Проектування алгоритму:

Початковий етап – це парсинг відео-потоку.Основна ідея – 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 1.



Рис.1 – зона, з якої необхідно зчитувати текст.

Для початку нам необхідно опрацювати відео файл. Розробка алгоритму буде відбуватись на мові програмуванні python 3.8.8. Середа розробки – jupyter notebook. Для початку необхідно завантажити відео в середу розробки jupyter notebook. Для цього будем використовувати бібліотку av (<https://pypi.org/project/av/>).

PyAV - це прив'язка Pythonic для FFmpeg. PyAV - це прямий доступ до медіа через контейнери, потоки, пакети, кодеки та кадри. Він надає кілька перетворень цих даних і допомагає отримувати дані в / з інших пакетів (наприклад, Numpy та Pillow).Щоб завантажити відеофайл будемо використовувати функцію open() з бібліотеки av. Далі, щоб отримати серію потоків із заданого пакета використовуємо функцію demux()з бібліотеки av.

Повний код зображений нижче:

Імпортуємо необхідну бібліотеку та завантажуємо відео файл у середу розробки.

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()

Необхідний відео файл завантажений. Далі необхідно обробити кожен кадр відеопотоку та конвертувати текст з картинки в текст. Для кожного існуючого пакету із відео використовуємо функцію decode() з бібліотеки av. Кожний кадр предсталяє 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 2.



Рис.2 – зона, з якої необхідно зчитувати текст.

Далі, встановлюємо змінну myfilter = 30, що означає, що ми будемо обробляти за 1 раз 30 кадрів. Відеофайли мають розмір 30 кадрів/ секунду, тому за 1 прогонку цикла ми будемо проводити обробку 1 секунди відеофайла. Оскільки, текст не встигає змінитись з такою швидкість, то зчитувати ми будемо 30 кадрів за секунду, але парсити будем лише кожен 3 кадр(щоб збільшити продуктивність). Для парсингу текста з картинки будемо використовувати pytesseract (https://pypi.org/project/pytesseract/). Документація по бібліотеці знаходиться тут <https://tesseract-ocr.github.io/tessapi/4.0.0/>. Tesseract - це механізм розпізнавання тексту з відкритим кодом (OCR) , доступний за ліцензією Apache 2.0. Поточний офіційний випуск - 4.1.1 . 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, але є кілька доступних на сторінці 3rdParty . За допомогою Tesseract будемо парсити текст з виділеної області та зберігати його в масив, до якого також будем додавати номер кадру в відеофайлі.

Повний код зображений нижче:

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 елементу масива зображений на рисунку 3.



Рис.3 – елемент масива, що містить кадр, текст з картинки без пробілів та текст картинки з пробілами.

Для годинного випуску новин створення масиву кадрів+ тексту займає приблизно 6 годин. Код можна покращити, розпалалелив процеси на кожне ядро окремо. Для цього будем використовувати бібліотеку multiprocessing. multiprocessing пакет, який підтримує нерестові процеси за допомогою API, подібного до модуля threading. 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. Документація до пакету знаходиться тут : [multiprocessing — Process-based parallelism — Python 3.9.4 documentation](https://docs.python.org/3/library/multiprocessing.html).

Оскільки найбільше часу займає обробка зображення(виділення тексту), тому винесемо цю частину кода в окрему функцію, та запустимо паралельними процесами на кожному із доступних ядер.

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. 1 година новинного випуску оброблюється за 1 годину. Співвідношення 1 до 1 з точністю до 5 хвилин.

В результаті, отрумаємо масив, що зображений на рисунку 4.

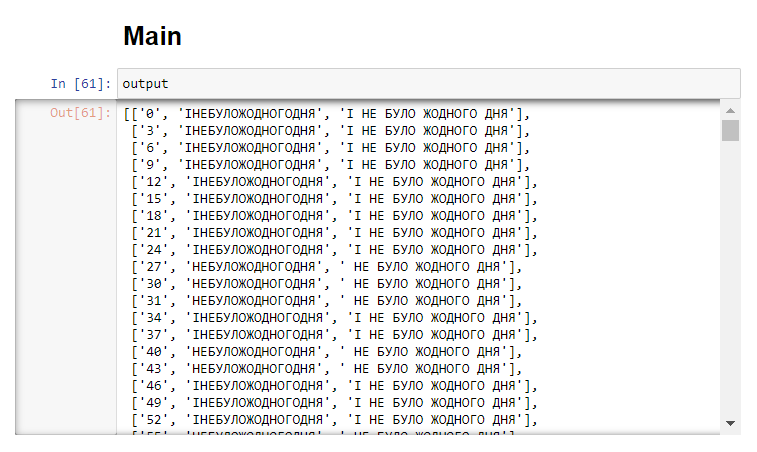


Рис.4 – елементи масива, що містять кадри, текст з картинки без пробілів та текст картинки з пробілами.

Наступним етапом буде обробка повторень та виділення повноцінних окремих частин кожного окремого сюжета(видеління початку сюжета, виделення кореспондентів, ведучих та гостей).

Література:

1. Інструкція по встановленню av -- <https://pypi.org/project/av/>
2. Документація до бібліотеки av -- <https://pyav.org/docs/stable/>
3. Інструкція по встановленню pytesseract - <https://pypi.org/project/pytesseract/>
4. Документація до бібліотеки pytesseract -- <https://tesseract-ocr.github.io/tessapi/4.0.0/>
5. Документація до бібліотеки multiprocessing -- multiprocessing — Process-based parallelism — Python 3.9.4 documentation