Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод фільтрації контекстно-вбудованої реклами в відео контент

(тема)

Виконав:
студент II курсу, групи КСМзм-19-1
Руденко Н.В.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Рубан І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Руденко Ніці Вячеславівни
(прізвище, ім'я, по батькові)

1. Тема роботи Метод фільтрації контекстно-вбудованої реклами в відео контент

затверджена наказом по університету від " 23 " жовтня 2020 р. № 168Стз

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020р.

3. Вхідні дані до роботи Публікації алгоритми узагальнення знань;
Приклад опису класів ситуацій (об'єктів), що змінюються з часом.

4. Перелік питань, що потрібно опрацювати в роботі
1) вивчення існуючих методів фільтрації реклами в відео;
2) розробка системи для ручної розмітки реклами у відеороликах з метою накопичення;
3) зібрання навчальної вибірки за допомогою розробленої системи;
4) розробка та навчання моделі розпізнавання рекламних вставок в відеоролик;
5) розробка та тестування REST API модуля фільтрації реклами;
6) розробка масштабованої серверної частини для обробки запитів браузерного розширення;
7) розробка розширення для браузера.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайди презентації – 11 слайдів.

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд і аналіз існуючих наукових підходів	27.10.20-05.11.20	
2	Огляд і аналіз програмних продуктів	06.11.20-13.11.20	
3	Опис алгоритмів перетворення тексту в	15.11.20-21.11.20	
	векторне подання		
4	Опис алгоритму дететкування рекламних	22.11.20-25.11.20	
	фрагментів		
5	Дослідження архітектури кожного складового	26.11.20-29.11.20	
	модуля системи		
6	Програмна реалізація кожного компонента	30.11.20-04.12.20	
	системи		
7	Тестування реалізованої системи	05.12.20-11.12.20	

Дата видачі завдання 26 жовтня 2020р.

Студент _____
(підпис)

Керівник роботи _____ проф. Рубан І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 81 с., 23 рис., 2 табл., 2 дод., 34 джерел.

НЕЙРОННА МЕРЕЖА, ІНТЕРНЕТ, ВЕКТОРИЗАЦІЯ, МОДЕЛЬ, СЕРВЕР, БРАУЗЕР, MATLAB, WI-FI, YOU-TUBE.

Метою атестаційної роботи є розробка браузерного розширення з функцією інтелектуального фільтра рекламних фрагментів в роликах відеохостингу YouTube.

У ході виконання атестаційної роботи були вирішенні такі завдання:

- вивчення існуючих методів фільтрації реклами в відео;
- розробка системи для ручної розмітки реклами у відеороликах з метою накопичення навчальної вибірки;
- зібрання навчальної вибірки за допомогою розробленої системи;
- розробка та навчання моделі розпізнавання рекламних вставок в відеоролик;
- розробка та протестування REST API модуля фільтрації реклами;
- розробка масштабованої серверної частини для обробки запитів браузерного розширення;
- розробка розширення для браузера.

ABSTRACT

Master’s thesis: 81 pages, 23 figures, 3 tables, 2 appendices, 34 sources.

NEURAL NETWORK, INTERNET, VECTORIZATION, MODEL, SERVER, BROWSER, MATLAB, WI-FI, YOU-TUBE.

The purpose of the certification work is to develop a browser extension with the function of intelligent filter of ad fragments in YouTube video hosting videos.

During the certification work the following tasks were solved:

- study of existing methods of filtering advertising in video;
- development of a system for manual marking of advertising in videos in order to accumulate a training sample;
- collection of a training sample with the help of the developed system;
- development and training of a model for recognizing advertising inserts in a video;
- development and testing of REST API module of advertising filtering;
- development of a scalable server part for processing browser extension requests;
- development of browser extensions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МГУ – мережа глибокого усереднення

LSTM – довга короткострокова пам'ять (англ., Long Short-Term Memory)

NLP – обробка природної мови (англ. Natural Language Processing)

ВСТУП

Машинне навчання – швидко розвивається сегмент області знання, пов'язаної зі штучним інтелектом – є каталізатором розвитку самих різних галузей, починаючи від медицини і фармакології і закінчуючи автопілотіруемимі автомобілями. По всьому світу різні наукові інститути та комерційні компанії намагаються застосовувати машинне навчання, щоб отримати якісно нові результати в своїх областях.

Індустрія розваг і, зокрема, відеоблогінг щороку стрімко набирає популярність. Найпопулярнішою майданчиком для відеохостингу по праву вважається YouTube, що належить компанії Google. Згідно з інформацією з їх офіційного сайту, аудиторія YouTube налічує більше одного мільярда зареєстрованих користувачів і більше одного мільярда годин відео переглядається користувачами щодня.

Разом зі зростанням популярності відеоблогів в Інтернеті з кожним роком також зростає сегмент небажаної відеореклами. Зараз вже досить важко знайти відеоролик без рекламних інтеграцій. Для відеоблогеров реклама є основним доходом, але для користувачів ця реклама рідко має користь, більшість не хотіли б відволікатися на рекламні вставки під час перегляду основного змісту. Цим обумовлена популярність браузерного розширення AdBlock (більше 100 мільйонів пристроїв), який крім блокування рекламних банерів на сайтах може також справлятися зі вставками рекламних роликів у відео. Однак його можливості обмежені блокуванням рекламних вставок, які надаються самою платформою YouTube і не є частиною відеоролика.

В окремий клас необхідно виділити рекламні вставки, які додаються самим блогером в відеоролик під час монтажу. Детектування і блокування такого виду реклами є складним завданням, на яку AdBlock не розрахований. Деякі відеоблогери намагаються інтегрувати рекламу в основний розповідь.

Такі рекламні інтеграції з одного боку часто неможливо прибрати, не зачепивши основний зміст, але з іншого боку, такі вставки набагато менше дратують користувачів. Тому в даній роботі ми сфокусуємось на типі реклами, яка є частиною відеоролика, яку можна виділити від основного змісту.

Варто відзначити, що рішення задачі детектування реклами актуально не тільки для користувачів, але і для самих рекламодавців. Система, здатна правильно детектувати і класифікувати рекламні вставки, може дозволити рекламодавцям зробити процес перевірки виконання замовлень на рекламу і отримання відповідних звітів автоматичним.

У першому розділі «Аналіз предметної області» наводиться огляд і аналіз існуючих наукових підходів і програмних продуктів по темі дослідження. У другому розділі «Теоретична частина» описуються алгоритми перетворення тексту в векторне подання та алгоритми машинного навчання, які використовуються для вирішення задач класифікації. У третьому розділі «Проектування» описується алгоритм детектування рекламних фрагментів, архітектура системи в цілому і архітектура кожного складового модуля системи. У четвертому розділі «Реалізаційна частина» описується програмна реалізація і тестування кожного компонента системи. У висновку наводяться основні результати роботи та напрямки подальших досліджень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд наукових підходів

Перші наукові роботи по темі детектування реклами в відео стали з'являтися в другій половині 90-х років. У більшості наукових статей автори фокусуються на телетрансляції. Формат реклами на телебаченні відрізняється від формату реклами в відеоблозі. По-перше, реклама на телебаченні зазвичай сильно контрастує з телепередачею по колірних і звуковим характеристикам. По-друге, на телебаченні дуже рідко показують один рекламний відеоролик, в переважній більшості випадків в рекламну паузу вміщується не менше трьох рекламних відеороликів. На відеохостингу YouTube, навпаки, рідкісний блогер стане рекламувати більше одного продукту поспіль. Найчастіше у всьому відеоролику зустрічається лише один рекламний фрагмент з рекламою одного продукту. Крім того, аудиторія конкретного каналу на YouTube менш різноманітна і реклама завжди вибирається з урахуванням специфіки цієї аудиторії. У зв'язку з чим, іноді реклама може бути майстерно вплетена в основний розповідь, що робить детектування без обрізки основного змісту неможливим. Однак окремі ідеї і методи, розроблені і протестовані для телетрансляцій, можуть бути застосовані і для відеоблогів. Тому, перш ніж почати розробку власного алгоритму, необхідно вивчити існуючі роботи по темі детектування реклами і зрозуміти, які методи є сенс застосувати до нашого завдання.

У багатьох наукових роботах використовується той факт, що перед і після кожного рекламного ролика присутні кілька чорних кадрів, супроводжуваних зниженням рівня аудіосигналу. Так, в статті [12] для детектування реклами використовують чорні безшумні кадри, визначення яких відбувається з використанням особливостей стандарту MPEG-1. Під

чорними безшумними кадрами маються на увазі ділянки відеоролика із середньою яскравістю кадру і рівнем аудіосигналу нижче заданих порогів.

Після вибору представляють інтерес кадрів застосовується набір статистичних і евристичних правил для визначення меж реклами. Зокрема, авторами було вирішено, що на кордоні реклами повинно знаходитися не менше 6 послідовних чорних безшумних кадрів, тривалість рекламного ролика не повинна перевищувати 90 секунд і протягом рекламної паузи показується не менше трьох рекламних роликів. У відеоблозі більше однієї реклами поспіль практично не зустрічається, ідея з чорними безшумними кадрами швидше за все також не може бути застосована. Еврістіки з відсіканням по типовою довжині потенційно може бути застосована для нашої задачі. Дійсно, тривалість рекламних фрагментів у відеоблозі зазвичай має тривалість від 15 секунд до 2 хвилин. Однак більш точні оцінки слід робити виходячи з тривалості всього відео і конкретного каналу.

В статті [8] крім ознак, пов'язаних з тривалістю реклами і наявністю чорних безшумних кадрів, наводяться такі ознаки, характерні для реклами, як часта зміна сцен, швидкі зміни кольору, статичні кадри і наявність тексту в кадрі. Для кількісного виміру активності в кадрі використовуються метрики Edge Change Ratio (ECR) і Motion Vector Length (MVL). Такий підхід відноситься до простору ознак методам. Також в статті згадуються методи, засновані на пошуку раніше відомої реклами. Такі методи спираються на базу рекламних роликів, для кожного з яких обчислюється «зліпок», який його описує. Для пошуку рекламного ролика в відео використовується метод ковзаючого вікна з обчисленням «зліпка» і порівнянням його з базою раніше відомих рекламних роликів. В кінці автори об'єднують обидва підходи в єдину самонавчальну систему. В першу чергу, для розпізнавання використали підхід на основі ознак. Фрагменти відео, що пройшли первинний відбір, потрапляють на обробку другим методом. Для автоматичного оновлення бази відомих рекламних роликів використовується припущення про те, що новий ролик з великою ймовірністю буде оточений

вже відомими рекламними роликами. Ознакові методи можуть бути застосовані до нашого завдання. Статичні кадри, швидка зміна колірних характеристик кадру і часта зміна сцени характерні для реклами в відеоблог, однак це також характерно і для основного змісту відеоролика. Наприклад, при огляді нового гаджета, камера також часто буде перемикатися між блогером і цим гаджетом. Тому такий підхід в меншій мірі підходить до нашого завдання, ніж до телетрансляцій.

Ознакові методи можуть бути застосовані до нашого завдання. Статичні кадри, швидка зміна колірних характеристик кадру і часта зміна сцени характерні для реклами в відеоблог, однак це також характерно і для основного змісту відеоролика. Наприклад, при огляді нового гаджета, камера також часто буде перемикатися між блогером і цим гаджетом. Тому такий підхід в меншій мірі підходить до нашого завдання, ніж до телетрансляцій.

Підхід, заснований на базі відомих рекламних роликів, однозначно непридатний для нашої задачі. Більшість популярних відеоблогеров рекламують продукт самі, а не вставляють готовий рекламний ролик від рекламодавця, адже так досягається велика ступінь нативної реклами. Крім того, в кожному новому відеоролику блогер зазвичай заново знімає рекламу, навіть якщо рекламований продукт не змінився.

В статті [5] відео розбивається на сцени, і для кожної сцени спочатку обчислюються 6 базових ознак, 4 з яких відносяться до візуальної складової і 2 – до аудіосигналу. Як візуальних ознак в статті використовують Average of Edge Change Ratio (A-ECR), Variance of Edge Change Ratio (V-ECR), Average of Frame Difference (A-FD) і Variance of Frame Difference (V-FD) . В якості ознак аудіосигналу автори використовують детектування зміни голосу на основі різниці мелкепстральних коефіцієнтів і ймовірності приналежності аудіо до кожного з чотирьох типів: мова, музика, тиша, фонова музика.

На основі цих базових ознак потім обчислюються контекстно-залежні ознаки, які є в деякому сенсі усередненням базових ознак по сусіднім сценам.

Обчислені таким чином ознаки подаються на вхід SVM класифікатора, який визначає приналежність сцени до реклами.

На етапі постобробки відбувається об'єднання прилеглих сцен. і видалення занадто коротких сцен. Такий підхід виглядає більш відповідним до нашого завдання, ніж описані раніше.

В статті [4] детектування реклами здійснюють тільки на підставі аудіосигналу. Такий підхід більш обчислювально ефективний, ніж підходи, які використовують відео. Детектування реклами відбувається в кілька стадій. На першій стадії з аудиосигнала витягуються мел-кепстральні коефіцієнти. На підставі обчислених коефіцієнтів визначаються моменти часу з мінімальною енергією сигналу. Для кожної такої точки за допомогою байєсівського критерію визначається, чи відбулася зміна диктора.

Таким чином, більшість існуючих статей адресовано проблеми детектування реклами в телетрансляції, і застосовуються в них методи мало підходять для завдання детектування реклами в відеоблозі. Можна виділити наступні основні відмінності телетрансляцій від відеоблогів, які створюють перешкод для застосування описаних методів:

- у відеоблозі при переході між рекламою і основним змістом відеоролика диктор залишається колишнім;
- в популярних відеоблозі реклама кожен раз унікальна, і застосування бази відомої реклами безглуздо;
- у відеоблозі вставляється тільки один рекламний ролик, а в теле-трансляціях зазвичай від трьох рекламних роликів поспіль;
- у відеоблозі не обов'язково відбувається зміна оточення і візуальних характеристик кадру при переході до рекламного фрагменту.

З описаних вище методів до нашого завдання має сенс застосовувати метод класифікації окремих сцен і евристики, пов'язані з обяеднання ням окремих сцен і відсіканням по тривалості.

1.2 Огляд програмних аналогів.

Тема реклами в Інтернеті і боротьби з нею безумовно є дуже актуальною. Саме тому так популярні і поширені різні інструменти для блокування різного виду рекламних банерів, спливаючих вікон, для анонімізації запитів з метою протидії персоналізованої реклами. Основним видом таких інструментів є розширення для браузера. Будь-який сучасний браузер підтримує роботу розширень, які являють собою скрипти, привносять додаткові можливості в браузер.

Найбільш популярними розширеннями для блокування реклами в Інтернеті є AdBlock+, uBlock, AdGuard, що мають кілька десятків мільйонів користувачів. Подібні розширення здатні блокувати рекламні банери на сайтах і рекламу в відеороликах на YouTube, яка додається самої платформою і не є частиною відеоролика. Інструментів, що дозволяють блокувати рекламу, яка є частиною відеоролика, знайти не вдалося.

В цьому розділі проведено огляд та аналіз існуючих наукових підходів і програмних рішень з теми дослідження. На підставі проведеного аналізу можна зробити висновок про те, що існуючі програмні рішення переважно спрямовані на блокування рекламних банерів на сайтах і не здатні фільтрувати рекламу в відеороликах, а існуючі наукові підходи, пов'язані з пошуком реклами в відео, націлені на телетрансляції. Були виявлені ключові особливості, що відрізняють рекламу в відеоблозі від реклами в телетрансляції.

2 ТЕОРЕТИЧНА ЧАСТИНА

В ході аналізу предметної області були виявлені перспективні методи, потенційно застосовні до нашого завдання, а саме, метод з класифікацією окремих сцен в комбінації з додатковими евристиками. На відміну від оригінальної статті, в даній роботі пропонується будувати признаковая опис сцен не на основі аудіовізуальної інформації, а на основі тексту субтитрів, так як в форматі відеоблогів ця модальність здається більш значущою для завдання пошуку реклами. У цьому розділі будуть розглянуті ключові алгоритми, що застосовуються в задачах класифікації та обробки текстової інформації.

2.1 Перетворення тексту в числовий вектор

Обробка природної мови (Natural Language Processing, NLP) – загальний напрямок штучного інтелекту і математичної лінгвістики. Воно вивчає проблеми комп'ютерного аналізу і синтезу природних мов. У NLP існує усталений набір кроків, які необхідно зробити для підготовки тексту перед навчанням і застосуванням моделей. Більшість алгоритмів машинного навчання призначене для роботи з числовими ознаками, тому перед навчанням і застосуванням будь-яких моделей текст спочатку проходить кілька етапів попередньої обробки і перетворюється в уявлення у вигляді числового вектора.

Для того, щоб застосувати будь-який алгоритм машинного навчання, треба перетворити всі нечислові дані в числові. Значення подвійної точності. Рекомендованим способом перетворення тексту у значення подвійної точності є використання функції str2double. Він може перетворювати символьні вектори, масиви рядків та масиви комірок символьних векторів.

Якщо вхідним аргументом є масив рядків або масив комірок символьних векторів, тоді str2double перетворює його в числовий масив, що має однаковий розмір. Ви можете створювати рядки, використовуючи подвійні лапки. (Рядки мають тип даних рядка, тоді як символьні вектори мають тип даних char.)

Функція str2double може перетворювати текст, який включає коми (як роздільники тисяч) та десяткові крапки. Наприклад, ви можете використовувати str2double для перетворення змінної Balance у таблиці нижче. Баланс представляє числа як рядки, використовуючи кому як роздільник тисяч.

Якщо str2double не може перетворити текст на число, тоді він повертає значення NaN.Хоча функція str2num також може перетворювати текст у числа, це не рекомендується. str2num використовує функцію eval, яка може спричинити ненавмисні побічні ефекти, коли введення тексту містить ім'я функції. Щоб уникнути цих проблем, використовуйте str2double.

Як альтернативу, ви можете перетворити рядки у значення з подвійною точністю, використовуючи функцію double. Якщо введенням є масив рядків, тоді double повертає числовий масив із однаковим розміром, як це робить str2double. Однак, якщо введеним символом є вектор, тоді подвійне перетворення окремих символів у числа, що представляють їх значення Unicode.

Цей список узагальнює найкращі практики перетворення тексту в числові значення:

- для перетворення тексту в числові значення використовувати функцію str2double. Він послідовно обробляє масиви рядків, вектори символів та масиви комірок векторів символів;

- можно використовувати функцію double для масивів рядків. Однак він по-різному трактує вектори символів;

- уникати str2num. Він викликає функцію eval, яка може мати непередбачені наслідки.

Шістнадцяткові та двійкові значення. Можно представляти шістнадцяткові та двійкові числа як текст або як літерали. Коли пишуться як літерали, треба використовувати префікси 0x та 0b. Наприклад, запишемо шістнадцяткове число як літерал. Префікс обов’язковий.Потім перетворюємо текст, що представляє те саме значення, за допомогою функції hex2dec. Він розпізнає префікс, але не вимагає його. Перетворюємо текст, що представляє двійкові значення, за допомогою функції bin2dec.

Дата та час. MATLAB забезпечує типи даних дати та тривалості, щоб зберігати дати та час та обробляти їх як числові значення. Щоб перетворити текст, що відображає дати та час, використовуйте функції дати та часу. Перетворити текст, що представляє дату, у значення дати та часу. Функція datetime розпізнає багато поширених форматів дат та часу. Перетворюємо масиви, що представляють дати та час.

Якщо ви перетворюєте текст у значення тривалості, використовуйте формати hh: mm: ss або dd: hh: mm: ss.

2.2 Етапи попередньої обробки текстових даних

До безпосереднього перетворення в числовий вектор текст проходить ряд етапів, які приводять його до деякого уніфікованого вигляду, відповідному розв'язуваної задачі. Набір і послідовність цих етапів майже не залежить від завдання. У цей набір входять наступні етапи:

- токенізація – це крок, який розбиває довші рядки тексту на менші шматки або лексеми. Більші фрагменти тексту можна токенізувати у речення, речення – у слова тощо. Подальша обробка, як правило, виконується після того, як фрагмент тексту був відповідним чином маркований. Токенізація також називається сегментацією тексту або лексичним аналізом. Іноді сегментація використовується для розбиття великої частини тексту на частини, більші за слова (наприклад, абзаци чи речення), тоді як токенізація зарезервована для процесу розбиття, результатом якого є виключно слова;

- нормалізація. Перед подальшою обробкою текст потрібно нормалізувати. Нормалізація, як правило, відноситься до ряду пов’язаних завдань, призначених для розміщення всього тексту на рівних ігрових полях: перетворення всього тексту в один і той же регістр (верхній чи нижній), видалення пунктуації, перетворення чисел у їх еквіваленти слова тощо. Нормалізація ставить усі слова на однакові основи і дозволяє обробці тривати рівномірно.

Нормалізація тексту може означати виконання ряду завдань, але для нашого середовища ми підійдемо до нормалізації у три окремі кроки: виведення, лематизація та все інше:

- стеммінг – це процес усунення афіксів (суфіксів, префіксів, інфіксів) зі слова, щоб отримати основу слова;

- лемматизація пов’язана із стеммінгом, відрізняючись тим, що лематизація здатна захоплювати канонічні форми на основі леми слова;

- видалення стоп-слів – процес видалення з тексту шумових слів, що не несуть ніякого смислового навантаження;

Стеммінг та лематизація є основними частинами зусиль з попередньої обробки тексту, це не проста маніпуляція з текстом; вони покладаються на детальне та точне розуміння граматичних правил та норм однак є численні інші кроки, які можуть допомогти поставити весь текст на однаковій основі, багато з яких включають порівняно прості ідеї заміни або видалення. Однак вони не менш важливі для загального процесу. До них належать:встановити всі символи в нижній регістр, видалити числа (або перетворити числа в текстові подання), видалити розділові знаки, зняти пробіли (також зазвичай є частиною токенізації),вилучити стоп-слова за замовчуванням (загальні англійські стоп-слова).

2.3 Векторизація методом мішка слів

У машинному навчанні всі ознаки діляться на наступні типи: бінарні, категоріальні, порядкові і кількісні. Слово є категоріальним ознакою. Для категоріальних ознак найкращим варіантом кодування вважається One-Hot-Encoding (OHE), які має на увазі створення вектора з розмірністю, рівної кількості можливих значень ознаки (в разі слів - це потужність словника), в якому всі елементи дорівнюють нулю, крім одного, відповідного кодованих слову. На рисунку 2.1 представлений приклад такого кодування для колірного ознаки.

color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

Рисунок 2.1 – One-Hot-Encoding колірної ознаки

Мішок слів (bag of words) – це один з класичних методів перекладу тексту в вектор. У цьому методі вектор, відповідний документу, виходить шляхом підсумовування One-Hot-Encoding векторів окремих слів. Нехай $w1$, ..., wm – безліч слів, які можуть бути присутніми в документах, інакше кажучи це використовуваний словник. Нехай ni (d) – це кількість входжень слова wi в документ d. Тоді кожен документ d представляється у вигляді вектора ($n1$ (d), ..., nm (d)). З опису зрозуміло, що такий метод ніяк не враховує порядок слів у документі. На рисунку 2.2 представлений приклад кодування документів методом мішка слів.

	it	is	puppy	cat	pen	a	this
it is a puppy	1	1	1	0	0	1	0
it is a kitten	1	1	0	0	0	1	0
it is a cat	1	1	0	1	0	1	0
that is a dog and this is a pen	0	2	0	0	1	2	1
it is a matrix	1	1	0	0	0	1	0

Рисунок 2.2 – Кодування методом мішка слів

2.4 Векторизація методом Doc2Vec

Метою doc2vec є створення числового подання документа, незалежно від його довжини. Але на відміну від слів, документи не входять у такі логічні структури, як слова, тому слід знайти інший метод.

Концепція, яку використовували Мікілов і Ле, була проста, але розумна: вони використали модель word2vec і додали ще один вектор (ID абзацу нижче), приблизно так:

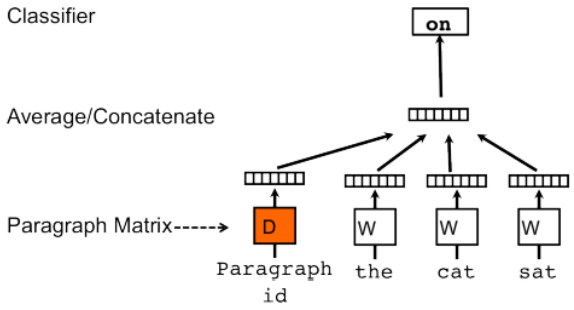

Рисунок 2.3 – Розподілена пам'ять версії абзацу вектора (PV-DM)

Отже, під час навчання словам W, навчається і вектор документа D, і в кінці навчання він містить числове представлення документа. Наведена вище модель називається розподіленою пам'яттю версії абзацу вектора (PV-DM). Він діє як пам’ять, яка запам’ятовує те, чого не вистачає у поточному

контексті – або як тема параграфа. Хоча вектори слів представляють поняття слова, вектор документа має намір представляти поняття документа. Розподілений пакет слів версії абзацу вектора (PV-DBOW).

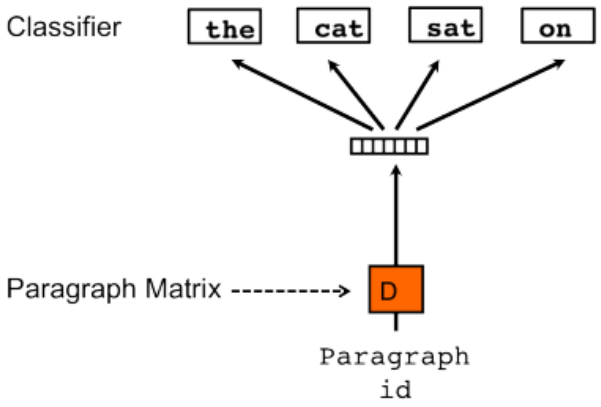

Рисунок 2.4 – Розподілений пакет слів версії абзацу вектора (PV-DBOW)

Цей алгоритм насправді швидший (на відміну від word2vec) і споживає менше пам'яті, оскільки немає необхідності зберігати вектори слів.

Моделі doc2vec можуть бути використані наступним чином: для навчання необхідний набір документів. Вектор слова W генерується для кожного слова, а вектор документа D генерується для кожного документа. Модель також тренує ваги для прихованого шару softmax. На етапі висновку може бути представлений новий документ, і всі коефіцієнти ваги встановлені для обчислення вектора документа.

2.5 Векторизація методом Universal Sentence Encoder

Не у всіх завданнях ключове значення має семантична близькість слів, тому і модель Word2Vec не є універсальною. Одним з найбільш ефективно виконуваних методів вбудовування речень зараз є Universal Sentence Encoder. Ключова особливість тут полягає в тому, що ми можемо використовувати його для багатозадачного навчання.

Це означає, що вбудовані нами речення можуть використовуватися для багатьох завдань, таких як аналіз настроїв, класифікація тексту, схожість речень тощо, а результати цих запитів потім передаються в модель, щоб отримати ще кращі вектори речень, ніж раніше.

Цей кодер заснований на двох моделях кодера, і ми можемо використовувати будь-яку з двох:

- трансформатор;
- мережа глибокого усереднення (МГУ).

Обидві ці моделі можуть взяти слово або речення як вхідні дані та створити вбудовування для них самих. Нижче наведено основний потік:

- токенізуйте речення після перетворення їх у малу літеру;
- залежно від типу кодера, речення перетворюється на 512-мірний вектор;
- вбудовані речення використовуються для різних непідконтрольних та контрольованих завдань.

Щоб почати використовувати вбудовування USE, спочатку потрібно встановити TensorFlow і TensorFlow hub в прикладі 2.1:

```
!pip3 install --upgrade tensorflow-gpu
# Install TF-Hub.
!pip3 install tensorflow-hub
Крок 1: Спочатку імпортуємо такі бібліотеки:
import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
Крок 2: Модель доступна через TFHub. Завантажимо модель:
module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
model = hub.load(module_url)
print ("module %s loaded" % module_url)
```

Приклад 2.1 – Встановлення TensorFlow і TensorFlow hub

Крок 1. Створюємо вбудовування для нашого списку речень, а також для нашого запиту (приклад 2.2):

```
sentence_embeddings = model(sentences)
query = "I had pizza and pasta"
query_vec = model([query])[0]
```

Приклад 2.2 – Створення вбудовування для списку речень та запиту

Крок 2. Обчислимо подібність між нашим тестовим запитом та списком речень (приклад 2.3):

```
for sent in sentences:
sim = cosine(query_vec, model([sent])[0])
print("Sentence = ", sent, "; similarity = ", sim)
```

Приклад 2.3 – Обчислення подібності між тестовим запитом та списком речень

2.6 Методи вирішення задачі класифікації

Методи класифікації корисні, щоб допомогти відповісти на такі запитання. Вони допомагають передбачити членство в групі осіб щодо заздалегідь визначеного членства в групі а також описати характеристики індивідів можуть передбачити їх членство в групі. Прикладами членства в групах / класах можуть бути: лояльні клієнти проти клієнтів, які будуть збиватися; чутливі до високої ціни проти низько цінових клієнтів; задоволені проти незадоволених клієнтів; покупці проти тих, хто не купує; активи, що збільшують вартість порівняно з ні; продукти, які можуть бути хорошими рекомендаціями для замовника, а не - і т. д.

Характеристики, корисні для класифікації осіб / даних у заздалегідь визначені групи / класи, можуть включати, наприклад, демографічні показники; психографія; минула поведінка; ставлення до конкретних продуктів, дані соціальних мереж тощо.

Існує багато методів вирішення класифікаційних проблем: класифікаційні дерева, логістична регресія, дискримінантний аналіз,

нейронні мережі, підсилені дерева, методи глибокого навчання, опорні векторні машини тощо. Існує також безліч пакетів R для всього, що було розроблено в минулому – включаючи «модні» методи глибокого навчання.

Важливим питанням при використанні методів класифікації є оцінка відносної ефективності всіх доступних методів / моделей, тобто для того, щоб використовувати найкращий за нашими критеріями.

2.7 Вирішальні дерева

Дерева рішень – один з методів автоматичного аналізу даних. Розбираємо загальні принципи роботи і області застосування.

Дерева рішень є одним з найбільш ефективних інструментів інтелектуального аналізу даних і самий корінь аналітики, які дозволяють вирішувати завдання класифікації і регресії.

Вони являють собою ієрархічні деревоподібні структури, що складаються з вирішальних правил виду «Якщо ..., то ...». Правила автоматично генеруються в процесі навчання на навчальній множині і, оскільки вони формулюються практично на природній мові ,дерева рішень як аналітичні моделі більш вербалізуемою і інтерпретованною, ніж, скажімо, нейронні мережі.

Власне, саме дерево рішень – це спосіб подачі вирішальних правил в ієрархічній структурі, що складається з елементів двох типів – вузлів (node) і листя (leaf). У вузлах знаходяться вирішальні правила і проводиться перевірка відповідності прикладів цього правила з якого-небудь атрибуту навчальної множини.

У найпростішому випадку, в результаті перевірки, безліч прикладів, які потрапили в вузол, розбивається на дві підмножини, в одне з яких потрапляють приклади, що задовольняють правилу, а в інше – що не задовольняють.

Потім до кожного підмножества знову застосовується правило і

процедура рекурсивно повторюється поки не буде досягнуто деяке умова зупинки алгоритму. В результаті в останньому вузлі перевірка і розбиття не проводиться і він оголошується листом. Лист визначає рішення для кожного потрапив в нього приклад.

Для дерева класифікації – це клас, що асоціюється з вузлом, а для дерева регресії – відповідний листу модальний інтервал цільової змінної.

Таким чином, на відміну від вузла, в листі міститься не правило, а підмножина об'єктів, які відповідають всім правилам гілки, яка закінчується даними листом.

Очевидно, щоб потрапити в лист, приклад повинен відповідати всім правилам, які лежать на шляху до цього листа. Оскільки шлях в дереві до кожного листу єдиний, то й кожен приклад може потрапити тільки в один лист, що забезпечує єдність розв'язку.

Основна сфера застосування дерев рішень – підтримка процесів прийняття управлінських рішень, використовувана в статистиці, аналізі даних і машинному навчанні. Завданнями, які розв'язуються за допомогою даного апарату, є:

- класифікація – віднесення об'єктів до одного з заздалегідь відомих класів. Цільова змінна повинна мати дискретні значення;
- регресія (чисельне пророкування) – прогноз числового значення незалежної змінної для заданого вхідного вектора;
- опис об'єктів – набір правил в дереві рішень дозволяє компактно описувати об'єкти. Тому замість складних структур, що описують об'єкти, можна зберігати дерева рішень.

2.8 Нейронні мережі

Штучна нейронна мережа – математична модель, побудована за принципом організації та функціонування біологічних нейронних мереж. Нейронна мережа являє собою один з методів машинного навчання і є

приватним методом вирішення завдання класифікації. На рисунку 2.5 показана модель нейрона, що лежить в основі штучних нейронних мереж.

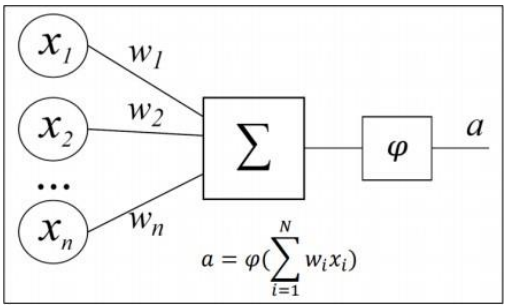

Рисунок 2.5 – Модель нейрона

Нейрони утворюють між собою зв'язки і вихідний сигнал одного нейрона надходить на вхід іншим нейронам. У більшості нейронних мереж сигнал поширюється в одному напрямку без зворотних зв'язків, як представлено на рисунку. Такі мережі називаються нейронними мережами прямого поширення.

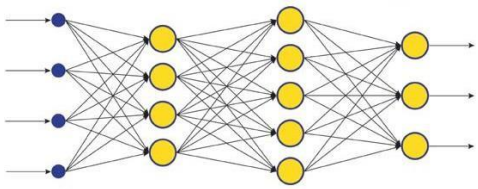

Рисунок 2.6 – Модель багатошарової нейронної мережі

У завданнях обробки послідовностей широко застосовуються рекуррентні нейронні мережі. Їх особливість полягає в тому, що вони мають зворотні дуги, тобто вихід нейрона може передаватися на вхід цього ж нейрона або нейронам з більш ранніх шарів мережі. На рисунку 2.5 представлена архітектура простої мережі з рекуррентним зв'язком.

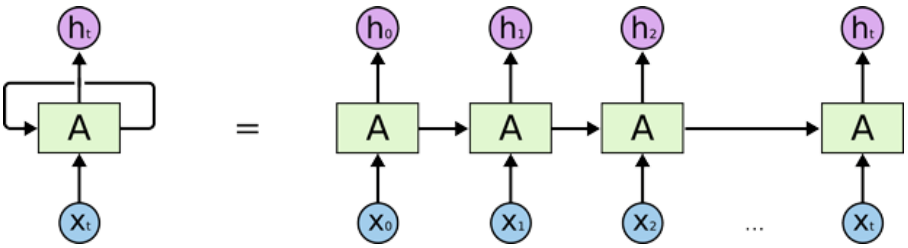

Рисунок 2.5 – Архітектура рекуррентной нейронної мережі

Однак класичні рекурентні нейронні мережі мають істотний недолік – вони мають дуже короткою пам'яттю, що не дозволяє вловлювати довгострокові залежності в послідовності. З цих причин на практиці застосовують такі модифікації рекурентних нейронних мереж, як LSTM і GRU. LSTM-мережа пристосована до завдань класифікації, обробки і прогнозуванні часових рядів і здатна до навчання довготривалим і короткочасним залежностям. Основним компонентом LSTM-мережі є стан осередку. Стан осередки є конвеєрну стрічку, яка проходить через всю послідовність модулів рекуррентної нейронної мережі, беручи участь лише в декількох лінійних перетвореннях.

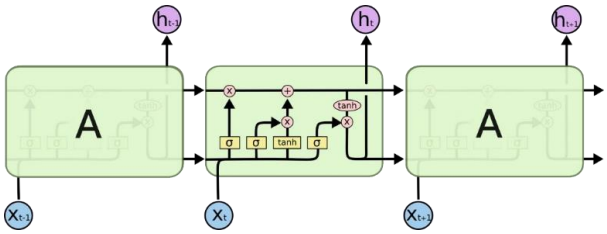

Рисунок 2.6 – Архітектура нейронної мережі LSTM

2.9 Метрики якості моделі

Вибір метрики якості – невід'ємна частина рішення будь-якої задачі машинного навчання. Метрика повинна відповідати тій меті, яку ми хочемо досягти, вирішуючи завдання. Наше завдання полягає в детектуванні

рекламних фрагментів у відео, що еквівалентно бінарної класифікації окремих кадрів, тому подальше пояснення використовуваних метрик буде вестися з припущення, що ми вирішуємо завдання бінарної класифікації кадрів.

Для початку опишемо проміжні величини, використовувані в метриках бінарної класифікації, а саме елементи матриці помилок (рисунок 2.7).

True Positives (TP) – кількість об'єктів, вірно віднесених моделлю до позитивного класу.

False Positives (FP) – кількість об'єктів, невірно віднесених моделлю до позитивного класу.

False Negatives (FN) – кількість об'єктів, невірно віднесених моделлю до негативного класу.

True Negatives (FN) – кількість об'єктів, вірно віднесених моделлю до негативного класу.

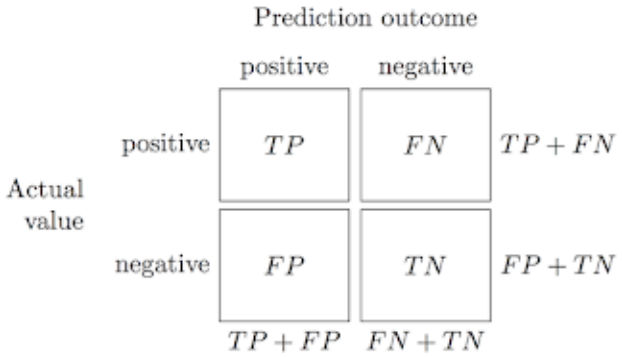

Рисунок 2.7 – Confusion Matrix

Завдання детекції реклами по природі відповіді близька до задачі детектування об'єктів на зображеннях, змінюється тільки розмірність відповіді. У завданнях з зображеннями традиційної метрикою є Intersection over *Union (IoU)*, тому в якості однієї з метрик пропонується використовувати саме *IoU*.

Для задач бінарної класифікації часто використовуваними і добре інтерпретуються метриками є *Accuracy, Precision і Recall*. Метрика *Accuracy* показує частку вірно класифікованих об'єктів, але вона абсолютно марна для задач з сильним дисбалансом класів. У задачі детектування реклами дисбаланс класів дуже великий – весь відеоролик може тривати більше години, при цьому реклама рідко перевищує пару хвилин. Метрика *Precision* показує, яка частка об'єктів, віднесених моделлю до позитивного класу, дійсно до нього ставляться. Метрика *Recall* показує, яку частку об'єктів позитивного класу моделі вдалося знайти.

Accuracy Metric. Цю метрику можна назвати базовою. Вона вимірює кількість правильно класифікованих об'єктів щодо загальної кількості всіх об'єктів. Майте на увазі, що accuracy має деякі недоліки: вона не ідеальна для незбалансованих класів, де може бути багато примірників одного класу і мало іншого

$$accuracy = \frac{TR + TN}{TP + TN + FP + FN}. \quad (2.1)$$

Recall/Sensitivity Metric. Скільки об'єктів наша модель змогла правильно класифікувати з позитивною міткою зі всієї безлічі позитивних

$$recall = \frac{TP}{TP + FN}. \quad (2.2)$$

Precision Metric. Скільки з усіх об'єктів, які класифікуються як позитивні, дійсно є позитивними, щодо загальної кількості отриманих від моделі позитивних міток

$$precision = \frac{TP}{TP + FP}. \quad (2.3)$$

Залежно від параметра β віддається пріоритет в сторону *Precision* або в сторону *Recall*. У нашій задачі ми хочемо знаходити якомога більше реклами, при цьому зовсім не хочемо вирізати основний контент. На даному етапі зупинимося на варіанті з рівними вагами для цих метрик, тобто формула буде виглядати наступним чином:

$$F1 = 2 * \frac{precision * recall}{precision + recall}. \quad (2.4)$$

Регресія. Mean Absolute Error (MAE). Метрика вимірює середню суму абсолютної різниці між фактичним значенням і прогнозованим значенням

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t|, where \quad e_t = original_t - predict_t. \quad (2.5)$$

Mean Squared Error (MSE). Вимірює середню суму квадратної різниці між фактичним значенням і прогнозованим значенням для всіх точок даних. Виконується зведення в другу ступінь, тому негативні значення не компенсують позитивними. А також в силу властивостей цієї метрики, посилюється вплив помилок, по квадратурі від початкового значення. Це означає, що якщо в у вихідних вимірах ми помилилися на 1, то метрика покаже 1, 2-4, 3-9 і так далі. Чим менше MSE, тим точніше наше передбачення. Оптимум досягається в точці 0, тобто ми ідеально передбачаємо у формулі (2.6):

$$MAE = \frac{1}{n}\sum_{t=1}^{n}e_t^2, where \quad e_t = original_t - predict_t. \quad (2.6)$$

У порівнянні з середньою абсолютною помилкою, MSE має деякі переваги:

- підкреслює великі помилки над менших помилках;
- є диференційовані, що дозволяє більш ефективно використовувати для пошуку мінімальних або максимальних значень за допомогою математичних методів.

Root Mean Squared Error (RMSE). Це корінь від квадрата помилки. Її легко інтерпретувати, оскільки він має ті ж одиниці, що і вихідні значення (на відміну від MSE). Також вона оперує меншими величинами за абсолютним значенням, що може бути корисно для обчислення на комп'ютері

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}, where \quad e_t = original_t - predict_t. \tag{2.7}$$

Ранжування. Проста метрика. Найкраще передбачене проти людської оцінки (Best Predicted vs Human, BPH): беруть найвищий за релевантністю елемент з отранжіровать алгоритмом, потім порівнюють з людської оцінкою. Ця метрика повертає бінарний вектор збігу або ж розбіжності оцінки алгоритму в порівнянні з людською.

Kendall's tau. Вимірює кореляцію між двома списками проранжовано елементів шляхом підрахунку узгоджених і неузгоджених парних порівнянь: для кожного екземпляра дано дві оцінки рангу (машинне пророкування і пророкування людини). Спочатку вони розкладаються на парні порівняння - розглядається знак відносини між поточним рангом і іншими. Узгодженої парою вважається ситуація, коли знак порівняння відповідає відповідному парному порівнянні з людською анотацією. В іншому випадку результат враховується як неузгоджена пара. Отже, tau обчислюється за формулою (2.8):

$$T = \frac{(number of\ concordant pairs) - (number of\ discordant pairs)}{n * (n-1)/2} \tag{2.8}$$

Зі значеннями від мінус один до одиниці. Чим ближче | τ | значення до одиниці, тим краще рейтинг. Зокрема, коли значення наближаються до мінус одиниці, рейтинг так само гарний, але порядок його елементів слід брати в зворотному порядку. Це типово для оціночних показників, які привласнюють більш високі бали кращим перекладам, тоді як оцінки людей зазвичай привласнюють більш низькі ранги найкращим. Значення нуля вказує на відсутність кореляції.

У цьому розділі були розглянуті і описані теоретичні аспекти перетворення тексту в векторної уявлення, алгоритми машинного навчання для вирішення задачі класифікації і метрики оцінки якості моделей.

3 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Проектування є невід'ємною частиною розробки програмних продуктів. У цьому розділі буде описано проектування системи в цілому і окремих її модулів.

3.1 Проектування системи

Загальна архітектура системи представлена у вигляді діаграми компонентів на рисунку 3.1 і складається з наступних компонентів:

- система накопичення вибірки – компонент, що надає інтерфейс для ручної розмітки рекламних фрагментів у відеороликах з метою накопичення даних для навчання і тестування моделі детекції;
- детектуюча модель – компонент, що містить в собі всю логіку детектування рекламних фрагментів у відеороликах. Використовує розмічені дані для навчання і надає API для пошуку реклами в відео;
- менеджер даних – компонент, що містить в собі логіку забезпечення доступу до даних відеоролика, відповідає за завантаження відео, субтитрів та іншої інформації з відеохостингу YouTube і надає необхідно дані детектуючої моделі;
- оброблювач запитів – компонент, який відповідає за обробку запитів на пошук реклами у відеороликах, взаємодіє з чергою запитів, детектуючої моделлю і базою даних для збереження результатів обробки. Компонент може бути запущений в декількох примірниках на різних фізичних серверах для прискорення швидкості обробки запитів;
- REST-server – компонент, який відповідає на запити пошуку реклами в відео від браузерного розширення. Взаємодіє з базою даних і чергою. При наявності в базі даних результату обробки відразу відправляє відповідь

розширенню. Інакше поміщає запит на обробку в чергу і повертає відповідь з відповідним статусом;

- черга запитів – компонент, що забезпечує асинхронну зв'язок між REST-сервером і обработчиками;

- база даних результатів – компонент, який відповідає за зберігання результатів обробки відеороликів;

- браузерні розширення – компонент, який працює в браузері користувача і відповідає за взаємодію з сервером, отримання інформації про місцезнаходження реклами у відеоролику і її фільтрації під час перегляду.

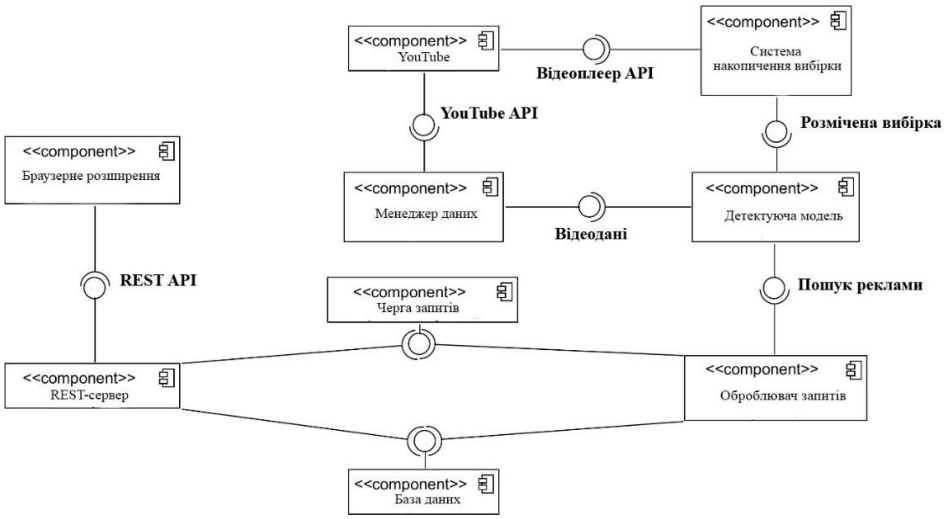

Рисунок 3.1 – Архітектура системи

3.2 Проектування системи для накопичення навчальної вибірки

Ручний збір даних для навчання і тестування моделей – це зазвичай довгий і трудомісткий процес. Тому для цього завдання необхідно застосовувати найбільш ефективні інструменти, які б не забирали час і не ускладнювали процес зайвим функціоналом і технічними особливостями.

У нашій задачі збір даних полягає в підборі відеороликів і виділення в них рекламних фрагментів. Для цього завдання не вдалося знайти існуючого інструменту, досить зручного для використання.

Було прийнято рішення розробити свій інструмент для збору даних, який задовольняв би наступним вимогам:

- дозволяє переглядати відеоролики YouTube;
- дозволяє відзначати місце розташування реклами в відеоролику;
- має мінімальний необхідний інтерфейс;
- є Web-додатком;
- має аутентифікацію користувачів.

3.2.1 Діаграма варіантів використання

На рисунку 3.2 представлена діаграма варіантів використання системи накопичення навчальної вибірки.

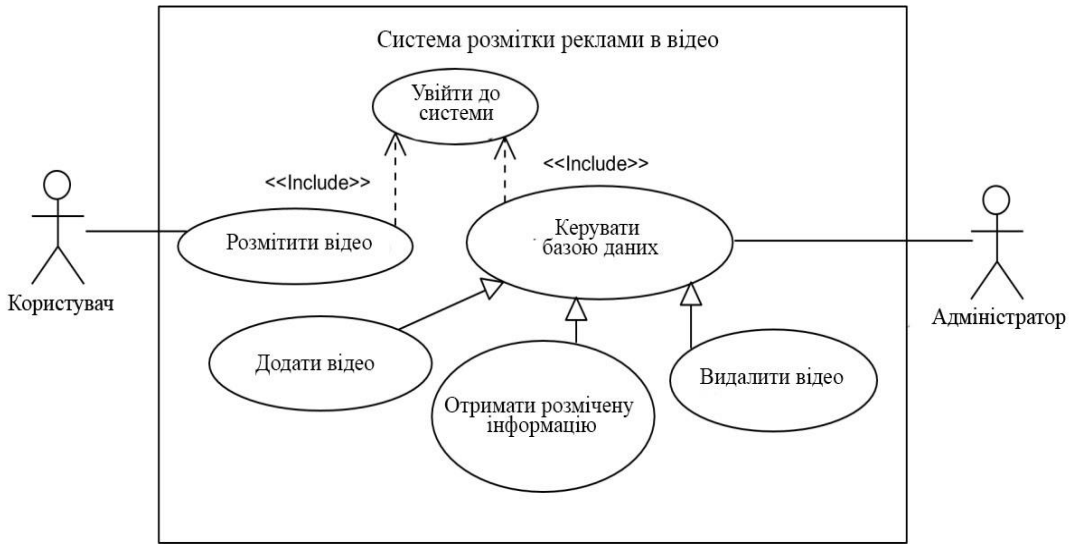

Рисунок 3.2 – Діаграма варіантів використання системи для накопичення навчальної вибірки

В системі визначено дві ролі: Користувач і Адміністратор, які мають по одному основному варіанту використання системи. Користувач може «Розмітити відео», а адміністратор може «Управляти базою даних». Обидва

ці варіанти використання включають в себе «Увійти в систему». варіант використання «Керувати базою даних» узагальнює наступні варіанти:

- «Додати відео»;
- «Отримати розмічену інформацію»;
- «Видалити відео».

3.2.2 Діаграма компонентів

Додаток складається з двох додатків – клієнтського та серверного. На рисунку 3.3 представлена діаграма компонентів.

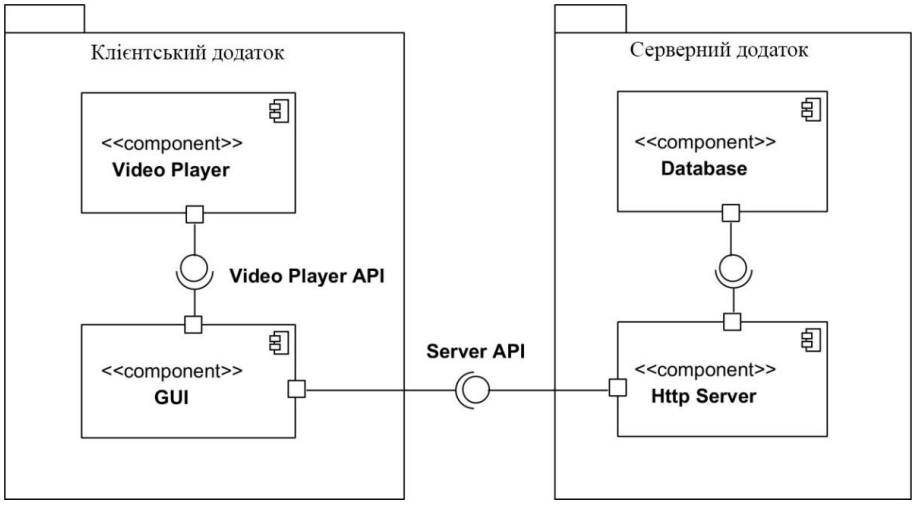

Рисунок 3.3 – Діаграма компонентів системи для накопичення навчальної вибірки

Клієнтську програму забезпечує перегляд відеоролика за допомогою компонента Video Player і надає графічний користувальницький інтерфейс (компонент GUI) для управління відеоплеєром за допомогою Video Player API, розставляння і збереження розмітки відеоролика на сервері через Server API. Серверний додаток складається з Http Server, який забезпечує видачу інформації про відеороликах і збереження розмітки в базі даних (компонент Database).

3.2.3 Діаграма діяльності користувача

На рисунку 3.4 представлена діаграма діяльності користувача в системі. В першу чергу, користувачеві необхідно авторизуватися в системі. Якщо користувач зайшов на цей сайт вперше, то він переводиться на сторінку з інструкцією. Після прочитання інструкції, або відразу після авторизації, завантажується основна сторінка з відеоплеєром і елементами управління для розмітки реклами у відеоролику. Користувач може пропустити відеоролик і відразу перейти до наступного або приступити до перегляду і розмітці завантаженого відео. В ході розмітки користувач відзначає початок і кінець рекламного фрагмента або відзначає, що рекламних вставок у відеоролику немає, і зберігає результат. Після обробки відеоролика користувач може або перейти до наступного, або завершити роботу в системі.

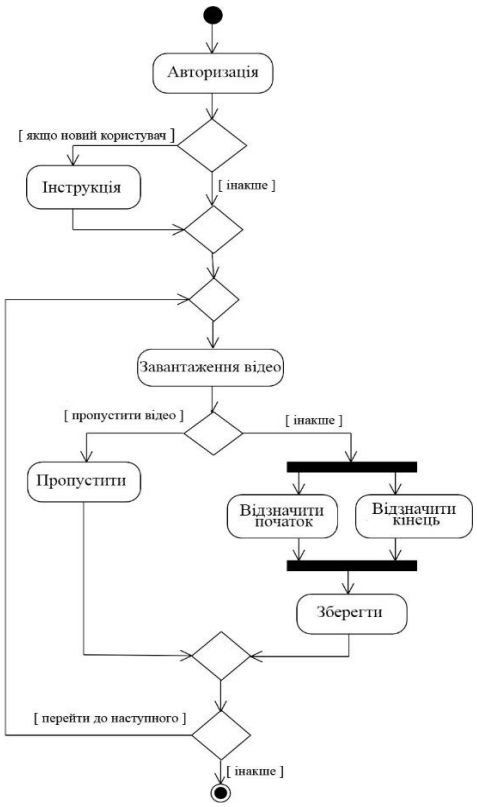

Рисунок 3.4 – Діаграма діяльності користувача в системі для накопичення навчальної вибірки

3.3 Проектування серверної частини

Серверна частина детектора реклами включає в себе наступні компоненти: REST-сервер, набір обробників, чергу для асинхронного взаємодії REST-сервера і обробників, база даних для зберігання статусу і результатів обробки.

REST-сервер в якості свого інтерфейсу надає лише один HTTP-метод: *GET / <video_id>*. Логіка обробки цього запиту залежить від того, чи був оброблений раніше відеоролик із зазначеним ідентифікатором. Діаграма діяльності сервера при обробці запиту представлена на рисунку 3.5.

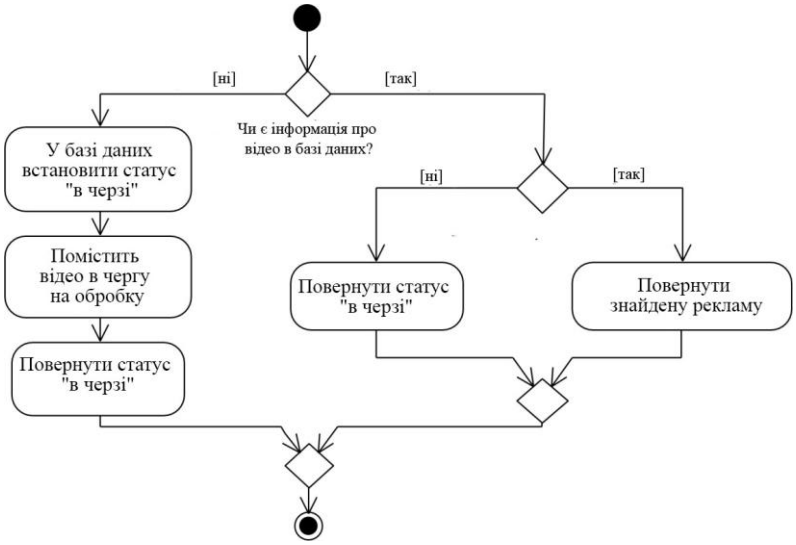

Рисунок 3.5 – Діаграма діяльності REST-сервера

Результат обробки повертається розширення в форматі JSON. Нижче представлений приклад 3.1 відповідей сервера:

```
{
 'videoId': videoId,
 'status': 'IN_QUEUE'
}
```

```
{
 'videoId': videoId,
 'status': 'IN_QUEUE',
 'ads': [[12.5, 34.7]]
}
```

Приклад 3.1 – Відповіді сервера

Завдання оброблювачів полягає в обробці завдань на пошук реклами в відеозаписах, що надходять в чергу від REST-сервера, і збереження результатів обробки в базу даних. Так як обробка кожного відеоролика займає тривалий час, вона повинна проводитися в асинхронному режимі. Також важливо мати можливість масштабувати рішення при зміні навантаження. Архітектура з чергою і набором оброблювачів є класичним способом забезпечити таку масштабованість в асинхронної системі (рисунок 3.6). Залежно від навантаження ми можемо збільшувати або зменшувати кількість активних оброблювачів.

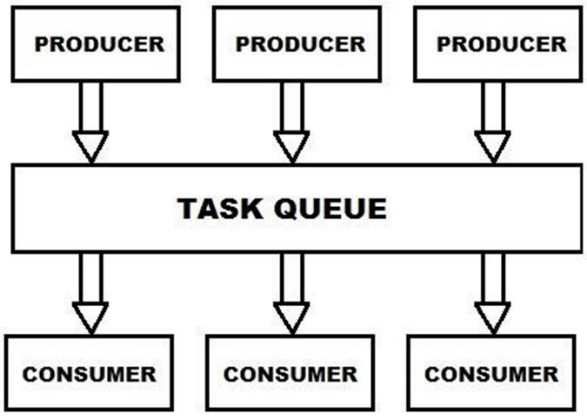

Рисунок 3.6 – Архітектура паттерна Producer-Consumer

3.4 Проектування браузерного розширення

Браузерні розширення буде працювати за наступним алгоритмом. При відкритті сторінки YouTube з відеороликом надсилається запит на REST-сервер з ідентифікатором відео, який витягується з адреси сторінки. У відповідь може прийти результат зі статусом «IN_QUEUE», в цьому випадку запит робиться повторно через деякий час. Спочатку цей час буде фіксованим і рівним 10 секундам, надалі можливо зробити цей час динамічним в залежності від довжини відео і завантаженості оброблювачів. Якщо отримана відповідь зі статусом «OK», запускається функція трекінгу

положення в відео, яка відстежує поточну позицію в відео і при попаданні в рекламний фрагмент змінює її на позиції, що відповідає кінця цього рекламного фрагмента. На рисунку 3.7 представлена діаграма діяльності браузерного розширення.

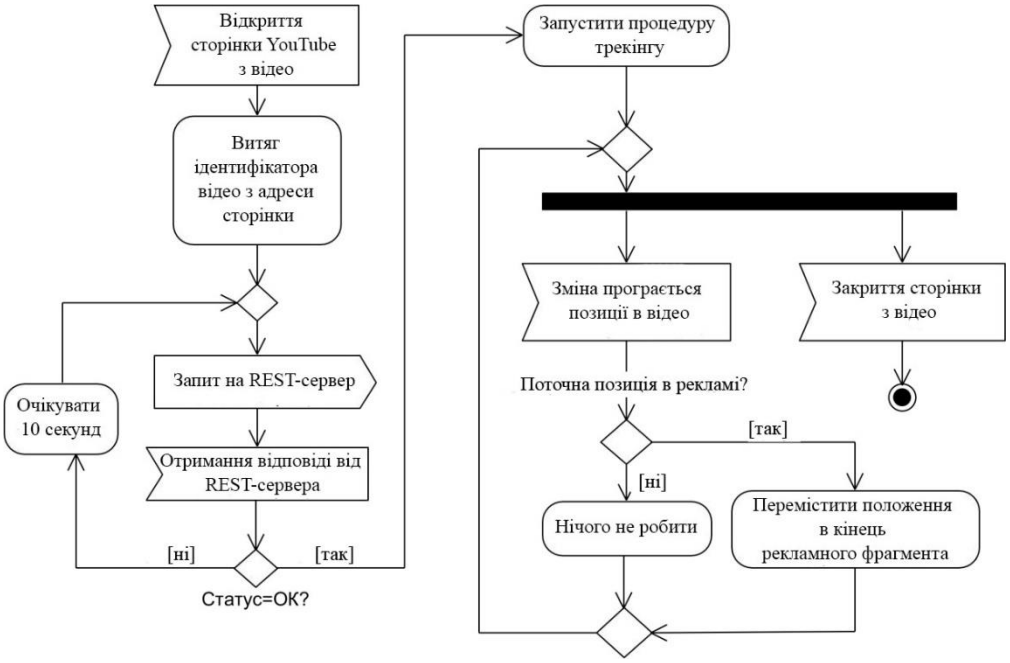

Рисунок 3.7 – Діаграма діяльності браузерного розширення

3.5 Проектування детектуючої моделі

Завдання детектування реклами в відеоролику може бути зведена до задачі бінарної класифікації окремих фрагментів відеоролика на класи «реклама» і «не реклама». Для формування простору ознак опису фрагментів було вирішено використовувати текст субтитрів, так як в форматі відеоблогів ця модальність здається найбільш значущою для завдання виявлення реклами. На рисунку 3.8 представлена загальна схема роботи детектуючої моделі, що включає в себе наступні кроки: розбиття відео на фрагменти, побудова простору ознак опису фрагментів, бінарна класифікація фрагментів і постобробка виділених фрагментів.

Рисунок 3.8 – Загальна схема роботи детектуючої моделі

3.5.1 Розбиття відео на фрагменти

В якості фрагментів відео можуть виступати окремі кадри, проте прив'язати додаткові осмислені ознаки на рівні окремого кадру важко. Тому більш розумним рішенням є поділ на фрагменти тривалістю в кілька секунд. Вибір кордонів таких фрагментів можна зробити статичним і не залежних від конкретного відеоролика. Більш логічним рішенням бачиться розподіл на підставі деякої контекстної інформації. У даній роботі буде використовуватися розбиття на основі переходів між сценами в відео.

У роботах з детектування реклами в телетрансляції часто застосовувався підхід на підставі чорних кадрів. У такому підході при визначенні меж розбивки обчислюється середня інтенсивність пікселів окремого кадру і порівнюється із заданим пороговим значенням.

Інший підхід заснований на обчисленні різниці послідовних кадрів і порівнянні цієї різниці з заданим порогом. Обчислення різниці відбувається попиксельно по кожній компоненті використовуваного колірного простору.

Обидва підходи будуть протестовані на якість розбиття, і кращий з них буде використаний в моделі. Під якістю розбиття будемо розуміти сукупність двох складових:

- середня відстань від кордону реклами до найближчого кордону сцени – чим менше вийде це значення, тим точніше ми зможемо визначати межі реклами;
- кількість знайдених сцен – чим менше це значень, тим швидше буде працювати алгоритм і більш осмисленими будуть отримані сцени.

3.5.2 Ознаковий опис фрагментів на основі субтитрів

На рисунку 3.9 представлена послідовність операцій, що перетворює текст субтитрів до уніфікованого вигляду.

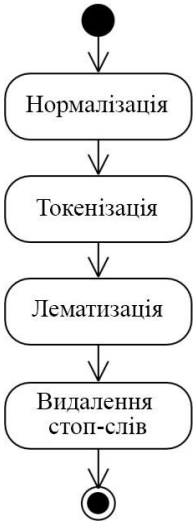

Рисунок 3.9 – Послідовність операцій попередньої обробки тексту

Першим кроком при роботі з текстовими даними в машинному навчанні є попередня обробка тексту. Моделі машинного навчання не можуть працювати зі звичайним текстом, їм необхідні вектора ознак, і для більшості моделей ці ознаки повинні бути числовими. Існує кілька варіантів перетворення тексту в уявлення у вигляді числового вектора. Вибір

конкретного з них часто неочевидний і вимагає проведення порівняльних експериментів на даних розв'язуваної задачі. У даній роботі будуть реалізовані і протестовані наступні методи: мішок слів, Doc2Vec і Universal Sentence Encoder. Кращий підхід буде використаний в підсумковій моделі.

Для методу мішка було вирішено використовувати не всі слова, а лише невеликий набір характерних рекламних слів. Для цього були пораховані частотні словники для рекламних і нерекламних фрагментів, з цих словників були обрані слова, які часто зустрічаються в рекламі і рідко – в основному змісті.

Також важливою ознакою є положення фрагмента в відеоролику. Більшість рекламних фрагментів розташовуються в першій половині відеоролика. Тому до простору ознак опису, який було збудовано на субтитрах, додається ознака рівних відношенню положення фрагмента до загальної тривалості відеоролика.

3.5.3 Класифікація сцен по ознаковому опису

Після складання ознаковому опису окремих фрагментів відеоролика необхідно побудувати класифікатор, то є певний алгоритм, який по ознаковому опису зможе коректно віднести фрагмент до одного з двох класів: «реклама» або «не реклама». Існує велика кількість алгоритмів машинного навчання, які вирішують задачу бінарної класифікації. Згідно з багатьма джерелами, в даний час в більшості завдань кращі результати показують вирішальні дерева і нейронні мережі. При цьому нейронні мережі абсолютно домінують в задачах з великою кількістю даних і однорідних ознак (особливо в задачах обробки аудіо та зображень), а вирішальні дерева відмінно поводяться на різнорідних ознаках (наприклад, зріст, вага і стать людини).

Вибір найкращого алгоритму для конкретного завдання неможливо зробити без порівняльного тестування. У даній роботі будуть реалізовані і

протестовані наступні підходи: градієнтний бустінг над вирішальними деревами, багатошаровий перцептрон і двунаправленна рекуррентна нейронна мережа LSTM. Кращий класифікатор буде використовуватися в підсумковій моделі.

У градієнтний бустінг і багатошаровий перцептрон будуть передаватися опису окремих фрагментів, а в LSTM послідовність з декількох сцен. Точні значення гіперпараметрів моделей будуть визначені на етапі тестування.

3.5.4 Постобробка обраних фрагментів

Будь-який класифікатор на реальних даних не може працювати ідеально і видавати 100% точність. Тому необхідний етап постобробки, на якому будуть усуватися окремі види помилок. Можна придумати різні евристики для застосування на цьому етапі. Застосування кожної з них має грунтуватися на результатах аналізу наявних для навчання даних.

На поточному етапі будуть використані дві прості евристики: видалення занадто коротких фрагментів і склеювання близьких фрагментів в один. Точні параметри цих евристик будуть визначені на етапі реалізації і тестування моделі. Надалі набір використовуваних евристик може збільшитися.

4 РЕАЛІЗАЦІЙНА ЧАСТИНА

4.1 Система накопичення навчальної вибірки

Система розмітки відеороликів реалізована у вигляді Web-додатки. Серверна частина реалізована на платформі Node.js [20] з використанням NoSQL бази даних MongoDB [21]. Клієнтська частина написана на мовах HTML, CSS і JavaScript з використанням відеоплеєра YouTube. Авторизація здійснюється через акаунт ВКонтакте за допомогою протоколу OAuth 2.0. Зображення головної сторінки додатка можна бачити на рисунку 4.1.

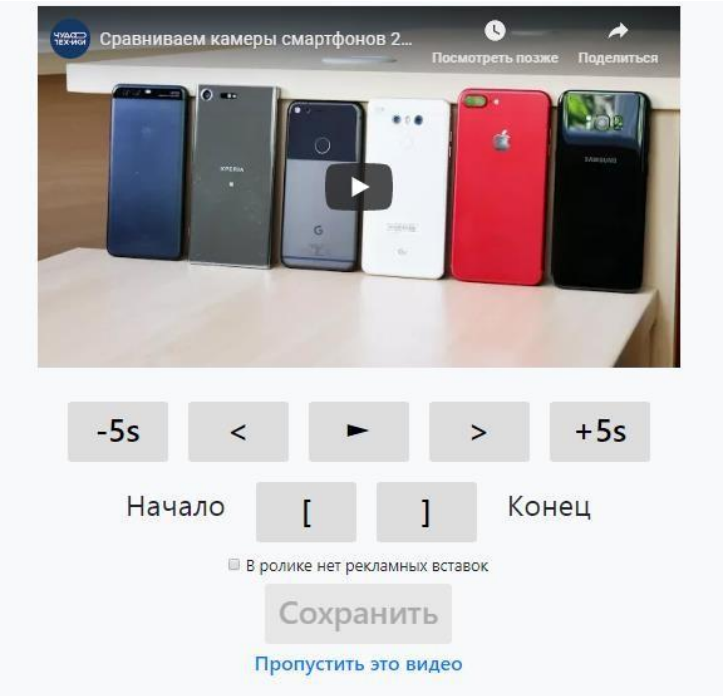

Рисунок 4.1 – Інтерфейс головної сторінки додатка

Інтерфейс містить кнопки для навігації по відео: по 5 секунд і покадрово для більш точного визначення меж рекламного фрагмента. Присутні кнопки для встановлення меж рекламного ролика і кнопка для збереження результатів.

4.2 Завантаження даних для навчання моделей

Для реалізації детектування реклами в відео нам необхідно мати доступ до відеоролика і субтитрами. YouTube не надає відкритого API або SDK для завантаження відеороликів та іншої необхідної інформації, але існують сторонніх рішення, що дозволяють це зробити.

Одним з найпопулярніших інструментів для скачування всілякої інформації про відео є youtube-dl [3]. Цей інструмент має CLI і Python API, дозволяє отримувати інформацію про відео і завантажувати аудіо, відео, субтитри і іншу інформацію.

Для всіх відео на YouTube за умовчанням автоматично генеруються субтитри для різних мов, якщо автор відеоролика спеціально це не заборонив. Відеоролики завантажувалися в форматі «mp4» з висотою кадру 360 пікселів, такі параметри були обрані для мінімального обсягу, займаного на диску, в поєднанні з найбільшою швидкістю завантаження. Відеоролики з меншим дозволом завантажувалися повільніше, що може бути пов'язано з тим, що YouTube віджимає в потрібний формат на льоту під час завантаження.

Субтитри надаються в декількох форматах, які мало чим відрізняються один від одного за рахунок використання бібліотек для роботи з субтитрами, що мають схожий інтерфейс. Для нашої розробки був вибран формат WebVTT і Python пакет webvtt-py [10]. Команда для скачування відеороликів в заданому вирішенні за допомогою програми youtube- dl наведена у прикладі 4.1:

```
youtube-dl –write-auto-sub --sub-format vtt --sub-lang ru –f
“(mp4)[height<=360]” --id https://www.youtube.com/watch?v=<video_id>.
```

Приклад 4.1 – Команда для скачування відеороликів

4.3 Реалізація серверної частини детектора

REST-сервер реалізований на мові Python з використанням фрейм-Ворку Flask [23]. Як було описано на етапі проектування, сервер надаставлять лише один метод: *GET / <video_id*, який повертає відповідь в форматі JSON. Як сховище результатів обробки відеороликів використовується NoSQL база даних MongoDB [21], а в якості черзі запитів брокер повідомлень RabbitMQ [24].

REST-сервер реалізований на мові Python з використанням фреймворку Flask [23]. Як було описано на етапі проектування, сервер надає лише один метод: GET / <video_id, який повертає відповідь в форматі JSON. Як сховище результатів обробки відеороликів використовується NoSQL база даних MongoDB [21], а в якості черзі запитів брокер повідомлень RabbitMQ [24].

4.4 Реалізація розширення для браузера

За різними рейтингами кількість широко відомих браузерів варіюється від 5 до 10. При цьому левову частку ринку довгий час утримує Google Chrome. Наприклад, по інформацій сайту [25] частка браузера Google Chrome за квітень 2019 року становить більше 60% **(**рисунок 4.2). Розробка крос-браузерного стандарту про розробку розширень ще не закінчена в повній мірі, хоча компанії Microsoft, Opera і Mozilla рухаються в цьому напрямку, розвиваючи стандарт, схожий зі специфікацією для браузера Google Chrome. Було вирішено на даному етапі розробляти розширення тільки під самий популярний браузер. Якщо стандарт буде прийнятий і дійсно буде сумісний зі специфікацією Google Chrome, то розроблене розширення буде працювати без доробок і для інших браузерів.

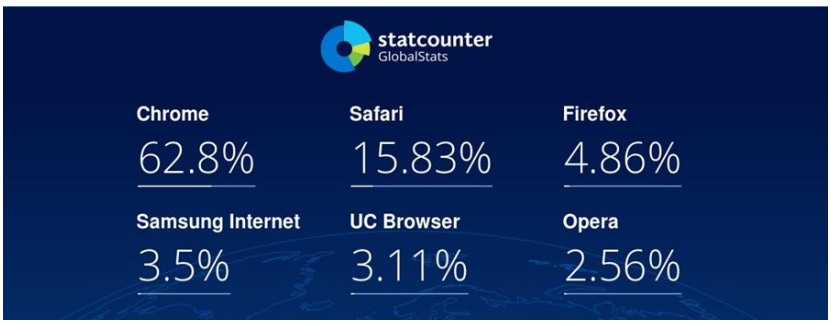

Рисунок 4.2 – Статистика використання різних браузерів

Головним елементом будь-якого розширення для браузера Google Chrome є файл *manifest.json*, в якому міститься основна інформація про розширення: назва, опис, список скриптів, дозволу.

Розширення можуть містять два види скриптів*: background* і *content*. Основна функціональність нашого розширення полягає в фільтрації реклами у відеороликах на YouTube. Фільтрація здійснюється шляхом зміни програється позиції в відео при попаданні в рекламний фрагмент, тобто нам потрібно взаємодія із завантаженою сторінкою, тому використовувати будемо *content* скрипт. Наведемо це в прикладі 4.2:

```
"content_scripts": [
      {
            "matches":
            ["https://www.youtube.com/*"], "js":
            ["contentScript.js"]
      }
]
```

Приклад 4.2 – *Content* скрипт

YouTube здійснює перехід між сторінками за допомогою технології AJAX, тому підписуватися на URL з відеороликом формату *"https://www.youtube.com/watch*"* не має сенсу, так як в цьому випадку у користувача, який зайшов спочатку на головну сторінку, розширення НЕ активується. Тому підписка йде на всі адреси домену www.youtube.com, а

всередині скрипта йде відстеження зміни адреси відображається сторінки. Оскільки нашому розширенню потрібно доступ до активної вкладці і можливість робити запити до REST-сервера, необхідно прописати в маніфесті необхідні дозволи наведені у прикладі 4.3.

```
"permissions": [
      "activeTab",
      "http://<restip>/"
]
```

Приклад 4.3 – Необхідні дозволи

Скрипт реалізований на мові JavaScript. Для запитів до REST-сервера використовується об'єкт типу *XMLHttpRequest*, періодичність перевірки адреси сторінка дорівнює одній секунді, періодичність запитів до сервера в разі повертається статусу «IN_QUEUE» дорівнює 10 секундам.

4.5 Реалізація детектуючої моделі

У розділі проектування був описаний алгоритм роботи детектуючої моделі, який включає в себе наступні кроки: розбиття відео на сцени, попередня обробка субтитрів, переклад тексту субтитрів в уявлення у вигляді числового вектора, класифікація і постобробка. У цьому розділі будуть описані деталі реалізації кожного етапу.

4.5.1 Розбиття на сцени

Для розбиття відеороликів на окремі сцени використовувалася програма PySceneDetect [1], що надає Python API. У PySceneDetect реалізовано два алгоритму детектування зміни сцени. Перший алгоритм обчислює середню яскравість кадру і порівнює її із заданим порогом, що в точності повторює ідею з чорними кадрами, використану у багатьох статтях,

описаних раніше. Другий алгоритм обчислює різницю між двома послідовними кадрами і порівнює її із заданим порогом. Цей метод фактично дозволяє виявляти не тільки зміну сцени, а й швидкі зміни ракурсу.

Щоб визначити кращий алгоритм і кращі параметри для нашої задачі, проведемо детектування сцен з різними настройками і подивимося на наступні метрики:

- середня відстань від кордону реклами до найближчого кордону сцени - чим менше вийде це значення, тим точніше ми зможемо визначати межі реклами;
- кількість знайдених сцен – чим менше це значень, тим швидше буде працювати алгоритм і тим більш осмисленими вийдуть фрагменти.

На рисунку 4.3 і 4.4 представлені результати проведеного порівняння. За ним видно, що кращі результати показує детектор, заснований на різниці сусідніх кадрів. При підвищенні порога детекції приблизно до 25, кількість сцен зменшується експоненціально з незначним зростанням відстані від кордонів реклами до кордонів сцени. При подальшому підвищенні порога детекції кількість сцен зменшується незначно, а відстань від кордонів реклами до кордонів сцени починає стрімко зростати.

На підставі цих результатів було вирішено використовувати алгоритм, заснований на різниці сусідніх кадрів з порогом 20, що забезпечує найкраще співвідношення кількості виділених сцен і середньої відстані до кордонів рекламних фрагментів.

У *PySceneDetect* є ще один важливий параметр – *min_scene_len*. Він відповідає за мінімальну довжину одержуваних на виході сцен. Після декількох експериментів було вирішено встановити цей параметр рівним 5 секундам.

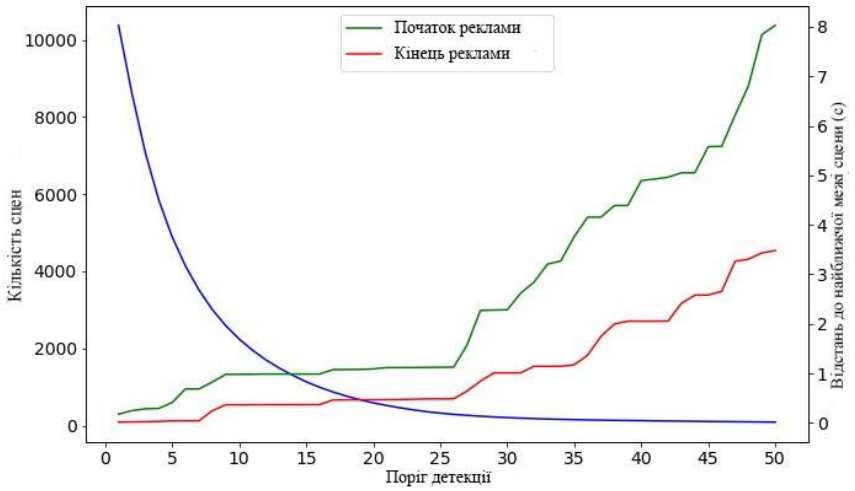

Рисунок 4.3 – Залежність кількості сцен і дальності кордонів реклами від кордонів сцен для детекції по різниці сусідніх кадрів

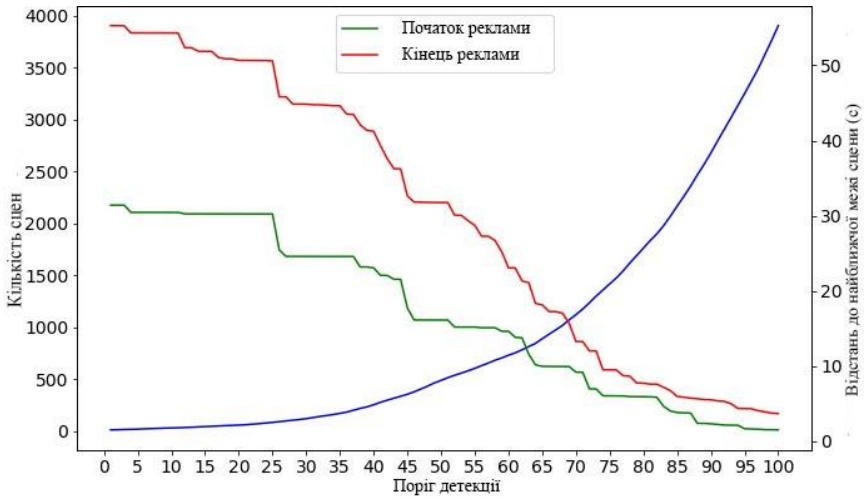

Рисунок 4.4 – Залежність кількості сцен і дальності кордонів реклами від кордонів сцен для детекції по яскравості одного кадру

4.5.2 Передобробка субтитрів

На етапі проектування було визначено, що на етапі попередньої обробки текст субтитрів пройде через такі операції: нормалізація, токенізація, лематизації і видалення стоп-слів.

Нормалізація і токенізація були реалізовані самостійно. На цих етапах текст субтитрів перекладається в нижній регістр, видаляються всі символи, що не входять в алфавіт російської мови, і отриманий текст розбивається на окремі слова. Для лематизації був використаний пакет мови Python *pymorphy* [6]. Стоп-слова взяті з пакету NLTK [26].

4.5.3 Переклад тексту в числовий вектор

Для формування вектора методом мішка слова був використаний об'єкт типу *CountVectorizer* з бібліотеки *scikit-learn* [27]. Як словника були використані тільки найхарактерніші рекламні слова. У цей список увійшли наступні слова: посилання, ссилочку, переходити, опис, партнер, знижка, купон, розпродаж, внизу, акція, рекомендуємо, відсоток, безкоштовно. Ці слова утворили словник для методу мішка слів, тобто на виході для кожного фрагмента виходив цілочисельний вектор розмірності 13, до якого додавався ще одна ознака, що відповідає за позицію фрагмента у відеоролику.

Для побудови векторного представлення тексту субтитрів методом Doc2Vec була використана реалізація з пакета *gensim* [28]. Модель навчалася на субтитрах фрагментів, отриманих після розбиття. Розмір векторів на виході визначався в ході експериментів. Оптимальні значення представлені в розділі тестування.

USE надається у вигляді модуля Tensorflow і доступний через пакет tensorflow-hub. Доступно декілька предобученних моделей різної розмірності і для різних мов. Для російської мови не надається предобученной моделі, тому була використана модель для англійської мови з розмірністю вихідного вектора дорівнює 512, доступна по посиланню [29]. Відповідно, додатково були завантажені і підготовлені субтитри англійською мовою. Для англійських субтитрів замість лематизації використовувався стемінг в складі пакету NLTK.

4.5.4 Класифікатор сцен

На етапі проектування було вирішено реалізувати і протестувати різні види алгоритмів класифікації: градієнтний бустінг над вирішальними деревами, багатошаровий перцептрон і двонаправлені рекурентні мережі LSTM.

Найпопулярнішими реалізаціями градиентного бустінга над вирішальними деревами на даний момент є бібліотеки XGBoost, LightGBM і CatBoost. Для даної роботи була обрана бібліотека CatBoost, так як, згідно з власного досвіду та інформації з їх офіційного сайту [30], CatBoost на більшості завдань працює краще без додаткової настройки параметрів і менше схильний до перенавчання. Оптимальні параметри представлені в розділі тестування.

Для використання багатошарового перцептронав в якості класифікатора була використана реалізація з бібліотеки scikit-learn [31]. В якості оптимізатора використовувався алгоритм Adam з адаптивним темпом навчання. Кількість ітерацій навчання і розміри прихованих шарів визначалися експериментально в залежності від методу векторизації. Оптимальні параметри представлені в розділі тестування.

Для реалізації двобічної мережі LSTM була використана бібліотека Keras [32]. В якості оптимізатора використовувався алгоритм Adam, в якості опції втрат – бінарна крос-ентропія. Архітектура складалася з одного шару двобічної LSTM і повнозв'язну шару з одним нейроном і сігмоідной функцією активації. На вхід надходять опису сцен в вигляді послідовності в рамках заданого вікна. На етапі застосування моделі для кожної сцени виходить кілька передбачень, так як вона потрапляє в кілька вікон. Підсумкове пророкування визначається шляхом голосування. Нижче представлений приклад 4.4 коду створення і навчання моделі:

```
model = Sequential()
model.add(Bidirectional(LSTM(100,
          return_sequences=True),
          input_shape=(self.window, X.shape[2])))
model.add(TimeDistributed(Dense(1,
activation='sigmoid'))) model.compile(optimizer='adam',
loss='binary_crossentropy') np.random.seed(0)
model.fit(X, Y, batch_size=len(X), shuffle=True, epochs=300)
```

Приклад 4.4 - Код створення і навчання моделі

Точна кількість епох і кількість нейронів в прихованому шарі залежить від використовуваного методу векторизації. Оптимальні значення наведені в розділі тестування.

4.5.5 Постобробка

Ніякої класифікатор не здатний видати 100% точність в реальних задачах. Незважаючи на те, що ми спочатку звели вихідну задачу детектування до задачі бінарної класифікації окремих фрагментів відео, варто пам'ятати про початкової постановці і властивості, які з неї випливають. В цьому випадку можна застосовувати ряд додаткових евристик, що поліпшують результати класифікатора.

У поточній реалізації моделі було вирішено використовувати всього два простих правила: склеювання близьких фрагментів в один і видалення занадто коротких фрагментів. Близькими почали вважати фрагменти, відстань між якими не перевищує 10 секунд, а надто короткими – тривалістю до 5 секунд. Такі параметри в ході декількох тестів показали найкращі результати.

4.6 Тестування детектуючої моделі

Розробка будь-якої програмної системи не может обходити без тестування. На цьом етапі перевіряється якість розроблення продукту и

відповідність виставленна на етапі проектування Вимоги. У цьому розділі тестування буде Розбита на две частини: тестування якості детектування розробленої моделі и тестування працездатності системи в цілому.

4.6.1 Опис процедури тестування

Головне завдання учнів алгоритмів – їх здатність узагальнюватися, тобто добре працювати на нових даних. Оскільки на нових даних ми відразу не можемо перевірити якість побудованої моделі, то треба пожертвувати невеликою порцією даних, щоб на ній перевірити якість моделі. Найчастіше це робиться одним з двох способів: відкладена вибірка або крос-валідація.

При використанні підходу з відкладеним вибіркою ми залишаємо і фіксуємо деяку частку навчальної вибірки (як правило від 20% до 40%), навчаємо модель на інших даних (60-80% вихідної вибірки) і вважаємо деяку метрику якості моделі на відкладеної вибірці. Такий спосіб рекомендується застосовувати при дуже великій кількості навчальних даних, так як при маленьких обсягах оцінка по одній відкладеної вибірці виявиться зміщеною.

Другий часто застосовується підхід – це крос-валідація, яку також називають перехресної перевіркою. При застосуванні даного підходу все наявні дані розбиваються на K рівних частин і виконується

K циклів навчання і застосування моделі. На кожному циклі одна з частин вихідного набору даних виступає в якості тестової вибірки, а решта (K - 1 частина) – в якості навчальної. На тестовій частині обчислюються цікавлять метрики якості, які усереднюються по всіх циклах.

Крос-валідація є обчислювально витратною процедурою, тому що модель доводиться навчати K раз, але такий підхід дозволяє отримати несмещенную оцінку якості застосовуваної моделі, особливо при невеликих обсягах даних для навчання. Загальна схема крос-валідації представлена на рисунку 4.5.

Так як в нашій задачі даних мало (100 розмічених відеороликів), в тестування для оцінки якості моделі буде застосовуватися крос-валідація з розбивкою на 4 частини. При розбитті важливо враховувати співвідношення класів в кожній частині. В ідеалі це співвідношення має бути однаковим в кожному розбитті (і дорівнює відношенню класів у всій вибірці). Процедура розбиття з урахуванням співвідношення класів називається стратифікація. Для розбиття зі стратифікацією був використаний клас Strati- fiedKFold з пакета scikit-learn [33].

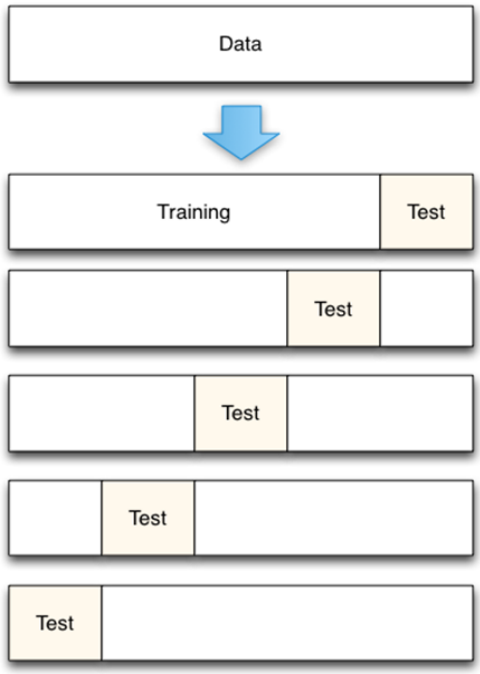

Рисунок 4.5 – Схема крос-валідації

Як метрик якості моделі використовуються *Intersection-over-Union*, *Precision*, *Recall* і *F1-score*.

4.6.2 Вибір відеороликів

Для проведення експериментів був обраний YouTube канал «Чудо техники» [34]. Даний канал був обраний з наступних причин:

- більшість відеороликів мають невелику тривалість (від 3 до 10 хвилин), що на ранньому етапі дозволяє швидше перевіряти гіпотези;

- відеоролики мають схоже за змістом зміст (огляд технологічних новинок), що дозволяє почати з простих підходів, які будуть працювати хоча б для обмеженого класу відеороликів, а потім масштабувати їх на інші класи;

- відеороликів досить багато (близько 300), причому присутні як ролики з рекламою, так і без неї;

- багато рекламних вставки схожі на основний зміст ролика, так як в обох випадках відбувається опис характеристик товару, що створює складності в застосуванні зовсім простих моделей.

За допомогою розробленого раніше інструменту для розмітки реклами у відеороликах були розмічені 100 роликів каналу «Чудо техніки», серед яких 37 роликів містять рекламу.

4.6.3 Результати тестування

В ході тестування детектуючої моделі було проведено 9 серій експериментів. У кожній серії експериментів використовувався один з трьох методів векторизації тексту: мішок слів, Doc2Vec або Universal Sentence Encoder, укупі з одним з трьох алгоритмів класифікації: градієнтний бустінг над вирішальними деревами за допомогою бібліотеки CatBoost, многослойний перцептрон за допомогою бібліотеки Scikit-Learn і двунаправленний LSTM за допомогою бібліотеки Keras.

У кожній серії експериментів проводилося багаторазове кількість запусків для визначення найкращих параметрів методу векторизації і класифікатора. Нижче в таблиці 4.1 представлені знайдені оптимальні параметри для кожної комбінації.

Таблиця 4.1 – Оптимальні параметри порівнюваних методів

		Метод векторизації		
		Bag of Words	Doc2Vec	Universal Sentene Encoder
Класифікатор	CatBoost	iterations=150 max_depth=3	iterations=150 vector_size=50	iterations=150
	MLP	layers=(15, 5)	layers=(50, 10) vector_size=50	layers=(32,)
	BiLSTM	lstm_cells=100 epochs=300 window=5	lstm_cells=100 epochs=500 window=3 vector_size=50	lstm_cells=100 epochs=300 window=3

Пояснення за параметрами:

- iterations / epochs – кількість ітерацій навчання;
- layers – розміри шарів;
- max_depth – максимальна глибина дерев;
- vector_size – розмір вектора на виході з Doc2Vec;
- window – розмір класифікується підпослідовності;
- lstm_cells – кількість нейронів з рекуррентном шарі.

У таблиці 4.2 представлені результати проведеного тестування детектуючої моделі на оптимальних значеннях параметрів. За результатами видно, що найкращі показники отримали при застосуванні підходу мішка слів для векторизації субтитрів. В цьому випадку всі три класифікатора показали хороші результати, при цьому найкращі показники у градиентного бустінга над вирішальними деревами. При використанні Doc2Vec нейронні мережі показали себе краще вирішальних дерев. Прикладом ня методу векторизації Universal Sentence Encoder не справдилося ні з одним з класифікаторів. Така кількість відеороликів мало для отримання максимального результату від нейронних мереж, тому вирішальні дерева показали кращі результати. В подальшій роботі вибірка буде збільшена.

Таблиця 4.2 – Результати тестування моделі

		Метод векторизації текста				
		Bag of Words	Doc2Vec	Universal Sentence Encoder		
Класифікатор	CatBoost	0.481	0.163	0.041	IoU	Метрики
		0.745	0.467	0.195	Precision	
		0.578	0.208	0.049	Recall	
		0.647	0.279	0.076	F1-score	
	MLP	0.471	0.246	0.100	IoU	
		0.686	0.291	0.244	Precision	
		0.606	0.624	0.140	Recall	
		0.639	0.392	0.176	F1-score	
	BiLSTM	0.477	0.365	0.040	IoU	
		0.725	0.702	0.592	Precision	
		0.589	0.437	0.041	Recall	
		0.642	0.523	0.074	F1-score	

4.7 Інтеграційне тестування системи

Інтеграційне тестування – вид тестування, при якому на відповідність вимог перевіряється інтеграція модулів, їх взаємодія між собою, а також інтеграція підсистем в одну загальну систему. Для інтеграційного тестування використовуються компоненти, вже перевірені за допомогою модульного тестування, які групуються в безлічі. Дані безлічі перевіряються відповідно до плану тестування, складеним для них, а об'єднуються вони через свої інтерфейси.

Тестування роботи системи проводилося в браузері Google Chrome з локально розгорнутим REST-сервером, чергою і базою даних. Результати тестування представлені в таблиці 4.3

Таблиця 4.3 – Специфікація інтеграційних тестів

Назва:	Загальний тест системи	
Функція:	Перевірити працездатність всієї системи в цілому	
Дія	Очікуваний результат	Результат тесту
Кроки тесту:		
Додати розширення в браузер згідно з документацією	Розширення успішно додалося	Пройдено
Зайти на сторінку YouTube з відео з рекламою	Відправився запит на сервер, в чергу додався запит, отримана відповідь зі статусом «IN_QUEUE»	Пройдено
Дочекатися відповіді зі статусом «ОК»	За кінцеве кількість спроб отриманий результат зі статусом «ОК»	Пройдено
Дочекатися моменту реклами	Реклама повинна пропустити	Пройдено
Оновити сторінку	Відправився запит на сервер і відразу отримана відповідь зі статусом «ОК»	Пройдено

У ході тестування перевірявся основний варіант використання системи – перегляд відеоролика на YouTube з фільтрацією рекламних фрагментів. Тест був пройдений успішно, всі компоненти системи працюють злагоджено і відповідно до специфікації, розробленої на етапі проектування.

ВИСНОВКИ

Мета даної роботи полягала в розробці розширення для браузера Google Chrome з функцією інтелектуального фільтрування рекламних фрагментів в роликах відеохостингу YouTube. Мета була досягнута, в ході досягнення цієї мети були вирішені наступні завдання:

- вивчені існуючі методи фільтрації реклами в відео;
- розроблена система для ручної розмітки реклами у відеороликах з метою накопичення навчальної вибірки;
- зібрана навчальна вибірка за допомогою розробленої системи;
- розроблена і навчена модель розпізнавання рекламних вставок в відеоролик;
- розроблений і протестований REST API модуля фільтрації реклами;
- розроблена масштабуєма серверна частина для обробки запитів браузерного розширення;
- розроблено розширення для браузера.

Надалі планується продовжити роботу по цій темі, збільшити розмір і різноманіття навчальної вибірки, застосувати сучасні методи роботи з текстом і відео, досліджувати можливість детектування більш складних форматів реклами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Castellano B. PySceneDetect [Electronic resource]. URL: https://pyscenedetect.readthedocs.io/en/latest/ (the date of access: 10.05.2019).

2. Cer D. et al. Universal Sentence Encoder [Electronic Resource]. // arXiv.org, 2018. URL: https://arxiv.org/abs/1803.11175 (the date of access: 07.06.2019).

3. Chih-Hsuan Y. youtube-dl [Electronic resource]. URL: https://github.com/ytdl-org/youtube-dl (the date of access: 06.05.2019).

4. Conejero D., Anguera X. TV advertisements detection and clustering based on acoustic information. // CIMCA, 2008. – P. 452–457.

5. Hua X.S., Lu L., Zhang H.J. Robust learning-based TV commercial detection. // IEEE ICME, 2005.

6. Korobov M. Морфологический анализатор pymorphy2 [Электронный ресурс]. URL: https://pymorphy2.readthedocs.io/en/latest/

7. Le Q. V., Mikolov T. Distributed Representations of Sentences and Documents. // ICML, 2014. – Vol. 32. – P. 1188–1196.

8. Lienhart R., Kuhmunch C., Effelsberg W. On the Detection and Recognition of Television Commercials. // IEEE, 1997.

9. YouTubeAdvertisementDetector – Detector of blogger adver- tisements in YouTube videos [Electronic resource]. URL: https://github.com/ YouTubeAdvertisementDetector (the date of ac- cess: 07.06.2019).

10. Mendez A. webvtt-py – PyPI [Electronic resource]. URL: https://pypi.org/project/webvtt-py/ (the date of access: 06.05.2019).

11. Mikolov T. et al. Efficient Estimation of Word Representations in Vector Space [Electronic resource]. // arXiv.org, 2013. URL: https://arxiv.org/abs/1301.3781 (the date of access: 07.06.2019).

12. Sadlier D.A. et al. Automatic TV advertisement detection from MPEG bitstream. // PRIS, 2002. – P. 14–25.

13. Schuster M., Paliwal K.K. Bidirectional Recurrent Neural Networks. // IEEE TRANSACTIONS ON SIGNAL PROCESSING, 1997. – Vol. 45. – No. 11. – P. 2673–2681.

14. Hochreiter S., Schmidhuber J. Long Short-Term Memory. // Neural Computation, 1997. – Vol. 9. – No. 8. – P. 1735–1780.

15. Радченко В., Кашницкий Ю. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес [Электронный ресурс]. URL: https://habr.com/ru/company/ods/blog/324402/ (дата обращения: 29.05.2019).

16. Хайкин С. Нейронные сети: полный курс / С. Хайкин, 2-е изд. – ООО «И.Д. Вильямс», 2016. – 1104 с.

17. Прессе – YouTube [Электронный ресурс]. URL: https://www.youtube.com/intl/ru/yt/about/press/ (дата обращения: 06.06.2018).

18. AdBlock Plus [Электронный ресурс]. URL: https://adblock-plus.org/ru/ (дата обращения: 06.06.2018).

19. Web-Queue-Worker architecture style [Electronic resource]. URL: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture- styles/web-queue-worker (the date of access: 15.05.2019).

20. Node.js [Electronic resource]. URL: https://nodejs.org/en/ (the date of access: 15.05.2019).

21. MongoDB [Electronic resource]. URL: https://www.mongodb.com/ (the date of access: 15.05.2019).

22. Flask [Electronic resource]. URL: http://flask.pocoo.org/ (the date of access: 07.05.2019).

23. RabbitMQ [Electronic resources]. URL: https://www.rabbitmq.com/ (the date of access: 30.05.2019).

24. StatCounter Global Stats [Electronic resource]. URL: http://gs.statcounter.com/ (the date of access: 15.05.2019).

25. Natural Language Toolkit [Electronic resource]. URL: https://www.nltk.org/ (the date of access: 06.05.2019).

26. Scikit-learn: CountVectorizer [Electronic resource]. URL: https://scikit-

learn.org/stable/modules/generated/sklearn.feature_extrac-
tion.text.CountVectorizer.html (the date of access: 16.05.2019).

27. Gensim: Doc2vec paragraph embeddings [Electronic resource]. URL: https://radimrehurek.com/gensim/models/doc2vec.html (the date of access: 07.06.2019).

28. TensorFlow Hub – Universal Sentence Encoder [Electronic resource]. URL: https://tfhub.dev/google/universal-sentence-encoder/2 (the date of access: 25.05.2019).

29. CatBoost – open-source gradient boosting library [Electronic re- source]. URL: https://catboost.ai/ (the date of access: 30.05.2019).

30. Scikit-learn: MLPClassifier [Electronic resource]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (the date of access: 30.05.2019).

31. Keras Documentation [Electronic resource]. URL: https://keras.io/ (the date of access: 30.05.2019).

32. Scikit-learn: StratifiedKFold [Electronic resource]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.Strati- fiedKFold.html (the date of access: 27.05.2019).

33. Чудо техники – YouTube [Электронный ресурс]. URL: https://www.youtube.com/channel/UCRP4EhX1Op-jL7D87PB3qhQ (дата обращения: 16.05.2019).

34. Мартовицький В.О., Руденко Н.В. Метод фільтрації контекстно-вбудованої реклами в відео контент. Міжнародна конференція «Інформаційні технології та взаємодії» стр. 40, 2–3 грудня 2020 року.