділі представлена запропонована технологія аналізу потоку текстової інформації в режимі реального часу, що реалізована в розробленому програмному комплексі. Представлена загальна архітектура розробленого програмного комплексу, визначені та описаны складові модулі програмного корпусу, представлені технології передобробки та алгоритм кластеризації для аналізу потоку текстової інформації в режимі реального часу. Визначені критерії оцінки якості, що можуть використовуватись для оцінки якості виявлення фальсифікованого текстового потоку новин або його аномалій.

Для технологій передобробки та алгоритму кластеризації розглянуті та виведені основні математичні моделі, що дозволяють отримувати найбільш якісний результат аналізу під час обробки великої кількості даних в реальному часі. Представлені математичні моделі технологій передобробкита вилучення інформації з текстового потоку TF-IDF, word2vec, doc2vec. Алгоритм кластеризації DBSCAN, що використовується для аналізу вилученої інформації, дає можливість більш точно виявляти аномалії серед корпусу документів, які в подальшому можна використовувати, як ознаки для виявлення фальсифікованих новин. Також, представлені критерії оцінки якості кластеризації, що доступні в розробленому програмному комплексі, які можуть бути використані як критерії оцінки якості виявлення фальсифікованих новин. Також, для оцінки якості виявлення фальсифікованих новин, використано коефіцієнт Cohen's kappa, який надає можливість виявити рівень узгодженості документів серед різних класів.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ АНАЛІЗУ ПОТОКУ ТЕКСТОВОЇ ІНФОРМАЦІЇ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

3.1 Програмний комплекс аналізу потоку текстової інформації в режимі реального часу

В процесі практичної реалізації технології аналізу потоку текстової інформації в режимі реального часу розроблено програмний комплекс за представленою в розділі 2 архітектурою, який надає можливість приймати, передобробляти, аналізувати та оцінювати якість обробки та виявлення, зберігати текстову інформацію новин для подальшої роботи.

Основною мовою програмування для реалізації програмного комплексу було обрано Java. Через те що вона є компільованою, строго типізованою, об’єктно орієнтованою, що спрощує налагодження в процесі розробки модулів і побудови архітектури програмного комплексу.

3.1.1 Сукупність інструментів, що використовуються модулями програмного комплексу

Аналіз потоку текстової інформації має на увазі обробку, аналіз та збереження досить великих об’ємів даних. Виходячи з цього для досягнення поставленої мети в розробленому програмному комплексі доступно застосування наступних інструментів.

Apache Spark – це система розподіленої обробки неструктурованих і слабо структурованих даних. Для розподілу обчислень великих об'ємів даних на кластері комп'ютерів використовується фреймворк Apache Hadoop.

Проект надає програмні інтерфейси для мов Java, Scala, Python, R. Складається з ядра і кількох розширень, таких як Spark SQL (дозволяє виконувати SQL-запити над даними), Spark Streaming (надбудова для обробки потокових даних), Spark MLlib (набір бібліотек машинного навчання), GraphX (призначене для розподіленої обробки графів). Може працювати як в середовищі кластера Hadoop під керуванням YARN, так і без компонентів ядра Hadoop, підтримує кілька розподілених систем зберігання - HDFS, OpenStack Swift, NoSQL-СУБД Cassandra, Amazon S3.

Apache Hadoop – це фреймворк, спроектований для пакетної обробки величезного обсягу даних. Для зберігання даних створена спеціальна файлова система Hadoop FS, яка автоматично

розподіляє дані по кластеру. Дані обробляються за допомогою моделі розподілених обчислень MapReduce.

MapReduce – складається з двох кроків: Map і Reduce, названих так за аналогією з однойменними функціями вищого порядку, map і reduce. На Map-кроці відбувається попередня обробка вхідних даних. Для цього один з комп'ютерів (званий головним вузлом - master node) отримує вхідні дані задачі, розділяє їх на частини і передає іншим комп'ютерам (робочим вузлам - worker node) для попередньої обробки. На Reduce-кроці відбувається згортка попередньо оброблених даних. Головний вузол отримує відповіді від робочих вузлів і на їх основі формує результат - рішення задачі, яка спочатку формувалася. Перевага MapReduce полягає в тому, що він дозволяє розподілений виробляти операції попередньої обробки і згортки. Операції попередньої обробки працюють незалежно один від одного і можуть проводитися паралельно (хоча на практиці це обмежено джерелом вхідних даних і / або кількістю використовуваних процесорів). Аналогічно, безліч робочих вузлів може здійснювати згортку - для цього необхідно лише щоб всі результати попередньої обробки з одним конкретним значенням ключа оброблялися одним робочим вузлом в один момент часу. Хоча цей процес може бути менш ефективним в порівнянні з більш послідовними алгоритмами, MapReduce може бути застосований до великих обсягів даних, які можуть оброблятися великою кількістю серверів. Так, MapReduce може бути використаним для сортування петабайта даних, що займе всього лише кілька годин. Паралелізм також дає деякі можливості відновлення після часткових збоїв серверів: якщо в робочому вузлі, що виробляє операцію попередньої обробки або згортки, виникає збій, то його робота може бути передана іншому робочому вузлу (за умови, що вхідні дані для проведеної операції доступні).

Apache Spark на відміну від класичного обробника з ядра Hadoop, що реалізує дворівневу концепцію MapReduce з дисковим сховищем, використовує спеціалізовані примітиви для рекурентної обробки в оперативній пам'яті, завдяки чому дозволяє отримувати значний виграш в швидкості роботи для деяких класів задач, зокрема, можливість багаторазового доступу до завантажених в пам'ять призначених для користувача даних робить бібліотеку привабливою для алгоритмів машинного навчання.

Для обробки потоку даних існує розширення основного ядра Apache Spark – Spark Streaming, яке дозволяє здійснювати масштабовану відмово стійку обробку потоку даних з високою пропускною здатністю. Модуль Spark Streaming отримує дані з Apache Kafka, ділить їх на пакети і відправляє в основний двигун Spark, який їх обробляє і генерує кінцевий потік результуючих пакетів. Така потокова модель називається мікро-пакетуванням. Spark Streaming

маніпулює абстракцією, званою DStream, який представляє з себе безперервну послідовність RDD.

Apache Kafka – розподілений програмний брокер повідомлень, написаний на мові програмування Scala. Відмінності від традиційних систем передачі повідомлень:

– спроектований спочатку як розподілена система, яку легко масштабувати;

– підтримує високу пропускну здатність як з боку джерел, так і систем-передплатників;

– підтримує об'єднання підписників в групи;

– забезпечує можливість тимчасового зберігання даних для подальшої пакетної обробки.

Однією з особливостей реалізації інструменту є застосування техніки, схожої з журналами транзакцій, використовуваними в системах управління базами даних.

RDD – відмово стійка і незмінна колекція елементів, яку можна обробляти паралельно як показано на рисунку. Кожен RDD в DStream містить дані з певного інтервалу часу. Схема основних можливостей роботи з RDD представлена на рисунку 3.1.

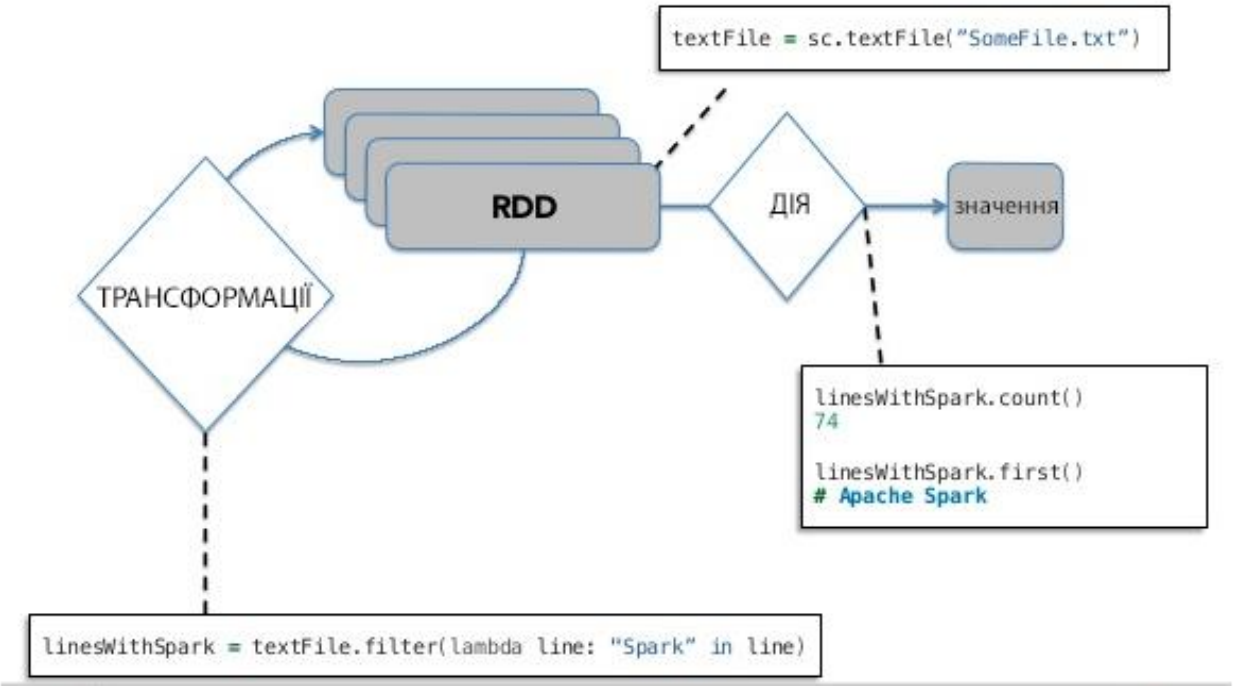


Рисунок 3.1 – Схема основних можливостей роботи з RDD

Для зберігання оброблених даних використано Apache Hive – це система управління базами даних на основі платформи Hadoop. Дозволяє виконувати запити, агрегувати і аналізувати дані, що зберігаються в Hadoop. Основні можливості цієї СУБД наступні:

– робота з даними використовуючи SQL-подібну мову запитів;

– підтримка різних форматів зберігання даних;

– робота безпосередньо з HDFS і Apache HBase;

– виконання запитів через Apache Tez, Apache Spark або MapReduce, що досить важливо у цьому випадку.

3.2.1 Модулі програмного комплексу

Програмний комплекс включає наступні модулі: модулі збору новин, модуль прийому новин у вигляді REST сервісу, модуль черги, модуль передобробки, модуль аналізу, модуль оцінки, база даних. Які в свою чергу відповідають шести типам програм:

– Збирачі.

– REST інтерфейс.

– Черга.

– Програми передобробки.

– Програми аналізу вилученої інформації.

– Програми оцінки.

3.2.1.1 Модуль прийому текстового потоку новин

Збирачі реалізовані на JavaScript та Java, в першому випадку це додаток для браузера в другому звичайний настільний додаток який дозволяє відправляти данні через REST метод.

3.2.1.2 REST Service

REST інтерфейс реалізований на мові Java та фреймворку Spring Boot MVC. Він має декілька методів для отримання та відправки даних: POST метод, GET

Використання POST методу для відправки документа на обробку виглядає наступним чином, як це представлно на рисунку 3.2.

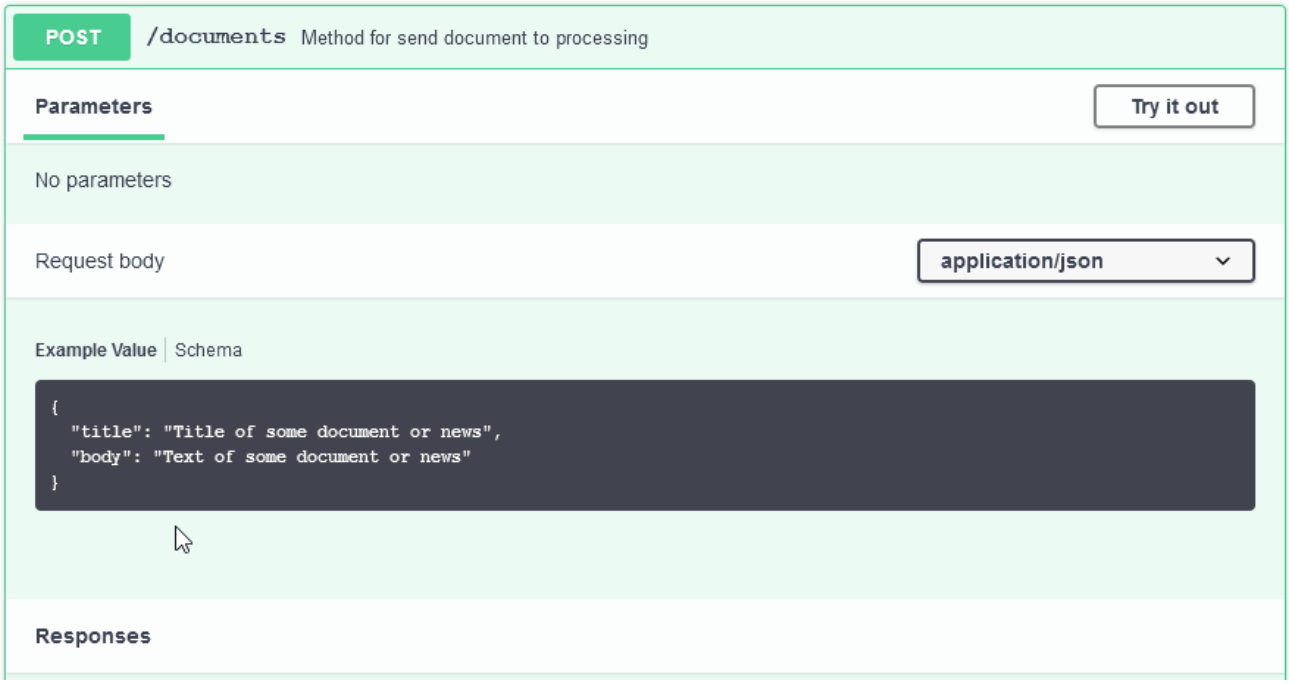


Рисунок 3.2 – Використання POST методу для відправки документа на обробку

Використання GET методу для запиту даних по документу виглядає наступним чином, як це представлно на рисунку 3.3.

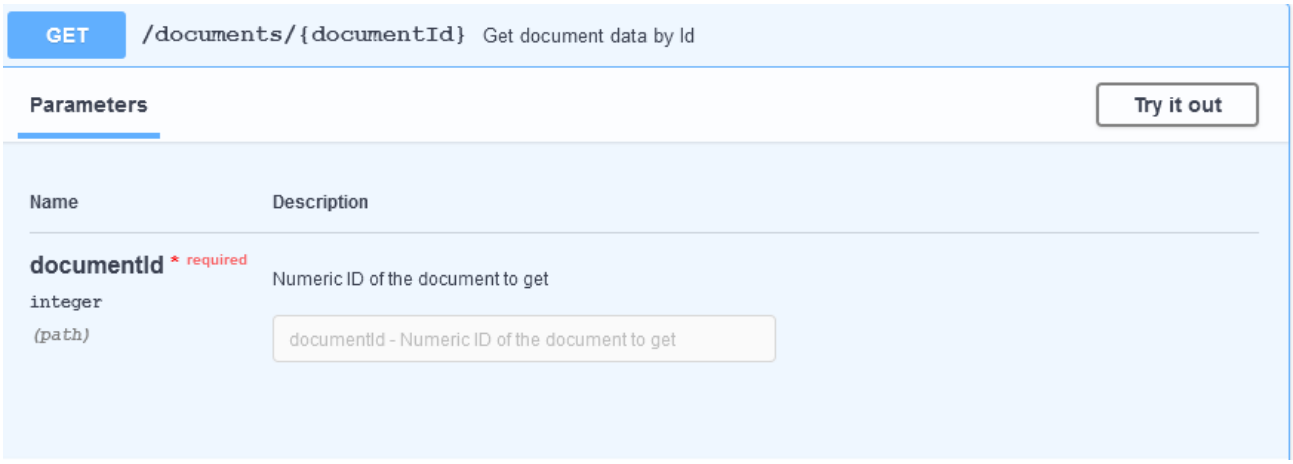


Рисунок 3.3 – Використання GET методу для відправки документа на обробку

У відповідь на використання методів для відправки документа на обробку повертається наступна модель:

```
{
  "id": 1,
  "status": "pending",
  "title": "A great news title",
```

```
  "body": {
    "raw": "A great big document body...",
    "preprocessed": "0.12132213 0.14, 0.1134...."
  },
  "result": {
    "label": "fake",
    "clustering": "string",
    "nn": "string"
  },
  "evaluations": {
    "silhouette_index": 0.345,
    "db_index": 0,
    "cohen_coef": 0.648
  }
}
```


Також доступний метод для отримування декількох документів за статусом, з можливістю пагінації через параметр Offset (рисунок 3.4).

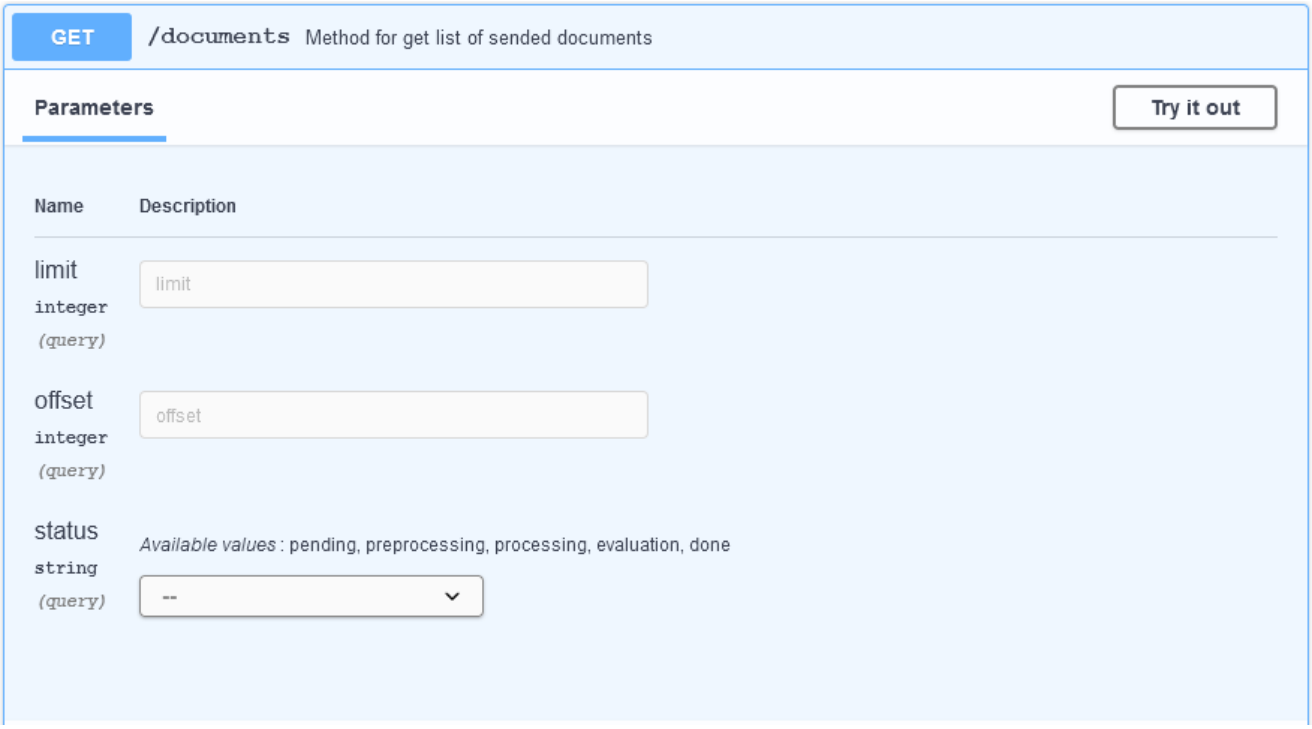


Рисунок 3.4 – Використання GET методу для відправки документа на обробку з можливістю пагінації через параметр Offset

Черга реалізована за допомогою Apache Kafka, що дозволяє масштабувати та балансувати обробку для великої кількості даних. На рисунку 3.5 можна бачити схему взаємодії інших модулів системи с чергою.

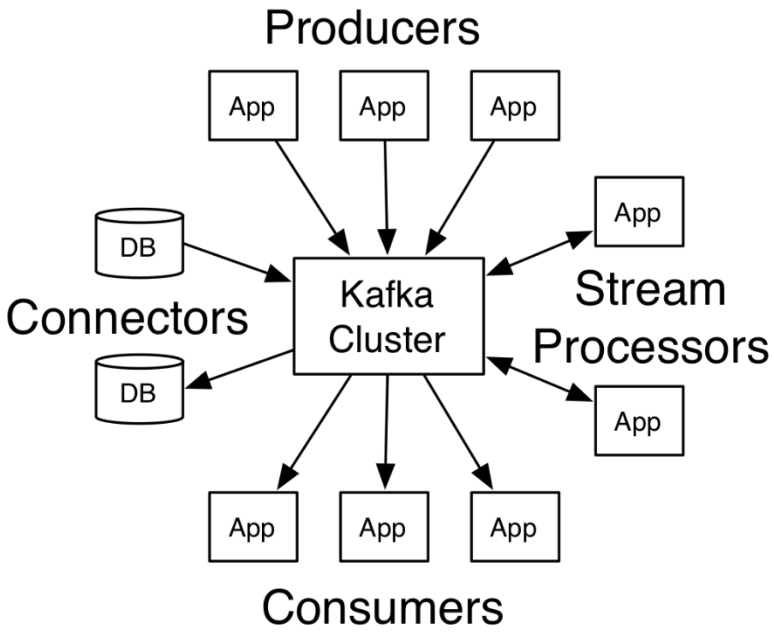


Рисунок 3.5 – Діаграма взаємодії черги Kafka з іншими модулями програмного комплексу

Тема - це категорія або назва каналу, до яких публікуються записи. Теми в Kafka завжди мають багато споживачів; тобто тема може мати нуль, одного або багатьох споживачів, які підписуються на дані, записані до неї. Для кожної теми черга Kafka підтримує журнал розділів, який виглядає приблизно так, як це представлено на рисунку 3.6.

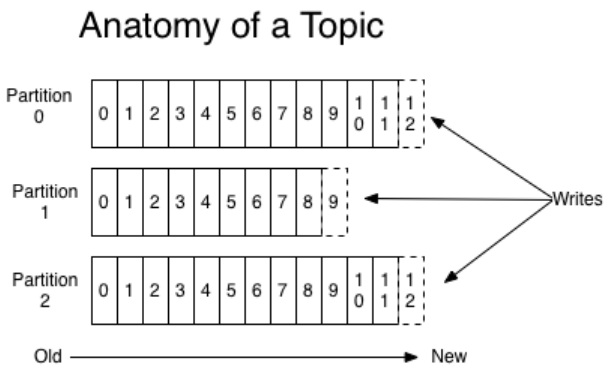


Рисунок 3.6 – Журнали розділів тем черги Kafka

Кожен розділ (partition) - це впорядкована, незмінна послідовність записів, яка постійно додається у структурований журнал фіксування. Кожному запису в розділах присвоюється послідовний ідентифікаційний номер, який називається зміщенням, який однозначно ідентифікує кожен запис у розділі. Продуктивність Kafka ефективно постійна щодо розміру даних, тому зберігання даних тривалий час не є проблемою. За рахунок архітектури можливе масштабування, як самої черги так і її споживачів (інших модулів системи), що зображено схематично на рисунку 3.7.

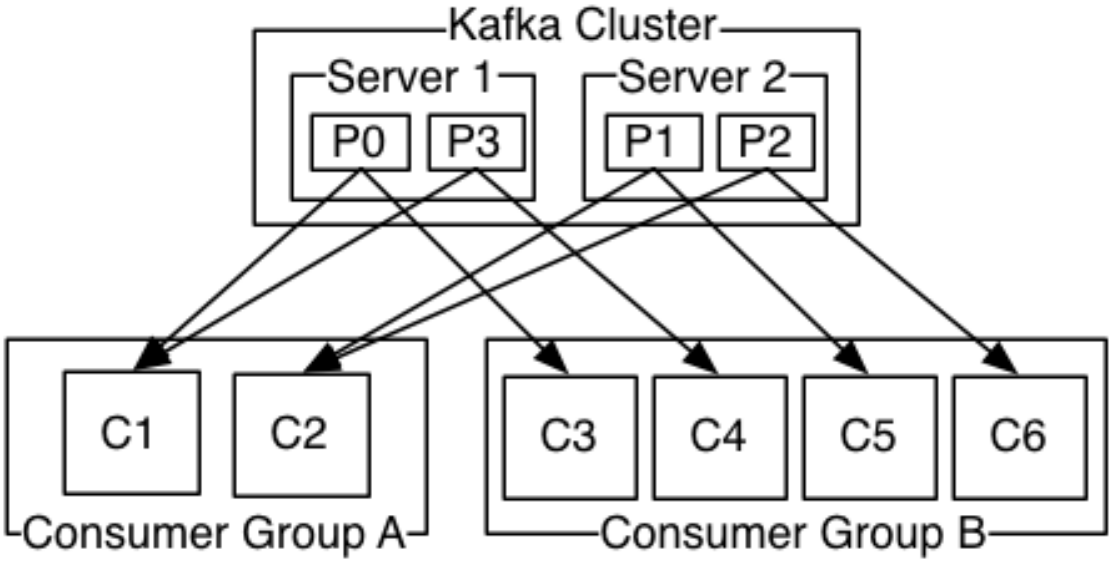


Рисунок 3.7 - Два серверних кластера Kafka, що розміщують чотири розділи (P0-P3) з двома групами споживачів

3.2.1.3 Модуль передобробки текстового потоку

Модулі передобробників реалізовані за допомогою зв’язки Spark Streaming та Apache Spark. Модулі забирають дані з черги через Spark Streaming після чого документ передобробляється та зберігається до бази даних. Та складається в іншу чергу вже для модулів аналізу вилученої інформації, як показано на рисунку 3.8.

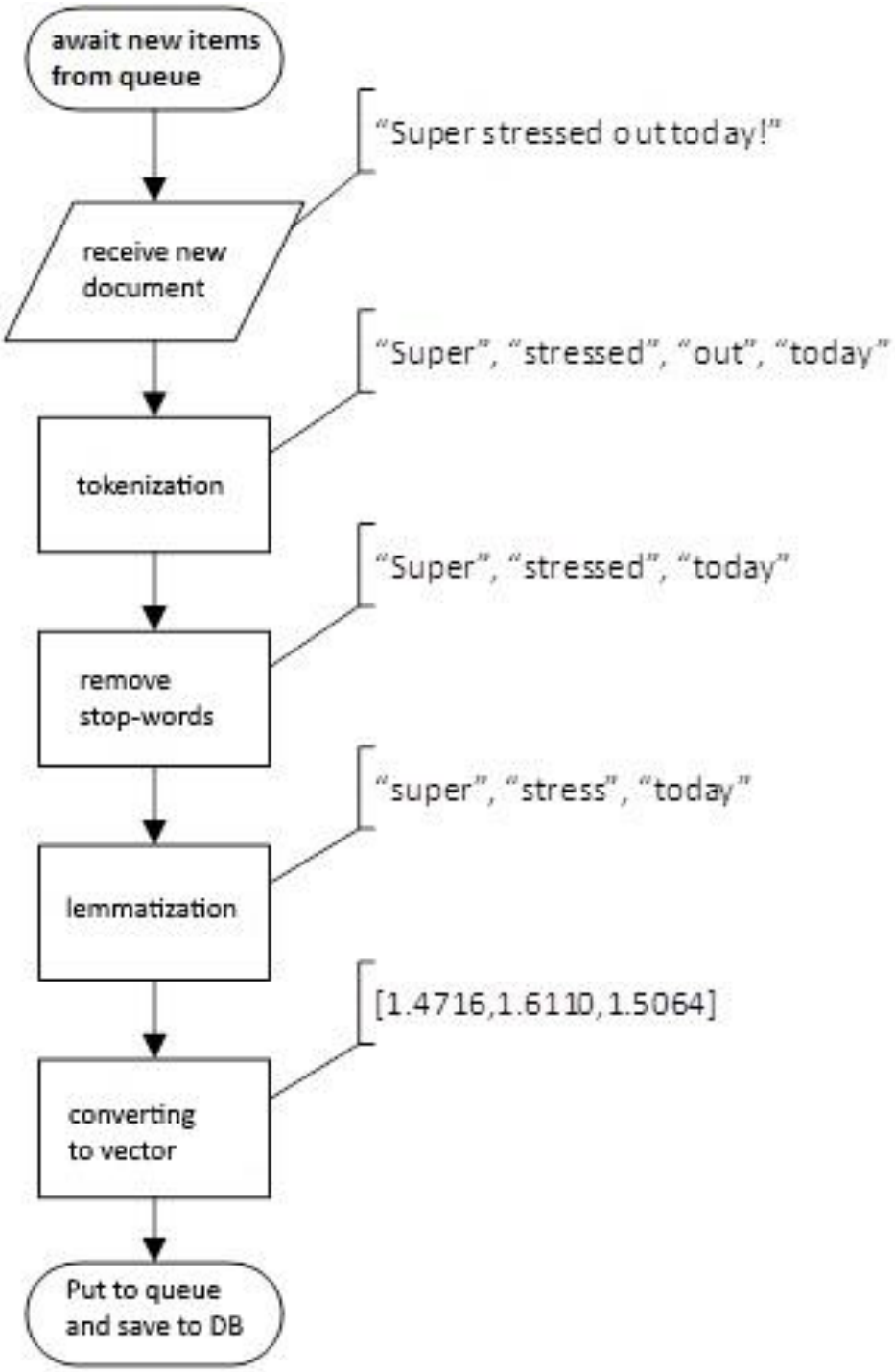


Рисунок 3.8 – Блок схема роботи модуля передобробки

3.2.1.4 Модуль аналізу вилученої інформації з текстового потоку

Модель аналізу вилученої інформації отримує дані з черги та бази даних, після чого проводить кластеризацію за допомогою методу DBSCAN, а за її результатами рекурсивна нейронна мережа опираючись на виділені ознаки назначає однин з класів що включає такі класи як правда, в основному правда, наполовину правда, ледь правда, брехня і безбожна брехня.

Після чого зберігає результати до бази даних та кладе документ у чергу на оцінювання, як це наведено в блок схемі на рисунку 3.9.

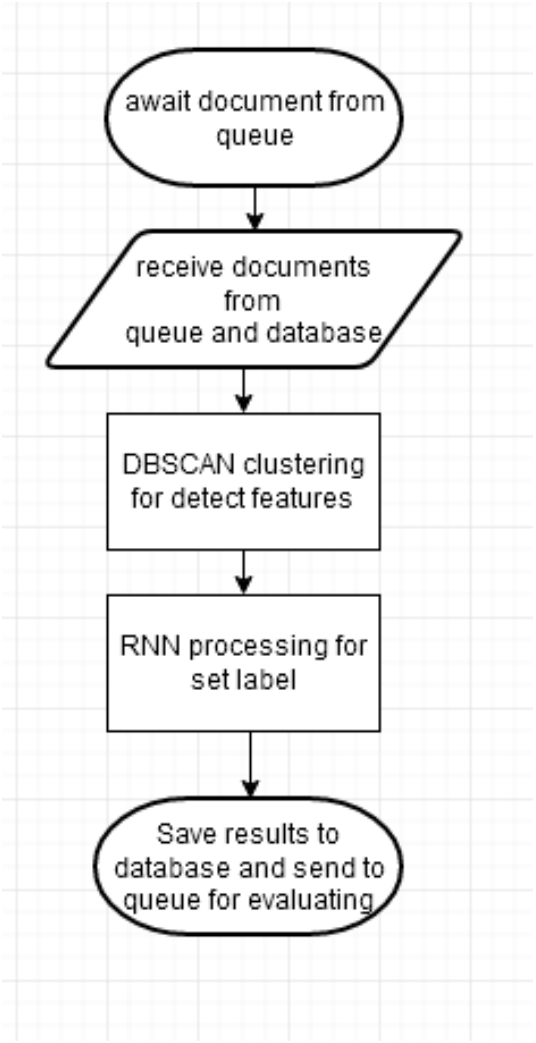


Рисунок 3.9 – Блок схема модулю аналізу

3.2.1.5 Модуль оцінки якості

Модуль оцінювання включає в себе три наступні алгоритми це індекси Silhouette та Davies–Bouldin для оцінки якості кластеризації та коефіцієнт Cohen’s kappa для оцінки узгодженості між документами та назначеними їм мітками. Після чого в залежності від оцінок, якщо результат задовольняє він зберігається до бази даних зі статусом “done” або документ з новими параметрами відправляється у чергу обробки.

3.3 Технології, методи аналізу потокової текстової інформації в режимі реального часу для виявлення фальсифікованих або аномальних новин

Розроблений програмний комплекс з використанням запропонованої технології аналізу текстового потоку з використанням методів кластеризації може використовуватися в якості веб-додатку. Приклад застосування веб-додатку для визначення фальсифікованості новини «Новий китайський вірус заразив сотні» з новинного сайту, що доступна за посиланням https://www.bbc.com/news/health-51148303, представлено на рисунку 3.10, а саме момент запуску програмного комплексу для перевірки текстового потоку.

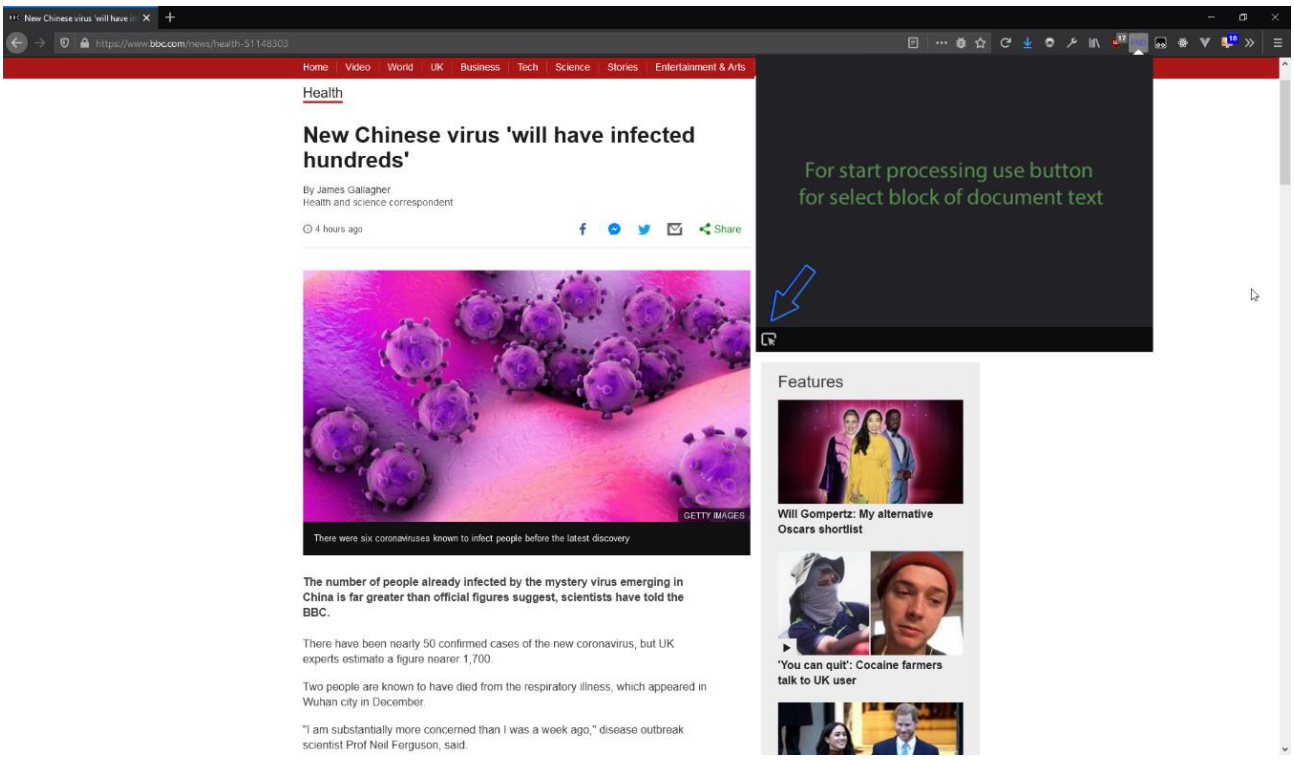


Рисунок 3.10 - Старт перевірки текстового потоку новини

Процес передобробки та аналізу текстового потоку може зайняти деякий час, як це представлено на рисунку 3.11.

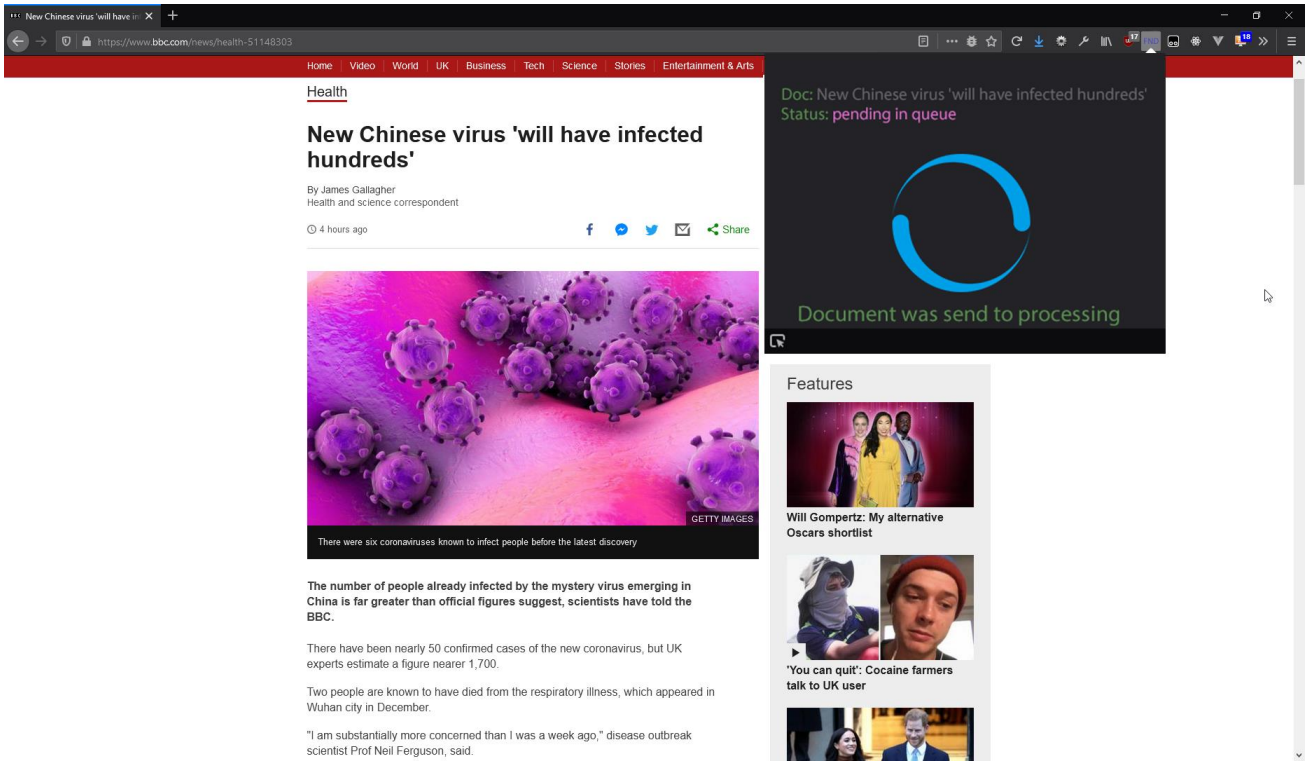


Рисунок 3.11 - Процес передобробки та аналізу текстового потоку

Вікна початку аналізу текстового потоку в браузері та його тривання представлені на рисунках 3.12 та 3.13 відповідно.

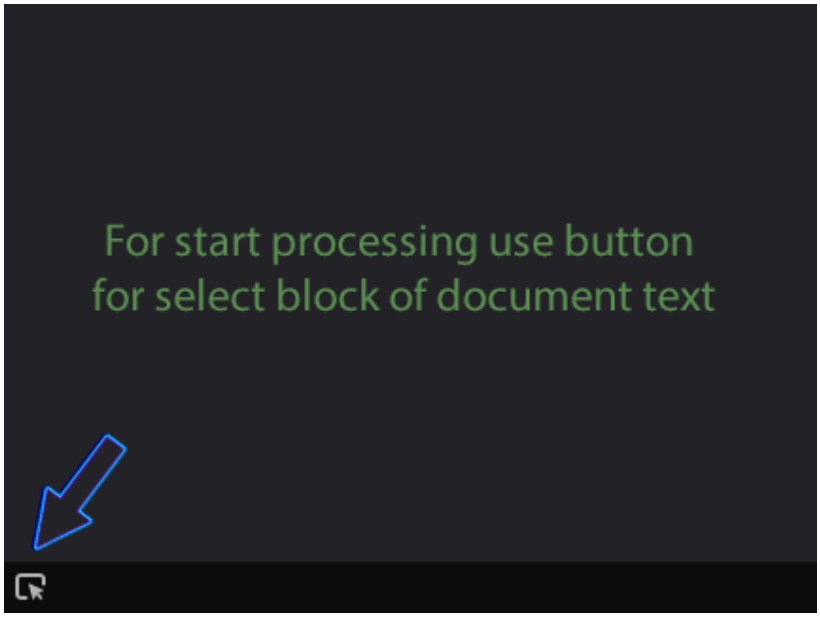


Рисунок 3.12 – Вікно початку аналізу текстового потоку

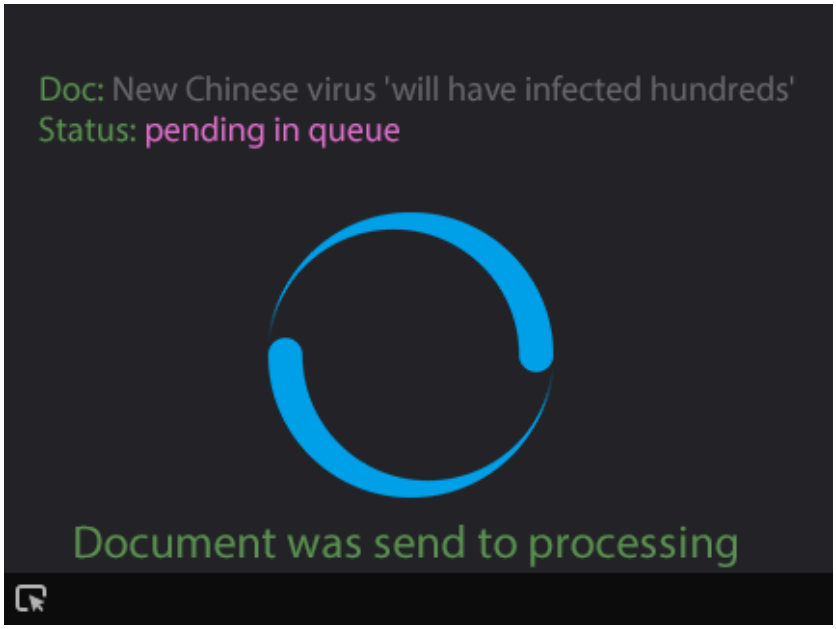


Рисунок 3.13 – Вікно тривання процесу аналізу текстового потоку

Результат аналізу текстового потоку з метою виявлення фальсифікованості новин преставлений на рисунку 3.14.

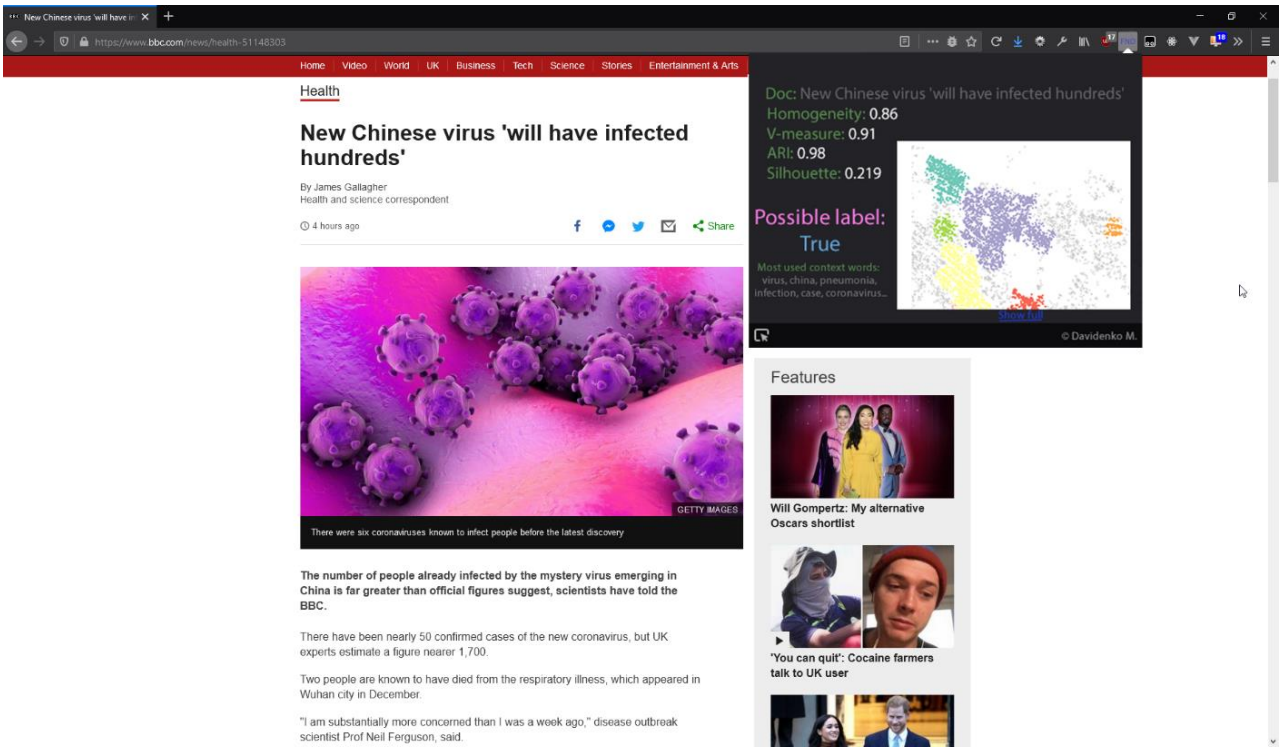


Рисунок 3.14 - Результат аналізу текстового потоку

Як можна побачити з рисунка 3.14, досліджувана новина новина визначена як правдива. Окрім оцінок якості кластеризації, які можна використовувати як оцінки якості виявлення

фальсифікованих новин, також доступна функція візуалізації кластерів текстового потоку, що досліджується, що представлена на рисунку 3.15.

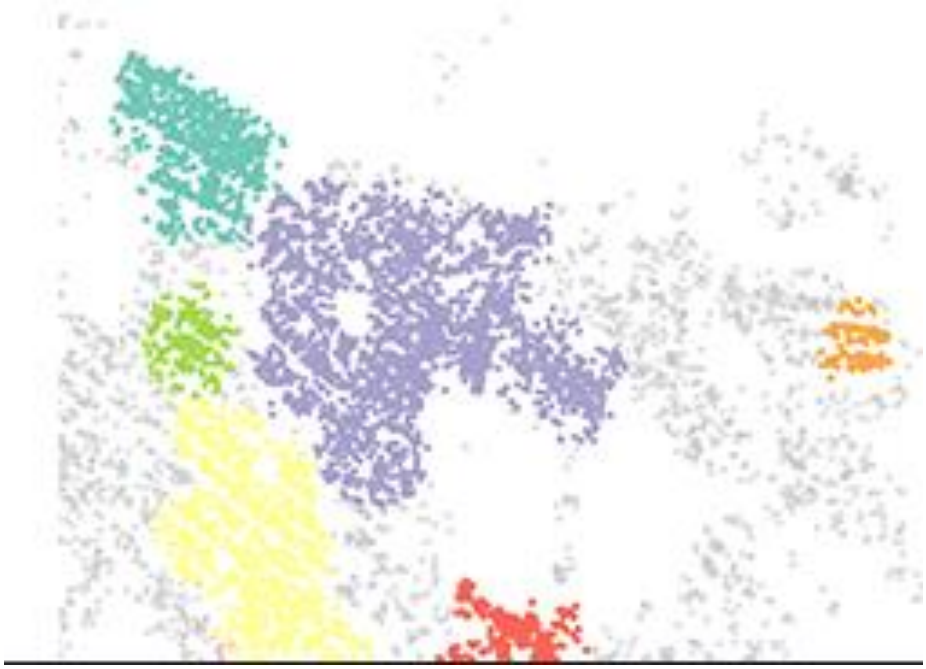


Рисунок 3.15 – Візуалізація точок даних текстового потоку та їх відповідність кластерам

Більш детально принцип роботи технологій, що були використані для етапів передобробки та аналізу вилученої інформації в розробленому програмному комплексі розглянуті далі.

3.3.1 Етап передобробки

Для передобробки текстового потоку в режимі реального часу в розробленому програмному комплексі використовуються наступні технології.

Лематизація, токенізація, видалення стоп слів використовуються для підготовки даних до векторизації, технологія Word2Vec використовується для надання словам документа векторного вигляду.

Лематизація, тобто процес перетворення слова в його базову форм, враховує контекст і перетворює слово в його значиму базову форму що дозволить виключити невірні значення і орфографічні помилки.

Токенізація, тобто процес поділу писемної мови на речення-компоненти, використовується для виокремлення речення кожен раз, коли знаходимо певний знак

пунктуації – крапку або інші розділові знаки або використати розділ по розділовому знаку між словами.

Стоп-слова (шумові) - слова, знаки, символи, які самостійно не несуть ніякого смислового навантаження просто ігноруються системою при обробці тексту.

Word2Vec включає в себе набір алгоритмів для розрахунку векторних уявлень слів, припускаючи, що слова, які використовуються в схожих контекстах, означають схожі речі, тобто семантично близькі.

3.3.2 Етап аналізу вилученної інформації

Алгоритм кластеризації DBSCAN використовується для виявлення аномалії та причетності новин до фальсифікованих. Заснована на щільності просторова кластеризація, для елементів з шумами є алгоритмом кластеризації даних і заснована на щільності точок в деякому просторі і виконує угруповання точок перебування яких самотнє в області їх з малою щільністю. Виявлення аномалій є методом пізнання під час інтелектуального аналізу даних де виявляються ознаки які рідко зустрічаються або мають ярко виражену стилістику, які можуть викликати підозри під час аналізу даних, зважаючи на істотні відмінності від основної частини даних в межах одного корпусу. Корпус може включати себе документи поєднані категоріально, за проміжки часу тощо.

3.3.3 Оцінювання результатів

Індекси Silhouette та Davies–Bouldin використовуються для оцінки якості кластеризації. В свою чергу для виявлення узгодженості між новиною та класами використовується коефіцієнт Cohen's kappa.

Значення Silhouette показує, наскільки об'єкт схожий на свій кластер в порівнянні з іншими кластерами.

Оцінка для всієї кластерної структури:

- середня відстань до інших об'єктів з кластера (компактність);

- середня відстань до об'єктів з іншого кластера (віддільність).

Можна помітити, що чим ближче дана оцінка до 1, тим краще. Є також спрощена варіація силуету і обчислюються через центри кластерів.

Davies-Bouldin індекс - одна з найбільш використовуваних засобів оцінки якості кластеризації.

Вона обчислює компактність, як відстань від об'єктів кластера до їх центроїд, а віддільність - як відстань між центроїдами.

Cohen's kappa була запропонована як характеристика згоди між двома ранжируваннями (наприклад, експертними оцінками). Формально ця величина є деякою характеристикою зв'язку двох випадкових величин. При цьому ці величини є числовими, при необхідності нечислові значення можуть формально бути «закодовані» числами. Cohen's kappa орієнтована на нечислові величини тобто на характеристики, виміряні в номінальній шкалі. Cohen's каппа з вагами дозволяють врахувати відносини, властиві рангових і числових шкал.

3.4 Проведення експерименту з виявлення фальсифікованих новин в потоці текстової інформації

Проведено екперимент з виявлення фальсицікованих новин потоку текстової інформації з використанням методів кластеризації та мови програмування Python.

3.4.1 Досліджувані дані

Був використаний набір даних Джорджа Макінтіра [15]. Набір даних був підготовлений у 2017 році. Він складається з даних із 5279 статей. Статті надходили від медіа-організацій, таких як New York Times, WSJ, Bloomberg, NPR та Guardian, і були опубліковані у 2015 або 2016 роках. Набір даних складається із заголовка та тексту новинної статті у вигляді вхідних змінних та вихідних змінних із двох класів: FAKE або REAL.

Виявлення фальсифікованих новин включає наступні кроки: передобробка даних текстового потоку, навчання моделі передобробки word2vec, використання алгоритму DBSCAN для кластеризації, оцінка якості кластеризації.

3.4.2 Передобробка даних текстового потоку

Висхідні дані текстового потоку були попередньо оброблені. Реалізація модулів які відповідають за це, наведена в додатку Б. Погані символи та стоп слова було видалено та все інше токенізовано. Вихідні дані наведені нижче (рисунок 3.16).

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Рисунок 3.16 - Дані текстового потоку до обробки

Передобробка висхідного текстового потоку проведена з видаленням поганих символів, що зашумлюють текстові дані, токінезацією, видаленням стоп-слів. Фрагмент коду, що використаний для передобробки текстового потоку представлений нижче.

```
from nltk.corpus import stopwords

def preprocessing(n):
    return re.sub(r'[^A-z\s]','',n).lower()

df['title']=df['title'].apply(preprocessing);
df['text']=df['text'].apply(preprocessing);

df_token = df
df_token['title'] = df.apply(lambda row: nltk.word_tokenize(row['title']), axi
s=1)
df_token['text'] = df.apply(lambda row: nltk.word_tokenize(row['text']), axis=
1)

stop_words = stopwords.words('english')
df_token['title'] = df_token['title'].map(lambda t: [word for word in t if wor
d not in stop_words])
df_token['text'] = df_token['text'].map(lambda t: [word for word in t if word
not in stop_words])
df_token.head(10)
```

Дані після попередньої обробки без поганих символів, після токенізації та видалення стоп слів можуть бути представлені наступним чином (рисунок 3.17).

	Unnamed: 0	title	text	label
0	8476	[smell, hillarys, fear]	[daniel, greenfield, shillman, journalism, fel...	FAKE
1	10294	[watch, exact, moment, paul, ryan, committed, ...	[google, pinterest, digg, linkedin, reddit, st...	FAKE
2	3608	[kerry, go, paris, gesture, sympathy]	[us, secretary, state, john, f, kerry, said, m...	REAL
3	10142	[bernie, supporters, twitter, erupt, anger, dn...	[kaydee, king, kaydeeking, november, lesson, t...	FAKE
4	875	[battle, new, york, primary, matters]	[primary, day, new, york, frontrunners, hillar...	REAL
5	6903	[tehran, usa]	[im, immigrant, grandparents, years, ago, arri...	FAKE
6	7341	[girl, horrified, watches, boyfriend, left, fa...	[share, baylee, luciani, left, screenshot, bay...	FAKE
7	95	[britains, schindler, dies]	[czech, stockbroker, saved, jewish, children, ...	REAL
8	4869	[fact, check, trump, clinton, commanderinchief...	[hillary, clinton, donald, trump, made, inaccu...	REAL
9	2909	[iran, reportedly, makes, new, push, uranium, ...	[iranian, negotiators, reportedly, made, lastd...	REAL

Рисунок 3.17 - Дані текстового потоку після предобробки

3.4.3 Навчання моделі word2vec

Фрагмент коду, що використаний для навчання моделі word2vec наведено нижче.

```
from gensim.test.utils import get_tmpfile
from gensim.models import Word2Vec

path = get_tmpfile("word2vec.model")

model = Word2Vec(df_token['title']+df_token['text'], size=150, window=5, min_c
ount=1, workers=4)
model.save("word2vec.model")
```

Після навчання моделі з даними що наведені вище, з розміром вікна 5 (з простим ослабленням швидкості навчання), можна бачити що ця модель може виводити найбільш схожі слова для кожного вхідного слова.

Наприклад, дано контекстне слово “go” та розмір вікна 5, потрібно щоб модель видавала одне з основних слів (одне з слів [stay, come, get, sit, throw, wait, happen, alone, roll, let] у наступному випадку).

model.wv.similar_by_word("go")

```
[
 ('stay', 0.7747250199317932),
 ('come', 0.7660486698150635),
 ('get', 0.751580536365509),
```

```
 ('sit', 0.7376989722251892),
 ('throw', 0.7257541418075562),
 ('wait', 0.7247833013534546),
 ('happen', 0.6911652088165283),
 ('alone', 0.6910232305526733),
 ('roll', 0.6893036365509033),
 ('let', 0.6857764720916748)
]
```

В іншому випадку, враховуючи контекстне слово «політик» та розмір вікна 5, потрібно щоб модель видавала одне з основних слів (одне із слів [destructive, healthcareaboveall, karma, potent, distributionthe, interlocking, ecologically, intolerable, divine, oneness]).

model.wv.similar_by_word("politic")

```
[
 ('destructive', 0.8911725282669067),
 ('healthcareaboveall', 0.888471782207489),
 ('karma', 0.8859128952026367),
 ('potent', 0.8848767876625061),
 ('distributionthe', 0.8825998902320862),
 ('interlocking', 0.8820247054100037),
 ('ecologically', 0.8805824518203735),
 ('intolerable', 0.8803092837333679),
 ('divine', 0.8801225423812866),
 ('oneness', 0.8792058229446411)
]
```

Наступними кроками після навчання моделі word2vec є конвертація слів з її допомогою в векторне уявлення та використання алгоритму DBSCAN для кластеризації.

Фрагмент коду, що застосовано для конвертації слів у вектор представлено нижче.

```
# map(lambda t:model.wv[])
df_vector_title = df_token['title'].map(lambda t: [model.wv[word] for word in
t])
df_vector_text = df_token['text'].map(lambda t: [model.wv[word] for word in t]
)
```

3.4.4 Оцінка якості кластеризації

Багато параметрів може використовуватися для оцінки кластеризації. Наступні параметри використовуються в цьому експеременті: однорідність, повнота, V-міра, Adjusted Rand index, Adjusted Mutual Information та Silhouette coefficient.

Метрика однорідності (homogeneity) маркування кластера дає основну істину. Результат кластеризації задовольняє однорідність, якщо всі його кластери містять лише точки даних, які є членами одного класу.

Метрика повноти (completeness) маркування кластера дає основну істину. Результат кластеризації задовольняє повноту, якщо всі точки даних, які є членами даного класу, є елементами одного кластеру.

V-міра (V-measure)- це гармонічне середнє значення між однорідністю та повнотою та обчислюється наступним чином.

$$V_{measure} = (1 + beta) * homogeneity * \frac{completeness}{beta*homogeneity + completeness} \quad (3.1)$$

Adjusted Rand index є виправленою для випадкості версії Rand index (RI). Таке виправлення випадковості встановлює базову лінію, використовуючи очікувану схожість усіх парних порівнянь між кластеризацією, визначеними випадковою моделлю.

$$ARI = \frac{RI - Expected_{RI}}{\max(RI) - Expected_{RI}} \quad (3.2)$$

Adjusted Mutual Information (AMI) це коригування оцінки Mutual Information (MI) для врахування випадковості. Це пояснює той факт, що показник MI, як правило, вище для двох кластеризацій із більшою кількістю кластерів, незалежно від того, чи є фактично більше інформації, що ділиться. Для двох кластеризацій U і V AMI задається як:

$$AMI(U,V) = \frac{[MI(U,V) - E(MI(U,V))]}{[avg(H(U),H(V)) - E(MI(U,V))]} \quad (3.3)$$

Silhouette coefficient відноситься до методу інтерпретації та валідації послідовності в кластерах даних. Методика дає стисле графічне зображення того, наскільки добре класифікований кожен об'єкт.

Значення Silhouette - це міра того, наскільки об'єкт схожий на власний кластер (згуртованість) порівняно з іншими кластерами (розділення). Silhouette коливається від -1 до +1, де високе значення вказує на те, що об'єкт добре відповідає власному кластеру і погано відповідає сусіднім кластерам. Якщо більшість об'єктів мають високе значення, то конфігурація кластеризації якісна. Якщо багато точок мають низьке або негативне значення, то в конфігурації кластеризації може бути занадто багато або занадто мало кластерів.

В результаті проведення кластеризації отримані наступні значення критеріїв якості кластеризації, що в даному випадку можуть використовуватись в якості критерів якості виявлення фальсифікованих новин.

Однорідність: 1.000
Повнота: 1.000
V-міра: 1.000
Adjusted Rand Index: 1.000
Adjusted Mutual Information: 0.000
Silhouette Coefficient: 0.326

Значення Silhouette Coefficient 0.326 вказує на задовільну якість кластеризації вилученої інформації текстового потоку, в той час, як значення Adjusted Rand Index, що дорівнює 1 вказує на високу якість кластеризації. Високі значення однорідності, повноти та V-міри, що позначає їх відношення, також свідчать про високу якість кластеризації. За більшостю критеріїв оцінки якості кластеризації можна зробити висновок про високу якість кластеризації та, виходячи з цього, про якісне виявлення фальсифікованих новин.

3.5 Висновки до розділу 3

В розділі представлено практичну реалізаціїю технології аналізу потоку текстової інформації в режимі реального часу з використанням розробленого програмного комплексу, який надає можливість приймати, передобробляти, аналізувати та оцінювати якість обробки та виявлення, зберігати текстову інформацію новин для подальшої роботи. Основною мовою програмування для реалізації програмного комплексу використано Java. Допоміжною Python.

Розглянуто використання існуючих інструментів в розробленому програмному комплексі для аналізу потоку текстової інформації в режимі реального часу. Розроблені модулі програмного комплексу: модулі збору новин, модуль прийому новин у вигляді REST сервісу,

модуль черги, модуль передобробки, модуль аналізу, модуль оцінки, база даних. Які в свою чергу відповідають шести типам програм: збирачі, REST інтерфейс, черга, програми передобробки, програми аналізу вилученої інформації і програми оцінки результатів.

Представлено проведення експерименту з виявлення фальсифікованих новин з використанням технології передобробки текстової інформації word2vec, алгоритму кластеризації DBSCAN, та запропонованих оцінок якості кластеризації однорідність, повнота, V-міра, Adjusted Rand index та Silhouette coefficient, що можуть бути використані в якості критеріїв оцінки якості виявлення фальсифікованих новин в текстовому потоці.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної магістерської роботи було розробка технології аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації. За цим програмним продуктом в подальшому розроблятиметься реальна система, яка значно полегшить виявлення фальсифікованих новин. Так як в процесі проектування використовувалося персональний комп’ютер, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера на якому буде розроблятися/використовуватися розроблена програма.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

При роботі з обчислювальною технікою змінюються фізичні і хімічні фактори навколишнього середовища: виникає статична електрика, електромагнітне випромінювання, змінюється температура і вологість, рівень вміст кисню і озону в повітрі. Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, які використовуються для обробки приміщень та обладнання. Неправильна організація робочого місця сприяє загальному і локальній напрузі м'язів шиї, тулуба, верхніх кінцівок, викривлення хребта і розвитку остеохондрозу. На всіх підприємствах, в установах, організаціях повинні створюватися безпечні і нешкідливі умови праці. Забезпечення цих умов покладається на власника або уповноважений ним орган (далі роботодавець). Умови праці на робочому місці, безпека технологічних процесів, машин,

механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. Роботодавець повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизмові, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників. Він не має права вимагати від працівника виконання роботи, поєднаної з явною небезпекою для життя, а також в умовах, що не відповідають законодавству про охорону праці. Працівник має право відмовитися від дорученої роботи, якщо створилася виробнича ситуація, небезпечна для його життя чи здоров'я або людей, які його оточують, і навколишнього середовища.

4.2 Аналіз стану умов праці

Робота над створенням системи автоматизованої обробки експериментальної інформації проходитиме в приміщенні квартири. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1.

Таблиця 4.1 – Розміри приміщення

Найменування	**Значення**
Довжина, м	5
Ширина, м	4
Висота, м	3
Площа, $м^2$	20
Об’єм, $м^3$	60

Згідно з ДСанПіН 3.3.2.007-98 [28] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [28] і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 – Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Робочий стіл на досліджуваному місці також містить достатньо простору для ніг. Крісло, що використовується в якості робочого сидіння, є підйомно поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки, також воно м'яке і виконане з екологічної шкіри, що дає можливість працювати у комфорті. Екран монітору знаходиться на відстані 0.8 м, клавіатура має можливість регулювання кута нахилу 5-15°. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам.

Приміщення кабінету знаходиться на другому поверсі трьох поверхової будівлі і має об’єм 60 м^3, площу – 20 м^2. У цьому кабінеті обладнано три місця праці, з яких два укомплектовані ПК.

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум в лабораторії знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

Розміщення вікон забезпечує природне освітлення з коефіцієнтом природного освітлення не менше 1,5%, а загальне штучне освітлення, яке здійснюється за допомогою восьми люмінесцентних ламп, забезпечує рівень освітленості не менше 200 Лк.

У кабінеті є електрична мережа з напругою 220 В, яка створює небезпеку ураження електричним струмом.

За ступенем пожежної безпеки приміщення належить до категорії В.

Наявна аптечка для надання долікарської допомоги, а також у кабінеті роблять вологе прибирання та щоденно провітрюють приміщення.

4.3 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями» [33], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої.

Основними робочими характеристиками персонального комп'ютера є наступні:

– робоча напруга U = +220В ± 5%;
– робочий струм I = 2А;
– споживана потужність P = 350 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 4 7 ДСанПіН 3.3.2-007-98 [28].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3).

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	ДСН 3.3.6.042-99 [30]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	ГОСТ 12.1.030-81[31] ГОСТ 13109-97[32]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	ДБН В.2.5-28:2015[29]
- підвищена яскравість світла	порушення умов праці (організації місця праці-налагодження моніторів)	1	ДСанПіН 3.3.2.00798[28]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	ДСанПіН 3.3.2.00798[28]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці-сидіння користувача,) та організації робочого часу - безпервна робота)	2	ДСанПіН 3.3.2.00798[28]

4.4 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія

електромережі для живлення ПК, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

4.5 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

Основний потік природного світла при цій повинен бути зліва. Не допускається спрямування основного світлового потоку природного світла праворуч, ззаду і спереду працівника на ПЕОМ.

Робота на ПЕОМ може здійснюватися за таких видах освітлення:

- загальному штучному освітленні, коли відео монітори розташовуються по периметру приміщення або при центральному розташуванні робочих місць у два ряди по довжині кімнати з екранами, звернені в протилежні сторони;

- суміщене освітлення (природне + штучне) тільки при одному і трьох рядном розташуванні робочих місць, коли екран і поверхню робочого столу знаходяться перпендикулярно світла несучій стіні. При цьому штучне освітлення буде виконане стельовими або підвісними люмінесцентними світильниками, рівномірно розміщеними по стелі рядами паралельно світловим прорізам так, щоб екран відео монітора знаходився в зоні захисного кута світильника, і його проекції не доводилися на екран. Працюючі на ПЕОМ не повинні бачити відображення світильників на екрані. Застосовувати місцеве освітлення при роботі на ПЕОМ не рекомендується.

Природне освітлення, коли робочі місця з ПЕОМ розташовуються в один ряд по довжині приміщення на відстані 0,8 - 1,0 м від стіни з віконними прорізами, і екрани знаходяться

перпендикулярно цієї стіни. Основний потік природного світла при цій повинен бути зліва. Не допускається спрямування основного світлового потоку природного світла праворуч, ззаду і спереду працює на ПЕОМ. Оптимальна відстань очей до екрана відео монітора повинна становити 60-70 см, допустиме не менше 50 см. Розглядати інформацію ближче 50 см не рекомендується.

У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації ЕОМ виконується зорова робота IVв розряду точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) рівна 200 лк. Джерелом природного освітлення є сонячне світло.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2015 [29]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2 + b^2} \cdot S_b = (1/8 \div 1/10) \cdot S_n \qquad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

S_n =·5∗4=20 м^2

$S_{вік}$ = 1/8∗20=2.5 м^2

Приймаємо 1 вікно площею S = 1,6 м^2 кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби

передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників *n* виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \tag{4.2}$$

де *E* – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 20 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400 лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \cdot 20 \cdot 1{,}15 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 1.6$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом, приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контура заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлювача η – це відношення діючої провідності цього заземлювача до найбільш можливої його провідності за нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів ηв в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4-0,99. Взаємну

екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача ηс.

Послідовність розрахунку.

а) Визначається необхідний опір штучних заземлювачів Rшт.з.:

$$R_{\text{шт.з}} \frac{R_{\text{д}} \cdot R_{\text{пр.з}}}{R_{\text{пр.з}} - R_{\text{д}}} \tag{4.3}$$

де Rпр.з. – опір природних заземлювачів;

Rд – допустимий опір заземлення.

Якщо природні заземлювачі відсутні, то Rшт.з.=Rд.

Підставивши числові значення у формулу (4.3), отримуємо:

$$R_{\text{шт.з}} = \frac{4 \cdot 40}{40 - 4} \approx 4\text{Ом}$$

б) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом*м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом*м (табличне значення).

в) Розрахунковий питомий опір ґрунту, ρрозр, Ом*м, визначається відповідно для вертикальних заземлювачів ρрозр.в, і горизонтальних ρрозр.г, Ом*м за формулою:

$$\rho_{\text{розр.}} = \psi \cdot \rho, \tag{4.4}$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів ρрозр.в=1,7 і горизонтальних ρрозр.г=5,5 Ом*м.

$$\rho_{\text{розр.в.}} = 1{,}7 \cdot 40 = 68\text{Ом} \bullet \text{м}$$

$$\rho_{\text{розр.г.}} = 5{,}5 \cdot 40 = 220\text{Ом} \bullet \text{м}$$

г) Розраховується опір розтікання струму вертикального заземлювача Rв, Ом, за (4.5).

$$R_{\text{в}} = \frac{\rho_{\text{розв}}}{2 \cdot \pi \cdot 1_{\text{в}}} \left(\frac{ln\, 2 \cdot 1}{d_{\text{ст}}} + \frac{1 \cdot ln}{2 \frac{4 \cdot t + 1_{\text{в}}}{4 \cdot t - 1_{\text{в}}}} \right) \tag{4.5}$$

де lв – довжина розр. вертикального заземлювача (для труб - 2−3 м; lв=3 м); dст – діаметр стержня (для труб - 0,03−0,05 м; dст=0,05 м);

t – відстаньвідповерхніземлі до серединизаземлювача, яка визначається за формулою (4.6):

$$t = h_{\text{в}} + \frac{1_{\text{в}}}{2} \tag{4.6}$$

де hв – глибина закладання вертикальних заземлювачів (0,8 м); тоді

$$t = 0{,}8 + \frac{3}{2} = 2{,}3\text{м}$$

$$R_{\text{в}} = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(ln\frac{2 \cdot 3}{0{,}05} + \frac{1}{2} ln\frac{4 \cdot 2{,}3 + 3}{4 \cdot 2{,}3 - 3} \right) = 18{,}50\text{м}$$

г) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахуваннякоефіцієнтавикористання η в:

$$n = \frac{2 \cdot R_{\text{в}}}{R_{\text{д}}} = \frac{2 \cdot 18{,}5}{4} = 9{,}25 \tag{4.7}$$

Визначаєтьсякоефіцієнтвикористаннявертикальнихелектродівгруповогозаземлювача без врахуваннявпливуз’єднувальноїстрічкиηв =0,57 (табличнезначення).

д) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання nв, шт:

$$n_{\text{в}} = \frac{2 \cdot R_{\text{в}}}{R_{\text{д}} \cdot \eta_{\text{в}}} = \frac{2 \cdot 18{,}5}{4 \cdot 0{,}57} = 16{,}2 \approx 16 \tag{4.8}$$

е) Визначається довжина з’єднувальної стрічки горизонтального заземлювача lс, м:

$$l_c = 1{,}05 \cdot L_{\text{в}} \cdot (\text{п}_{\text{в}} - 1) \tag{4.9}$$

де Lв – відстань між вертикальними заземлювачами, (прийняти за Lв = 3м); nв – необхідна кількість вертикальних заземлювачів.

$$1_c = 1{,}05 \cdot 3 \cdot (16 - 1) \approx 84\text{м}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}<4$ Ом, а саме:

$$R_{\text{заг}} = \frac{18{,}5 \cdot 8{,}1}{18{,}5 \cdot 0{,}3 + 8{,}1 \cdot 16 \cdot 0{,}57} = 1{,}9 \leq R_{\text{д}}$$

При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

-іскри і дуги коротких замикань;

-перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

-іскри при розмиканні і розмиканні ланцюгів;

-розряди статичної електрики;

-необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. В приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися згідно з нормами. У приміщеннях повинна бути пожежна сигналізація, вогнегасник.

У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків таусуненню пожежі.

4.6 Вплив на навколишнє середовища

На даний момент найбільш суворим з існуючих світових стандартів екологічності для комп'ютерної техніки є стандарт TCO99.

У порівнянні з попередніми він містить додаткові обмеження по частині екології, ергономіки, енергоспоживання і емісії пристроїв.

Найбільш значимі ярлики, такі як «Блакитний ангел», що видається Німецької

сертифікаційної організацією як знак відповідності екологічним стандартам, є великою рідкістю в сфері електроніки. навпаки, широке поширення отримав логотип «Energy Star», якого удостоюються енергозберігаючі пристрої. Однак у випадку з ним проблема полягає в тому, що кожен виробник має право самостійно маркувати свою продукцію, не проходячи при цьому перевірок.

З огляду на те що дана емблема не несе ніяких відомостей про дійсний енергоспоживанні пристроїв, її цілком можна ігнорувати. ЖК-екрани - один з джерел парникових газів, які набагато шкідливіше діоксиду вуглецю. Рідкокристалічні монітори швидко знайшли популярність, прийшовши на зміну громіздким ЕПТ-моделями. І це не дивно, адже вони мають тонкими корпусами і споживають значно менше електроенергії. За іншим аспектам екологічної безпеки дисплеї на основі рідких кристалів також вважалися проривом, тому що в них не використовувався газ, що містить свинець. Досить довго ніхто не звертав уваги на застосовуваний для чищення РК-панелей трехфторістий азот (NF3), і тільки в середині 2008 року вченими було доведено наявність даного хімічної речовини в атмосфері. Відкриття було вражаючим: по порівняно з діоксидом вуглецю (CO2) NF3 є в 17 000 разів більше активним парниковим газом, а його атмосферний час напіврозпаду може складати від 550 до 740 світлових років (у CO2 - від 30 до 40 років). Закону, який обмежував би рівень викиду NF3, поки не існує.

Виявлення енерговитрат є таким же проблематичним процесом, як і визначення кількості матеріалів, придатних для вторинної переробки, і важких металів, що містяться в пристроях. Дивовижний результат був отриманий організацією Greenpeace в ході порівняльного аналізу декількох моделей ідентичних ноутбуків з різних країн. У тачпаде Dell Vostro V13, доступного на китайському ринку, були виявлені сліди брому. В моделі з Німеччини ця речовина теж присутнє, тільки не в тачпаде, а кнопках. У лептопі, купленому в США, бром був знайдений в блоці живлення.

Схожа картина спостерігається і у інших виробників: при дослідженні продукції компанії Apple експерти виявили, що в кабелі ноутбука MacBook Pro 13 з США і Нідерландів міститься в три рази більше брому, ніж в пристроях з Філіппін і Росії. При аналізі іншого кабелю сліди брому виявили вже в пристроях з Росії та Нідерландів, а в моделі з США їх не було. Таким чином, надійним показником екологічності залишається тільки рівень енергоспоживання - серед субноутбуков першість належить лише декільком моделям, а решта різко відрізняються від лідерів за своїми характеристиками

4.7 Висновки до розділу 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Було визначено параметри факторів, що впливають на екологічний стан навколишнього середовища, при порушенні викидання сміття і небезпечних відходів. Комп'ютери та інша електроніка, лампи освітлення завдадуть великої шкоди екології при неправильній утилізації.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

Метою дипломної роботи визначено підвищення точності аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації в умовах нерозмічених даних, дослідження методів вилучення, обробки та аналізу текстової інформації.

В ході дослідницької частини роботи були отримані наступні результати:

1. Проведено аналіз галузі застосування та технології аналізу текстового потоку в режимі реального часу.
2. Визначені етапи, сукупність методів та инструментів технології аналізу текстового потоку.

В ході практичної частини роботи були отримані наступні результати:

1. Розроблено програмний комплекс для аналізу текстового потоку в режимі реального часу.
2. Протестовано запропоновану технологію аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці.
3. Визначено оцінку якості виявлення фальсифікованих новин в текстовому потоці.

Таким чином, вирішено поставлене завдання - аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації в умовах нерозмічених даних, досліджено методи, технології вилучення, передобробки та аналізу текстового потоку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Anandarajan, M., Hill, C., & Nolan, T. (2018). Introduction to Text Analytics. Advances in Analytics and Data Science, 1–11. doi:10.1007/978-3-319-95663-3_1

2. Hearst, M. A. (1999a, June). Untangling text data mining. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (pp. 3–10). Association for Computational Linguistics.

3. Gupta, V., & Lehal, G. S. (2009). A survey of text mining techniques and applications. Journal of Emerging Technologies in Web Intelligence, 1(1), 60–76.

4. Tips for spotting false news. Access mode: https://techcrunch.com/2017/04/06/facebook-puts-link-to-10-tips-for-spotting-false-news-atop-feed/

5. The Independent Shih, Gerry. Boston marathon bombings: How twitter and reddit got it wrong. Available online: http://www.independent.co.uk/news/world/americas/boston-marathon-bombings-how-twitter-and-reddit-got\-it-wrong-8581167.html, April 2013.

6. Kenneth Rapoza. Can ‘fake news’ impact the stock market? Available online: https://www.forbes.com/sites/kenrapoza/2017/02/26/can-fake-news-impact-the-stock-market/#6584de9d2fac, Feb 2017

7. Мультимедійна платформа іномовлення України «УКРІНФОРМ» Режим доступу: https://www.ukrinform.ua/rubric-society/2803594-lise-11-ukrainciv-mozut-vidrizniti-spravzni-novini-vid-fejkiv.html

8. Мультимедійна платформа іномовлення України «УКРІНФОРМ» Режим доступу: https://www.ukrinform.ua/rubric-society/2803659-rosijski-zmi-ak-dzerelo-novin-vikoristovuut-13-ukrainciv.html

9. News and analysis - Ukraine Under Information Fire. Access mode: https://euvsdisinfo.eu/ukraine-under-information-fire/

10. Метод кластеризации текстов, учитывающий совместную встречаемость ключевых терминов, и его применение к анализу тематической структуры новостного потока, а также ее динамики. Режим доступу: http://elar.urfu.ru/bitstream/10995/1421/1/IMAT_2005_22.pdf

11. Efficient Estimation of Word Representations in Vector Space Access mode: https://arxiv.org/pdf/1301.3781.pdf

12. Fake news stance detection using stacked ensemble of classifiers, James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, Andreas Vlachos. Access mode: https://arxiv.org/pdf/1405.4053.pdf

13. Quoc Le, Tomas Mikolov: Distributed Representations of Sentences and Documents. Access mode: https://cs.stanford.edu/~quocle/paragraph_vector.pdf

14. Natali Ruchansky, Sungyong Seo, Yan Liu, CSI: A Hybrid Deep Model for Fake News Detection. Access mode: https://arxiv.org/pdf/1703.06959.pdf

15. Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, Huan Liu, Fake News Detection on Social Media: A Data Mining Perspective. Access mode: https://www.kdd.org/exploration_files/19-1-Article2.pdf.

16. Kelly Stahl, Fake news detection in social media Access mode: https://www.csustan.edu/sites/default/files/groups/University%20Honors%20Program/Journals/02_stahl.pdf.

17. Наївний баєсів класифікатор. Режим доступу: https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%97%D0%B2%D0%BD%D0%B8%D0%B9_%D0%B1%D0%B0%D1%94%D1%81%D1%96%D0%B2_%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%82%D0%BE%D1%80

18. Jiawei Zhang , Bowen Dong , Philip S. Yu, FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network. Access mode: https://arxiv.org/pdf/1805.08751.pdf

19. Киселев М. В. Пивоваров В. С. Шмулевич М. М. , Метод кластеризации текстов, учитывающий совместную встречаемость ключевых терминов, и его применение к анализу тематической структуры новостного потока, а также ее динамики. Режим доступу: http://elar.urfu.ru/bitstream/10995/1421/1/IMAT_2005_22.pdf

20. A review on outlier detection techniques on data stream by using different approaches of K-Means algorithmP Chauhan, M Shukla - 2015 International Conference on …, 2015 - ieeexplore.ieee.org

21. Outlier Detection for a 2D Feature Space in Python. Access mode: https://towardsdatascience.com/outlier-detection-python-cd22e6a12098

22. Seyedmehdi Hosseinimotlagh, Evangelos E. Papalexakis, Access mode: https://www.cs.ucr.edu/~epapalex/papers/wsdm18-mis2-fakenews.pdf

23. Potthast, M., Kiesel, J., Reinartz, K., Bevendorff, J. And Stein, B., 2017. A stylometric inquiry into hyperpartisan and fake news. arXiv preprint arXiv:1702.05638.

24. Tacchini, E., Ballarin, G., Della Vedova, M.L., Moret, S. and de Alfaro, L., 2017. Some like it hoax: Automated fake news detection in social networks. arXiv preprint arXiv:1704.07506.

25. Dietterich, T.G., 2000, June. Ensemble methods in machine learning. In International workshop on multiple classifier systems (pp. 1-15). Springer, Berlin, Heidelberg.

26. Shu, K., Sliva, A., Wang, S., Tang, J. and Liu, H., 2017. Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), pp.22-36.

27. Наказ Комітету по нагляду за охороною праці міністерства праці та соціальної політики України від 29 січня 1998 року N 9, НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці. Режим доступу: https://dnaop.com/html/64/doc-НПАОП_0.00-4.15-98.

28. Затверджено постановою головного державного санітарного лікаря України 10 грудня 1998 р. № 7, ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Режим доступу: https://dnaop.com/html/2297/doc-ДСанПін_3.3.2.007-98.

29. Наказ Мінітерства регіонального розвитку, будівництва та житлового-комунального господарства України 03.10.2018 №264 ДБН В.2.5-28:2018 Природнє і штучне освітлення. Режим доступу: https://okna.ua/img_all/oknaua/dbn-V-2-5-28-2018-ed.pdf.

30. Головний державний санітарний лікар України постанову від 1 грудня 1999 року № 42, ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих. Режим доступу: https://dnaop.com/html/31678/doc-ДСН_3.3.6.042-99.

31. Затверджено і введено в дію Ухвалою Державного комітету СРСР по стандартах від 15.05.81 № 2404, ГОСТ 12.1.030-81 ССБТ Електробезпечність. Захисне заземлення. Занулення. Режим доступу: https://dnaop.com/html/31493/doc-ГОСТ_12.1.030-81.

32. Прийнято Міждержавною Радою із стандартизації, метрології та сертифікації протокол № 12-97 від 21 листопада 1997 р., . ГОСТ 13109-97 „ Електрична енергія. Сумісність технічних засобів електромагнітних. Норми якості електроенергоснабженія загального призначення”. Режим доступу: https://dnaop.com/html/42313/doc-ГОСТ_13109-97.

33. Наказ Міністерства соціальної політики України "Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями" від 14.02.2018 № 207, НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями. Режим доступу: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php.

Додаток А Слайди презентації

Рисунок А.1 – Слайд №1

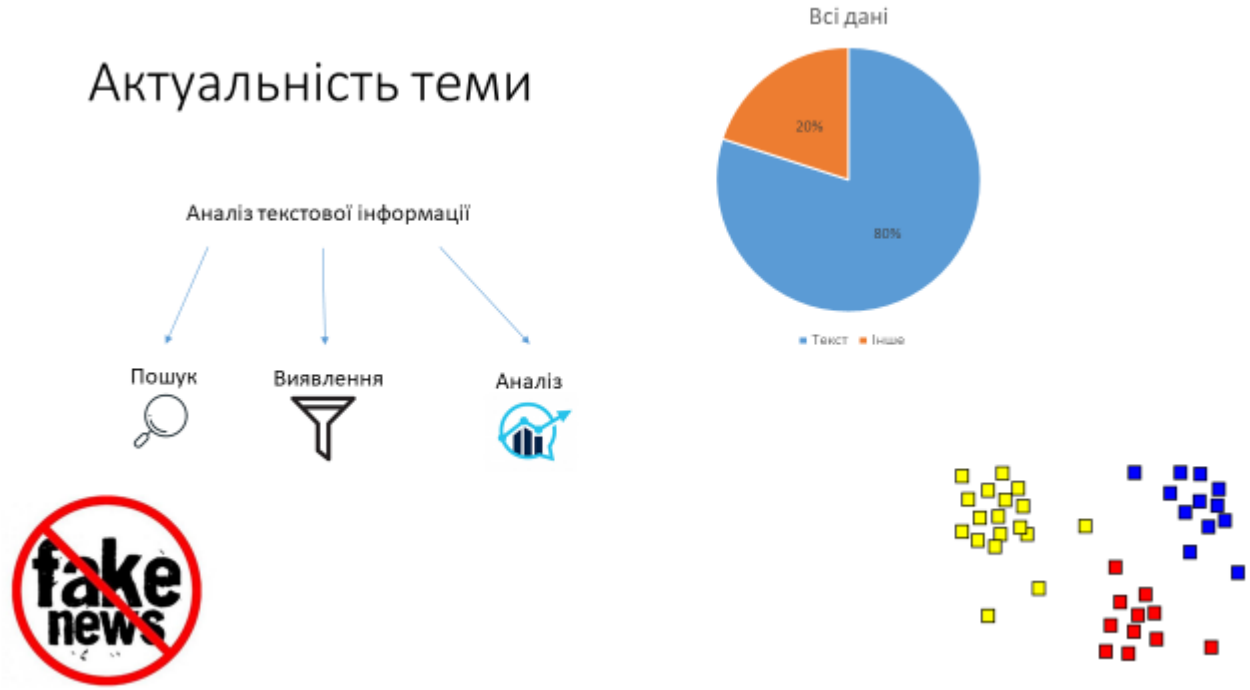


Рисунок А.2 – Слайд №2

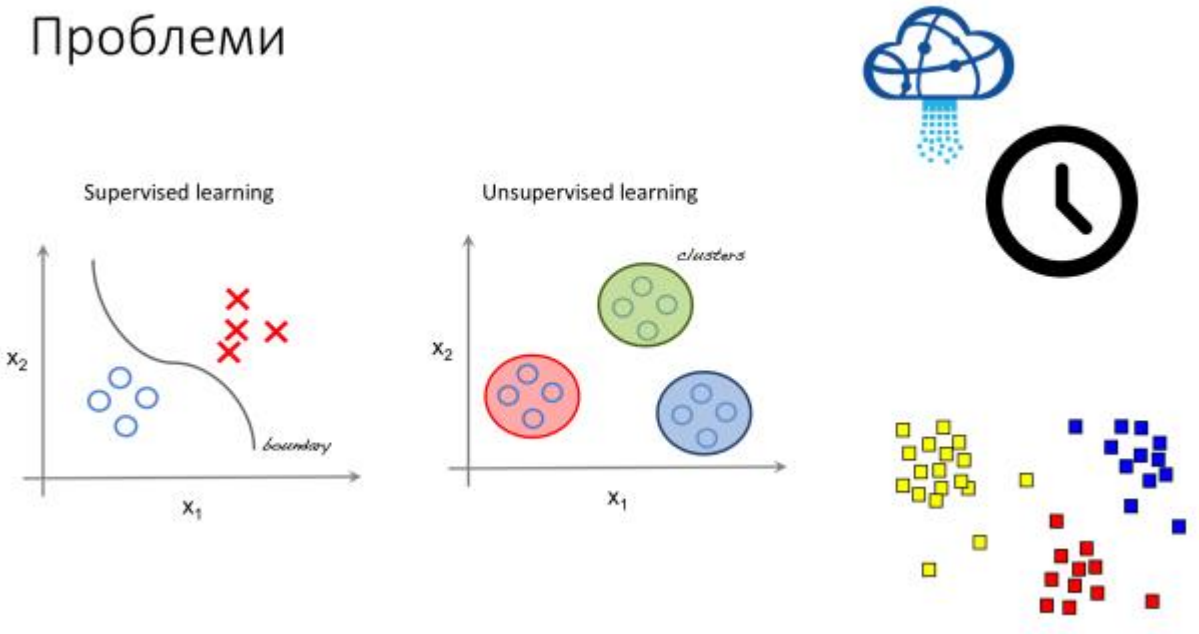


Рисунок А.3 – Слайд №3

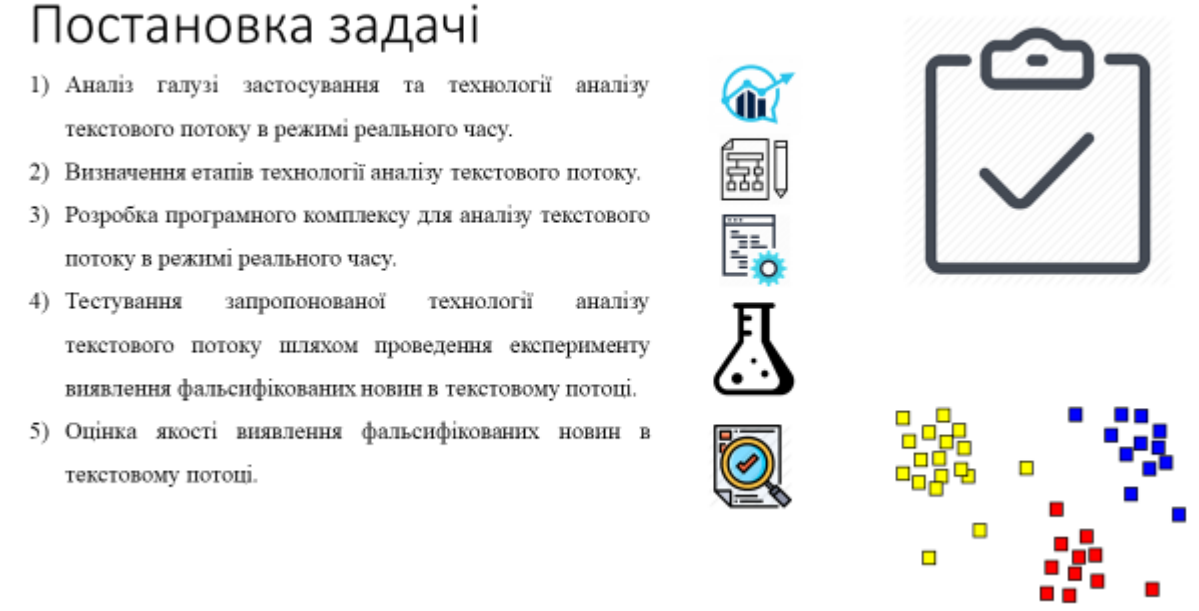


Рисунок А.4 – Слайд №4

Дослідження

Об'єкт дослідження: процес вилучення, обробки, аналізу потоку текстової інформації.

Предмет дослідження: сукупність методів та інструментів для аналізу потоку текстової інформації в режимі реального часу.

Мета і завдання дослідження. Метою дослідження є підвищення точності аналізу потоку текстової інформації в режимі реального часу з використанням методів кластеризації в умовах нерозмічених даних, дослідження методів, технологій вилучення, передобробки та аналізу текстового потоку.

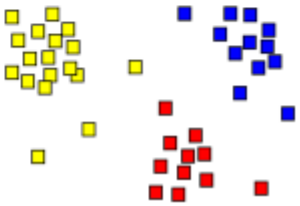


Рисунок А.5 – Слайд №5

Дослідження

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

- Аналіз галузі застосування та технології аналізу текстового потоку в режимі реального часу.
- Визначення етапів, сукупності методів та інструментів технології аналізу текстового потоку.
- Розробка програмного комплексу для аналізу текстового потоку в режимі реального часу.
- Тестування запропонованої технології аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці.
- Оцінка якості виявлення фальсифікованих новин в текстовому потоці.

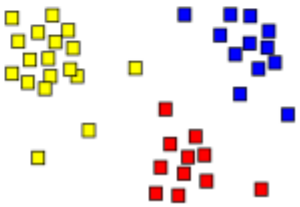


Рисунок А.6 – Слайд №6

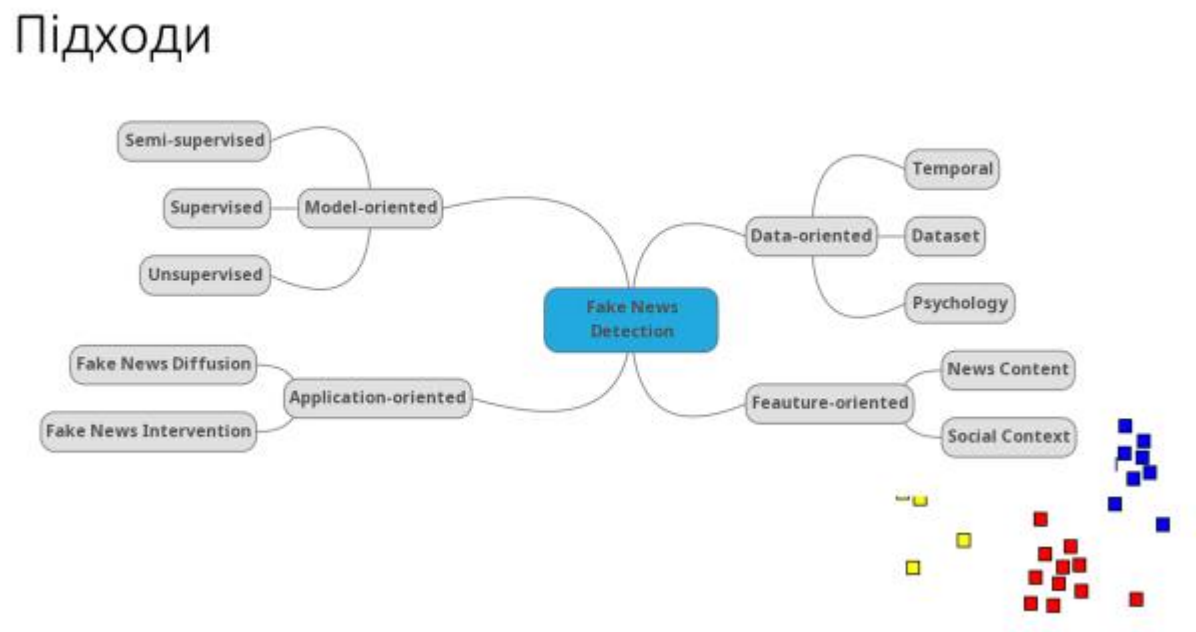


Рисунок А.7 – Слайд №7

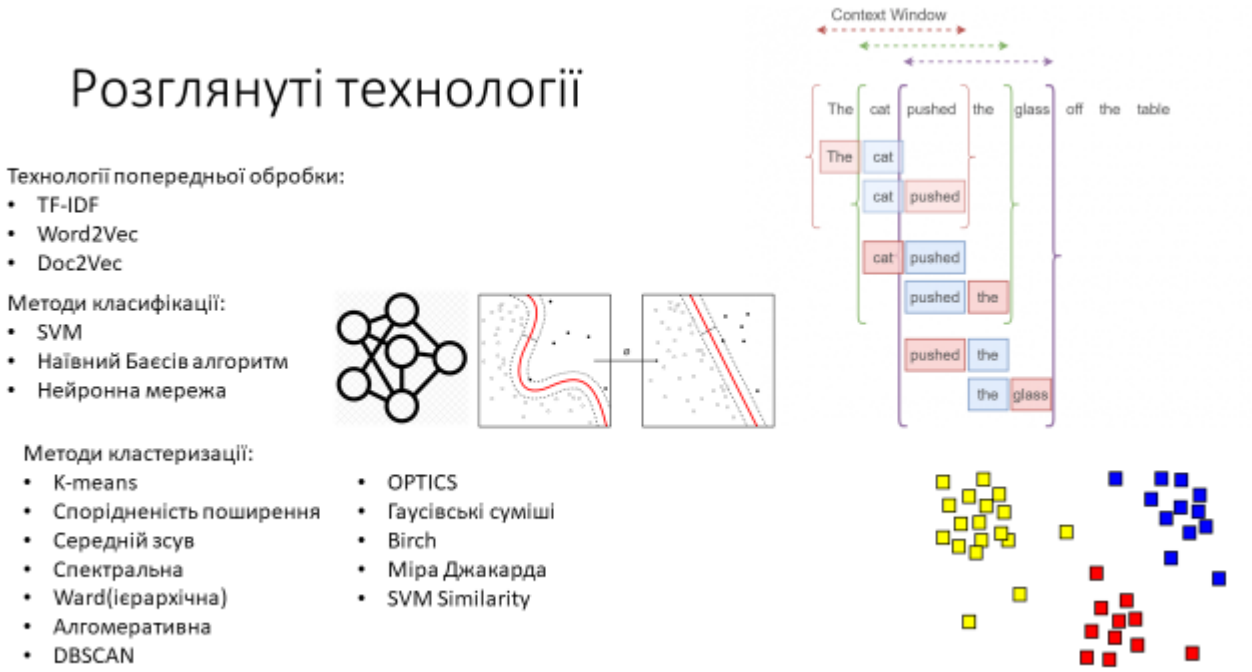
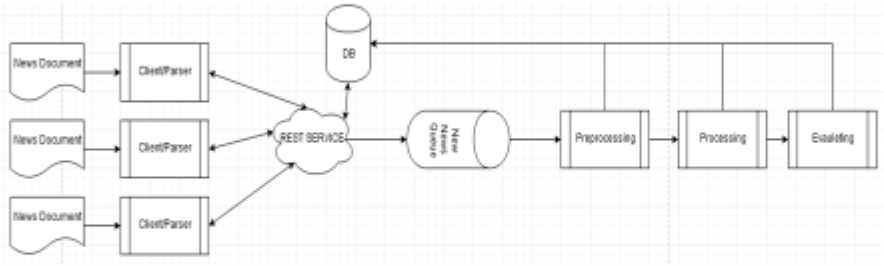


Рисунок А.8 – Слайд №8

Архітектура програмного комплексу



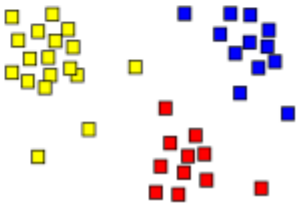


Рисунок А.9 – Слайд №9

TF-IDF

$$tf(t,d) = \frac{n_t}{\sum_k n_k},$$

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$

	Doc 1	Doc 2	...	Doc *n*
Term(s) 1	12	2	...	1
Term(s) 2	0	1	...	0
...	...	...	...	
Term(s) *n*	0	6	...	3

Вага деякого слова пропорційний частоті вживання цього слова в документі і обернено пропорційна частоті вживання слова в усіх документах колекції.

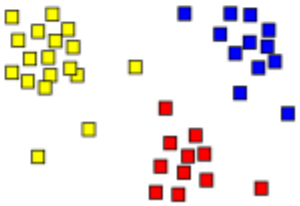


Рисунок А.10 – Слайд №10

Рисунок А.11 – Слайд №11

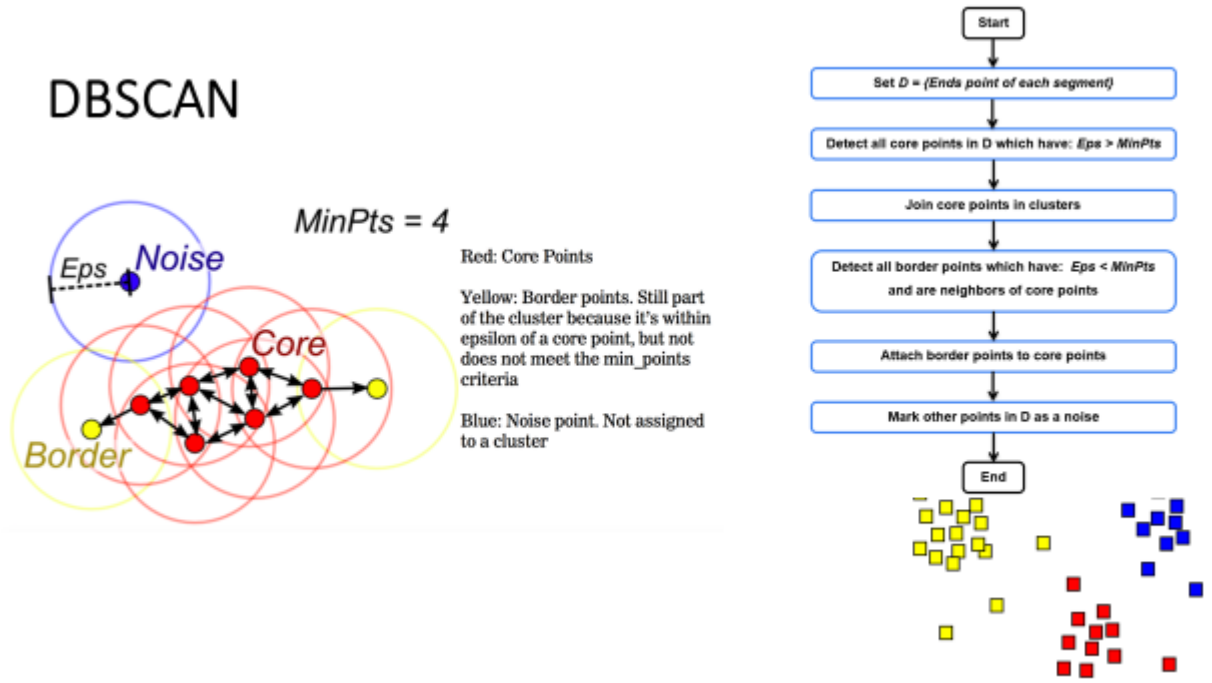


Рисунок А.12 – Слайд №12

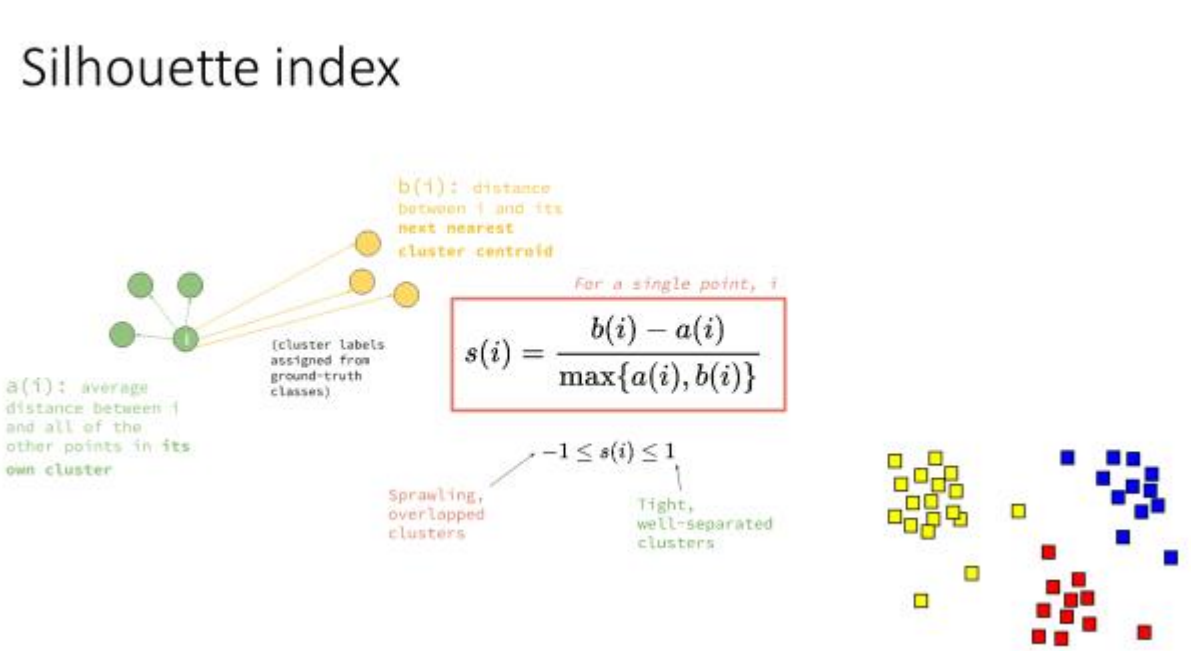


Рисунок А.13 – Слайд №13

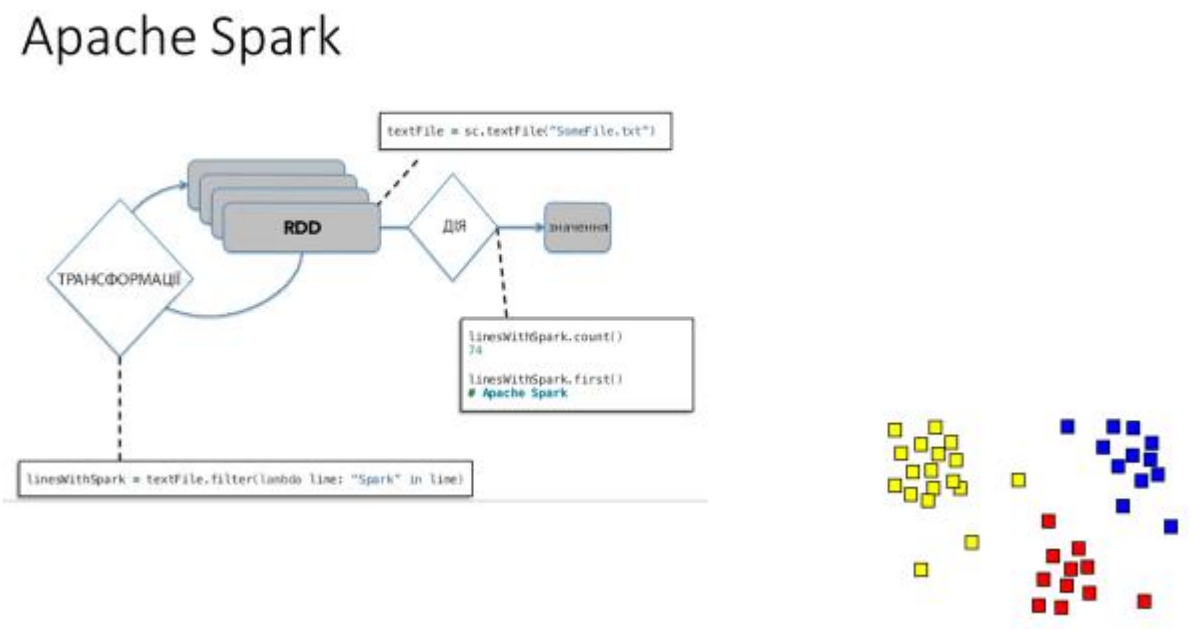


Рисунок А.14 – Слайд №14

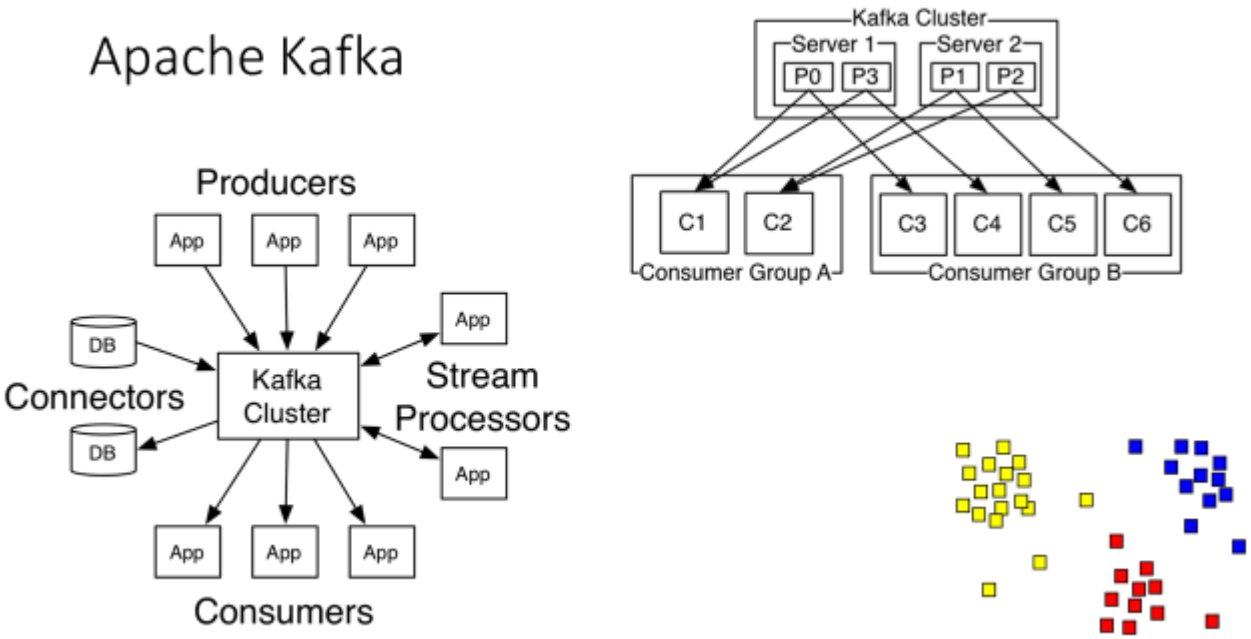


Рисунок А.15 – Слайд №15

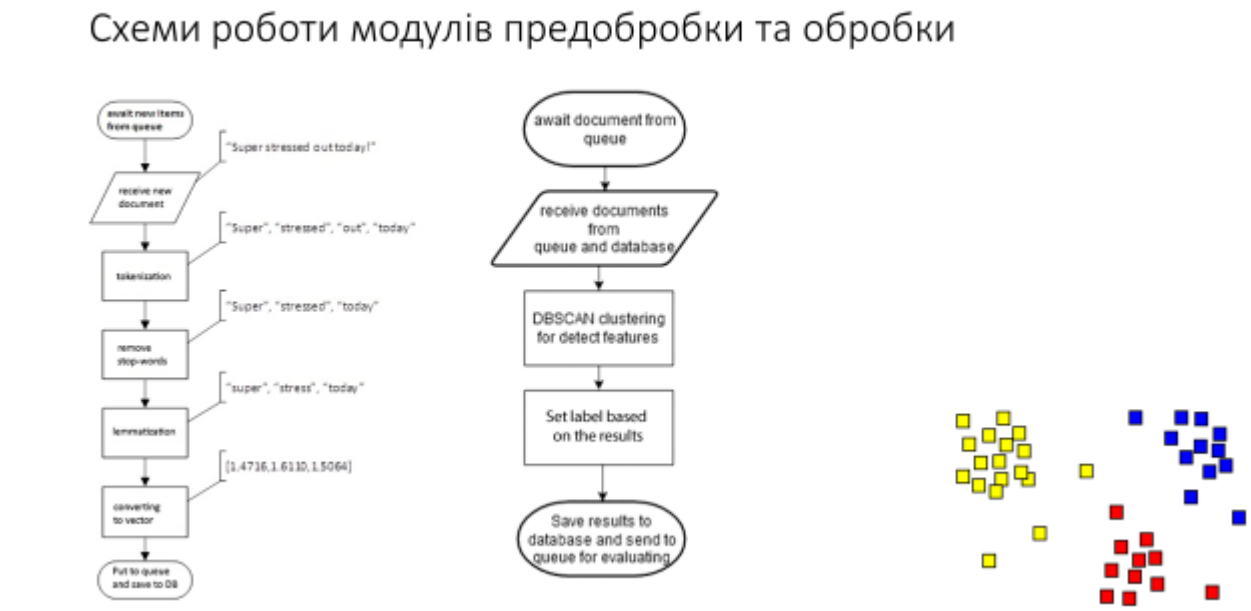


Рисунок А.16 – Слайд №16

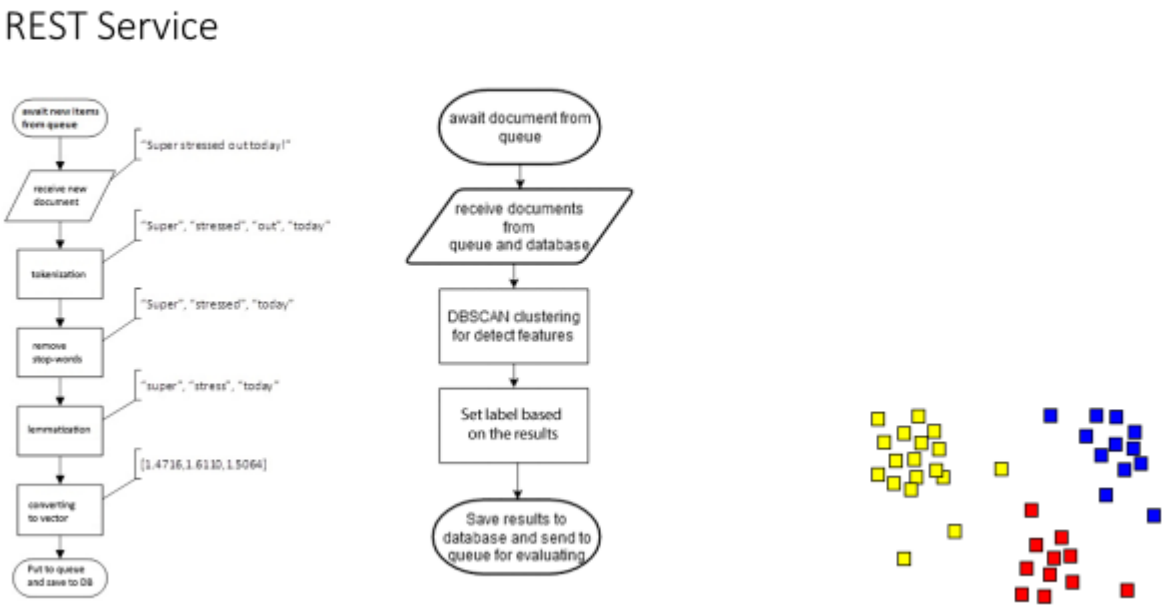


Рисунок А.17 – Слайд №17

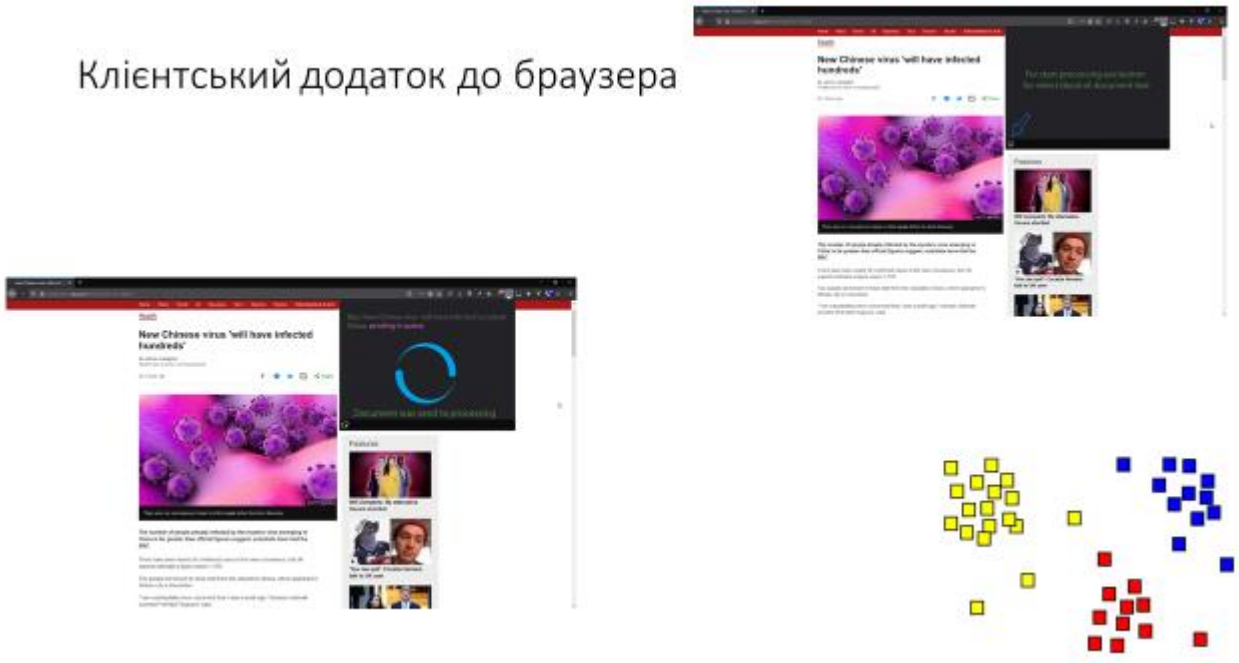


Рисунок А.18 – Слайд №18

Клієнтський додаток до браузера

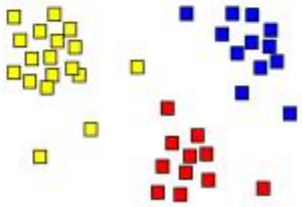


Рисунок А.19 – Слайд №19

Тестування та результат

Однорідність: 1.000

Повнота: 1.000

V-міра: 1.000

Adjusted Rand Index: 1.000

Adjusted Mutual Information: 0.000

Silhouette Coefficient: 0.326

Набір даних для тестування включає в себе 5279 розмічених новини (FAKE, REAL). Що є зручно для проведення експерименту.

Значення Silhouette Coefficient 0.326 вказує на задовільну якість кластеризації вилученої інформації текстового потоку, в той час, як значення Adjusted Rand Index, що дорівнює 1 вказує на високу якість кластерізації.

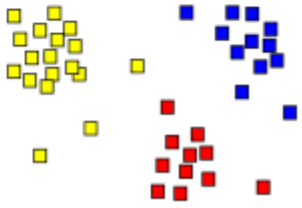


Рисунок А.20 – Слайд №20

Висновки

В ході дослідницької частини роботи були отримані наступні результати:

- Проведено аналіз галузі застосування та технології аналізу текстового потоку в режимі реального часу.
- Визначені етапи, сукупність методів та инструментів технології аналізу текстового потоку.

В ході практичної частини роботи були отримані наступні результати:

- Розроблено програмний комплекс для аналізу текстового потоку в режимі реального часу.
- Протестовано запропоновану технологію аналізу текстового потоку шляхом проведення експерименту виявлення фальсифікованих новин в текстовому потоці.
- Визначено оцінку якості виявлення фальсифікованих новин в текстовому потоці.

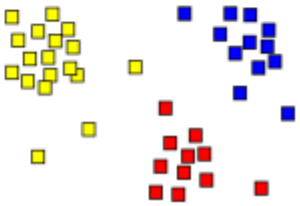


Рисунок А.21 – Слайд №21

Додаток Б Лістинги файлів програмного комплексу

```
// Лістинг файлу swagger.yml

openapi: 3.0.0
info:
  title: Fake news detection API
  description: REST interface for work with fake news detection complex
  version: 0.0.1
components:
  schemas:
    Errors:
      type: object
      properties:
        field:
          type: array
          items:
            type: string
        non_field_errors:
          type: array
          items:
            type: string
    DocumentList:
      type: object
      properties:
        offset:
          type: integer
        items:
          type: array
          items:
            $ref: '#/components/schemas/Document'

    Document:
      type: object
      properties:
        id:
          type: integer
          format: int64
          example: 1

        status:
          type: string
          enum: [pending, preprocessing, processing, evaluation, done]

        title:
          type: string
          nullable: true
          example: "A great news title"
        body:
          type: object
          properties:
            raw:
              type: string
              example: "A great big document body..."
            preprocessed:
              type: string
              example: "0.12132213 0.14, 0.1134...."
              nullable: true
        result:
          type: object
          nullable: true
          properties:
```

```
              label:
                type: string
                enum: [fake, 'true']
              clustering:
                type: string
              nn:
                type: string


          evaluations:
            type: object
            properties:
              silhouette_index:
                type: integer
                example: 0.345
                nullable: true
              db_index:
                type: integer
                example: 0.0
                nullable: true
              cohen_coef:
                type: integer
                example: 0.648
                nullable: true


paths:
  /documents:
    get:
      summary: 'Method for get list of sended documents'
      parameters:
      - in: query
        name: limit
        schema:
          type: integer
          maximum: 5
      - in: query
        name: offset
        schema:
          type: integer
      - in: query
        name: status
        schema:
          type: string
          enum: [pending, preprocessing, processing, evaluation, done]
      operationId: GetDocuments
      responses:
        200:
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/DocumentList'

          description: List document by filter
    post:
      summary: Method for send document to processing
      description: ''
      operationId: AddDocument
      requestBody:
        content:
          application/json:
            schema:      # Request body contents
              type: object
```

```
123                 properties:
124                   title:
125                     type: string
126                   body:
127                     type: string
128                 example:   # Sample object
129                   title: Title of some document or news
130                   body: Text of some document or news
131         responses:
132           201:
133             content:
134               application/json:
135                 schema:
136                   $ref: '#/components/schemas/Document'
137             description: return document after creating
138           400:
139             content:
140               application/json:
141                 schema:
142                   $ref: '#/components/schemas/Errors'
143             description: If client send some wrong fields to server
144
145     /documents/{documentId}:
146       get:
147         summary: Get document data by Id
148         parameters:
149           - in: path
150             name: documentId
151             schema:
152               type: integer
153             required: true
154             description: Numeric ID of the document to get
155         responses:
156           200:
157             content:
158               application/json:
159                 schema:
160                   $ref: '#/components/schemas/Document'
161             description: return document by id if document exist
162           404:
163             description: document not found
164
165   // Лістинг файлу src/main/java/io/api/DocumentsApi.java

166   /**
167    * NOTE: This class is auto generated by the swagger code generator program
168   (3.0.16).
169    * https://github.com/swagger-api/swagger-codegen
170    * Do not edit the class manually.
171    */
172   package io.swagger.api;
173
174   import io.swagger.model.Body;
175   import io.swagger.model.Document;
176   import io.swagger.model.DocumentList;
177   import io.swagger.model.Errors;
178   import io.swagger.annotations.*;
179   import org.springframework.http.ResponseEntity;
180   import org.springframework.validation.annotation.Validated;
181   import org.springframework.web.bind.annotation.PathVariable;
182   import org.springframework.web.bind.annotation.RequestBody;
183   import org.springframework.web.bind.annotation.RequestHeader;
184   import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RequestPart;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.bind.annotation.CookieValue;

import javax.validation.Valid;
import javax.validation.constraints.*;
import java.util.List;
import java.util.Map;
@javax.annotation.Generated(value =
"io.swagger.codegen.v3.generators.java.SpringCodegen", date = "2020-01-
18T21:05:26.239Z[GMT]")
@Api(value = "documents", description = "the documents API")
public interface DocumentsApi {

    @ApiOperation(value = "Method for send document to processing", nickname =
"addDocument", notes = "", response = Document.class, tags={  })
    @ApiResponses(value = {
        @ApiResponse(code = 201, message = "return document after creating",
response = Document.class),
        @ApiResponse(code = 400, message = "If client send some wrong fields to
server", response = Errors.class) })
    @RequestMapping(value = "/documents",
        produces = { "application/json" },
        consumes = { "application/json" },
        method = RequestMethod.POST)
    ResponseEntity<Document> addDocument(@ApiParam(value = ""  )  @Valid
@RequestBody Body body
);


    @ApiOperation(value = "Get document data by Id", nickname =
"documentsDocumentIdGet", notes = "", response = Document.class, tags={  })
    @ApiResponses(value = {
        @ApiResponse(code = 200, message = "return document by id if document
exist", response = Document.class),
        @ApiResponse(code = 404, message = "document not found") })
    @RequestMapping(value = "/documents/{documentId}",
        produces = { "application/json" },
        method = RequestMethod.GET)
    ResponseEntity<Document> documentsDocumentIdGet(@ApiParam(value = "Numeric ID
of the document to get",required=true) @PathVariable("documentId") Integer
documentId
);


    @ApiOperation(value = "Method for get list of sended documents", nickname =
"getDocuments", notes = "", response = DocumentList.class, tags={  })
    @ApiResponses(value = {
        @ApiResponse(code = 200, message = "List document by filter", response =
DocumentList.class) })
    @RequestMapping(value = "/documents",
        produces = { "application/json" },
        method = RequestMethod.GET)
    ResponseEntity<DocumentList> getDocuments( @Max(5) @ApiParam(value = "",
allowableValues = "") @Valid @RequestParam(value = "limit", required = false)
Integer limit
,@ApiParam(value = "") @Valid @RequestParam(value = "offset", required = false)
Integer offset
,@ApiParam(value = "", allowableValues = "pending, preprocessing, processing,
evaluation, done") @Valid @RequestParam(value = "status", required = false) String
status
```

```
);

}

// Лістинг файлу src/main/java/Clusterer.java

import clustering.ClusteringResult;
import clustering.GeneticAlgorithmProcessor;
import configuration.SparkConfiguration;
import configuration.StartAppConfiguration;
import configuration.DocsConfiguration;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.mllib.linalg.Vector;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.streaming.Durations;
import org.apache.spark.streaming.api.java.JavaDStream;
import org.apache.spark.streaming.api.java.JavaStreamingContext;
import org.joda.time.DateTime;
import preprocessing.Preprocessor;
import singletons.SparkSingleton;
import utils.ClusterInfoWriter;

import java.util.List;

/**
 * Класс, отвечающий за потоковую кластеризацию сообщений
 */
public class Clusterer {
    /**
     * Запустить потоковую кластеризацию сообщений
     */
    public static void startWork(){
        JavaSparkContext context = SparkSingleton.getContext();
        SparkSession session = SparkSingleton.getSparkSession();
        JavaStreamingContext streamingContext =
SparkSingleton.getStreamingContext();

        JavaDStream<String> lines = getDocsJavaDStream(streamingContext);
        JavaDStream<String> window =
lines.window(Durations.seconds(SparkConfiguration.windowDuration),
Durations.seconds(SparkConfiguration.butchDuration));

        window.foreachRDD(rdd -> startClustering(rdd));

        try {
            streamingContext.start();
            streamingContext.awaitTermination();
        } catch (InterruptedException e) {
            System.out.println("Error while awaitTermination");
        }
        streamingContext.close();
    }

    /**
     * Получить поток сообщений
     * @param streamingContext Контекст потока
     * @return Поток сообщений
     */
    private static JavaDStream<String> getDocsJavaDStream(JavaStreamingContext
streamingContext) {
        Configuration configuration = new ConfigurationBuilder()
                .setOAuthConsumerKey(DocsConfiguration.consumerKey)
```

```
                .setOAuthConsumerSecret(DocsConfiguration.consumerSecret)
                .setOAuthAccessToken(DocsConfiguration.accessToken)
                .setOAuthAccessTokenSecret(DocsConfiguration.accessTokenSecret)
                .build();

        Authorization DocsAuth  = AuthorizationFactory.getInstance(configuration);

        return DocsUtils.createStream(streamingContext, DocsAuth)
                .filter(status -> !status.isRetweet() &&
status.getLang().equals("en"))
                .map(status -> status.getText());
    }

    /**
     * Запустить кластеризацию набора сообщений
     * @param docs-сообщения
     */
    public static void startClustering(JavaRDD<String> docs){
        if (docs.count() == 0) return;

        String sessionPath = StartAppConfiguration.pathOut +
DateTime.now().toString("dd'.'MM'.'yyyy HH.mm.ss") + "/";

        JavaRDD<List<String>> words = Preprocessor.process(docs);
        JavaRDD<Vector> vectors = SparkSingleton.getContext().emptyRDD();

        ClusteringResult result = GeneticAlgorithmProcessor.process(vectors);
        String resultPath = String.format("%s%s clusters/", sessionPath,
result.ClustersNumber);

        ClusterInfoWriter.WriteInfo(docs, words, vectors,
result.GetClustersNumbers(vectors), result.GetCenters(), resultPath);
    }
}
// Лістинг файлу src/main/java/Main.java

import clustering.evaluation.EvaluatingMethodEnum;
import configuration.*;

public class Main {
    /**
     * Точка входа в приложение
     * @param args аргументы которые получает программа при старте
     */
    public static void main(String[] args){
        int currentArg = 0;
        SparkConfiguration.appName = "FakeNewsClusterer";
        SparkConfiguration.butchDuration = Integer.parseInt(args[currentArg++]);
        SparkConfiguration.windowDuration = Integer.parseInt(args[currentArg++]);

        StartAppConfiguration.pathOut = args[currentArg++];
        StartAppConfiguration.evaluatingMethod =
EvaluatingMethodEnum.values()[Integer.parseInt(args[currentArg++])];
        StartAppConfiguration.isLocalStart = args.length > currentArg &&
args[currentArg++].equals("-local");

        FSConfiguration.FileSystemType = StartAppConfiguration.isLocalStart ?
FileSystemTypeEnum.WindowsFS : FileSystemTypeEnum.HadoopFS;

        Clusterer.startWork();
    }
}
```

```
// Лістинг файлу src/main/java/clustering/evaluation/AbstractKMeansEvaluation.java

package clustering.evaluation;

import com.cloudera.oryx.app.kmeans.ClusterInfo;
import com.cloudera.oryx.app.kmeans.DistanceFn;
import com.cloudera.oryx.app.kmeans.EuclideanDistanceFn;
import com.google.common.base.Preconditions;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;
import org.apache.spark.mllib.linalg.Vector;
import scala.Tuple2;

import java.io.Serializable;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Базовый класс оценки KMeans
 */
abstract class AbstractKMeansEvaluation implements Serializable {

    private final DistanceFn<double[]> distanceFn;
    private final Map<Integer,ClusterInfo> clusters;

    AbstractKMeansEvaluation(List<ClusterInfo> clusterList) {
        this.distanceFn = new EuclideanDistanceFn();
        this.clusters = new HashMap<>();
        for (ClusterInfo info : clusterList) {
            clusters.put(info.getID(), info);
        }
    }

    final DistanceFn<double[]> getDistanceFn() {
        return distanceFn;
    }

    final Map<Integer,ClusterInfo> getClustersByID() {
        return clusters;
    }

    public abstract double evaluate(JavaRDD<Vector> evalData);

    JavaPairRDD<Integer,ClusterMetric> fetchClusterMetrics(JavaRDD<Vector>
evalData) {
        return evalData.mapToPair(new PairFunction<Vector,Integer,ClusterMetric>()
{
            @Override
            public Tuple2<Integer,ClusterMetric> call(Vector vector) {
                double closestDist = Double.POSITIVE_INFINITY;
                int minClusterID = Integer.MIN_VALUE;
                double[] vec = vector.toArray();
                for (ClusterInfo cluster : clusters.values()) {
                    double distance =
distanceFn.applyAsDouble(cluster.getCenter(), vec);
                    if (distance < closestDist) {
                        closestDist = distance;
                        minClusterID = cluster.getID();
                    }
                }
```

```
                Preconditions.checkState(!Double.isInfinite(closestDist) &&
!Double.isNaN(closestDist));
                return new Tuple2<>(minClusterID, new ClusterMetric(1L,
closestDist, closestDist * closestDist));
            }
        }).reduceByKey(new Function2<ClusterMetric,ClusterMetric,ClusterMetric>()
{
            @Override
            public ClusterMetric call(ClusterMetric a, ClusterMetric b) {
                return a.add(b);
            }
        });
    }

}
// Лістинг файлу src/main/java/clustering/evaluation/ClusterMetric.java

package clustering.evaluation;

import java.io.Serializable;

/**
 * Модель кластера, содержащая расстояния, необходимые для оценки
 */
public class ClusterMetric implements Serializable {

    private final long count;
    private final double sumDist;
    private final double sumSquaredDist;

    ClusterMetric(long count, double sumDist, double sumSquaredDist) {
        this.count = count;
        this.sumDist = sumDist;
        this.sumSquaredDist = sumSquaredDist;
    }

    long getCount() {
        return count;
    }

    double getSumDist() {
        return sumDist;
    }

    double getSumSquaredDist() {
        return sumSquaredDist;
    }

    double getMeanDist() {
        return sumDist / count;
    }

    ClusterMetric add(ClusterMetric other) {
        return new ClusterMetric(count + other.getCount(),
                sumDist + other.getSumDist(),
                sumSquaredDist + other.getSumSquaredDist());
    }
}
// Лістинг файлу src/main/java/clustering/evaluation/DaviesBouldinIndex.java

package clustering.evaluation;

import com.cloudera.oryx.app.kmeans.ClusterInfo;
```

```
import com.cloudera.oryx.app.kmeans.DistanceFn;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;

import java.util.List;
import java.util.Map;

/**
 * Класс для оценки кластера по принципу Davies Bouldin
 */
public class DaviesBouldinIndex extends AbstractKMeansEvaluation {

    public DaviesBouldinIndex(List<ClusterInfo> clusters) {
        super(clusters);
    }

    /**
     * Оценить кластеризацию
     * @param evalData Векторы
     * @return Оценка кластеризации
     */
    @Override
    public double evaluate(JavaRDD<Vector> evalData) {
        Map<Integer,ClusterMetric> clusterMetricsByID =
fetchClusterMetrics(evalData).collectAsMap();
        double totalDBIndex = 0.0;
        Map<Integer,ClusterInfo> clustersByID = getClustersByID();
        DistanceFn<double[]> distanceFn = getDistanceFn();
        for (Map.Entry<Integer,ClusterInfo> entryI : clustersByID.entrySet()) {
            double maxDBIndex = 0.0;
            Integer idI = entryI.getKey();
            double[] centerI = entryI.getValue().getCenter();
            double clusterScatter1 = clusterMetricsByID.get(idI).getMeanDist();
            for (Map.Entry<Integer,ClusterInfo> entryJ : clustersByID.entrySet())
{
                Integer idJ = entryJ.getKey();
                if (!idI.equals(idJ)) {
                    double[] centerJ = entryJ.getValue().getCenter();
                    double clusterScatter2 =
clusterMetricsByID.get(idJ).getMeanDist();
                    double dbIndex = (clusterScatter1 + clusterScatter2) /
distanceFn.applyAsDouble(centerI, centerJ);
                    maxDBIndex = Math.max(maxDBIndex, dbIndex);
                }
            }
            totalDBIndex += maxDBIndex;
        }

        return totalDBIndex / clustersByID.size();
    }

}
// Лістинг файлу src/main/java/clustering/evaluation/DunnIndex.java

package clustering.evaluation;

import com.cloudera.oryx.app.kmeans.ClusterInfo;
import com.cloudera.oryx.app.kmeans.DistanceFn;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;

import java.util.ArrayList;
import java.util.List;
```

```
557
558 /**
559  * Класс для оценки кластера по принципу Dunn Index
560  */
561 public class DunnIndex extends AbstractKMeansEvaluation {
562
563     public DunnIndex(List<ClusterInfo> clusters) {
564         super(clusters);
565     }
566
567     /**
568      * Оценить кластеризацию
569      * @param evalData Векторы
570      * @return Оценка кластеризации
571      */
572     @Override
573     public double evaluate(JavaRDD<Vector> evalData) {
574         double maxIntraClusterDistance = 0.0;
575         for (ClusterMetric metric :
576 fetchClusterMetrics(evalData).values().collect()) {
577             maxIntraClusterDistance = Math.max(maxIntraClusterDistance,
578 metric.getMeanDist());
579         }
580
581         double minInterClusterDistance = Double.POSITIVE_INFINITY;
582         List<ClusterInfo> clusters = new ArrayList<>(getClustersByID().values());
583         DistanceFn<double[]> distanceFn = getDistanceFn();
584         for (int i = 0; i < clusters.size(); i++) {
585             double[] centerI = clusters.get(i).getCenter();
586             for (int j = i + 1; j < clusters.size(); j++) {
587                 double[] centerJ = clusters.get(j).getCenter();
588                 minInterClusterDistance = Math.min(minInterClusterDistance,
589 distanceFn.applyAsDouble(centerI, centerJ));
590             }
591         }
592
593         return minInterClusterDistance / maxIntraClusterDistance;
594     }
595 }
596 // Лістинг файлу src/main/java/clustering/evaluation/EvaluatingMethodEnum.java

597 package clustering.evaluation;
598
599 /**
600  * Перечисление, хранящее режимы оценки класетризации
601  */
602 public enum EvaluatingMethodEnum {
603     Dunn,
604     Davies,
605     SilhouetteCoefficient
606 }
607 // Лістинг файлу
608 src/main/java/clustering/evaluation/SilhouetteCoefficientIndex.java

609 package clustering.evaluation;
610
611 import com.cloudera.oryx.app.kmeans.ClusterInfo;
612 import com.cloudera.oryx.app.kmeans.DistanceFn;
613 import com.google.common.base.Preconditions;
614 import com.google.common.collect.Lists;
615 import org.apache.spark.api.java.JavaPairRDD;
616 import org.apache.spark.api.java.JavaRDD;
617 import org.apache.spark.api.java.function.PairFunction;
```

```
618 import org.apache.spark.mllib.linalg.Vector;
619 import scala.Tuple2;
620
621 import java.util.List;
622 import java.util.Map;
623
624 /**
625  * Класс для оценки кластера по принципу Dunn Index
626  */
627 public class SilhouetteCoefficientIndex extends AbstractKMeansEvaluation {
628
629     private static final long MAX_SAMPLE_SIZE = 100000;
630
631     public SilhouetteCoefficientIndex(List<ClusterInfo> clusters) {
632         super(clusters);
633     }
634
635     /**
636      * Оценить кластеризацию
637      * @param evalData Векторы
638      * @return Оценка кластеризации
639      */
640     @Override
641     public double evaluate(JavaRDD<Vector> evalData) {
642         return
643 silhouetteCoefficient(fetchClusteredPoints(fetchSampleData(evalData)).collectAsMap
644 ());
645     }
646
647     double silhouetteCoefficient(Map<Integer,Iterable<double[]>>
648 clusterPointsByID) {
649         double totalSilhouetteCoefficient = 0.0;
650         long sampleCount = 0L;
651
652         for (Map.Entry<Integer, Iterable<double[]>> entry :
653 clusterPointsByID.entrySet()) {
654             List<double[]> clusteredPoints = iterableToList(entry.getValue());
655             long clusterSize = clusteredPoints.size();
656             sampleCount += clusterSize;
657             if (clusterSize > 1) {
658                 for (double[] point : clusteredPoints) {
659                     double pointIntraClusterDissimilarity =
660 clusterDissimilarityForPoint(point, clusteredPoints, true);
661                     double pointInterClusterDissimilarity =
662                             minInterClusterDissimilarityForPoint(entry.getKey(),
663 point, clusterPointsByID);
664                     totalSilhouetteCoefficient += silhouetteCoefficient(
665                             pointIntraClusterDissimilarity,
666 pointInterClusterDissimilarity);
667                 }
668             }
669
670         }
671
672         return sampleCount == 0 ? 0.0 : totalSilhouetteCoefficient / sampleCount;
673     }
674
675     static JavaRDD<Vector> fetchSampleData(JavaRDD<Vector> evalData) {
676         JavaRDD<Vector> data = evalData;
677         long count = evalData.count();
678         if (count > MAX_SAMPLE_SIZE) {
679             data = evalData.sample(false, (double) MAX_SAMPLE_SIZE / count);
680         }
```

```
681             return data;
682         }
683
684         private JavaPairRDD<Integer, Iterable<double[]>>
685     fetchClusteredPoints(JavaRDD<Vector> evalData) {
686             return evalData.mapToPair(new PairFunction<Vector, Integer, double[]>() {
687                 @Override
688                 public Tuple2<Integer, double[]> call(Vector vector) {
689                     double closestDist = Double.POSITIVE_INFINITY;
690                     int minClusterID = Integer.MIN_VALUE;
691                     double[] vec = vector.toArray();
692                     DistanceFn<double[]> distanceFn = getDistanceFn();
693                     Map<Integer,ClusterInfo> clusters = getClustersByID();
694                     for (ClusterInfo cluster : clusters.values()) {
695                         double distance =
696     distanceFn.applyAsDouble(cluster.getCenter(), vec);
697                         if (distance < closestDist) {
698                             closestDist = distance;
699                             minClusterID = cluster.getID();
700                         }
701                     }
702                     Preconditions.checkState(!Double.isInfinite(closestDist) &&
703     !Double.isNaN(closestDist));
704                     return new Tuple2<>(minClusterID, vec);
705                 }
706             }).groupByKey();
707         }
708
709         private double clusterDissimilarityForPoint(double[] point,
710                                                     List<double[]> clusterPoints,
711                                                     boolean ownCluster) {
712             DistanceFn<double[]> distanceFn = getDistanceFn();
713             double totalDissimilarity = 0.0;
714             for (double[] clusterPoint : clusterPoints) {
715                 totalDissimilarity += distanceFn.applyAsDouble(point, clusterPoint);
716             }
717
718             if (ownCluster) {
719                 return totalDissimilarity / (clusterPoints.size() - 1);
720             } else {
721                 return totalDissimilarity / clusterPoints.size();
722             }
723         }
724
725         private double minInterClusterDissimilarityForPoint(
726                 int otherClusterID,
727                 double[] point,
728                 Map<Integer, Iterable<double[]>> clusteredPointsMap) {
729             double minInterClusterDissimilarity = Double.POSITIVE_INFINITY;
730             for (Map.Entry<Integer, Iterable<double[]>> entry :
731     clusteredPointsMap.entrySet()) {
732                 if (!entry.getKey().equals(otherClusterID)) {
733                     List<double[]> clusteredPoints = iterableToList(entry.getValue());
734                     minInterClusterDissimilarity =
735     Math.min(minInterClusterDissimilarity,
736                             clusterDissimilarityForPoint(point, clusteredPoints,
737     false));
738                 }
739             }
740             return minInterClusterDissimilarity;
741         }
742
743         static double silhouetteCoefficient(double ai, double bi) {
```

```
            if (ai < bi) {
                return 1.0 - (ai / bi);
            }
            if (ai > bi) {
                return (bi / ai) - 1.0;
            }
            return 0.0;
        }

        @SuppressWarnings("unchecked")
        private static <T> List<T> iterableToList(Iterable<T> it) {
            return it instanceof List ? (List<T>) it : Lists.newArrayList(it);
        }
    }
```

```
// Лістинг файлу src/main/java/clustering/evaluators/ClusterEvaluator.java
```

```
package clustering.evaluators;

import com.cloudera.oryx.app.kmeans.ClusterInfo;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;

import java.util.ArrayList;
import java.util.List;

/**
 * Оценщик кластерищации
 */
public abstract class ClusterEvaluator {
    /**
     * Получение данных о кластеризации, необходимых для оценки
     * @param data Векторыне данные
     * @param centers Центры кластеров
     * @return Данные о кластеризации
     */
    List<ClusterInfo> GetClustersInfo (JavaRDD<Vector> data, Vector[] centers){
        long dataCount = data.count();

        List<ClusterInfo> clustersInfo = new ArrayList<>();

        for (int i = 0; i < centers.length; i++) {
            clustersInfo.add(new ClusterInfo(i, centers[i].toArray(), dataCount));
        }
        return clustersInfo;
    }

    /**
     * Оценить кластеризацию
     * @param data Векторные данные
     * @param centers Центры кластеров
     * @return Результат оценки кластеризации
     */
    public abstract double Evaluate(JavaRDD<Vector> data, Vector[] centers);

    /**
     * Сравнить оценку кластеризации
     * @param nextEval Новая оценка
     * @param oldEval Старая оценка
     * @return Результат сравнения
     */
    public abstract boolean IsBetter(double nextEval, double oldEval);
```

```
}

// Лістинг файлу src/main/java/clustering/evaluators/ClusterEvaluatorFactory.java

package clustering.evaluators;

import clustering.evaluation.EvaluatingMethodEnum;

/**
 * Фабрика для создания оценки кластеризации
 */
public class ClusterEvaluatorFactory {
    /**
     * Получить экземпляр ClusterEvaluator
     * @param evaluatingMethodEnum Метод сравнения
     * @return Экземпляр ClusterEvaluator
     */
    public static ClusterEvaluator get(EvaluatingMethodEnum evaluatingMethodEnum){
        switch (evaluatingMethodEnum)
        {
            case Davies:
                return new DaviesClusterEvaluator();
            case Dunn:
                return new DunnClusterEvaluator();
            case SilhouetteCoefficient:
                return new SilhouetteCoefficientEvaluator();
            default:
                return null;
        }
    }
}

// Лістинг файлу src/main/java/clustering/evaluators/DaviesClusterEvaluator.java

package clustering.evaluators;

import clustering.evaluation.DaviesBouldinIndex;
import com.cloudera.oryx.app.kmeans.ClusterInfo;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;

import java.util.List;

/**
 * Оценщик кластерицации методом Davies Bouldin
 */
public class DaviesClusterEvaluator extends ClusterEvaluator {
    /**
     * Оценить кластеризацию по методу Davies Bouldin
     * @param data Векторные данные
     * @param centers Центры кластеров
     * @return Результат оценки кластеризации
     */
    @Override
    public double Evaluate(JavaRDD<Vector> data, Vector[] centers) {
        List<ClusterInfo> clustersInfo = GetClustersInfo(data, centers);

        DaviesBouldinIndex daviesBouldinIndex = new
DaviesBouldinIndex(clustersInfo);

        return daviesBouldinIndex.evaluate(data);
    }
```

```
865
866         /**
867          * Сравнить оценку кластеризации
868          * @param nextEval Новая оценка
869          * @param oldEval Старая оценка
870          * @return Результат сравнения
871          */
872         @Override
873         public boolean IsBetter(double nextEval, double oldEval) {
874             return nextEval < oldEval;
875         }
876     }
877
878     // Лістинг файлу src/main/java/clustering/evaluators/DunnClusterEvaluator.java
879     package clustering.evaluators;
880
881     import clustering.evaluation.DunnIndex;
882     import com.cloudera.oryx.app.kmeans.ClusterInfo;
883     import org.apache.spark.api.java.JavaRDD;
884     import org.apache.spark.mllib.linalg.Vector;
885
886     import java.util.List;
887
888     /**
889      * Оценщик кластерицации методом Dunn
890      */
891     public class DunnClusterEvaluator extends ClusterEvaluator {
892         /**
893          * Оценить кластеризацию по методу Dunn
894          * @param data Векторные данные
895          * @param centers Центры кластеров
896          * @return Результат оценки кластеризации
897          */
898         @Override
899         public double Evaluate(JavaRDD<Vector> data, Vector[] centers) {
900             List<ClusterInfo> clustersInfo = GetClustersInfo(data, centers);
901
902             DunnIndex dunnIndex = new DunnIndex(clustersInfo);
903
904             return dunnIndex.evaluate(data);
905         }
906
907         /**
908          * Сравнить оценку кластеризации
909          * @param nextEval Новая оценка
910          * @param oldEval Старая оценка
911          * @return Результат сравнения
912          */
913         @Override
914         public boolean IsBetter(double nextEval, double oldEval) {
915             return nextEval > oldEval;
916         }
917     }
918
919     // Лістинг файлу
920     src/main/java/clustering/evaluators/SilhouetteCoefficientEvaluator.java
921     package clustering.evaluators;
922
923     import clustering.evaluation.SilhouetteCoefficientIndex;
```

```
import com.cloudera.oryx.app.kmeans.ClusterInfo;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;

import java.util.List;

public class SilhouetteCoefficientEvaluator extends ClusterEvaluator {
    /**
     * Оценить кластеризацию по методу SilhouetteCoefficient
     * @param data Векторные данные
     * @param centers Центры кластеров
     * @return Результат оценки кластеризации
     */
    @Override
    public double Evaluate(JavaRDD<Vector> data, Vector[] centers) {
        List<ClusterInfo> clustersInfo = GetClustersInfo(data, centers);

        SilhouetteCoefficientIndex silhouetteCoefficientIndex = new
SilhouetteCoefficientIndex(clustersInfo);

        return silhouetteCoefficientIndex.evaluate(data);
    }

    /**
     * Сравнить оценку кластеризации
     * @param nextEval Новая оценка
     * @param oldEval Старая оценка
     * @return Результат сравнения
     */
    @Override
    public boolean IsBetter(double nextEval, double oldEval) {
        return nextEval > oldEval;
    }
}
```

```
// Лістинг файлу src/main/java/clustering/ClusteringResult.java

package clustering;

import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.clustering.KMeansModel;
import org.apache.spark.mllib.linalg.Vector;

/**
 * Модель данных о результате кластеризации
 */
public class ClusteringResult {
    public int ClustersNumber;
    public double Evaluation;

    private KMeansModel _model;

    public JavaRDD<Integer> GetClustersNumbers(JavaRDD<Vector> javaRDD){
        return _model.predict(javaRDD);
    }

    public Vector[] GetCenters(){
        return _model.clusterCenters();
    }

    public ClusteringResult(KMeansModel model, int clustersNumber, double
evaluation){
```

```
985             _model = model;
986             ClustersNumber = clustersNumber;
987             Evaluation = evaluation;
988         }
989     }
990
991     // Лістинг файлу src/main/java/clustering/GeneticAlgorithmProcessor.java
992     package clustering;
993
994     import org.apache.spark.api.java.JavaRDD;
995     import org.apache.spark.mllib.linalg.Vector;
996     import org.jenetics.*;
997     import org.jenetics.engine.Engine;
998     import org.jenetics.engine.EvolutionResult;
999
1000    import java.util.*;
1001
1002    /**
1003     * Кластеризатор с генетическим алгоритмом
1004     */
1005    public class GeneticAlgorithmProcessor {
1006
1007        /**
1008         Провести кластеризацию с гибким количеством кластеров при помощи ГА
1009         * @param data Вектора для кластеризации
1010         * @return Результат кластеризации
1011         */
1012        public static ClusteringResult process(JavaRDD<Vector> data) {
1013            int     min = 2,
1014                    max = 20,
1015                    count = 5;
1016            boolean toMax = true;
1017            int maxLength = binar(max).size();
1018
1019            Map<Integer, ClusteringResult> dict = new HashMap<>();
1020            Map<Integer, Integer> dictEval = new HashMap<>();
1021
1022            IntegerChromosome integerChromosome = IntegerChromosome.of(min, max,
1023    count);
1024
1025            List<BitChromosome> list = new ArrayList<>();
1026            integerChromosome.forEach(g ->
1027    list.add(BitChromosome.of(BitSet.valueOf(new long[] { g.intValue() }),
1028    maxLength)));
1029
1030            Engine<BitGene, Integer> engine = Engine
1031                    .builder(g -> {
1032                        int clustersNumber =
1033    (int)g.getChromosome().as(BitChromosome.class).longValue();
1034
1035                        if (clustersNumber < 0) clustersNumber *= -1;
1036                        if (clustersNumber < min) clustersNumber = min;
1037                        if (clustersNumber > max) clustersNumber = max;
1038                        if (dictEval.containsKey(clustersNumber)) {
1039                            return dictEval.get(clustersNumber);
1040                        }
1041                        ClusteringResult result = KMeansProcessor.process(data,
1042    clustersNumber);
1043                        int eval = (int)(result.Evaluation * 10000);
1044                        dict.put(clustersNumber, result);
1045                        dictEval.put(clustersNumber, eval);
```

```
                    return eval;
                }, Genotype.of(list))
                .offspringSelector(new RouletteWheelSelector<>())
                .survivorsSelector(new TournamentSelector<>())
                .populationSize(5)
                .optimize(toMax ? Optimize.MAXIMUM : Optimize.MINIMUM)
                .alterers(new Mutator<>(0.03), new SinglePointCrossover<>(0.05))
                .build();

        Genotype<BitGene> result = engine.stream()
                .limit(10)
                .collect(EvolutionResult.toBestGenotype());

        int clustersNumberResult =
(int)result.getChromosome().as(BitChromosome.class).longValue();

        if (clustersNumberResult < min) clustersNumberResult = min;
        if (clustersNumberResult > max) clustersNumberResult = max;

        ClusteringResult clusteringResult = dict.get(clustersNumberResult);

        return clusteringResult;
    }

    /**
     * Переводит число в двоичный набор
     * @param value число
     * @return двоичный набор
     */
    public static List<Byte> binar(int value) {
        List<Byte> list = new ArrayList<>();
        int i = value, b;
        while(i != 0) {
            b = i % 2;
            list.add((byte)b);
            i = i / 2;
        }
        return list;
    }
}
```

// Лістинг файлу src/main/java/clustering/KMeansProcessor.java

```
package clustering;

import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.clustering.KMeans;
import org.apache.spark.mllib.clustering.KMeansModel;
import org.apache.spark.mllib.linalg.Vector;
import singletons.ClusterEvaluatorSingleton;

/**
 * Кластеризатор с алгоритмом KMeans
 */
public class KMeansProcessor {
    static KMeans kmeans = new KMeans().setInitializationMode("k-means||");

    /**
     * Провести кластеризацию с указанным количеством кластеров
     * @param data Вектора для кластеризации
     * @param numClusters Количество кластеров
```

```
     * @return Результат кластеризации
     */
    public static ClusteringResult process(JavaRDD<Vector> data, int numClusters){
        int numIterations = 20;

        KMeansModel clusters = KMeans.train(data.rdd(), numClusters,
numIterations);
        Vector[] centers = clusters.clusterCenters();
        double evaluation = ClusterEvaluatorSingleton.getInstance().Evaluate(data,
centers);

        return new ClusteringResult(clusters, numClusters, evaluation);
    }
}
```

Лістинг файлу src/main/java/clustering/DBSCANProcessor.java

```
package clustering;

import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.clustering.DBSCAN;
import org.apache.spark.mllib.clustering.DBSCANModel;
import org.apache.spark.mllib.linalg.Vector;
import singletons.ClusterEvaluatorSingleton;

/**
 * Кластеризатор с алгоритмом DBSCAN
 */
public class DBSCANProcessor {
    static DBSCAN dbscan = new DBSCAN ().setInitializationMode("DBSCAN||");

    /**
     * Провести кластеризацию с указанным количеством кластеров
     * @param data Вектора для кластеризации
     * @param numClusters Количество кластеров
     * @return Результат кластеризации
     */
    public static ClusteringResult process(JavaRDD<Vector> data, int numClusters){
        int numIterations = 20;

        DBSCANModel clusters = DBSCAN.train(data.rdd(), numClusters,
numIterations);
        Vector[] centers = clusters.clusterCenters();
        double evaluation = ClusterEvaluatorSingleton.getInstance().Evaluate(data,
centers);

        return new ClusteringResult(clusters, numClusters, evaluation);
    }
}
```

```
// Лістинг файлу src/main/java/configuration/FSConfiguration.java

package configuration;

/**
 * Настройки файловой системы
 */
public class FSConfiguration {
    public static FileSystemTypeEnum FileSystemType =
FileSystemTypeEnum.WindowsFS;
}
```

```
// Лістинг файлу src/main/java/configuration/FileSystemTypeEnum.java

package configuration;

/**
 * Перечисление, хранящее файловые системы
 */
public enum  FileSystemTypeEnum {
    WindowsFS,
    HadoopFS
}


// Лістинг файлу src/main/java/configuration/SparkConfiguration.java

package configuration;

public class SparkConfiguration {
    public static String appName = "Default";
    public static Integer butchDuration = 1000;
    public static Integer windowDuration = 1000;
}


// Лістинг файлу src/main/java/configuration/StartAppConfiguration.java

package configuration;

import clustering.evaluation.EvaluatingMethodEnum;

/**
 * Настройки конфигурации запуска приложения
 */
public class StartAppConfiguration {
    public static String pathOut;
    public static boolean isLocalStart;
    public static EvaluatingMethodEnum evaluatingMethod;
}


//Лістинг файлу src/main/java/configuration/DocsConfiguration.java

package configuration;

/**
 * Настройки конфигурации подключения к Docs Api
 */
public class DocsConfiguration {
    public static String accessToken;
    public static String accessTokenSecret;
    public static String consumerKey;
    public static String consumerSecret;
}


// Лістинг файлу src/main/java/visualizing/Pca.java

package visualizing;

import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Matrix;
import org.apache.spark.mllib.linalg.Vector;
import org.apache.spark.mllib.linalg.distributed.RowMatrix;
```

```
import singletons.SparkSingleton;

import java.util.Arrays;
import java.util.List;

/**
 * Реализует алгоритм PCA
 */
public class Pca {
    /**
     * Привести вектора к двумерным
     * @param vectors Вектора
     * @return Двумерные вектора
     */
    public static JavaRDD<Vector> execute(JavaRDD<Vector> vectors){
        RowMatrix mat = new RowMatrix(vectors.rdd());

        Matrix pc = mat.computePrincipalComponents(2);
        RowMatrix projected = mat.multiply(pc);

        return projected.rows().toJavaRDD();
    }

    /**
     * Привести вектора к двумерным
     * @param vectors Вектора
     * @return Двумерные вектора
     */
    public static List<Vector> execute(List<Vector> vectors){
        return
execute(SparkSingleton.getContext().parallelize(vectors)).collect();
    }

    /**
     * Привести вектора к двумерным
     * @param vectors Вектора
     * @return Двумерные вектора
     */
    public static Vector[] execute(Vector[] vectors){
        List<Vector> list = execute(Arrays.asList(vectors));
        return list.toArray(new Vector[list.size()]);
    }
}
```

```
// Лістинг файлу src/main/java/visualizing/Visualizer.java

package visualizing;

import de.erichseifert.gral.data.DataTable;
import de.erichseifert.gral.graphics.DrawingContext;
import de.erichseifert.gral.io.plots.DrawableWriter;
import de.erichseifert.gral.io.plots.DrawableWriterFactory;
import de.erichseifert.gral.plots.Plot;
import de.erichseifert.gral.plots.XYPlot;
import de.erichseifert.gral.plots.colors.RandomColors;
import de.erichseifert.gral.plots.points.DefaultPointRenderer2D;
import de.erichseifert.gral.plots.points.PointRenderer;
import de.erichseifert.gral.util.MathUtils;
import models.SimpleClusterData;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;
import singletons.FileManagerSingleton;
```

```
import utils.ClusterDataMapper;

import java.awt.*;
import java.awt.geom.Ellipse2D;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import java.util.Random;

/**
 * Визуализация результатов кластеризации
 */
public class Visualizer {
    static RandomColors randomColors = new RandomColors();

    /**
     * Диаграмма кластеров
     * @param path Путь к файлу
     * @param vectors Вектора
     * @param clustersNumbers Номер кластера каждого твит-сообщения
     * @param clustersCentres Центры кластеров
     */
    public static void plot(String path, JavaRDD<Vector> vectors, JavaRDD<Integer>
clustersNumbers, Vector[] clustersCentres){
        Vector[] centres = Pca.execute(clustersCentres);
        JavaRDD<Vector> pca = Pca.execute(vectors);

        JavaRDD<SimpleClusterData> model =
ClusterDataMapper.getSimpleClusterData(pca, clustersNumbers, centres);

        JavaRDD<DataTable> rdd1 = model.map(m -> {
            DataTable table = new DataTable(Double.class, Double.class);
            m.items.forEach(i -> {
                double[] arr = i.toArray();
                table.add(arr[0], arr[1]);
            });
            return table;
        });

        List<DataTable> list1 = rdd1.collect();

        JavaRDD<DataTable> rdd2 = model.map(m -> {
            DataTable table = new DataTable(Double.class, Double.class);
            double[] arr = m.center.toArray();
            table.add(arr[0], arr[1]);
            return table;
        });

        List<DataTable> list2 = rdd2.collect();

        List<DataTable> list = new LinkedList<>();
        list.addAll(list1);
        list.addAll(list2);

        DataTable[] arr = list.toArray(new DataTable[list.size()]);

        XYPlot plot = new XYPlot(arr);

        Color[] colors = getRandomColors(clustersCentres.length);

        for (int i = 0; i < list1.size(); i++) {
```

```
1344            Color color = colors[i];
1345            PointRenderer pointRenderer = new DefaultPointRenderer2D();
1346            pointRenderer.setColor(color);
1347            plot.setPointRenderers(list1.get(i), pointRenderer);
1348        }
1349
1350        for (int i = 0; i < list2.size(); i++) {
1351            Color color = colors[i];
1352            PointRenderer pointRenderer = new DefaultPointRenderer2D();
1353            pointRenderer.setColor(color);
1354            pointRenderer.setShape(new Ellipse2D.Double(-10.0, -10.0, 20.0,
1355  20.0));
1356            plot.setPointRenderers(list2.get(i), pointRenderer);
1357        }
1358
1359        try {
1360            FileManagerSingleton.getInstance().write(path, getJpg(plot));
1361        } catch (IOException e) {
1362            e.printStackTrace();
1363        }
1364    }
1365
1366    /**
1367     * Возвращает указанное количество случайных цветов
1368     * @param count Количество цветов
1369     * @return Случайные цвета
1370     */
1371    private static Color[] getRandomColors(int count) {
1372        Color[] res = new Color[count];
1373        for (int i = 0; i < res.length; i++) {
1374            res[i] = getRandomColor();
1375        }
1376        return res;
1377    }
1378
1379    /**
1380     * Возвращает случаный цвет
1381     * @return Случайный цвет
1382     */
1383    private static Color getRandomColor() {
1384        Random random = new Random();
1385        float[] colorVariance =  randomColors.getColorVariance();
1386        float hue = colorVariance[0] + colorVariance[1]*random.nextFloat();
1387        float saturation = colorVariance[2] + colorVariance[3]*random.nextFloat();
1388        float brightness = colorVariance[4] + colorVariance[5]*random.nextFloat();
1389        return Color.getHSBColor(
1390            hue,
1391            MathUtils.limit(saturation, 0f, 1f),
1392            MathUtils.limit(brightness, 0f, 1f)
1393        );
1394    }
1395
1396    /**
1397     * Формирует преобразует диаграмму в изображение
1398     * @param plot Диаграмма
1399     * @return Байтовый набор изображения
1400     * @throws IOException
1401     */
1402    private static byte[] getJpg(Plot plot) throws IOException {
1403        BufferedImage bImage = new BufferedImage(800, 600,
1404  BufferedImage.TYPE_INT_ARGB);
1405        Graphics2D g2d = (Graphics2D) bImage.getGraphics();
1406        DrawingContext context = new DrawingContext(g2d);
```

```
1407            plot.draw(context);
1408            ByteArrayOutputStream baos = new ByteArrayOutputStream();
1409            DrawableWriter wr = DrawableWriterFactory.getInstance().get("image/jpeg");
1410            wr.write(plot, baos, 800, 600);
1411            baos.flush();
1412            byte[] bytes = baos.toByteArray();
1413            baos.close();
1414            return bytes;
1415        }
1416    }
1417
1418    // Лістинг файлу src/main/java/preprocessing/Preprocessor.java
1419    package preprocessing;
1420
1421    import org.apache.spark.api.java.JavaRDD;
1422
1423    import java.io.Serializable;
1424    import java.util.List;
1425
1426    /**
1427     * Препроцессинг данных для кластеризации
1428     */
1429    public class Preprocessor implements Serializable {
1430        /**
1431         * Очистить сообщения от гиперссылок и символов других языков
1432         * @param docs сообщения
1433         * @return Очищенные сообщения
1434         */
1435        private static JavaRDD<String> clearDocs(JavaRDD<String> docs) {
1436            return docs.map(x -> x
1437                .replaceAll("(https?:\\/\\/)?([\\da-z\\.-]+)\\.([a-z\\.]{2,6})([\\/\\w
1438    \\.-]*)*\\/?", "")
1439                .replaceAll("[^A-z ]", "").toLowerCase()
1440            );
1441        }
1442
1443        /**
1444         * Токинизация и лемматизация сообщений
1445         * @param docs сообщения
1446         * @return сообщения, разбитые на слова
1447         */
1448        private static JavaRDD<List<String>> tokenizeAndLemmatize(JavaRDD<String>
1449    docs) {
1450            return docs.map(x -> TokenizerAndLemmatizer.execute(x));
1451        }
1452
1453        /**
1454         * Удалить стоп-слова
1455         * @param words Слова
1456         * @return Очищенные слова
1457         */
1458        private static JavaRDD<List<String>> removeStopWords(JavaRDD<List<String>>
1459    words) {
1460            return StopWordsRemover.execute(words).filter(row -> !row.isEmpty());
1461        }
1462
1463        /**
1464         * Выполнить препроцессинг для сообщений
1465         * @param docs сообщения
1466         * @return Слова
1467         */
```

```
    public static JavaRDD<List<String>> process (JavaRDD<String> docs) {
        JavaRDD<String> clear = clearDocs(docs);
        JavaRDD<List<String>> split = tokenizeAndLemmatize(clear);
        JavaRDD<List<String>> filtered = removeStopWords(split);
        return filtered;
    }
}
```

```
// Лістинг файлу src/main/java/preprocessing/StopWordsRemover.java
package preprocessing;

import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.types.DataTypes;
import org.apache.spark.sql.types.Metadata;
import org.apache.spark.sql.types.StructField;
import org.apache.spark.sql.types.StructType;
import singletons.SparkSingleton;

import java.util.List;

/**
 * Обеспечивает удаление стоп-слов
 */
public class StopWordsRemover {
    private static StructType schema = new StructType(new StructField[]{
        new StructField("raw", DataTypes.createArrayType(DataTypes.StringType),
false, Metadata.empty())
    });

    static org.apache.spark.ml.feature.StopWordsRemover stopWordsRemover = new
org.apache.spark.ml.feature.StopWordsRemover()
            .setInputCol("raw")
            .setOutputCol("filtered");

    /**
     * Выполнить удаление стоп-слов
     * @param words Слова
     * @return Очищенные слова
     */
    public static JavaRDD<List<String>> execute(JavaRDD<List<String>> words){
        SparkSession session = SparkSingleton.getSparkSession();

        JavaRDD<Row> rows = words.map(x -> RowFactory.create(x));
        Dataset<Row> dataset =  session.createDataFrame(rows.collect(), schema);

        return
stopWordsRemover.transform(dataset).select("filtered").toJavaRDD().map(row ->
row.getList(0));
    }
}
```

```
// Лістинг файлу src/main/java/preprocessing/TokenizerAndLemmatizer.java
package preprocessing;

import edu.stanford.nlp.ling.CoreAnnotations.LemmaAnnotation;
```

```
import edu.stanford.nlp.ling.CoreAnnotations.SentencesAnnotation;
import edu.stanford.nlp.ling.CoreAnnotations.TokensAnnotation;
import edu.stanford.nlp.ling.CoreLabel;
import edu.stanford.nlp.pipeline.Annotation;
import edu.stanford.nlp.pipeline.StanfordCoreNLP;
import edu.stanford.nlp.util.CoreMap;
import singletons.StanfordCoreNLPSingleton;

import java.util.LinkedList;
import java.util.List;

/**
 * Обеспечивает токинизацию и лемматизацию
 */
public class TokenizerAndLemmatizer {
    /**
     * Выполнить токинизацию и лемматизацию
     * @param docs сообщение
     * @return Слова
     */
    public static List<String> execute(String docs)
    {
        List<String> lemmas = new LinkedList<>();
        StanfordCoreNLP pipeline = StanfordCoreNLPSingleton.getInstance();
        Annotation annotation = new Annotation(docs);
        pipeline.annotate(annotation);
        List<CoreMap> sentences = annotation.get(SentencesAnnotation.class);

        for (CoreMap sentence : sentences) {
            for (CoreLabel token : sentence.get(TokensAnnotation.class)) {
                lemmas.add(token.get(LemmaAnnotation.class));
            }
        }

        return lemmas;
    }
}
// Лістинг файлу src/main/java/singletons/ClusterEvaluatorSingleton.java

package singletons;

import clustering.evaluators.ClusterEvaluator;
import clustering.evaluators.ClusterEvaluatorFactory;
import configuration.StartAppConfiguration;

/**
 * Обеспечивает один экземпляр ClusterEvaluator для всего проекта
 */
public class ClusterEvaluatorSingleton
{
    static ClusterEvaluator evaluator;

    /**
     * Получить экземпляр ClusterEvaluator
     * @return Экземпляр ClusterEvaluator
     */
    public static ClusterEvaluator getInstance(){
        if(evaluator == null) {
            evaluator =
ClusterEvaluatorFactory.get(StartAppConfiguration.evaluatingMethod);
        }
        return evaluator;
    }
```

```
}

// Лістинг файлу src/main/java/singletons/FileManagerSingleton.java

package singletons;

import configuration.FSConfiguration;
import io.FileManager;
import io.FileManagerFactory;

/**
 * Обеспечивает один экземпляр FileManager для всего проекта
 */
public class FileManagerSingleton {
    private static FileManager fileManager;

    /**
     * Получить экземпляр FileManager
     * @return Экземпляр FileManager
     */
    public static FileManager getInstance() {
        if(fileManager == null) {
            fileManager = FileManagerFactory.get(FSConfiguration.FileSystemType);
        }
        return fileManager;
    }
}

// Лістинг файлу src/main/java/singletons/GsonSingleton.java

package singletons;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

/**
 * Обеспечивает один экземпляр Gson для всего проекта
 */
public class GsonSingleton {
    public static Gson instance = new GsonBuilder().create();
}

// Лістинг файлу src/main/java/singletons/SparkSingleton.java

package singletons;

import configuration.SparkConfiguration;
import configuration.StartAppConfiguration;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.streaming.Durations;
import org.apache.spark.streaming.api.java.JavaStreamingContext;

/**
 * Обеспечивает однин экземпляр каждой нстройки Spark для всего проекта
 */
public class SparkSingleton {
    private static JavaSparkContext sparkContext;

```

```
1646        /**
1647         * Получить экземпляр JavaSparkContext
1648         * @return Экземпляр JavaSparkContext
1649         */
1650        public static synchronized JavaSparkContext getContext() {
1651            if(sparkContext == null) {
1652                try {
1653                    SparkConf conf = new SparkConf()
1654                            .setAppName("Spark user-activity")
1655                            .set("spark.streaming.blockInterval",
1656    SparkConfiguration.butchDuration + "ms");
1657                    if (StartAppConfiguration.isLocalStart)
1658                        conf = conf
1659                                .setMaster("local[4]")
1660                                .set("spark.driver.host", "localhost");
1661                    sparkContext = new JavaSparkContext(conf);
1662                } catch (Exception e) {
1663                    throw new RuntimeException("При создании JavaSparkContext
1664    произошла ошибка:\n" + e.toString());
1665                }
1666            }
1667            return sparkContext;
1668        }
1669
1670        private static JavaStreamingContext sparkStreamingContext;
1671
1672        /**
1673         * Получить экземпляр JavaStreamingContext
1674         * @return Экземпляр JavaStreamingContext
1675         */
1676        public static synchronized JavaStreamingContext getStreamingContext(){
1677            if(sparkStreamingContext == null){
1678                try{
1679                    sparkStreamingContext = new JavaStreamingContext(getContext(),
1680    Durations.milliseconds(SparkConfiguration.butchDuration));
1681                }catch (Exception e){
1682                    throw new RuntimeException("При создании JavaStreamingContext
1683    произошла ошибка:\n"+e.toString());
1684                }
1685            }
1686            return sparkStreamingContext;
1687        }
1688
1689        private static SparkSession sparkSession;
1690
1691        /**
1692         * Получить экземпляр SparkSession
1693         * @return Экземпляр SparkSession
1694         */
1695        public static synchronized SparkSession getSparkSession(){
1696            if(sparkSession == null){
1697                try{
1698                    sparkSession = SparkSession.builder()
1699                            .config("spark.sql.warehouse.dir", "spark-warehouse")
1700                            .getOrCreate();
1701                }catch (Exception e){
1702                    throw new RuntimeException("При создании SparkSession произошла
1703    ошибка\n"+e.toString());
1704                }
1705            }
1706            return sparkSession;
1707        }
1708    }
```

```
// Лістинг файлу src/main/java/singletons/StanfordCoreNLPSingleton.java

package singletons;

import edu.stanford.nlp.pipeline.StanfordCoreNLP;

import java.util.Properties;

/**
 * Обеспечивает один экземпляр StanfordCoreNLP для всего проекта
 */
public class StanfordCoreNLPSingleton {
    private static StanfordCoreNLP pipeline;
    private static Properties props;

    /**
     * Получить экземпляр StanfordCoreNLP
     * @return Экземпляр StanfordCoreNLP
     */
    public static StanfordCoreNLP getInstance() {
        if(props == null){
            try {
                props = new Properties();
                props.put("annotators", "tokenize, ssplit, pos, lemma");
            } catch (Exception e){
                throw new RuntimeException("При создании Properties для pipline
произошла ошибка\n"+e.toString());
            }
        }
        if(pipeline == null){
            try {
                pipeline = new StanfordCoreNLP(props);
            } catch (Exception e){
                throw new RuntimeException("При создании StanfordCoreNLP произошла
ошибка\n"+e.toString());
            }
        }
        return pipeline;
    }
}
```

```
// Лістинг файлу src/main/java/utils/ClusterDataMapper.java

package utils;

import models.SimpleClusterData;
import models.DocsClusterData;
import models.DocsData;
import models.DocsVectorData;
import models.rw.DocsClusterDataRW;
import models.rw.VectorDataRW;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;
import scala.Tuple2;

import java.util.LinkedList;
import java.util.List;

/**
```

```
 * Формирует данные в модели
 */
public class ClusterDataMapper {
    /**
     * Получить SimpleClusterData
     * @param vectors Вектора
     * @param clustersNumbers Номер кластера каждого сообщения
     * @param clustersCenters Центры кластеров
     * @return Сформированный SimpleClusterData
     */
    public static JavaRDD<SimpleClusterData> getSimpleClusterData(JavaRDD<Vector>
vectors, JavaRDD<Integer> clustersNumbers, Vector[] clustersCenters) {
        return zip(clustersNumbers, vectors).groupByKey().sortByKey().map(tuple ->
new SimpleClusterData(tuple._1(), clustersCenters[tuple._1], tuple._2));
    }

    /**
     * Получить DocsData
     * @param docs сообщения
     * @param words Слова сообщений
     * @return Сформированный DocsData
     */
    public static JavaRDD<DocsData> getDocsData(JavaRDD<String> docs,
JavaRDD<List<String>> words){
        return zip(docs, words).map(tuple -> new DocsData(tuple._1, tuple._2));
    }

    /**
     * Получить DocsClusterData
     * @param docsData Данные сообщений
     * @param vectors Вектора
     * @param clustersCenters Центры кластеров
     * @param clustersNumbers Номер кластера каждого сообщения
     * @return Сформированный DocsClusterData
     */
    public static JavaRDD<DocsClusterData> getDocsClusterData(JavaRDD<DocsData>
docsData, JavaRDD<Vector> vectors, Vector[] clustersCenters, JavaRDD<Integer>
clustersNumbers)
    {
        JavaRDD<DocsVectorData> docsVectorData = getDocsVectorData(docsData,
vectors);
        return zip(clustersNumbers,
docsVectorData).groupByKey().sortByKey().map(tuple -> new
DocsClusterData(tuple._1, clustersCenters[tuple._1], tuple._2));
    }

    /**
     * Возвращает DocsClusterDataRW
     * @param docsData Данные сообщений
     * @param vectors Вектора
     * @param clustersCenters Центры кластеров
     * @param clustersNumbers Номер кластера каждого сообщения
     * @return Сформированный DocsClusterDataRW
     */
    public static JavaRDD<DocsClusterDataRW>
getDocsClusterDataRW(JavaRDD<DocsData> docsData, JavaRDD<Vector> vectors, Vector[]
clustersCenters, JavaRDD<Integer> clustersNumbers)
    {
        JavaRDD<VectorDataRW> vectorsDataRW = getVectorDataRW(docsData, vectors);
        return zip(clustersNumbers,
vectorsDataRW).groupByKey().sortByKey().map(tuple ->  {
            List<VectorDataRW> vectorsList = new LinkedList<>();
            tuple._2.forEach(v -> vectorsList.add(v));
```

```
            return new DocsClusterDataRW(tuple._1,
clustersCenters[tuple._1].toArray(), vectorsList);
        });
    }

    /**
     * Возвращает DocsVectorData
     * @param docsData Данные сообщений
     * @param vectors Вектора
     * @return Сформированный DocsVectorData
     */
    public static JavaRDD<DocsVectorData> getDocsVectorData(JavaRDD<DocsData>
docsData, JavaRDD<Vector> vectors)
    {
        return zip(docsData, vectors).map(tuple -> new DocsVectorData(tuple._2(),
tuple._1()));
    }

    /**
     * Возвращает VectorDataRW
     * @param docsData Данные сообщений
     * @param vectors Вектора
     * @return Сформированный VectorDataRW
     */
    public static JavaRDD<VectorDataRW> getVectorDataRW(JavaRDD<DocsData>
docsData, JavaRDD<Vector> vectors)
    {
        return zip(vectors, docsData).map(tuple -> new
VectorDataRW(tuple._1.toArray(), tuple._2));
    }

    /**
     * Функция скрещивания двух rdd списков одинаковой длинны
     * @param rdd1 Первый rdd список
     * @param rdd2 Второй rdd список
     * @param <T> Тип первого списка
     * @param <U> Тип второго списка
     * @return Скрещёный список
     */
    public static <T, U> JavaPairRDD<T, U> zip(JavaRDD<T> rdd1, JavaRDD<U> rdd2){
        return
JavaPairRDD.fromJavaRDD(JavaPairRDD.fromJavaRDD(rdd1.zipWithIndex().map(x -> new
Tuple2<>(x._2, x._1))).join(JavaPairRDD.fromJavaRDD(rdd2.zipWithIndex().map(x ->
new Tuple2<>(x._2, x._1)))).map(x -> x._2));
    }
}
```

```
// Лістинг файлу src/main/java/utils/ClusterInfoWriter.java

package utils;

import models.DocsData;
import models.rw.DocsClusterDataRW;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Vector;
import visualizing.Visualizer;

import java.util.List;

/**
 * Запись информации о кластеризации
 */
```

```
public class ClusterInfoWriter {
    /**
     * Записать информацию о кластеризации
     * @param docs сообщения
     * @param words Слова
     * @param vectors Вектора
     * @param clustersNumbers Количество кластеров
     * @param clustersCenters Центры кластеров
     * @param path Путь к папке записи
     */
    public static void WriteInfo(JavaRDD<String> docs, JavaRDD<List<String>>
words, JavaRDD<Vector> vectors, JavaRDD<Integer> clustersNumbers, Vector[]
clustersCenters, String path){
        JavaRDD<DocsData> docsData = ClusterDataMapper.getDocsData(docs, words);

        String jsonFilePath = path + "clusters.json";
        JavaRDD<DocsClusterDataRW> docsClusterDataRW =
ClusterDataMapper.getDocsClusterDataRW(docsData, vectors, clustersCenters,
clustersNumbers);
        JsonSerializer.dump(jsonFilePath, docsClusterDataRW.collect());

        String imgFilePath = path + "clusters.png";
        Visualizer.plot(imgFilePath, vectors, clustersNumbers, clustersCenters);
    }
}
```

```
// Лістинг файлу src/main/java/utils/JsonSerializer.java

package utils;

import singletons.FileManagerSingleton;
import singletons.GsonSingleton;

/**
 * Обеспечивает сериализацию данных в Json формат
 */
public class JsonSerializer {
    /**
     * Загрузить объект из файла
     * @param path Путь к файлу
     * @param tClass Класс объекта
     * @param <T> Тип объекта
     * @return Объект указанного типа
     */
    public static <T> T load(String path, Class<T> tClass)
    {
        String line;
        if ((line = FileManagerSingleton.getInstance().readString(path)) == null)
return null;
        return GsonSingleton.instance.fromJson(line, tClass);
    }

    /**
     * Сохранить объект в файл
     * @param path Путь к файлу
     * @param object Объект для сериализации
     */
    public static void dump(String path, Object object){
        FileManagerSingleton.getInstance().write(path,
GsonSingleton.instance.toJson(object));
    }
}
```

```
// Лістинг файлу src/main/java/models/rw/ClusterDataRW.java

package models.rw;

import java.io.Serializable;
import java.util.List;

/**
 * Модель данных кластера в формате чтения\записи
 * @param <C> Тип центра кластера
 * @param <T> Тип входящих элементов
 */
public class ClusterDataRW<C, T> implements Serializable {
    public Integer Number;
    public C Center;
    public List<T> Items;

    public ClusterDataRW(Integer number, C center, List<T> items)
    {
        Number = number;
        Center = center;
        Items = items;
    }
}

// Лістинг файлу src/main/java/models/rw/DocsClusterDataRW.java

package models.rw;

import java.util.List;

/**
 * Векторная модель данных кластера в формате чтения\записи
 */
public class DocsClusterDataRW extends ClusterDataRW<double[], VectorDataRW> {
    public DocsClusterDataRW(Integer number, double[] center, List<VectorDataRW>
items){
        super(number, center, items);
    }
}

// Лістинг файлу src/main/java/models/rw/VectorDataRW.java

package models.rw;

import models.DocsData;
import models.VectorData;

/**
 * Модель данных вектора в формате для чтения\записи
 */
public class VectorDataRW extends VectorData<double[]> {
    public VectorDataRW(double[] vector, DocsData document){
        super(vector, document);
    }
}

// Лістинг файлу src/main/java/models/ClusterData.java
```

```
package models;

import java.io.Serializable;

/**
 * Модель данных кластера
 * @param <C> Тип центра кластера
 * @param <T> Тип входящих элементов
 */
public class ClusterData<C, T> implements Serializable {
    public Integer number;
    public C center;
    public Iterable<T> items;

    public ClusterData(Integer number,C center, Iterable<T> items)
    {
        this.number = number;
        this.center = center;
        this.items = items;
    }
}
```

```
// Лістинг файлу src/main/java/models/SimpleClusterData.java

package models;

import org.apache.spark.mllib.linalg.Vector;

/**
 * Проставя векторная модель данных кластера
 */
public class SimpleClusterData extends ClusterData<Vector, Vector> {
    public SimpleClusterData(Integer number, Vector center, Iterable<Vector>
items){
        super(number, center, items);
    }
}
```

```
// Лістинг файлу src/main/java/models/DocsClusterData.java

package models;

import org.apache.spark.mllib.linalg.Vector;

/**
 * Модель данных кластера сообщений
 */
public class DocsClusterData extends ClusterData<Vector, DocsVectorData> {
    public DocsClusterData(Integer number, Vector center, Iterable<DocsVectorData>
items){
        super(number, center, items);
    }
}
// Лістинг файлу src/main/java/models/DocsData.java

package models;

import java.io.Serializable;
import java.util.List;

/**
```

```
 * Модель данных сообщения
 */
public class DocsData implements Serializable {
    public String text;
    public List<String> words;

    public DocsData(String text, List<String> words) {
        this.text = text;
        this.words = words;
    }
}


// Лістинг файлу src/main/java/models/DocsVectorData.java

package models;

import org.apache.spark.mllib.linalg.Vector;

/**
 * Модель данных вектора
 */
public class DocsVectorData extends VectorData<Vector> {
    public DocsVectorData(Vector vector, DocsData docsData){
        super(vector, docsData);
    }
}


// Лістинг файлу src/main/java/models/VectorData.java

package models;

import java.io.Serializable;

/**
 * Модель данных вектора
 */
public class VectorData<T> implements Serializable {
    public T vector;
    public DocsData docsData;

    public VectorData(T vector, DocsData docsData){
        this.vector = vector;
        this.docsData = docsData;
    }
}


// Лістинг файлу src/main/java/io/FileManager.java

package io;

/**
 * Отвечает за работу с файлами
 */
public abstract class FileManager {
    /**
     * Проверка наличия пути к файлу
     * @return Путь существует
     */
    public abstract boolean isPathExists(String path);

```

```
2124        /**
2125         * Создать путь к файлу
2126         * @param path Путь к файлу
2127         */
2128        public abstract void createPath(String path);
2129
2130        /**
2131         * Создать пустой файл
2132         * @param path Путь к файлу
2133         */
2134        public abstract void createFile(String path);
2135
2136        /**
2137         * Записать текст в файл
2138         * @param path Путь к файлу
2139         * @param text Текст
2140         */
2141        public abstract void write(String path, String text);
2142
2143        /**
2144         * Записать данные в файл
2145         * @param path Путь к файлу
2146         * @param bytes Данные
2147         */
2148        public abstract void write(String path, byte[] bytes);
2149
2150        /**
2151         * Считать текст файла
2152         * @param path Путь к файлу
2153         * @return Текст файла
2154         */
2155        public abstract String readString(String path);
2156
2157        /**
2158         * Считать данные файла
2159         * @param path Путь к файлу
2160         * @return Данные файла
2161         */
2162        public abstract byte[] readBytes(String path);
2163
2164        /**
2165         * Создать файл, если его нету
2166         * @param path Путь к файлу
2167         */
2168        public void checkFile(String path){
2169            if (!isPathExists(path)) createPath(path);
2170        }
2171    }
2172
2173    // Лістинг файлу src/main/java/io/FileManagerFactory.java
2174    package io;
2175
2176    import configuration.FileSystemTypeEnum;
2177
2178    /**
2179     * Фабрика для создания FileManager
2180     */
2181    public class FileManagerFactory {
2182        /**
2183         * Получить экземпляр FileManager
2184         * @param fileSystemType Файловая система
```

```
2185         * @return Экземпляр FileManager
2186         */
2187        public static FileManager get(FileSystemTypeEnum fileSystemType){
2188            switch (fileSystemType){
2189                case WindowsFS:
2190                    return new WindowsFileManager();
2191                case HadoopFS:
2192                    return new HadoopFileManager();
2193                default:
2194                    return null;
2195            }
2196        }
2197    }
2198
2199    // Лістинг файлу src/main/java/io/HadoopFileManager.java
2200    package io;
2201
2202    import org.apache.hadoop.conf.Configuration;
2203    import org.apache.hadoop.fs.FSDataInputStream;
2204    import org.apache.hadoop.fs.FSDataOutputStream;
2205    import org.apache.hadoop.fs.FileSystem;
2206    import org.apache.hadoop.fs.Path;
2207    import org.apache.hadoop.io.IOUtils;
2208
2209    import java.io.IOException;
2210    import java.io.UnsupportedEncodingException;
2211
2212    public class HadoopFileManager extends FileManager {
2213        static FileSystem _fs;
2214
2215        public HadoopFileManager() {
2216            if (_fs == null) {
2217                try {
2218                    Configuration config = new Configuration();
2219                    _fs = FileSystem.get(config);
2220                } catch (Exception e) {
2221                    e.printStackTrace();
2222                }
2223            }
2224        }
2225
2226        @Override
2227        public boolean isPathExists(String path) {
2228            try {
2229                _fs.exists(new Path(path).getParent());
2230            } catch (Exception e) {
2231                e.printStackTrace();
2232            }
2233            return false;
2234        }
2235
2236        @Override
2237        public void createPath(String path) {
2238            try {
2239                _fs.create(new Path(path).getParent());
2240            } catch (Exception e) {
2241                e.printStackTrace();
2242            }
2243        }
2244
2245        @Override
```

```
2246        public void createFile(String path) {
2247            createPath(path);
2248            try {
2249                _fs.createNewFile(new Path(path));
2250            } catch (Exception e) {
2251                e.printStackTrace();
2252            }
2253        }
2254
2255        @Override
2256        public void write(String path, String text) {
2257            try {
2258                write(path, text.getBytes("UTF-8"));
2259            } catch (UnsupportedEncodingException e) {
2260                e.printStackTrace();
2261            }
2262        }
2263
2264        @Override
2265        public void write(String path, byte[] bytes) {
2266            FSDataOutputStream out = null;
2267            try {
2268                out = _fs.create(new Path(path));
2269                out.write(bytes);
2270                out.close();
2271            } catch (IOException e) {
2272                IOUtils.closeStream(out);
2273                e.printStackTrace();
2274            }
2275        }
2276
2277        @Override
2278        public String readString(String path) {
2279            FSDataInputStream in = null;
2280            try {
2281                in = _fs.open(new Path(path));
2282                String result = in.readUTF();
2283                in.close();
2284                return result;
2285            } catch (IOException e) {
2286                IOUtils.closeStream(in);
2287                e.printStackTrace();
2288            }
2289            return null;
2290        }
2291
2292        @Override
2293        public byte[] readBytes(String path) {
2294            FSDataInputStream in = null;
2295            try {
2296                in = _fs.open(new Path(path));
2297                byte[] result = in.readUTF().getBytes();
2298                in.close();
2299                return result;
2300            } catch (IOException e) {
2301                IOUtils.closeStream(in);
2302                e.printStackTrace();
2303            }
2304            return null;
2305        }
2306    }
2307
```

```
// Лістинг файлу src/main/java/io/WindowsFileManager.java

package io;

import java.io.*;

public class WindowsFileManager extends FileManager {

    @Override
    public boolean isPathExists(String path) {
        File file = new File(path);
        return file.exists() && !file.isDirectory();
    }

    @Override
    public void createPath(String path) {
        new File(path).getParentFile().mkdirs();
    }

    @Override
    public void createFile(String path) {
        createPath(path);
        try {
            new File(path).createNewFile();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void write(String path, String text) {
        checkFile(text);
        try {
            PrintWriter writer = new PrintWriter(new File(path), "UTF-8");
            writer.println(text);
            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void write(String path, byte[] bytes) {
        checkFile(path);
        try {
            FileOutputStream fos = new FileOutputStream(new File(path));
            fos.write(bytes);
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public String readString(String path) {
        byte[] bytes = readBytes(path);
        try {
            return bytes != null ? new String(bytes, "UTF-8") : null;
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return null;
    }
```

```
    @Override
    public byte[] readBytes(String path) {
        try {
            File file = new File(path);
            FileInputStream fis = new FileInputStream(file);
            byte[] data = new byte[(int) file.length()];
            fis.read(data);
            fis.close();
            return data;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

// Основні лістинги файлів додатку ModelTrainer

// Лістинг файлу src/main/java/Trainer.java

```
import configuration.StartAppConfiguration;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.ml.feature.Word2Vec;
import org.apache.spark.ml.feature.Word2VecModel;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.types.*;
import preprocessing.Preprocessor;
import singletons.SparkSingleton;

import java.io.IOException;
import java.util.List;

class Trainer {
    private static final StructType schema = new StructType(new StructField[]{
        new StructField("trainData", new ArrayType(DataTypes.StringType, true),
false, Metadata.empty())
    });

    static void StartWork(){
        JavaSparkContext context = SparkSingleton.getContext();
        SparkSession session = SparkSingleton.getSession();

        JavaRDD<String> docs = context.textFile(StartAppConfiguration.pathIn);
        JavaRDD<List<String>> words = Preprocessor.process(docs);
        JavaRDD<Row> trainData = words.map(x -> RowFactory.create(new String[][]
{x.toArray(new String[x.size()])}));

        Dataset<Row> trainDataFrame = session.createDataFrame(trainData, schema);

        Word2Vec word2Vec = new Word2Vec()
                .setInputCol("trainData")
                .setOutputCol("result")
                .setVectorSize(5)
                .setMinCount(0);

        Word2VecModel w2vModel = word2Vec.fit(trainDataFrame);
        try {
```

```
            w2vModel.write().overwrite().save(StartAppConfiguration.pathOut);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

// Лістинг файлу src/main/java/Main.java

```
import configuration.SparkConfiguration;
import configuration.StartAppConfiguration;

public class Main {
    /**
     * Точка входа в приложение
     * @param args аргументы которые получает программа при старте
     */
    public static void main(String[] args){
        int currentArg = 0;

        System.setProperty("hadoop.home.dir", "C:\\winutils\\");

        SparkConfiguration.appName = "ModelTrainer";

        StartAppConfiguration.pathIn = args[currentArg++];
        StartAppConfiguration.pathOut = args[currentArg++];

        StartAppConfiguration.isLocalStart = args.length > currentArg &&
args[currentArg++].equals("-local");

        Trainer.StartWork();
    }
}
```

// Лістинг файлу src/main/java/configuration/StartAppConfiguration.java

```
package configuration;

/**
 * Настройки конфигурации запуска приложения
 */
public class StartAppConfiguration {
    public static String pathIn;
    public static String pathOut;
    public static boolean isLocalStart;
}
```

// Основні лістинги файлів додатку DocsGetter

// Лістинг файлу src/main/java/Main.java

```
import configuration.StartAppConfiguration;
import configuration.DocsConfiguration;

public class Main {
    /**
     * Точка входа в приложение
     * @param args Параметры запуска
     */
    public static void main(String[] args){
        int currentArg = 0;
```

```
2487
2488            DocsConfiguration.accessToken = args[currentArg++];
2489            DocsConfiguration.accessTokenSecret = args[currentArg++];
2490            DocsConfiguration.consumerKey = args[currentArg++];
2491            DocsConfiguration.consumerSecret = args[currentArg++];
2492
2493            StartAppConfiguration.docsCount = Integer.parseInt(args[currentArg++]);
2494            StartAppConfiguration.pathOut = args[currentArg++];
2495
2496            DocsObserver.startCollecting();
2497        }
2498    }
2499
2500    // Лістинг файлу src/main/java/DocsObserver.java
2501    import configuration.StartAppConfiguration;
2502    import configuration.DocsConfiguration;
2503    import Docs4j.*;
2504    import Docs4j.conf.ConfigurationBuilder;
2505
2506    import java.io.FileWriter;
2507    import java.io.IOException;
2508    import java.util.ArrayList;
2509    import java.util.List;
2510
2511    /**
2512     * Отвечает за сбор и сохранение сообщений
2513     */
2514    public class DocsObserver {
2515
2516        /**
2517         * Начать сбор сообщений
2518         */
2519        public static void startCollecting()
2520        {
2521            ConfigurationBuilder configurationBuilder = DocsAuth();
2522            DocsStream DocsStream = DocsStreamInit(configurationBuilder);
2523            DocsStreamFilter(DocsStream);
2524            DocsStreamListener(DocsStream, StartAppConfiguration.docsCount);
2525        }
2526
2527        private static ConfigurationBuilder DocsAuth(){
2528            ConfigurationBuilder configurationBuilder = new ConfigurationBuilder();
2529            configurationBuilder
2530                    .setOAuthConsumerKey(DocsConfiguration.consumerKey)
2531                    .setOAuthConsumerSecret(DocsConfiguration.consumerSecret)
2532                    .setOAuthAccessToken(DocsConfiguration.accessToken)
2533                    .setOAuthAccessTokenSecret(DocsConfiguration.accessTokenSecret);
2534            return configurationBuilder;
2535        }
2536
2537        private static DocsStream docsStreamInit(ConfigurationBuilder
2538    configurationBuilder){
2539            return new DocsStreamFactory(configurationBuilder.build()).getInstance();
2540        }
2541
2542        private static void DocsStreamFilter(DocsStream DocsStream){
2543            FilterQuery filterQuery = new FilterQuery();
2544            filterQuery.track("a", "the");
2545            filterQuery.language("en");
2546
2547            DocsStream.filter(filterQuery);
```

```
2548        }
2549
2550        private static void docsStreamListener(DocsStream DocsStream, int docsCount){
2551
2552            StatusListener listener = new StatusListener() {
2553                int nowcount = 0;
2554                List<String> docs = new ArrayList<>();
2555
2556                public void onStatus(Status status) {
2557                    if (nowcount <= docsCount && !status.isRetweet()) {
2558                        nowcount++;
2559                        docs.add(status.getText());
2560
2561                    } else if (nowcount > docsCount){
2562                        try {
2563                            saveDocs(docs);
2564                        } catch (IOException e) {
2565                            e.printStackTrace();
2566                        }
2567                        DocsStream.shutdown();
2568                    }
2569                }
2570
2571                public void onDeletionNotice(StatusDeletionNotice
2572    statusDeletionNotice) {
2573
2574                }
2575
2576                public void onTrackLimitationNotice(int numberOfLimitedStatuses) {
2577
2578                }
2579
2580                public void onScrubGeo(long userId, long upToStatusId) {
2581
2582                }
2583
2584                public void onStallWarning(StallWarning stallWarning) {
2585
2586                }
2587
2588                public void onException(Exception ex) {
2589                    ex.printStackTrace();
2590                }
2591            };
2592
2593            DocsStream.addListener(listener);
2594        }
2595
2596        private static void saveTerms(List<String> docs) throws IOException {
2597            FileWriter writer = new FileWriter(StartAppConfiguration.pathOut);
2598            for (String str: docs) {
2599                writer.write(str + "\n");
2600            }
2601            writer.close();
2602        }
2603    }
2604
2605    // Лістинг файлу src/main/java/configuration/StartAppConfiguration.java
2606    package configuration;
2607
2608
```

```
2609  /**
2610   * Настройки конфигурации запуска приложения
2611   */
2612  public class StartAppConfiguration {
2613      public static String pathOut;
2614      public static Integer docsCount;
2615  }
```