ВСТУП

Перший пристрій для аналізу голосу з'явився ще в 1952 році, завдяки йому можна було розпізнавати проговорені вголос цифри. З тих пір розвиток штучного інтелекту в напрямку natural language processing та computer vision досяг фантастичних маштабів. Завдяки сучасним технологіям можливо якісно оброблювати людську мову та відеофайли. Можливості штучного інтелекту використовуються у всіх сферах людського життя, виключенням не є і медіакомпанії.

На сьогоднішній день сучасним медіакомпаніям є необхідним мати розмічений архів новинних відеосюжетів для подальшої ефективної його обробки, отримання аналітичних даних, успішного спілкування з замовниками. Сама така задача класифікації новинних репортажів виникла на телеканалі «Україна». Новинні канали виконують класифікацію та обробку новинних сюжетів вручну або за допомогою підрядників, проте цю задачу можна виконати завдяки штучному інтелекту.

На дипломну роботу було вибрано завдання створити програмне забезпечення, яке дозволяє визначити start-end time кожного окремо сюжету, хронометраж сюжету, виділити назву сюжету, виділити текст сюжету, виділити тип (новини, анонс або реклама), визначити журналістів та гостей, визначити ведучих, визначити тегів та класифікувати сюжет – позитивний або негативний.

Розроблена система могла б забезпечити економію ресурсів працівників та звільнення людей від рутинної роботи. Також програмний код допомагає уникнути людських помилок.

На сьогоднішній день розроблено багато алгоритмів аналізу тексту, звуку та відеопотоку. Тому, використовуючи ці напрацювання, можна створити алгоритм та розробити систему для аналізу і класифікації новинних репортажів на телеканалі «Україна». Проблема є актуальною, адже такої системи немає у використанні ні в однієї медіа компанії України.

Під час дипломної роботи було проаналізовано доступні математичні методи та програмні рішення для розв’язання поставленої задачі. Серед них обрано Tesseract ORC, Google Speech Recognition, Spacy та NLTK. З використанням даних бібліотек розроблено програмне забезпечення для обробки відеосюжетів.

# ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи є створення математичного та програмного забезпечення для обробки й класифікації архіву новинних відеорепортажів.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

* проведення порівняльного аналізу існуючих методів nlp та cv;
* вибір та адаптація існуючого методу для вирішення задачі nlp та cv;
* розробка програмного забезпечення на базі вибраного математичного методу;
* тестування розробленої автоматизованої системи.

Реалізована система має задовольняти такі вимоги:

* визначати день тижня та номер тижня з назви відеофайлу;
* визначати start-end time кожного окремо сюжету;
* визначати хронометраж сюжету;
* визначати назву сюжету;
* визначати текст сюжету;
* визначати тип (новини, анонс або реклама);
* визначати журналістів та гостей;
* визначати ведучих;
* визначати теги;
* класифікувати сюжет на позитивний чи негативний;
* мати високі показники ефективності розпізнавання start-end time сюжету (точність повинна бути до 1 секунди);
* представляти таблицю обробки даних новинного випуску у вигляді файлу формату Microsoft Excel;
* додавати до результуючої таблиці канал та назву новинної програми.

# АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

Методи розв'язання поставленої задачі відносяться до таких сучасних напрямків роботи з інформацією, як computer vision та natural language processing. Далі наведено короткий огляд основних існуючих рішень, що стосуються даних напрямків.

## Computer vision (CV)

### Парсинг відеопотоку

* + - 1. Tesseract ORC

Tesseract є програмою, що розроблялася компанією Hewlett-Packard з середини 1980-х по середину 1990-х років. Потім програма близько 10 років «пролежала на полиці» і в серпні 2006 року її купила Google. Google відкрив вихідний код під ліцензією Apache 2.0 для продовження розробки [1]. На сьогоднішній день бібліотека є найбільш крупним рішенням, якщо потрібно обробити дані з будь-якого фото. Оскільки в роботі досліджується обробка відеофайлу, то подамо його як набір кадрів і будемо зчитувати текст з кожного кадра окремо. Основні переваги бібліотеки Tesseract - це висока точність та легкість впровадження у власний проект.

* + - 1. OCRopus

OCRopus - OCR-система для розпізнавання текстів на базі Tesseract. Використовує код мови моделювання з проекту OpenFST.

В даний час OCRopus використовує тільки інтерфейс командного рядка, приймаючи на вхід зображення з текстом, і виводячи дані в форматі hOCR (відкритий формат на основі HTML). Опції командного рядка дозволяють виконувати окремо конкретні операції (наприклад, розпізнавання одного рядка).

Основні переваги - такі ж, як і в Tesseract ORC, проте спосіб OCRopus важчий в впроваджені в код, ніж Tesseract ORC, адже Tesseract ORC можна імпортувати як бібліотеку, а OCRopus потрібно вставляти напряму в код [1].

* + - 1. CNN

CNN, або згорткові нейронні мережі, можуть бути використані для вирішення задачі розпізнавання тексту з картинки [2]. Основна перевага використання CNN – можна досягти більш високої точності. Основні недоліки – навчання нейронної мережі займає багато часу, потрібно збирати дуже багато даних, розмічувати їх, вибрати архітектуру та проводити навчання, тому швидше буде використати та налаштувати напереднавчену модель під свої завдання.

### Voise to text system

* + - 1. Google Speech Recognition

Щоб працювати з текстом, для початку необхідно його отримати. Для цього необхідно розпізнати текст зі звуку. Google Speech Recognition - це бібліотека для розпізнавання мови з підтримкою API, бібліотека працює онлайн та офлайн. Загальна кількість мов, які можна розпізнати завдяки Google Speech Recognition, дорівнює 119. Бібліотека Google Speech Recognition є найбільш точною та якісною при роботі зі звуком на українській мові. Також бібліотека непогано справляється із звуковими доріжками, що перевантажені шумом. SpeechRecogntion має низький бар’єр для входу та сумісність із багатьма доступними API розпізнавання мов [3].

* + - 1. Wav2letter++

Wav2letter ++ це швидкий інструментарій для обробки мовлення з відкритим кодом від the Speech у Facebook AI Research, створений для полегшення досліджень у наскрізних моделях розпізнавання мови. Модель швидко працює і зручна у використанні, проте SpeechRecognition краще оброблює шуми на аудіодоріжці [4].

* + - 1. CMUSphinx

CMUSphinx на ринку понад 20 років досліджень CMU. Деякі переваги цієї бібліотеки: інструменти CMUSphinx розроблені спеціально для малоресурсних платформ, гнучкого дизайну та зосереджені на практичній розробці додатків, а не на дослідженні. Точність визначення тексту менша, ніж з використанням SpeechRecognition [5].

## Natural language processing (nlp)

### Визначення тегів

* + - 1. TF-IDM

TF-IDF - це простий і зручний спосіб оцінити важливість терміну для будь-якого документа відносно всіх інших документів. Принцип такий: якщо слово зустрічається в будь-якому документі часто, при цьому зустрічаючись рідко у всіх інших документах, то це слово має велику значимість для того самого документа. Слова, неважливі для взагалі всіх документів, наприклад, прийменники - отримають дуже низьку вагу TF-IDF, бо часто зустрічаються у всіх документах, а важливі - високу.

* + - 1. NLTK

NLTK – одна із найпоширеніших бібліотек на мові Python для обробки тексту. Найвідоміша NLP бібліотека, створена дослідниками в цій галузі. Вона популярна в академічних колах і в основному використовується для навчання або створення різних методів обробки тексту [6]. Так само вона досить повільна в силу того, що написана повністю на Python і працює з рядками [7].

* + - 1. SpaCy

SpaСy – друга популярна бібліотека на мові Python для обробки тексту. Вона значно швидше, ніж NLTK, так як вона написана на Cython і працює з об'єктами. SpaCy надає в основному кращі інструменти для вирішення конкретного завдання. В цілому, SpaCy з її напереднавченими моделями, швидкістю і зручним API набагато краще підходить для розробників, що створюють готові рішення. Для створення власних моделей ні NLTK, ні SpaCy не підходить. Для цього існують Tensorflow, PyTorch та інші [6, 8].

### Класифікація настрою сюжету

* + - 1. SentimentIntensityAnalyzer

Найкращим знайденим рішенням для класифікації тексту на позитивний чи негативний виявилась функція SentimentIntensityAnalyzer з бібліотеки NLTK, яку було визначено вище.

Бібліотека дозволяє визначити настрій сюжету по ключовим словам у тексті, результат роботи моделі – число, від 0 до 1, по кожному з наступних параметрів: positive, negative, neutral. Це число означає, на скільки відсотків текст позитивний, негативний чи нейтральний [6, 7, 9].

## Висновки до розділу

У цьому розділі були розглянути основні технології, які будуть використовуватись у роботі. Визначені їх основні переваги та недоліки. Для виконання задач natural language processing будуть використовуватись бібліотеки NLTK, SpaCy, Google Speech Recognition та метод TF-IDM. Для задач computer vision було обрано метод Tesseract ORC.

# МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

## Згорткові нейронні мережі

Усі розглянути рішення є напереднатренованими нейронними мережами, які мають архітектуру згорткових нейронних мереж.

Згорткові нейронні мережі (convolutional neural network) або коротко CNN були вперше представлені у 1980-х роках Яном ЛеКуном, докторським дослідником інформатики [10]. У своїй роботі ЛеКун спирався на роботу, яка була виконана Кунігіко Фукусімою, вченим з Японії, який кількома роками раніше відкрив неокогнітрон – базову нейронну мережу для розпізнавання зображень. Перша версія CNN мала назву LeNet, від ім'я ЛеКуна. Вона могла розпізнавати рукописні цифри. CNN вперше використовували у банківських справах та в поштових послугах, де було необхідно розпізнавати цифри на чеках та зчитувати поштові індекси. Аж до 2012 року CNN не використовувались масово, оскільки були складні в масштабуванні. Для навчання CNN необхідний великий масив даних та багато обчислювальних ресурсів. У 2012 році завдяки AlexNet почалось повернення до ідеї використовувати CNN, оскільки з’явились великі набори даних. Цим набором данних був ImageNet. ImageNet містить мільйони розмічених зображень. Обчислювальні ресурси також значно зросли з 1980-х років. З цього моменту можна було виконувати завдання комп'ютерного зору, які раніше були неможливі [10, 11].

## Архітектура згорткових нейронних мереж

Згорткові нейронні мережі складаються з величезної кількості шарів штучних нейронів. Поведінка кожного нейрона залежить від його ваги. При подачі пікселів штучні нейрони CNN здатні виділяти різні візуальні елементи. При введені зображення в CNN, кожен його шар генерує декілька карт активації. Карти активації виділяють деякі особливості зображення. Кожен з нейронів використовує в якості вхідного сигналу область пікселів, перемножує його вагу на значення кольору, підсумовує їх та запускає через функцію активації. Перший (або як його також називають нижній) шар CNN зазвичай здатний виявити основні ознаки, такі як горизонтальні, вертикальні та діагональні штрихи. Вихід першого шару подається як вхід наступного шару, який спробує виділити більш складні функції, такі як кути та комбінації штрихів. По мірі проходженння по згортковій нейронній мережі шари починають виявляти функції вищого рівня, такі як об’єкти, обличчя тощо [10].

## Шари згорткових нейронних мереж

### Шар згортки

Основною особливістю для згорткової нейронної мережі є згортковий шар, який дає мережі свою назву. Цей шар виконує операцію, яка називається «згортка».

Згортка – це лінійна операція, яка множить набір вагів на вхідні дані подібно до традиційної нейронної мережі. Згортка використовує фільтри, які виконують операції згортки під час сканування вхідних даних щодо його розмірів. Отриманий результат називається картою об’єктів або картою активації [10].

У кожному шарі є набір фільтрів m1. Кількість застосованих на одному етапі фільтрів еквівалентна глибині обсягу вихідних карт функцій. Кожен фільтр виявляє певну особливість у кожному місці на вході. Вихід шару *l* складається з характеристичних карт розміром . I-та карта особливостей, позначена як , обчислюється за форулою.

|  |  |
| --- | --- |
|  | (3.1) |

– матриця зміщення, – фільтр розміру , що з'єднує j-ту карту об'єктів у шарі (*l*-1) з *i*-тою картою об'єктів у шарі.

### Шар об'єднання

Шар об'єднання відповідає за зменшення просторового розміру карт активації. Як правило, вони використовуються після декількох етапів згорткових та нелінійних шарів з метою поступового зменшення обчислювальних ресурсів через мережу, а також мінімізації ймовірності перенавчання. Об'єднуючий шар *l* має два гіперпараметри, просторову протяжність фільтра *F*(*l*) і проміжок *S*(*l*). Він приймає вхідний шар розміром і забезпечує вихідний шар розміром де,

|  |  |
| --- | --- |
|  | (3.2) |

Ключова концепція шару об’єднання полягає у забезпеченні незмінного зменшення розміру шарів, оскільки в завданнях розпізнавання зображень виявлення об’єкта є більш важливим порівняно з точним розташуванням об’єкта. Тому операція об’єднання спрямована на збереження виявлених об’єктів у меншому поданні, і робить це, відкидаючи менш значущі дані ціною просторового стиску.

Існує декілька способів використовувати шар об'єднання, зокрема, максимальне та середнє, де береться максимальне та середнє значення відповідно.

### Повнозв'язний шар

Повнозв`язний шар – це шар, що об'єднує входи, де кожен вхід підключений до всіх нейронів. Повнозв'язні шари зазвичай знаходяться на кінці архітектури CNN і можуть бути використані для оптимізації оцінки класів.

Повністю зв’язані шари в згортковій мережі є практично багатошаровим персептроном, метою якого є відображення активації від поєднання попередніх різних шарів до розподілу ймовірностей класу. Таким чином, вихідний шар багатошарового персептрона матиме виходів. Ключовою відмінністю від стандартного багатошарового персептрону є вхідний рівень, де замість вектора в якості вхідного приймається об'єм активації. В результаті повністю пов'язаний шар визначається наступним чином.

Якщо l-1 шар повнозв'язний то

|  |  |
| --- | --- |
|  | (3.3а) |

Якщо l-1 шар не повнозв'язний то

|  |  |
| --- | --- |
|  | (3.3b) |
|  |  |
|  |  |

### Функції активації

Функції активації визначають результуючі значення нейронів в залежності від результатів зваженої суми входу нейрона і порогового значення. Функція активації перевіряє видане нейроном значення на факт того, чи повинні зовнішні зв'яки розглядати цей нейрон як активований або його можна ігнорувати.

Для навчання нейронних мереж використовують серед популярних функцій активації сигмоїду, гіперболічний тангенс, ReLu та інші.

### Сигмоїда

Функція сигмоїди визначається наступним чином:

|  |  |
| --- | --- |
|  | (3.4) |

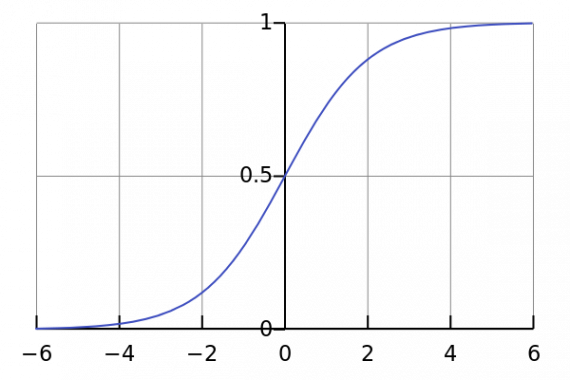


Рисунок 3.1 – Функція сигмоїди

Сигмоїда є нелінійною функцією, також вона небінарна. Сигмоїда є підходящею функцією для задач класифікаціїї. Вона намагається привести значення Y до значення 0 або до 1. Така поведінка дозволяє знаходити чіткі границі при використанні її в нейронних мережах. Інший плюс сигмоїди є її область визначення - від мінус нескінченності до плюс нескінченності. Така властивість сигмоїди не приводить до помилок в випадку великих значень активації. Сьогодні сигмоїда є однією з найбільш частих активаційних функцій в нейронних мережах. Але і у неї є недоліки, на які варто звернути увагу.

При наближенні до кінців сигмоїди значення Y мають тенденцію слабо реагувати на зміни в X. Це означає, що градієнт в таких областях приймає маленькі значення. А це, в свою чергу, призводить до проблем з градієнтим спуском. В такому випадку значення градієнта мале або зникає (не може зробити істотної зміни через надзвичайно мале значення Х). Нейронна мережа відмовляється навчатися далі чи робить це вкрай повільно.

### Гіперболічний тангенс

Ще одна функція активації, яка часто використовується, - це гіперболічний тенгенс.

Гіперболічний тангенс визначається як:

|  |  |
| --- | --- |
|  | (3.5) |

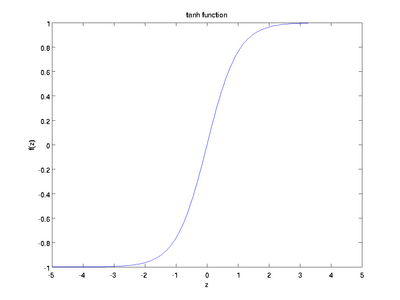


Рисунок 3.2 – Функція гіперболічного тангенсу

Гіперболічний тангенс схожий на сигмоїду, оскільки визначається через неї.

|  |  |
| --- | --- |
|  | (3.6) |

Тому, функція активації гіперболічний тангенс має такі самі характеристики як і сигмоїда. Гіперболічний тангенс нелінійний і має діапазон значень від -1 до 1. Градієнт гіперболічного тангенсу більше, ніж у сигмоїди. При прийнятті рішення про вибір функції активації потрібно звернути увагу на вимоги до амплітуди градієнта. Також, як і сигмоїда, гіперболічний тангенс має проблему зникнення градієнта. Гіперболічний тангенс також є дуже популярною активаційною функцією.

### ReLu

Ще одна функція активації, яка часто використовується, - це ReLu.

ReLu визначається як:

|  |  |
| --- | --- |
|  | (3.7) |

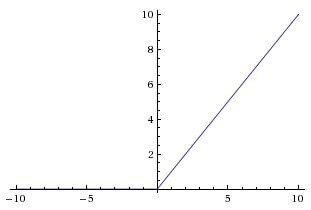


Рисунок 3.3 – Функція ReLu

Фунція ReLu нелінійна, комбінація декількох ReLu - також нелінійна. ReLu є хорошим аппроксіматором, так як будь-яка функція може бути апроксимована комбінацією ReLu. Область допустимих значень ReLu - [0, inf). Через те, що частина ReLu є горизонтальною лінією (для негативних значень X), градієнт на цій частині дорівнює 0. Через рівність нулю градієнта, ваги не будуть корегуватися під час градієнтного спуску. Така ситуація називається проблемою помираючого ReLu (Dying ReLu problem). Через цю проблему деякі нейрони занулюються і не будуть відповідати, роблячи значну частину нейромережі пасивною. Існують варіації ReLu, які допомагають цю проблему вирішити. Наприклад, можна замінити горизонтальну частину функції на лінійну. Якщо вираз для лінійної функції задається виразом *y =* 0.01*x* для області *x* < 0, тоді лінія злегка відхиляється від горизонтального положення. Існують й інші способи уникнути нульового градієнта. Основна ідея - зробити градієнт нерівним нулю.

ReLu менш вимогливий до обчислювальних ресурсів, ніж гіперболічний тангенс або сигмоїда, так як виробляє більш прості математичні операції. Тому має сенс використовувати ReLu при створенні глибоких нейронних мереж.

Використовуючи шари згортки, шари об'єднання, повнозв'язні шари та функції активації можна записати загальна схему роботи згорткової нейронної мережі, що представлена на рисунку 3.4.

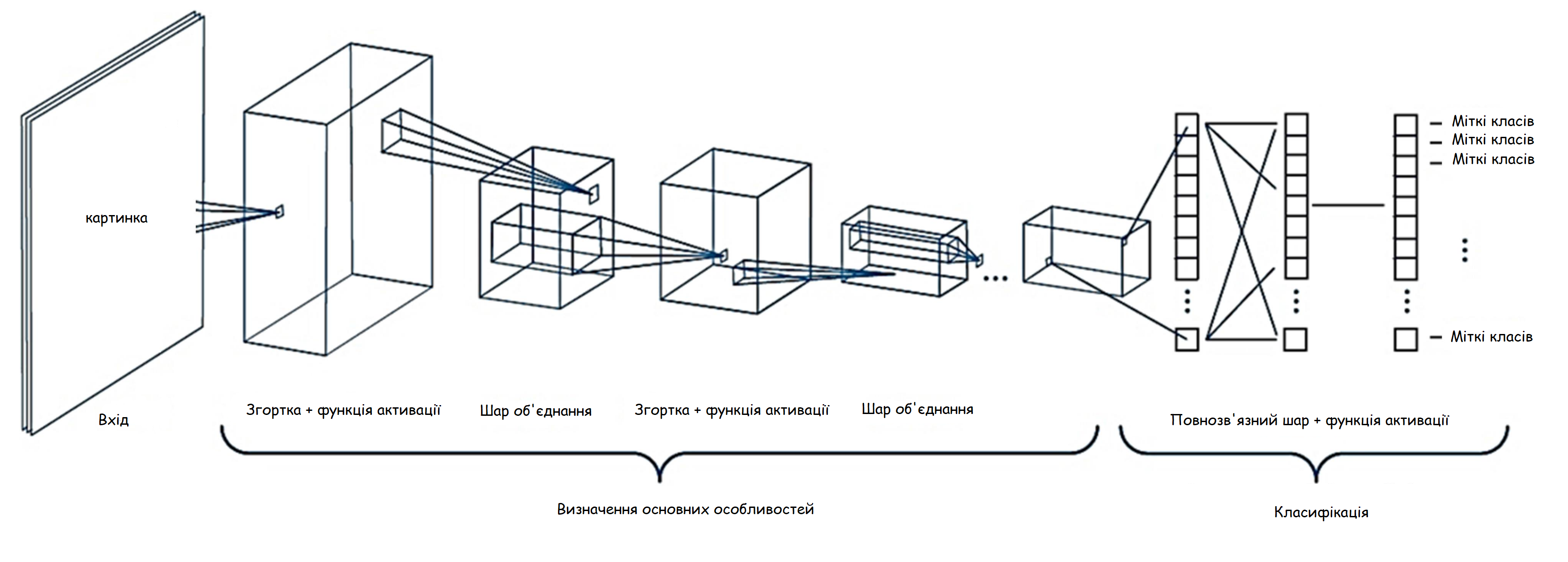


Рисунок 3.4 – Архітектура CNN

## Висновки до розділу

У цьому розділі була розглянута історія розвитку згорткових нейронних мереж та їх архітектура. Також були розглянуті основні складові CNN, а саме: шар згортки, шар об'єднання, повнозв'язний шар. Було розглянуто основні функції активації. Сигмоїда добре показує себе в задачах класифікації. Гіперболічний тангенс також чудово справляється з цією задачею, його відмінність визначається у вимогах до амплітуди градієнта. ReLu, в свою чергу, працює як хороший аппроксиматор.

# ПРОЦЕС ОБРОБКИ ВІДЕОСЮЖЕТУ

## Задача парсингу тексту з використанням Tesseract ORC

Початковий етап процесу обробки відеосюжету – це парсинг відео-потоку. Основна ідея – парсити текст та звук в новинному випуску. В тексті зосереджується ключова інформація, а саме: імена ведучих, назви сюжетів, імена кореспондентів та гостів. Приклад зони, з якої потрібно зчитувати текст, зображено на рисунку 4.1.



Рисунок 4.1 – Зона розташування тексту з іменами ведучих.

Оскільки відео - це набір кадрів, тому необхідно зчитувати текст з вибраної області певну кількість разів за секунду. Для новинних репортажів fps = 30, тому необхідно отримувати текст 30 разів за секунду. Кожний кадр представляє собою 3х-мірний масив даних. Далі цей масив подається на вхід бібліотеки Tesseract ORC, яка виділяє український текст з вибраного кадра. Tesseract - це механізм розпізнавання тексту з відкритим кодом (OCR), доступний за ліцензією Apache 2.0. Поточний офіційний випуск - 4.1.1, який і використовувався в роботі. Tesseract можна використовувати безпосередньо через командний рядок або за допомогою API для вилучення друкованого тексту із зображень. Він підтримує широкий спектр мов. Tesseract не має вбудованого графічного інтерфейсу. За допомогою Tesseract будемо парсити текст з виділеної області та зберігати його в масив, до якого також будемо додавати номер кадру в відеофайлі.

## Розпізнавання тексту з використанням Google Speech Recognition

Розпізнавання мови розпочалось з досліджень, проведенних в Bell Labs на початку 1950-х років. Ранні системи обмежувались одним оратором і мали обмежений словниковий запас близько десятка слів. Сучасні системи розпізнавання мови пройшли довгий шлях від своїх давніх аналогів. Вони можуть розпізнавати мовлення кількох мовців і мають величезний словниковий запас на багатьох мовах. Перший компонент розпізнавання мови - це, звичайно, мова. Мова повинна бути перетворена з фізичного звуку в електричний сигнал за допомогою мікрофона, а потім у цифрові дані за допомогою аналого-цифрового перетворювача. Обробивши кілька моделей, їх можна використовувати для транскрипції звуку в текст. Більшість сучасних систем розпізнавання мовлення покладаються на Hidden Markov Model (HMM). Цей підхід працює на припущенні, що мовленнєвий сигнал при розгляді на досить короткому часовому масштабі може бути розумно апроксимований як стаціонарний процес - тобто процес, в якому статистичні властивості не змінюються з часом. У типовому HMM мовний сигнал розділений на 10-мілісекундні фрагменти. Для декодування мови в текст групи векторів узгоджуються з однією або кількома фонемами - фундаментальною одиницею мови. Цей розрахунок вимагає підготовки, оскільки звук фонеми різниться залежно від динаміка і навіть різниться від одного до іншого висловлювання того самого динаміка. Потім застосовується спеціальний алгоритм для визначення найбільш вірогідного слова (або слів), що виробляють задану послідовність фонем. У багатьох сучасних системах розпізнавання мови нейронні мережі використовуються для спрощення мовного сигналу з використанням методів перетворення ознак та зменшення розмірності перед розпізнаванням HMM. Більш детальна інформація про архітектуру та можливості Google Speech Recognition знаходиться в документації [3].

## Визначення тегів

### TF-IDM

При визначенні тегів важливим моментом є частота входження слів в документ. Якщо слово зустрічається в будь-якому документі часто, при цьому зустрічаючись рідко у всіх інших документах - це слово має велику значимість для того самого документа. У зв'язку з цим вводяться величини: *TF* – частота зустрічі слова у всіх документах; *IDF* – показник кількості документів, в яких зустрічається дане слово. На основі даних величин розраховується нова величина TF-IDM:

|  |  |
| --- | --- |
|  | (4.1) |

Для розрахунку *TF* використовуємо наступну формулу:

|  |  |
| --- | --- |
|  | (4.2) |

де – число входжень слова в документ, а – загальна кількість слів в документі.

Для розрахунку *IDF* використовуємо наступну формулу:

|  |  |
| --- | --- |
|  | (4.3) |

де – кількість документів в базі, а – кількість документів зі словом (при ).

Основа логарифма у формулі не має значення (може бути будь-яке число), оскільки інша основа призведе до зміни вагів кожного окремого слова на сталий коефіцієнт, тобто вагові співвідношення залишаються незмінними.

Більшу вагу *TF-IDF* приймають слова з високою кількістю входжень в один документ та низькою частотою використання в інших документах бази.

### NLTK та SpaCy

NLTK та SpaCy є напереднатреновані дві нейромережі, архітектурою яких є CNN. Завдяки NLTK можна отримати токенізацію текстів, отримати теги, отримати іменовані сутності, відобразити дерево синтаксичного аналізу [7,9,11,12]. Більш детальна інформація про архітектуру та можливості NLTK знаходиться в документації [9].

Бібліотека SpaCy має наступні особливості [13]:

1. підтримує більше 64 мов;
2. має 55 навчених пайплайнів для 17 мов;
3. характеризується багатозадачним навчанням за допомогою попередньо навчених трансформаторів, таких як BERT;
4. має високу швидкість роботи;
5. має токенізацію з мовною мотивацією;
6. має компоненти для розпізнавання іменованих сутностей, позначення частини мови, розбору залежностей, сегментації речень, класифікації тексту , лематизації, морфологічний аналіз, зв’язування сутності тощо;
7. SpaСy легко розширюється за допомогою спеціальних компонентів - PyTorch, TensorFlow та інших фреймворків. Більш детальна інформація про архітектуру та можливості SpaCy знаходиться в документації [8, 13].

### SentimentIntensityAnalyzer для визначення настрою сюжету

SentimentIntensityAnalyzer є функцією з бібліотеки NLTK, яка дозволяє визначити настрій сюжету по ключовим словам у тексті. Результатом роботи моделі є число від 0 до 1, по кожному з наступних параметрів: positive, negative, neutral. Це число означає на скільки відсотків текст позитивний, негативний чи нейтральний.

Модель працює на напереднавченній нейронній мережі, яка аналізує токени в кожному реченні і визначає речення позитивне, негативне чи нейтральне. Для оцінки тексту береться сума параметрів оцінки кожного речення і виводиться фінальний коефіцієнт для всього тексту. Більш детальна інформація про можливості SentimentIntensityAnalyzer знаходиться в документації [9].

## Висновки до розділу

У цьому розділі розглянуто основний процес обробки відеосюжету, розглянуто логіку обробки кадрів та розпізнавання тексту з них. Розглянуто роботу з бібліотекою Google Speech Recognition. Розглянуто логіку виділення тегів завдяки бібліотекам NLTK та SpaCy та роботу функції SentimentIntensityAnalyzer з бібліотеки NLTK.

# ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

## Загальна схема роботи програмного забезпечення

Розглянемо повну схему взаємодії програмних засобів, зображену на рисунку 5.1. З початкового відео-файла виділяється звук та текст. Завдяки тексту з окремих кадрів можна виділити та отримати start-end time кожного сюжета, імена ведучих, кореспондентів та гостей, а також класифікувати сюжет на новини, анонс, рекламу, привітання або прощання.

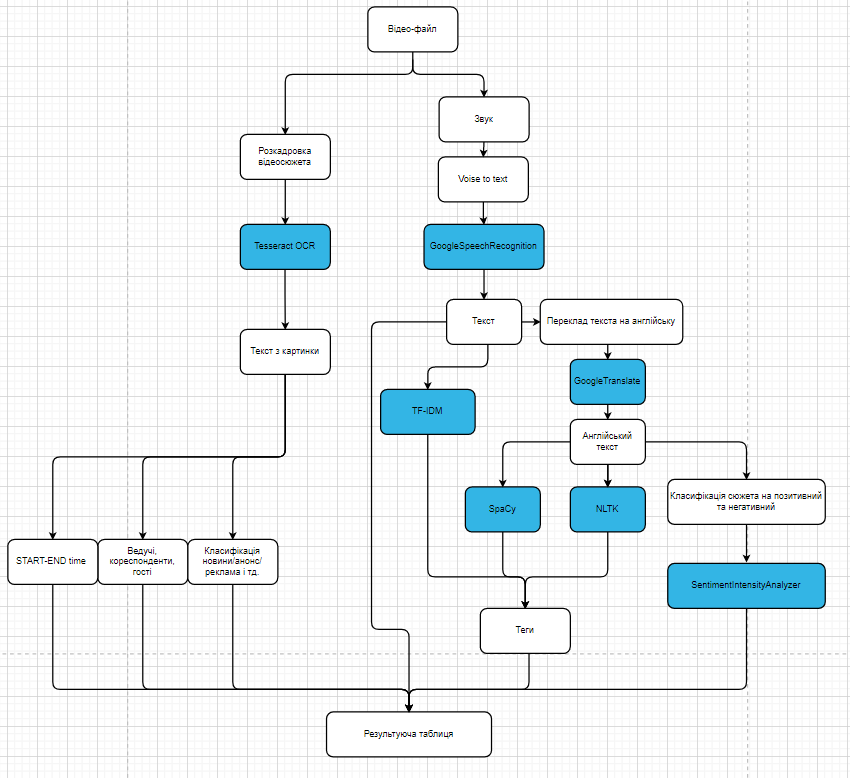


Рисунок 5.1 – Схема взаємодії програмних засобів

Завдяки звуку, виділеного з відео-файлу, можна отримати розпізнанний текст. Цей текст передаємо в результуючу змінну «повний текст». Також текст передаємо в блок TF-IDM, звідки отримуємо першу частину тегів. Далі текст перекладаємо на англійську мову для виділення тегів за допомогою SpaCy та NLTK. Ці три блоки об'єднуємо та записуємо в результуючу змінну «теги». Останній етап – передача англійського тексту на вхід блоку для класифікації сюжету. Після цього записуємо в спеціальну результуючу змінну відсоток, що відповідає позитивному чи негативному сюжету та формуємо результуючу таблицю.

Кодування алгоритму буде відбуватись на мові програмуванні Python 3.8.8. Середа розробки – Jupyter Notebook. Код розробленого програмного забезпечення наведено в додатку А.

## Обробка відеопотоку

Основна ідея обробки відеопотоку – парсити текст та звук в новинному випуску. Почнемо з тексту. В тексті зосереджується ключова інформація, а саме: імена ведучих, назви сюжетів, імена кореспондетів та гостів. Приклад зони, з якої потрібно зчитувати текст, зображено на рисунку 5.2.



Рисунок 5.2 – Зона кадру, з якої необхідно зчитувати текст

Для початку нам необхідно опрацювати відеофайл. Для цього будемо використовувати бібліотеку av [14].

Щоб завантажити відеофайл, будемо використовувати функцію open() з бібліотеки av. Далі, щоб отримати серію потоків із заданого пакета, використовуємо функцію demux() з бібліотеки av. Наступний фрагмент коду завантажує відеофайл і розбиває його на серію контейнерів.

import av

road = "C:/Users/lysyi/Desktop/TRKu/maintest/video/"

name = "Ukraina\_110\_20210206\_190000\_20210206\_200000\_archive.110.1612630800.mp4"

file = road + name

input\_container = av.open(file)

input\_packets = input\_container.demux()

Далі необхідно обробити кожен кадр відеопотоку та конвертувати текст, отриманий із кадру, в текст. Для кожного існуючого пакету із відео використовуємо функцію decode() з бібліотеки av. Кожний кадр представляє собою 3х-мірний масив даних. Конвертуємо зображення av в формат numpy array. Для цього використовуємо функцію to\_nd\_array(). Далі необхідно вибрати лише необхідну нам область з картинки, а саме область, зображену на рисунку 5.3.



Рисунок 5.3 – Зона кадру, з якої необхідно зчитувати текст

Далі встановлюємо змінну myfilter = 30, що означає, що ми будемо обробляти за 1 раз 30 кадрів. Відеофайли мають fps 30 кадрів на секунду, тому за 1 прогонку цикла ми будемо проводити обробку 1 секунди відеофайла. Оскільки текст не встигає змінитись з такою швидкістю, то зчитувати ми будемо 30 кадрів за секунду, але парсити будем кожен третій кадр, щоб збільшити продуктивність. Для парсингу текста з картинки будемо використовувати бібліотеку Pytesseract. За допомогою Tesseract будемо парсити текст з виділеної області та зберігати його в масив, до якого також будем додавати номер кадру в відеофайлі [1].

Описану вище послідовність дії реалізує наступний фрагмент коду:

for packet in input\_packets:

if isinstance(packet.stream, VideoStream):

frames = packet.decode()

for raw\_frame in frames:

array\_list += [frame.to\_nd\_array()[320:365,187:575]]

check +=1

if myfilter < check:

if m%3==0:

var1 = str(pytesseract.image\_to\_string(array\_list[m], lang = "ukr")).split('\n\n \n\x0c')[0].replace('#', '').replace('#', '').replace('%', '').replace('\*', '').replace('!', '').replace('@', '').replace('/', '').replace("'",'"').replace('|','').replace('\n', '').replace(' ', '').replace('\x0c', '').replace('.', '').replace(',', '').replace('(', '').replace(')', '').upper()

var2 = str(pytesseract.image\_to\_string(array\_list[m], lang = "ukr")).replace('#', '').replace('%', '').replace('\*', '').replace('!', '').replace('@', '').replace('/', '').replace('|','').replace('\n', '').replace('\x0c', '').replace('.', '').replace(',', '').replace("'",'"').replace('(', '').replace(')', '').upper()

if var1 != '' and var1 != '\x0c' and var1 != ' \n\x0c' and var1 != '\n\x0c' and var1 != ' ':

mass = [str(check2+m), var1, var2]

for k in mass:

output.append([k[0], k[1], k[2]])

answer\_mass = []

mass = []

check2 += check

check = 0

array\_list = []

На виході отримуємо массив, в якому збережений номер кадру, текст з картинки з пробілами та текст без пробілів. Приклад одного елементу масиву зображений на рисунку 5.4. Тут присутні номер кадру, текст з картинки без пробілів та текст картинки з пробілами.



Рисунок 5.4 – Елемент масива з номером кадра та текстом

Для годинного випуску новин створення масиву кадрів та тексту займає приблизно 6 годин. Код можна покращити, запустивши процеси на кожному ядрі ОЗП окремо. Для цього будемо використовувати бібліотеку multiprocessing. Multiprocessing - це пакет, який підтримує паралельні процеси за допомогою API. Multiprocessing пакет пропонує як локальну, так і віддалену паралельність, ефективно крокуючи в сторону глобального блокування інтерпретатора за допомогою підпроцесів замість потоків. Завдяки цьому multiprocessing модуль дозволяє програмісту повністю використовувати кілька процесорів на даній машині. Він працює як на Unix, так і на Windows [15].

Оскільки обробка зображення (виділення тексту) займає найбільшу частку часу, тому винесемо цю частину кода в окрему функцію, та запустимо окремими процесами на кожному із доступних ядер. Наступний фрагмент коду реалізує вказану функцію та запуск на паралельних ядрах процес парсингу відеопотоку.

def processInput(m,check2=check2):

if m%3==0:

var1 = str(pytesseract.image\_to\_string(array\_list[m], lang = "ukr")).split('\n\n \n\x0c')[0].replace('#', '').replace('#', '').replace('%', '').replace('\*', '').replace('!', '').replace('@', '').replace('/', '').replace("'",'"').replace('|','').replace('\n', '').replace(' ', '').replace('\x0c', '').replace('.', '').replace(',', '').replace('(', '').replace(')', '').upper()

var2 = str(pytesseract.image\_to\_string(array\_list[m], lang = "ukr")).replace('#', '').replace('%', '').replace('\*', '').replace('!', '').replace('@', '').replace('/', '').replace('|','').replace('\n', '').replace('\x0c', '').replace('.', '').replace(',', '').replace("'",'"').replace('(', '').replace(')', '').upper()

if var1 != '' and var1 != '\x0c' and var1 != ' \n\x0c' and var1 != '\n\x0c' and var1 != ' ':

text = [str(check2+m), var1, var2]

return text

for packet in input\_packets:

if isinstance(packet.stream, VideoStream):

frames = packet.decode()

for raw\_frame in frames:

array\_list += [frame.to\_nd\_array()[320:365,187:575]]

check +=1

if myfilter < check:

mass = Parallel(n\_jobs=-1)(delayed(processInput)(m,check2=check2) for m in range(0,len(array\_list)))

mass = [x for x in mass if x is not None]

for k in mass:

output.append([k[0], k[1], k[2]])

answer\_mass = []

mass = []

check2 += check

check = 0

array\_list = []

Після цієї оптимізації ми отруємо код, що працює в 6 разів швидше, ніж початковий - одна година новинного випуску оброблюється за одну годину з точністю до 5 хвилин.

В результаті отримаємо масив, фрагмент якого зображений на рисунку 5.5.

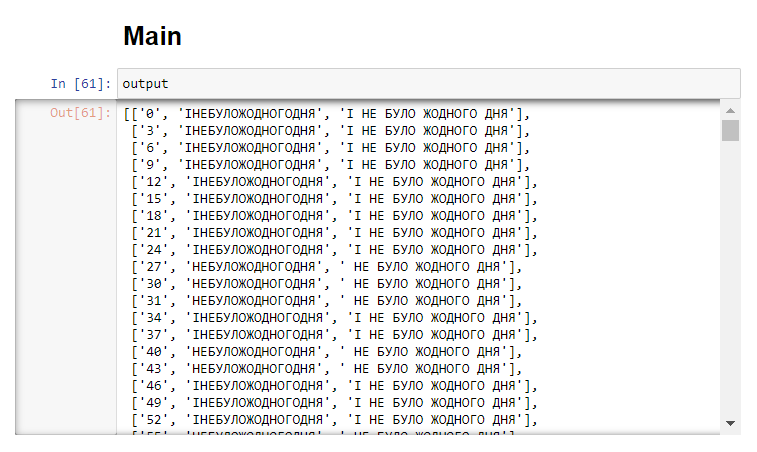


Рисунок 5.5 – Масив з номерами кадрів та відповідним текстом

Наступним етапом буде обробка повторень та виділення повноцінних окремих частин кожного окремого сюжета - виділення початку сюжета, виділення кореспондентів, ведучих та гостей.

## Обробка масиву кадрів

Для обробки кадрів об'єднаємо кадри, що містять однаковий текст. Оскільки текст може мати незначні похибки в написанні, через помилку обробки бібліотекою Tesseract ORC, необхідно фільтрувати помилки.

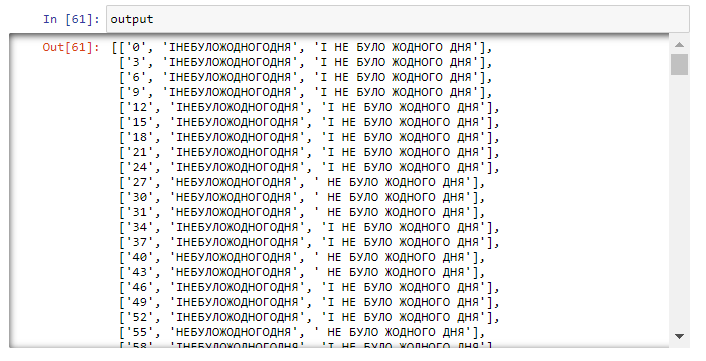


Рисунок 5.6 – Можливі відмінності при віделенні текста

Для оцінки подібності тексту будемо використовувати метрику Джаро-Вінклера.

Подібність Джаро між двома рядками *l*1 та *l*2 визначається як:

– довжина  *i*-го рядка;

*n* – кількість співпадінь символів;

*t* – кількість транспозицій.

Два символи від *l*1 і *l*2 відповідно, вважаються співпадінням тільки тоді, коли вони однакові і розташовані не далі ніж один від одного.

Метрика Джаро-Вінклера представлена в бібліотеці jellyfish методом jaro\_distance. Детальніше про бібліотеку jellyfish можна дізнатись в документації [16].

Сюжетний ефір завжди має 3 частини – анонс, основний ефір та рекламна частина. Виділення анонсу знаходимо шляхом співставлення кожного кадру з кадром заставки.

Перед рекламною частиною в тексті буде знаходитись напис ОТОВ«НОВИННАГРУПАУКРАЇНА». Після виділення цього напису алгоритм буде розуміти, що наступна частина є рекламною частиною.

Також додамо напис «Привітання» та «Прощання» на основі тексту з відеопотоку.

Основний ефір – це решта відеофайлу. Доповнюємо результуючий масив четвертим елементом, який буде відображати тип новин, а саме: реклама, анонс, новини, заставка або прощання.

Для виявлення заставки необхідно аналізувати та знаходити картинку заставки.

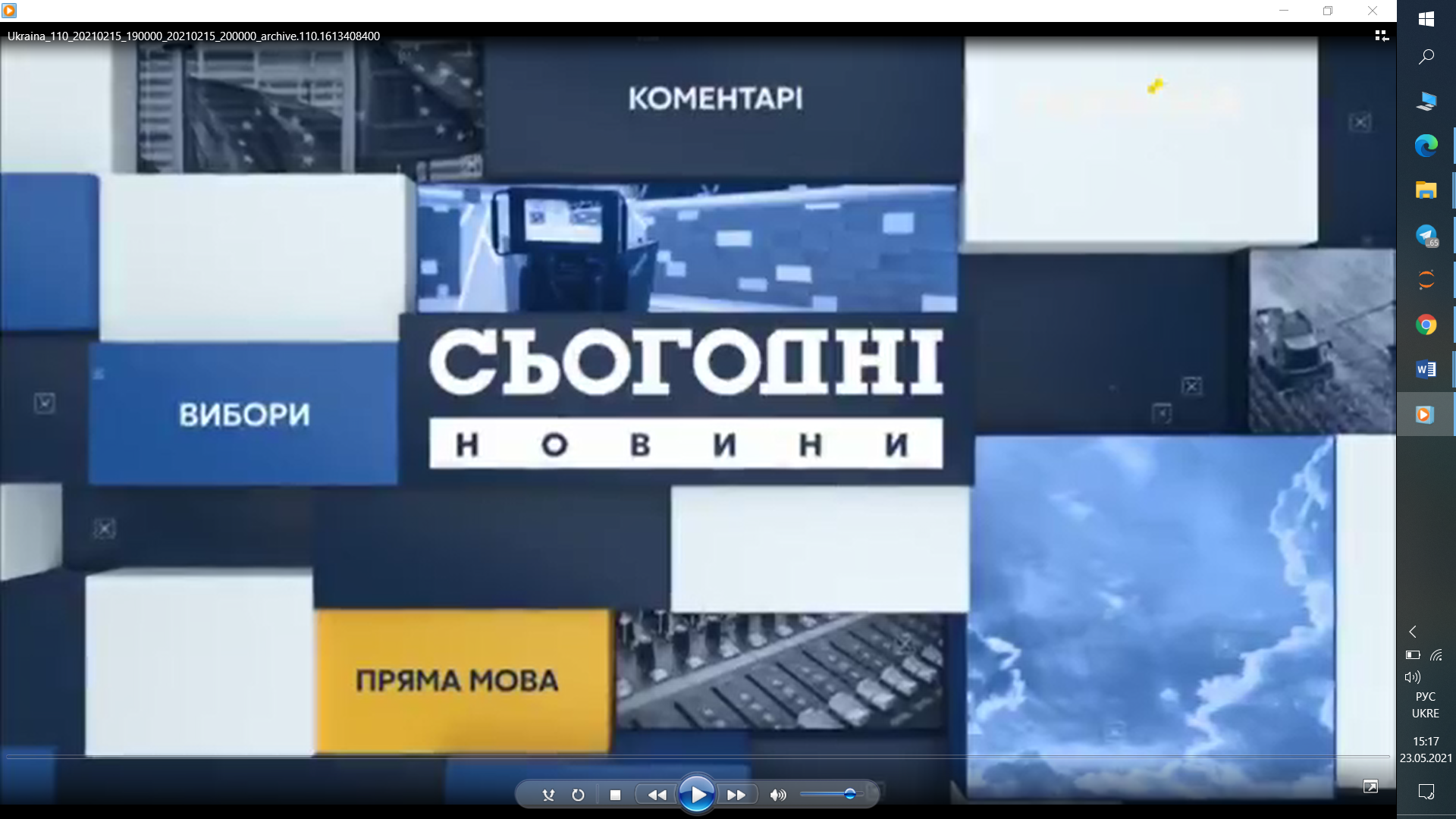


Рисунок 5.7 – Зображення заставки

Так само визначаються анонси. Анонси в новинному випуску – це короткий огляд новинних репортажів, що будуть розглядатися у випуску. Зазвичивай анонси займають від 5 до 15 секунд для кожної новини. Анонси, як і повноцінні новинні сюжети, мають напис внизу, тому назву сюжету в анонсі можна виділити. Xарактерною особливістю анонсів є спеціальний кадр, після якого починаються анонси, як це показано на рисунку 5.8.

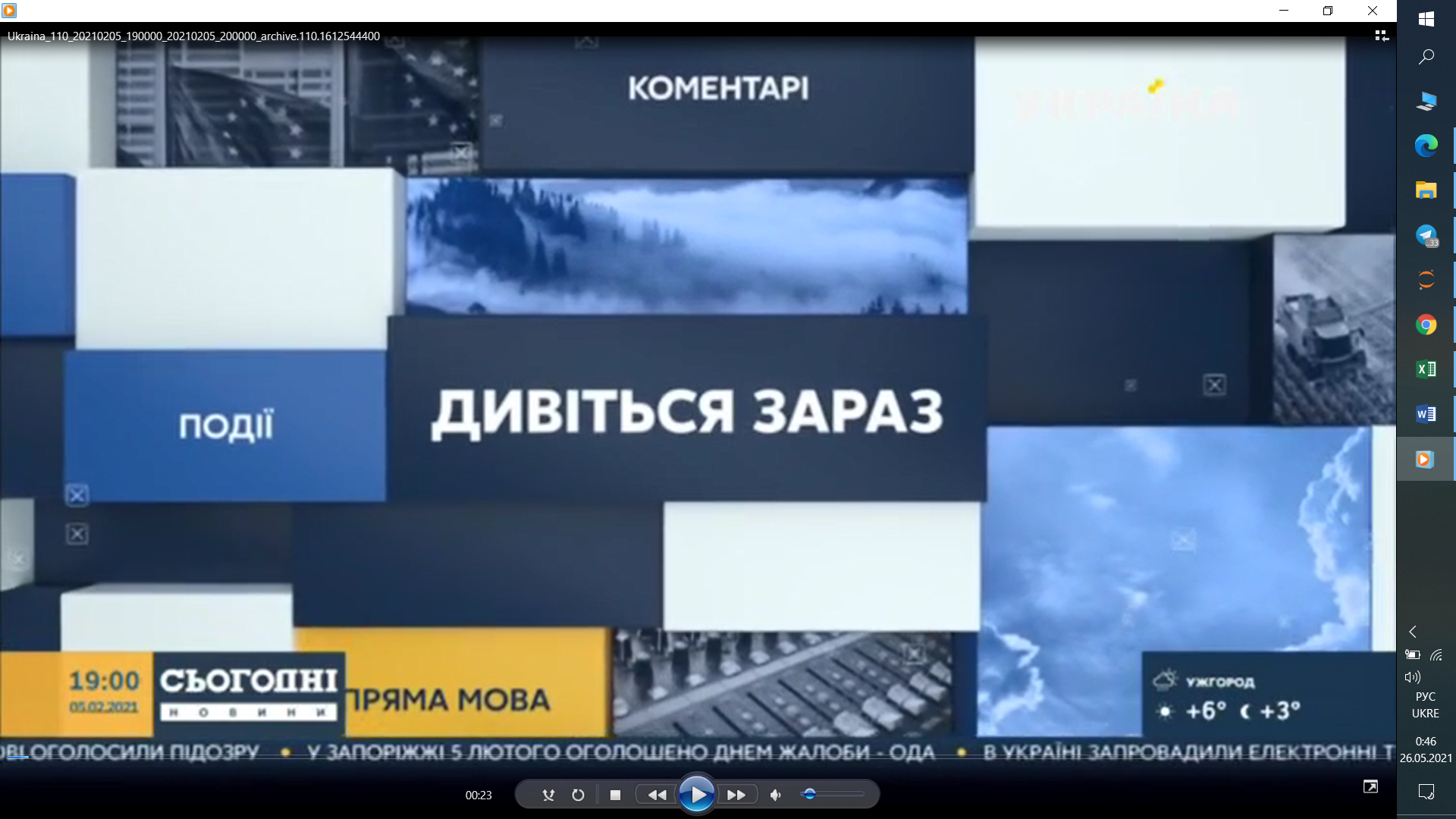


Рисунок 5.8 – Кадр, після якого починаються анонси

Так само, як і при виділенні кадру заставки, необхідно перевіряти та порівнювати кожен кадр. Для цього будемо використовувати різницю по модулю кожного кадру із заданим для перевірки. Це можна зробити за допомогою функції fabs() з бібліотеки math. Дослідницьким шляхом було встановлено, що величина допустимого відхилення при відніманні двох кадрів не може перевищувати 8. Якщо це виконується, то додаємо до результуючого масива наступні кадри з типом «Анонс». Анонс закінчується появою в відеопотоці такого ж самого кадру як і на рисунку 5.8. Після цього наступні новинні репортажі мають тип «Новини». Код для виявлення анонсів зображений нижче:

if math.fabs(hash0 - hash1) < 8 or math.fabs(hash3 - hash1) < 8:

output.append([str(index), ‘text1’, ‘text2’, 'АНОНС', 'АНОНС'])

Результат виконання коду представлений на рисунку 5.9.

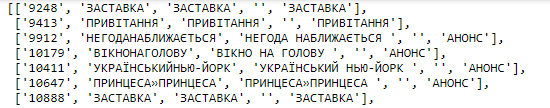


Рисунок 5.9 – Результат виконання коду після виявлення заставки та анонсів

Також до результуючого масиву додаємо кореспондентів та гостей. Написи кореспондентів та гостей з'являються на екрані між написами сюжетів, наприклад: «Маско хто ти?» - «Оксана Байрак» - «Маско хто ти?». Тоді кореспондентів та гостей додаємо до масиву. Приклад частини масиву результата, що містить кореспонднтів та гостей, зображений на рисунку 5.10.



Рисунок 5.10 – Результат виконання коду після виявлення кореспондентів та гостей

Наступна проблема – визначити імена ведучих. Імена ведучих з'являються на екрані під час привітання на початку новинного випуску. Приклад появи зони з іменами ведучих зображений на рисунку 5.11.

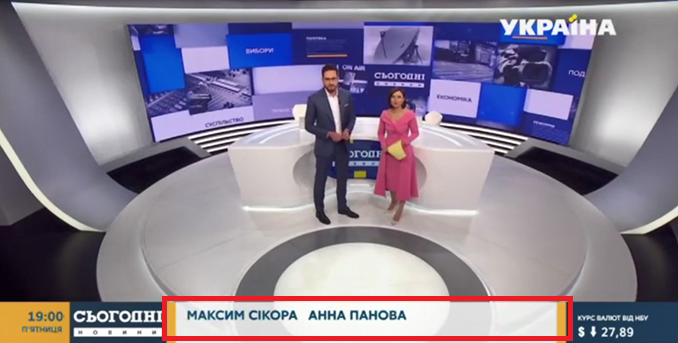


Рисунок 5.11 – Зона кадру з іменами ведучих

Для визначення імен ведучих серед тексту новинних заголовків, кореспондентів, анонсів та реклами необхідно аналізувати текст та порівнювати його з базою ведучих. Ведучі зазвичай рідко змінюються, а при необхідності базу ведучих можна буде доповнити новими іменами. База ведучих для телеканалу «Україна» містить 6 ведучих, зокрема: Анна Панова, Максим Сікора, Олена Кот, Олег Панюта, Ігор Пупков та Ольга Гагрицик.

Для виявлення ведучих та частини в новинному сюжеті, що містить привітання, необхідно порівнювати кожен кадр з розпізаним текстом в базі ведучих. Після цього додаємо до основного циклу обробки відеосюжету перевірку на входження розпізнаного тексту в представленний масив. При виявленні імені ведучого зберігаємо його в окрему змінну. Масиву, що містить кадр, текст з пробілами та без пробілів, присвоюємо тип «Привітання». Код для виявлення ведучих та присвоєння відеофрагменту тип «Привітання» зображений нижче:

for name in leading:

if name == first\_part[i][1] or mm in first\_part[i][1]:

news\_anchor\_mass += name

first\_part[i][1] = 'ПРИВІТАННЯ'

first\_part[i][2] = 'ПРИВІТАННЯ'

first\_part[i][4] = 'ПРИВІТАННЯ'

Результат виконання коду представлений на рисунку 5.12.



Рисунок 5.12 – Результат виконання коду для виявлення ведучих та присвоєння відеофрагменту тип «Привітання»

## Отримання start-end time кожного сюжету

Масив результату містить наступні елементи: номер кадру, текст без пробілів, текст з пробілами, ведучі або гості, тип сюжету. Для отримання start-end time кожного сюжету необхідно конвертувати номер кадру в час від початку відеофайлу та додати до нього час виходу програми в ефір.

Відеофайл має 30fps, тому для конвертування номеру кадру в час початку окремого сюжету неохідно номер кадру поділити на 30fps. Після цього ми отримаємо результат в секундах. Для отримання результату в форматі хвилин та годин необхідно від отриманого значення взяти остачу та цілу частину при діленні на 60 відповідно. Після цього необхідно об'єднати секунди, хвилини та години в одну змінну. Код для отримання start-end time кожного сюжету зображений нижче:

number\_of\_frames = 30

for i in range(0,len(input)):

hours = str(int((int(input[i][0]))/number\_of\_frames/60/60/60))

minutes = str(int(int(input[i][0])/number\_of\_frames)//60)

seconds = str(int(int(input[i][0])/number\_of\_frames%60))

if len(minutes) == 1:

minutes = '0' + str(minutes)

if len(seconds) == 1:

seconds = '0' + str(seconds)

end\_time = str(hours)+':'+ str(minutes)+':'+str(seconds)

## Отримання хронометражу

Для отримання хронометражу кожної частини відеофайлу необхідно відняти від номера кадра кінця сюжета номер кадра початку сюжета. Отриманий результат показуватиме хронометраж сюжету в кадрах. Відеофайл має 30fps, тому для конвертування хронометражу в кадрах в хронометраж в секундах неохідно хронометраж поділити на 30fps. Після цього ми отримаємо результат в секундах. Для отримання результату в форматі хвилин та годин необхідно від отриманого значення взяти остачу та цілу частину при діленні на 60 відповідно. Після цього необхідно об'єднати секунди, хвилини та години в одну змінну. Код для отримання хронометражу кожного сюжету зображений нижче:

duration = int(input[i+1][0]) - int(input[i][0])

hours = str(int((int(duration))/number\_of\_frames/60/60/60))

minutes = str(int(duration)/number\_of\_frames)//60)

seconds = str(int(int(duration)/number\_of\_frames%60))

if len(minutes) == 1:

minutes = '0' + str(minutes)

if len(seconds) == 1:

seconds = '0' + str(seconds)

duration\_seconds = str(hours)+':'+ str(minutes)+':'+str(seconds)

## Отримання звукової доріжки з відеофайлу

Для отримання звукової доріжки з відеофайлу використовуємо бібліотеку subprocess. Модуль subprocess дозволяє підключатись до консолі з програми Python. Детальніше про бібліотеку subprocess можна дізнатись в документації [17].

Виділяємо звукові доріжки за start-end time кожного окремого сюжету. Subprocess має метод call(), який використовується для запуску консольної команди.

В консольній команді використовується FFmpeg. FFmpeg - це безкоштовний програмний проект із відкритим кодом, що складається з великого набору бібліотек та програм для обробки відео, аудіо та інших мультимедійних файлів та потоків. Її основою є сама програма FFmpeg, призначена для обробки відео та аудіофайлів на основі командного рядка. FFmpeg широко використовується для перекодування формату, базового редагування (обрізка та конкатенація), масштабування відео та додання ефектів постпродукції відео. Детальніше про продукт FFmpeg можна дізнатись в документації [18]. Код виділення окремих звукових доріжок наведено нижче:

import subprocess

for i in range(0,len(input)-1):

command = "ffmpeg -i "+file+" -ss "+start\_time+" -t "+end\_time+"-q:a 0 -map a video/sample"+str(i)+".wav"

subprocess.call(command, shell=True)

## Отримання тексту зі звукової доріжки

Насамперед імпортуємо бібліотеку Speech\_recognition та викликаємо з неї метод Recognizer(). Після цього передаємо звукову доріжку в метод AudioFile(). Далі використовуємо метод adjust\_for\_ambient\_noise(), щоб прибрати шуми із звукової доріжки. Отриманий текст перекладаємо на англійську, оскільки для виділення тегів та настрою сюжету напереднавченні нейронні мережі використовують англійську мову. Для цього використовуємо google\_translator() з бібліотеки google\_trans\_new [3]. Нижче наведено фрагмент коду, який реалізує вказану послідовність дій:

from google\_trans\_new import google\_translator

import speech\_recognition as sr

r = sr.Recognizer()

harvard = sr.AudioFile(road+'sample'+str(i)+'.wav')

while 1:

with harvard as source:

r.adjust\_for\_ambient\_noise(source)

audio = r.record(source)

text = r.recognize\_google(audio, language="uk-UA")

translator = google\_translator()

english\_text = translator.translate(text,lang\_tgt='en')

## Отримання тегів

Для отримання тегів використовується об'єднання бібліотек SpaCy та NLTK та методу TF-IDM.

### TF-IDM

Щоб використовувати метод TF-IDM необхідно завантажити базу статей в корпус системи. Далі запрограмуємо формули (4.1), (4.2), (4.3) в функції та завернемо весь блок в одну функцію. Код для отримання тегів за допомогою метрики TF-IDM представлено нижче:

#tf\_idm

database.append(english\_text)

length = 0

for l in database:

length+=len(l)

print(length)

corpus = []

stop\_words = set(stopwords.words('english'))

for text\_block in database:

data = []

for jj in text\_block.lower().replace('.','').replace(',','').replace('-','').split(' '):

data.append(compare\_stemmer\_and\_lemmatizer(PorterStemmer(), WordNetLemmatizer(), word = lemmatizer.lemmatize(jj), pos = wordnet.VERB))

without\_stop\_words = [word for word in data if not word in stop\_words]

corpus.append(without\_stop\_words)

length = 0

for l in corpus:

length+=len(l)

print(length)

vocabulary = list(compute\_tfidf(corpus)[0].items())

tf\_idm = sorted(vocabulary, key=lambda tup: tup[1], reverse = True)

print('\ntf\_idm')

#tf\_idm end

### Застосування SpaCy

Бібліотека SpaCy може виділяти власні імена, назви брендів, час, дати та інші ключові слова [8]. Оскільки час, дати та цифри є менш характерними для опису тексту, тому видалимо їх з виділених тегів, а залишимо лише власні імена. Код виділення ключових слів, а саме, власних назв, зображений нижче:

while 1:

LowPrior = []

delimiter2 += 1

try:

for ent in doc.ents:

if ent.label\_ !='DATE' and ent.label\_ !='ORDINAL' and ent.label\_ !='CARDINAL' and ent.label\_ !='TIME' and ent.label\_ !='QUANTITY':

LowPrior.append(ent.text.lower())

except Exception as e: print(e)

if delimiter2 == 10 or len(LowPrior) != 0:

break

### Застосуваня NLTK

Бібліотека NLTK необхідна для виділення нормалізованих слів з тексту [9]. Необхідно розбити весь текст на слова та нормалізувати їх. Для цього будемо використовувати метод WordNetLemmatizer() з бібліотеки NLTK. Далі завантажуємо в корпус базу текстів, прибираємо всі символи (коми, крапки, крапки з комою, лапки і т.д.) та нормалізуємо слова. Для очистки тексту від зайвих символів використоується функція sub() з бібліотки re.

Для розбиття тексту на слова будемо використовувати метод split() для текстових даних. В лапки необхідно передати символ, по якому необхідно розбивати текст. У нашому випадку - це пробіл. Код розбиття тексту на слова та нормалізація цих слів зображений нижче:

lemmatizer = WordNetLemmatizer()

text = text.split(‘ ‘)

mass\_text = [lemmatizer.lemmatize(j) for j in text]

## Класифікація сюжету на позитивний чи негативний

Для класифікації сюжету необхідно виконати наступну послідовнісь дій. Імпортувати функцію SentimentIntensityAnalyzer() з бібліотки NLTK [9]. Створити об'єкт класу SentimentIntensityAnalyzer() та оримати настрій сюжету методом polarity\_scores(text), де замість змінної text вибрати весь текст з сюжету. Також необхідно записати текст в файл csv. Наступний фрагмент коду реалізує вказану послідовність дій:

from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

mood = sid.polarity\_scores(text)

data14 += 'neg:' + str(ss['neg']) + ' pos:' + str(ss['pos'])

sheet['O'+str(i+k)] = data14

## Запис отриманих даних в Excel-файл

Останній етап роботи програмного забезпечення – запис результуючої таблиці в файл csv. Для цього будемо використовувати бібліотеку openpyxl [19]. Для завантаження файлу використовуємо функцію load\_workbook(). Обираємо сторінку з необхідною нам назвою та починаємо записувати дані построково. Для присвоєння комірці ексель данних необхідно вказати колонку та рядок клітинки, в яку ми бажаємо записти данні, та самі дані. Наприклад, для запису слова «text» в комірку A1 необхідно написати :

sheet['A1')] = «text»

Таким чином, створюємо цикл, що проходить по всій стореній базі новинних сюжетів та переносить результат в таблицю Excel. Після змін вихідного файлу Excel його необхідно зберігти. Для зберігання результату запису даних в Excel-таблицю використовуємо функцію save(). Повний код запису результуючої таблиці в файл csv наведений нижче:

for i in range(0,len(input)-1):

xfile = openpyxl.load\_workbook('my.xlsx')

sheet = xfile.get\_sheet\_by\_name('Sheet1')

sheet['A'+str(i+k)] = week

sheet['B'+str(i+k)] = programm

sheet['C'+str(i+k)] = day

sheet['D'+str(i+k)] = data

sheet['E'+str(i+k)] = channel

….

xfile.save('my.xlsx')

Фрагменти csv-файлу – результат обробки одного новинного випуску – представлено на рисунку 5.13 та 5.14. На рисунку 5.13 наведено тиждень, програму, день тижня, дату, канал, start-end time кожного сюжету, хронометраж, назву сюжету, текст сюжету. На рисунку 5.14. наведено тип сюжету (анонс, новини, реклама, привітання або прощання), імена ведучих та гостей, визначено тему сюжету за ключовими тегами та визначено частку позитивного та негативного тексту в сюжеті.

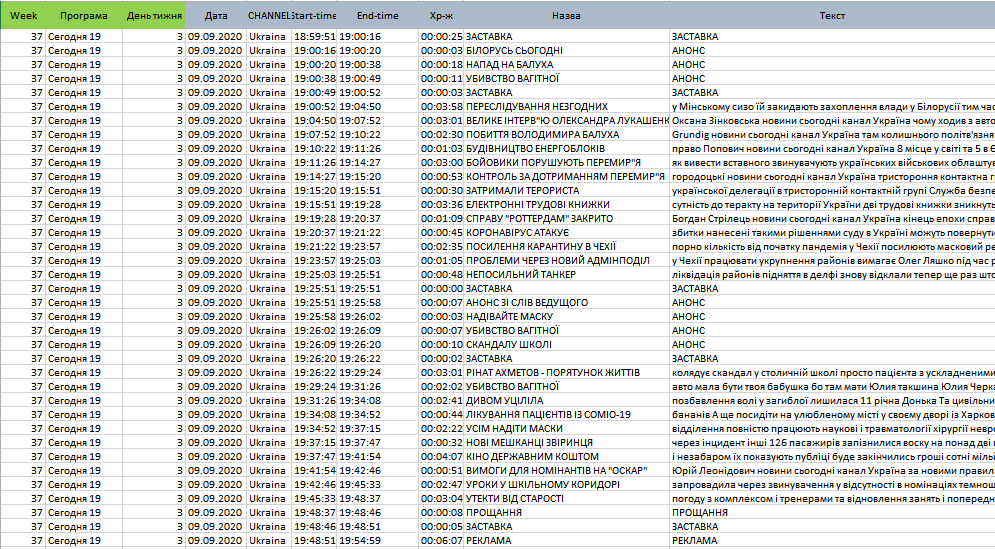


Рисунок 5.13 – Фрагмент А результуючого csv-файлу

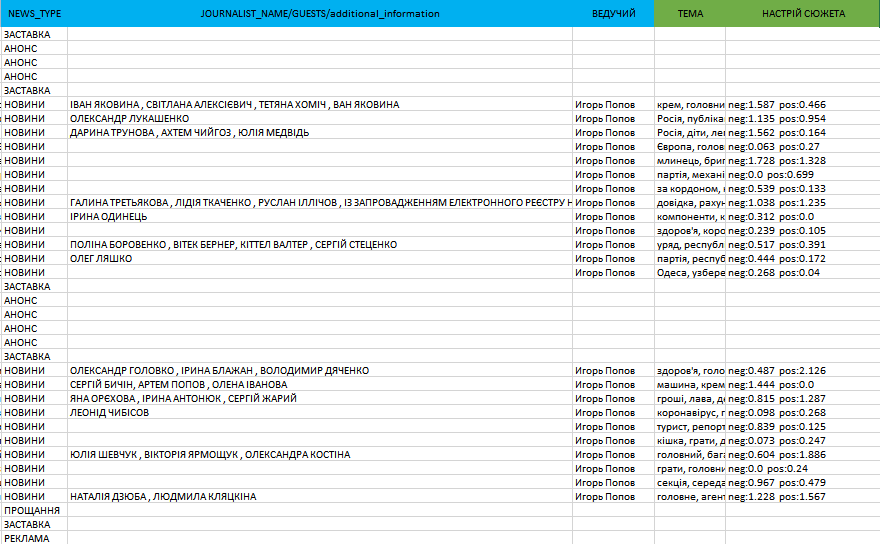


Рисунок 5.14 – Фрагмент B результуючого csv-файлу

## Показники точності програмного забезпечення

Фінальна csv-таблиця містить наступні поля: порядковий номер тижня, програму, день тижня, дату, телеканал, час початку новинного сюжета, час кінця новинного сюжета, хронометраж, текст сюжета, тип новин, ім'я журналістів та гостей, ведучих, теги (тему) та класифікації сюжету на позитивний та негативний.

За допомогою розробленого програмного забезпечення було опрацьовано більше 80 годин новинних відеоефірів. Отримані результати були порівняні з аналогічними, одержаними вручну. Як показало співставлення цих результатів, досягнуто таку точність при визначені полів:

* Тиждень – 100% точності;
* Програма – 100% точності;
* День тижня – 100% точності;
* Дата – 100% точності;
* Телеканал – 100% точності;
* Початок новинного сюжету – 96.72% точності. Для тестування були відібрані новинні випуски за один місяць. З точністю до 1 секунди правильно визначений час початку сюжету в 96,72% новинних репортажів. Помилки з'являються в результаті появи тексту з затримкою на екрані;
* Кінець новинного сюжету – 96.72% точності. Для тестування були відібрані новинні випуски за один місяць. З точністю до 1 секунди правильно визначений час початку сюжету в 96,72% новинних репортажів. Помилки з'являються в результаті появи тексту з затримкою на екрані;
* Хронометраж – 100% точності;
* Текст сюжета – 95% тексту розпізнається правильно, помилки виникають при роботі з англійською та російською мовами, адже початковий алгоритм настроєний на роботу з українською мовою. Також при зйомках на відкритому просторі виникає багато шумів в записі, в таких випадках точність розпізнавання тексту зменшується до 60%;
* Тип новин – 100% точності;
* Імена журналістів, гостей та ведучих – 100% точності;
* Теги – не існує однозначної метрики оцінки якості тегів, оскільки в мові існує велика кількість синонімів, тому ключові слова в кожному тексті можна представити по-різному. В цілому, теги повністю розкривають тему сюжета, проте інколи до тегів можуть попадати загальні слова, які не несуть характерної інформації, хоча є важливими, наприклад: людина, погода, тварина, тощо. Рівень відповідності тегів становить приблизно 85%;
* Класифікація сюжету на позитивний чи негативний – не існує однозначної метрики оцінки якості класифікації сюжетів на позитивний чи негативний. Було проведено тестування, в якому реальні учасники повинні були оцінити рівень позитиву чи негативу у сюжеті. Деякі сюжети було важко оцінити, наприклад: сюжет про в'язниці, в яких в'язні отримували другий шанс на соціальне життя. Учасники тестування в середньому виявили 80% позитиву у сюжеті та 20% негативу, в той час, як класифікатор вказав на значення 50% на 50%. Рівень відповідності класифікації сюжету на позитивний чи негативний складає 90%.

Таблиця 5.1 – Показники точності програмного забезпечення

|  |  |  |  |
| --- | --- | --- | --- |
| Тиждень | 100% | Хронометраж | 100% |
| Програма | 100% | Текст сюжета | 95% |
| День тижня | 100% | Тип новини | 100% |
| Дата | 100% | Ім'я журналістів та гостей | 100% |
| Телеканал | 100% | Ведучий | 100% |
| Початок новинного сюжету | 96,72% | Теги | 85% |
| Кінець новинного сюжету | 96,72% | Класифікація сюжету на позитивний чи негативний | 90% |

## Висновки до розділу

У цьому розділі було описано загальну схему роботи програмного забезпечення для обробки й класифікації архіву новинних відеорепортажів. Було представлено схему взаємодії програмних засобів, прописана логіка обробки відеопотоку, було представлено детальний опис отримання тексту з відеосюжету, були прописані правила, завдяки яким визначаються ведучі, журналісти, гості, визначається тип сюжету (новини, анонс, реклама і т.д.), визначено алгоритм отримання start-end time кожного сюжету, хронометражу кожного сюжету, одержання тегів за допомогою TF-IDM, SpaCy та NLTK. Сформульована послідовність дій для класифікації сюжету на позитивний чи негативний та запис результату в Excel-файл.

За допомогою розробленого програмного забезпечення було опрацьовано більше 80 годин відеорепортажів. Порівняння отриманих даних з аналогічними, одержаними вручну, показало, що за більшістю вимірюваних параметрів точність становить 100%. Точність визначення тегів становить 85%, що обумовлено включенням загальних слів у результат. Точність визначення повного тексту сюжету складає 95% через помилки в розпізнаванні іноземних мов та фонових шумів. Точність класифікації сюжету на позитивний чи негативний становить 90%.

# ВИСНОВКИ

У рамках дипломного проектування було створено програмне забезпечення для обробки архіву новинних відеорепортажів, яке дозволяє визначити start-end time кожного окремо сюжету, його хронометраж, визначити назву сюжету, виділити тип (новини, анонс, реклама та інші), визначити ведучих, журналістів та гостей, визначити текст і теги сюжету та встановити його настрій – позитивний чи негативний.

Для розв'язання поставленої задачі в роботі були використані методи computer vision (CV) та natural language processing (NLP). Було проаналізовано математичний апарат згорткових нейронних мереж, що використовуються в CV та NLP, та існуючі програмні рішення.

Розроблено алгоритм обробки окремого відеосюжету, який включає: виділення тексту та його обробку засобами NLP; аналіз звукової доріжки та отримання її у вигляді тексту за допомогою voise to text system; переклад тексту за допомогою Google Speech Recognition; виділення тексту за допомогою computer vision та визначення назви, ведучих, гостей та кореспондентів відеосюжету; визначення тегів за допомогою SpaСy, NLTK та TF-IDF та встановлення на основі отриманої інформації тематики відеосюжету; визначення настрою новинного сюжету – позитивний чи негативний за допомогою функції SentimentIntensityAnalyzer бібліотеки NLTK; збереження усіх отриманих результатів у вигляді csv-файлу.

Спроектований алгоритм реалізовано програмно на мові Python.

За допомогою створеного програмного забезпечення було оброблено більше 80 годин новинних відеорепортажів та проведено порівняння отриманих результатів з аналогічними, одержаними вручну. Точність визначення переважної більшості параметрів становить 100%. Для тексту відеосюжету, тегів та настрою відеосюжету точність їх визначення коливається в межах від 85% до 95%.