НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із переддипломної практики

на тему:  
«Математичне та програмне забезпечення для обробки й класифікації архіву новинних відеорепортажів»

|  |  |
| --- | --- |
| Виконав: | Керівник від університету: |
| студент групи КМ-71  Лисий П.О. | асистент кафедри ПМА  Ковальчук-Хімюк Л.О.  Керівник від підприємства:  старший викладач кафедри ПМА  Бай Ю.П. |

Київ – 2021

Анотація

Даний звіт присвячено результатам проходження переддипломної практики на тему «Математичне та програмне забезпечення для обробки й класифікації архіву новинних відеорепортажів», яка відбувалася з 12 квітня по 16 травня 2021 року на базі наукової лабораторії інформаційно-пошукових та інформаційно – аналітичних систем при ФПМ. У рамках практики поставлено задачу на дипломне проектування, а саме, розробити програмне забезепечення для обробки відеосюжету, 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 класифікувіти сюжет – позитивний або негативний. Для розв'язання поставленої задачі використовуються методи computer vision та natural language processing. Були розглянуті існуючі рішення, що застосовуватимуться при обробці відеосюжетів, зокрема: Tesseract ORC, Google Speech Recognition, Spacy та NLTK. З використанням даних бібліотек розроблено програмне забезепечення для обробки відеосюжетів, проведено його тестування, встановлено достатньо високу точність отриманих результатів.

ABSTRACT

This report is devoted to the results of undergraduate practice on "Mathematical and software for processing and classification of news video archives", which took place from April 12 to May 16, 2021 on the basis of the scientific laboratory of information retrieval and information-analytical systems at FPM. As part of the practice, the task is set for diploma design, namely, to develop software for video processing, which allows you to determine the start-end time of each plot, timing of the plot, highlight the title of the plot, select the text of the plot, select the type (news, announcement or advertising) , identify journalists and guests, identify presenters, define tags and classify the plot - positive or negative. Computer vision and natural language processing methods are used to solve this problem. Existing solutions that will be used in video processing were considered, in particular: Tesseract ORC, Google Speech Recognition, Spacy and NLTK. Using these libraries, software was developed for video processing, its testing was carried out, and a sufficiently high accuracy of the obtained results was established.

ЗМІСТ

[Перелік умовних позначень, скорочень і термінів 5](#_Toc72188883)

[Вступ 6](#_Toc72188884)

[1 Постановка задачі 7](#_Toc72188885)

[2 Аналіз існуючих методів розв'язання задачі 8](#_Toc72188886)

[2.1.1 Tesseract ORC 8](#_Toc72188887)

[2.1.3 CNN 9](#_Toc72188888)

[2.2 Voice to text system 9](#_Toc72188889)

[2.1.1 Google Speech Recognition 9](#_Toc72188890)

[2.1.2 Wav2letter ++ 10](#_Toc72188891)

[2.1.3 CMU Sphinx 10](#_Toc72188892)

[2.2 Natural language processing (nlp) 10](#_Toc72188893)

[2.2.1 Визначення тегів 10](#_Toc72188894)

[2.2.2 Класифікація настрою сюжету 11](#_Toc72188895)

[3. Процес обробки відеосюжету 13](#_Toc72188896)

[3.2 Розпізнавання тексту з використанням Google Speech Recognition 14](#_Toc72188897)

[3.3 Визначення тегів 15](#_Toc72188898)

[3.3.1 TF-IDM 15](#_Toc72188899)

[3.3.2 NLTK та Spacy 16](#_Toc72188900)

[3 Програмне забезпечення 18](#_Toc72188901)

[Висновки 19](#_Toc72188902)

[Перелік посилань 20](#_Toc72188903)

[Додаток А 21](#_Toc72188904)

[Лістинги програм 21](#_Toc72188905)

Перелік умовних позначень, скорочень і термінів

NLP — natural language processing

CV — computer vision

CNN — convolution neural network

start-end time — час початку та кінця кожного окремого відеосюжету з новинного репортажу, починаючи відлік від початку сюжету.

Fps — frames per second, кількість кадрів за секунду

OCR — Optical Character Recognition, оптичне розпізнавання символів

Вступ

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.

Тому на переддипломну практику мені було поставлено 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 однієї медіа компанії України.

Під час практики 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 переддипломної практики 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:

включати визначення start-end time кожного окремо сюжету, визначення хронометражу сюжету, виділення назви сюжету, виділення тексту сюжету, виділення типу (новини, анонс та реклама), визначення журналістів та гостей, визначення ведучих, визначення тегів та класифікація сюжету(позитивний/негативний).

мати високі показники ефективності розпізнавання start-end time сюжету(точність повинна бути до 1 секунди, >85% правильності);

Записувати таблицю обробки даних для новинного випуску в файл формату Microsoft Excel;

# 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.

2.1 Computer vision(CV)

### 2.1.1 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. На сьогоднішній день бібліотека є найбільш крупним рішенням, якщо потрібно обробити дані з будь-якого фото. Оскільки у нас обробка відеофайлу - відеофайл представим як набір кадрів і будемо зчитувати текст з кожного кадра окремо. Основні переваги це висока точність та легкість впровадження у власний проект.

2.1.2 OCRopus

OCRopus - OCR-система для розпізнавання текстів на базі tesseract. Використовує код для 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.

### CNN

CNN, або згорткові нейронні мережі, можуть бути використані для вирішення задачі розпізнавання тексту з картинки. Основна перевага – 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.

## 2.2 Voice to text system

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 = 119. Бібліотека Google Speech Recognition є найбільш точною та якісною при роботі зі звуком на українській мові. Також, бібліотека непогано справляється з звуковими доріжками, що перевантажені шумом. SpeechRecogntion має низький бар’єр для входу та сумісність із багатьма доступними API розпізнавання мов. (<https://pypi.org/project/SpeechRecognition/>).

### 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 працює і зручна у використанні, проте SpeechRecognition краще оброблює шуми на аудіодоріжці. (https://github.com/flashlight/wav2letter).

### CMU Sphinx

CMU Sphinx на ринку понад 20 років досліджень CMU. Деяка перевага 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. (https://cmusphinx.github.io/).

## 2.2 Natural language processing (nlp)

### 2.2.1 Визначення тегів

2.2.1.1 TF-IDM

TF-IDF це простий і зручний спосіб оцінити важливість терміну для будь-якого документа щодо всіх інших документів. Принцип такий: 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. Слова, неважливі для взагалі всіх документів, наприклад, прийменники - отримають дуже низьку вагу TF-IDF, бо часто зустрічаються у всіх документах, а важливі - високу.

2.2.1.2 NLTK

NLTK – одна із найпоширеніших бібліотека 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. Так само вона досить повільна в силу того, що написана повністю на Python і працює з рядками.

2.2.1.3 Spacy

Spacy – 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.

### 2.2.2 Класифікація настрою сюжету

2.2.2.1 SentimentIntensityAnalyzer

Найкращим знайденим рішенням виявилась функція SentimentIntensityAnalyzer з бібліотеки NLTK, яку було визначено вище.

(<https://www.nltk.org/api/nltk.sentiment.html>)

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.

# 3. процес обробки відеосюжету

3.1 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 рис. 1.



Рис.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 будем додавати номер кадру в відеофайлі.

## 3.2 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. Спектр потужності кожного фрагмента, який по суті є графіком потужності сигналу як функції від частоти, відображається на вектор дійсних чисел, відомий як цепстральні коефіцієнти. Розмір цього вектора, як правило, невеликий - іноді до 10, хоча більш точні системи можуть мати розмір 32 і більше. Кінцевий вихід HMM є послідовністю цих векторів. 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. Детектори голосової активності (VAD) також використовуються для зменшення звукового сигналу лише до тих частин, які можуть містити мову. Це запобігає марнуванню часу декодера на аналіз непотрібних частин сигналу. Більш детальна інформація про архітектуру та можливості Google Speech Recognition знаходиться в документації [10].

## 3.3 Визначення тегів

### 3.3.1 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, з якою слово зустрічається у всіх документах; *IDF* – показник кількості документів, в яких зустрічається дане слово. На основі даних величин розраховується нова величина

*TF-IDM:*

TF-IDM = TF \* IDF

Для розрахунку TF використовуємо наступну формулу:

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

Для розрахунку IDF використовуємо наступну формулу:

де – кількість документів в базі, а – кількість документів, в яких зустрічається слово (при ).

Вибір основи логарифму у формулі не має значення, адже зміна основи призведе до зміни ваги кожного слова на постійний множник, тобто вагове співвідношення залишиться незмінним.

Більшу вагу TF-IDF отримають слова з високою частотою появи в межах документа та низькою частотою вживання в інших документах бази.

### 3.3.2 NLTK та Spacy

NLTK та Spacy є напереднатреновані дві нейромережі, архітектурою яких є CNN. Завдяки NLTK можна отримати токенезацію текстів, отримати теги, отримати іменовані сутності, відобразити дерево синтаксичного аналізу [1-3]. Більш детальна інформація про архітектуру та можливості NLTK знаходиться в документації [8].

Бібліотека Spacy має наступні особливості:

1. Підтримка 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. Spacy легко розширюється за допомогою спеціальних компонентів - PyTorch, TensorFlow та інші фреймворки. Більш детальна інформація про архітектуру та можливості Spacy знаходиться в документації [9].

3.3.3 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 [8].

# 

# 4 Програмне забезпечення

4.1 Cхема взаємодії програмних засобів

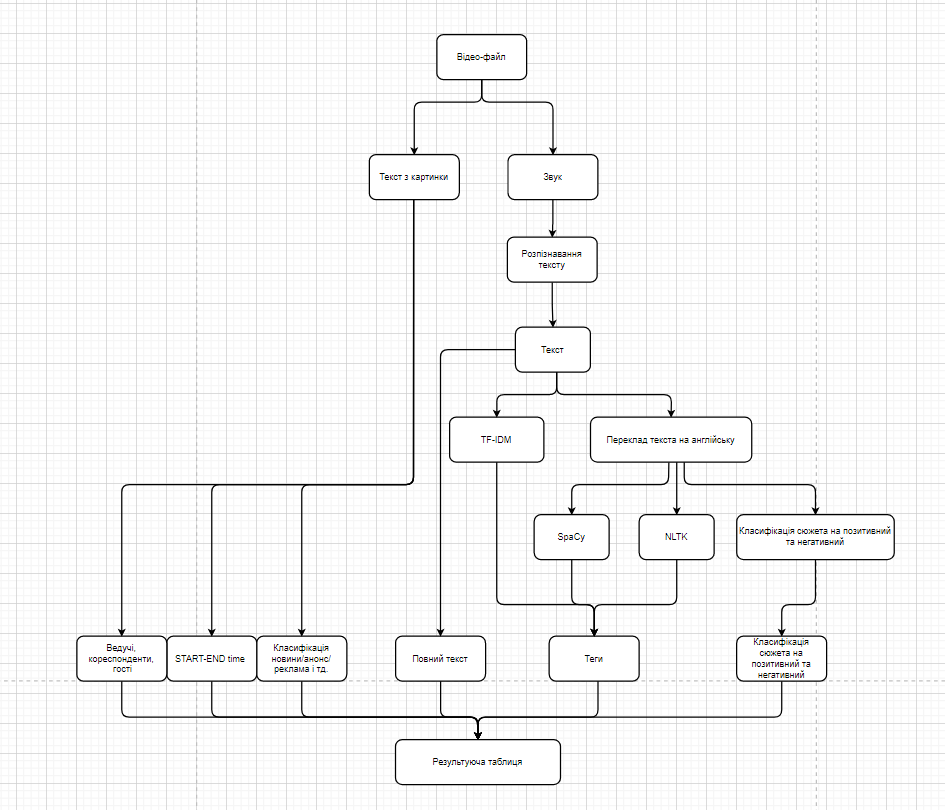


Рис.1. Схема взаємодії програмних засобів.

На рис.1. зображене схема взаємодії програмних засобів.З початкового відео-файла виділяється звук та текст з картинки. Завдяки тексту з картинки виділяємо та отримуємо start-end time кожного сюжета, імена ведучих, кореспондентів та гостей, також класифікуємо сюжет на новини, анонс, рекламу, привітання та прощання. Завдяки звуку, виділеного з відео-файлу, отримуємо 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 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 позитивний чи негативний сюжет у результуючу змінну 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.

## 4.1 Обробка відеопотоку

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. Для цього будем використовувати бібліотка av[].

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 на 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 кожен 3 кадр (щоб збільшити продуктивність). Для парсингу текста з картинки будемо використовувати pytesseract[]. 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 та тексту займає приблизно 6 годин. Код можна покращити, запустивши процеси на кожному ядрі 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 [].

Оскільки найбільше часу займає обробка зображення(виділення тексту), тому винесемо цю частину кода в окрему функцію, та запустимо окремими 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 годин з точністю до 5 хвилин.

В результаті отримаємо масив, що зображений на рисунку 4.

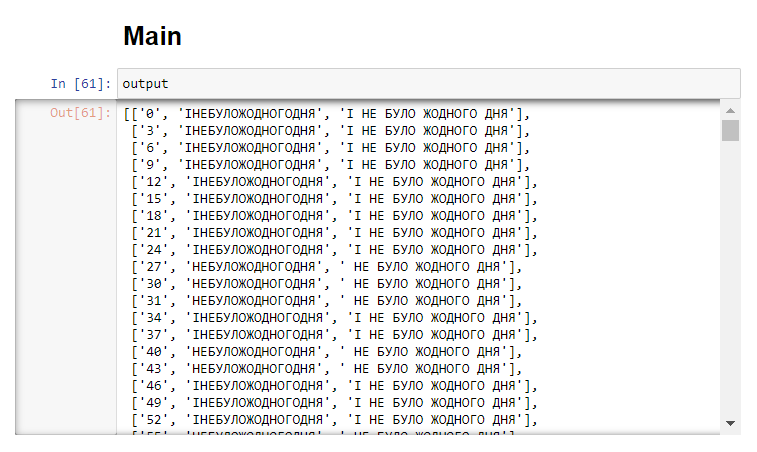


Рис.4 – елементи масива, що містять номери кадрів, текст з картинки без пробілів та текст картинки з пробілами.

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

## 4.2 Обробка масиву кадрів, що містять номери кадрів, текст з картинки без пробілів та текст картинки з пробілами

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

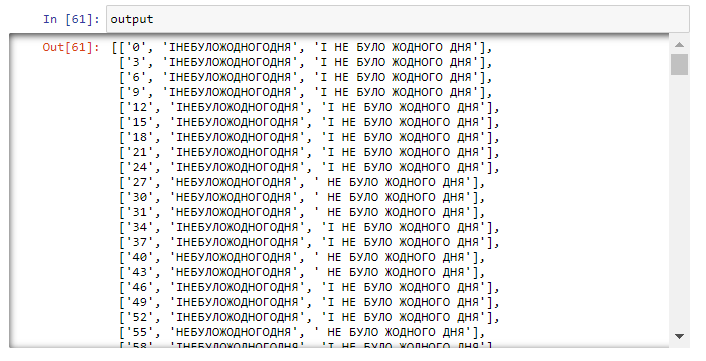


Рис. 5. – Можливі відмінності при віделенні текста

Для оцінки подібності тексту будемо використовувати метрику Джаро-Вінклера.

Подібність Джаро між двома рядками l1 та l2 визначається як:

– довжина i-го рядка

n – кількість співпадінь символів

t – кількість транспозицій

Два символи від l1 і l2 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[].

Сюжетний ефір поділений на 3 частини – анонс, основний ефір та рекламна частина. Виділення анонсу знаходимо шляхом співзіставленням кожного кадру з кадром заставки.

Перед рекламною частиною в тексті буде знаходитись напис ОТОВ«НОВИННАГРУПАУКРАЇНА». Після цього, 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 та прощання.

Для виявлення заставки необхідно аналізувати за знаходити картинку заставки.

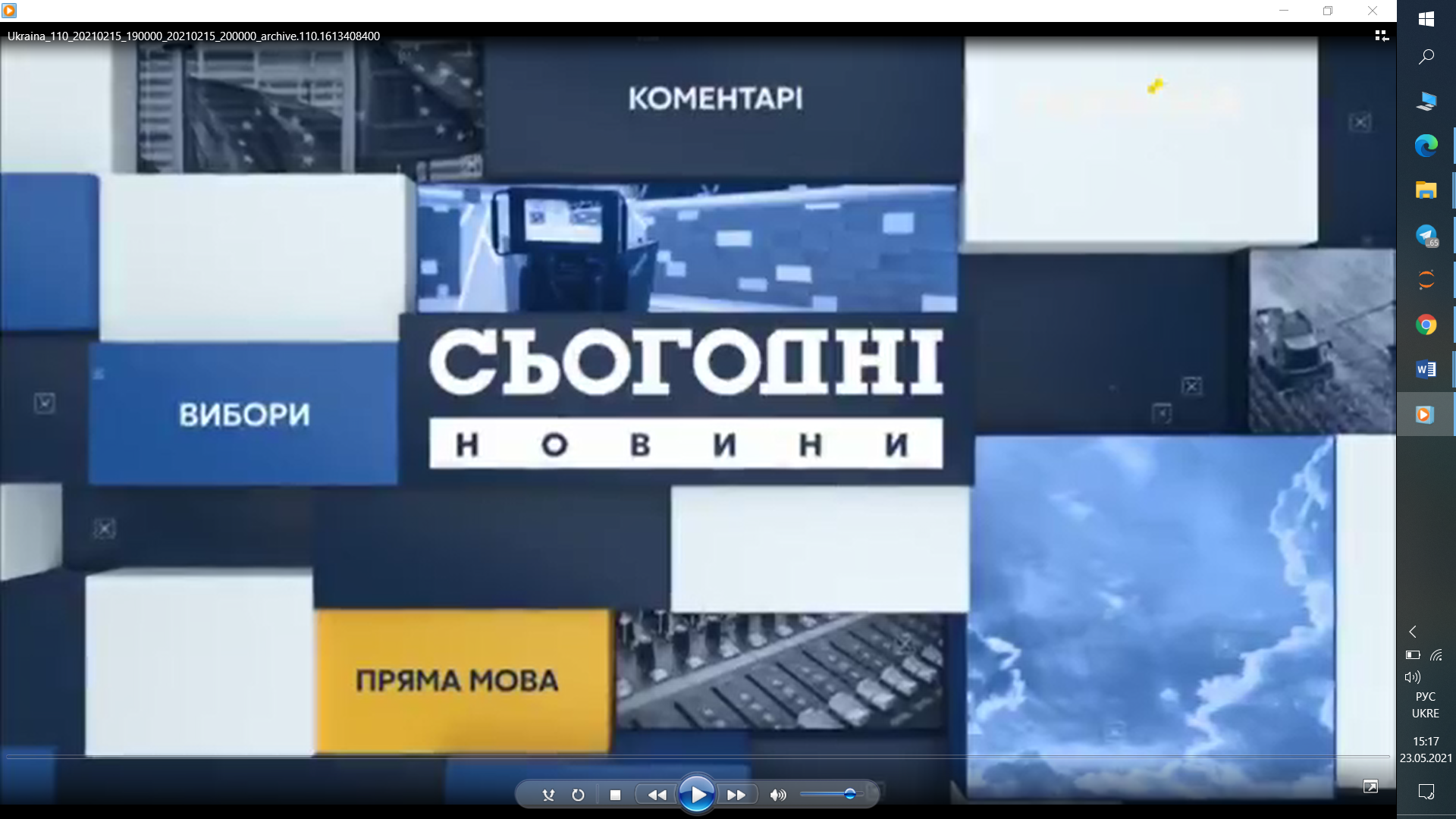


Рис. 6. – Картинка заставки

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 новину. 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. Кадр, після якого починаються анонси, зображений на рисунку 10.

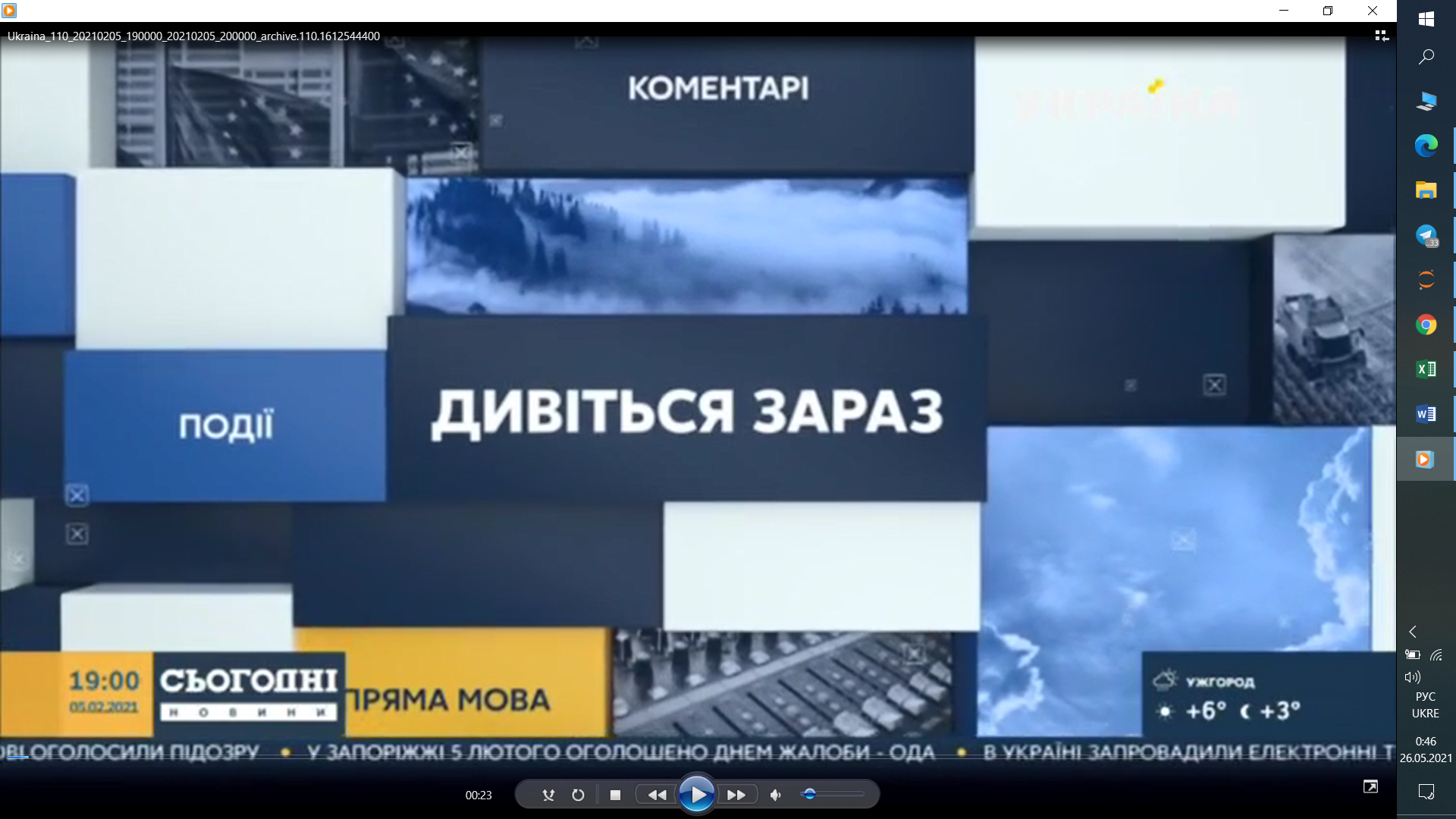


Рис. 10. – Кадр після якого починаються анонси

Так само як і при виділенні кадру заставки, необхідно перевіряти та порівнювати кожен кадр. Для цього будемо використовувати різницю по модулю кожного кадру з заданим для перевірки. Функція модуль представлена в бібліотеці math функцією fabs(). Встановимо дослідницьким шляхом, 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 10. 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 11.

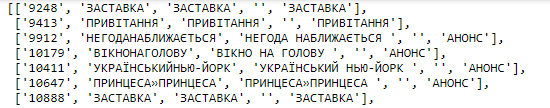


Рис. 11. – Результат виконання коду для виявлення заставки та анонсів.

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 зображений на рисунку 7.



Рис. 7. – Приклад результату роботи програми – додання кореспондентів до масиву.

Наступна проблема – визначити імена ведучих. Імена ведучих з'являються на екрані при привітанні на початку новинного випуску. Приклад появи зони з іменами ведучих зображений на рисунку 8.

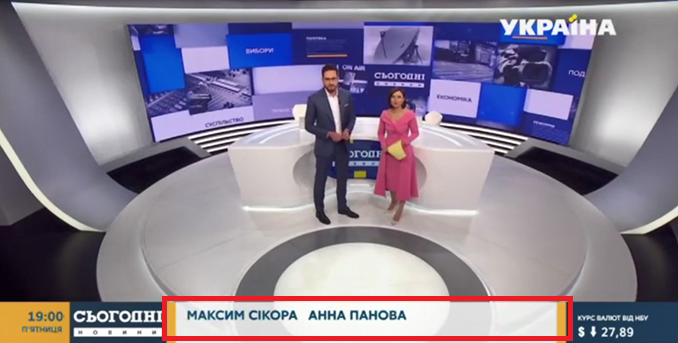


Рис. 8. – Імена ведучих

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, з привітанням необхідно порівнювати кожен кадр з розпізаним текстом з базою ведучих. Код для задання бази ведучих наведений нижче:

leading=['ІГОРПУПКОВ','ОЛЕНАКОТ','МАКСИМСІКОРА', 'ОЛЕГПАНЮТА','АННАПАНОВА','ОЛЬГАГРИЦИК']

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, а 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 9.



Рис. 9. – Результат виконання коду для виявлення ведучих та присвоєння відеофрагменту тип «Привітання»

4.3 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[].

Виділяємо звукові доріжки по 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. Він 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[].

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)

4.4 Отримання тексту зі звукової доріжки

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[].

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')

4.5 Класифікація сюжету на позитивний чи негативний

Імпортуємо функцію SentimentIntensityAnalyzer() з бібліотки nltk[]. Створюємо об'єкт класу SentimentIntensityAnalyzer() та оримуємо настрій сюжету методом polarity\_scores(text), де замість змінної text – вибираємо весь текст з сюжету. Та записуємо текст в файл csv.

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

4.6 Запис в Excel файл

Останній етап – запис результуючої таблиці в файл csv.

Для цього будемо використовувати бібліотеку openpyxl []. 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. Приклад запису в клітинку А1 слова «text» зображений нижче:

sheet['A1')] = «week»

Таким чином, створюємо цикл, що проходить по всій стореній базі новинних сюжетів та переносить результат в таблицю ексель. Після змін вихідного файлу ексель його необхідно зберігти.

Для зберігання результату запису даних в ексель 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')

Результат роботи програми представлено на рис. 8.

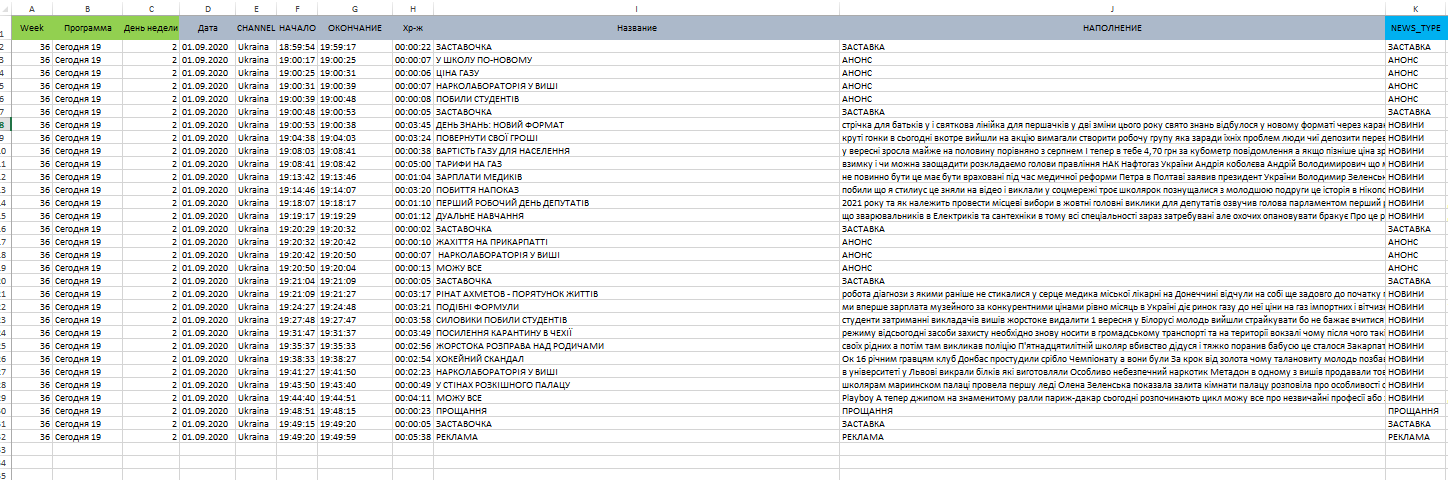


Рис. 12. Фрагмент csv-файлу – результат обробки одного новинного випуску. Виділений тиждень, програма, день тижня, дата, канал, start-ent time кожного сюжету, хронометраж, назва сюжету, текст сюжету та тип сюжету.

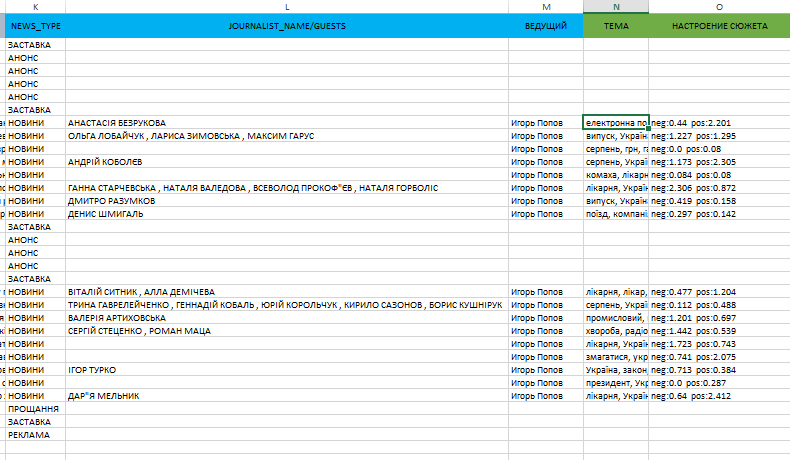


Рис. 13. Фрагмент csv-файлу - результат обробки одного новинного випуску. Виділений тип сюжету, імену журналістів та гостей, імена ведучих, теги та класифікація сюжету на позитивний чи негативний.

Висновки

На переддипломну практику було поставлено 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.

Для розв'язання поставленої задачі використовуються методи computer vision та natural language processing. Під час проходження практики було проаналізовано доступні математичні методи та програмні рішення. Було розроблено алгоритм для обробки одного відеосюжету, який складається з:

1. Виділення тексту та його обробка за допомогою natural language processing.
2. Аналіз звукової доріжки і отримання її у вигляді тексту за допомогою natural language processing.
3. Перекладання тексту за допомогою Google Speech Recognition.
4. Виділення тексту за допомогою computer vision та визначення назви, ведучих, гостей та кореспонденів відеосюжету.
5. Визначення тегів відеосюжету за допомогою Spacy, NLTK та TF-IDF та встановлення на основі отриманої інформації тематики відеосюжету.
6. Класифікація настрою новинного сюжету positive, negative або neutral за допомогою функції SentimentIntensityAnalyzer з бібліотеки NLTK.

Розроблено програмне забезпечення на мові python, що реалізує даний алгоритм, протестована його робота, отримана висока точність результатів.

# Перелік посилань

1. Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit / Steven Bird, Ewan Klein, and Edward Loper. – [Електронний ресурс]. – Режим доступу: <https://www.nltk.org/book>
2. Jurafsky D. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition / D. Jurafsky, J. H. Martin. — 3rd edition. — Prentice Hall. — 2019. — 621 p.
3. . Близнюк Б.О. Современные методы обработки естественного языка / Б. О. Близнюк, Л. В. Васильева, И. Д. Стрельников, Д. С. Ткачук // Вісник Харківського нац. ун-ту ім. В. Н. Каразіна. — 2017. — Вип. 36. — С. 14–26.
4. Хайрова Н.Ф. Современные технологии обработки текстовых данных на базе пакета NLTK Python : Учеб. пособ. / Н. Ф. Хайрова, О. Ж. Мамырбаев, С. В. Петрасова, К. Ж. Мухсина. — Харьков : НТУ «ХПИ». — 2020. — 134 с.
5. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / Пер. с англ. А. А. Слинкина. — 2-е изд., испр. – М.: ДМК Пресс. — 2018. — 652 с.
6. Vasiliev Y. Natural Language Processing with Python and spaCy: A Practical Introduction. — 2020. – 226 с. [Електронний ресурс]. – Режим доступу: <https://nostarch.com/NLPPython>
7. Blei D. M., Ng A. Y., Jordan M. I.Latent Dirichlet allocation // Journal of MachineLearning Research. — 2003. — Vol. 3. — P. 993–1022.
8. Документація бібліотеки NLTK для роботи з текстом. [Електронний ресурс]. – Режим доступу: <https://www.nltk.org/>
9. Документація бібліотеки SpaCy для роботи з текстом . [Електронний ресурс]. – Режим доступу: <https://spacy.io/>
10. Документація бібліотеки SpeechRecognition для розпізнання звуку. [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/ru/docs/Web/API/SpeechRecognition>
11. Документація бібліотеки av для розпізнання звуку. [Електронний ресурс]. – Режим доступу: <https://pypi.org/project/av/>
12. Документація бібліотеки tesseract-ocr для розпізнання звуку. [Електронний ресурс]. – Режим доступу: <https://tesseract-ocr.github.io/tessapi/4.0.0/>
13. Документація бібліотеки jellyfish для розпізнання звуку. [Електронний ресурс]. – Режим доступу: <https://pypi.org/project/jellyfish>

Лістинги програм

import av

from av.video.frame import VideoFrame

from av.video.stream import VideoStream

import pytesseract

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

import subprocess

import math

import matplotlib as mpl

import speech\_recognition as sr

import jellyfish

mpl.rcParams['agg.path.chunksize'] = 100000

number\_of\_frames = 30

leading = ['ІГОРПУПКОВ', 'ОЛЕНАКОТ', 'МАКСИМСІКОРА', 'ОЛЕГПАНЮТА', 'АННАПАНОВА', 'ІГОРПУПКОВОЛЕНАКОТ', 'ІГОРПУПКОВОЛЬГАГРИЦИК', 'МАКСИМСІКОРААННАПАНОВА']

multiplication\_hours = 18

multiplication\_minutes = 55

from moviepy.editor import VideoFileClip, concatenate\_videoclips

clip1 = VideoFileClip("video/crop.mp4")

clip2 = VideoFileClip("video/" + name + '30fps.mp4')

final\_clip = concatenate\_videoclips([clip1,clip2])

final\_clip.write\_videofile("video/"+name+"done.mp4")

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

name = name +"done.mp4"

path = "C:/Users/lysyi/Desktop/TRKu/maintest/"

file = road + name

myfilter = 6

mmm=0

check = 0

check2 = 0

i = 0

array\_list = []

output = []

input\_container = av.open(file)

input\_packets = input\_container.demux()

del input\_container

from PIL import Image

import imagehash

#hash0 = imagehash.average\_hash(Image.open(road + 'data/splash\_screen/' + '1' + ".jpeg"))

#hash3 = imagehash.average\_hash(Image.open(road + 'data/splash\_screen/' + '3' + ".jpeg"))

hash0 = imagehash.average\_hash(Image.open(road + 'data/splash\_screen/' + '6' + ".jpeg"))

hash3 = imagehash.average\_hash(Image.open(road + 'data/splash\_screen/' + '7' + ".jpeg"))

zastavka = 0

for packet in input\_packets:

if isinstance(packet.stream, VideoStream):

frames = packet.decode()

for raw\_frame in frames:

frame = raw\_frame.reformat(720,396, 'gray')

splash = frame.to\_nd\_array()[:300,]

splash = Image.fromarray(splash)

splash = splash.convert("RGB")

hash1 = imagehash.average\_hash(splash)

cutoff = 10

zastavka = 0

if math.fabs(hash0 - hash1) < cutoff:

output.append([str(i+check2), 'ЗАСТАВКА', 'ЗАСТАВКА'])

zastavka = 1

elif math.fabs(hash3 - hash1) < cutoff:

output.append([str(i+check2), 'ЗАСТАВКА', 'ЗАСТАВКА'])

zastavka = 1

#from PIL import Image

#im = Image.fromarray(splash)

#im.save(road + 'data/splash\_screen/' + str(mmm) + ".jpeg")

#mmm+=1

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

array\_list += [array]

if zastavka == 0:

check +=1

if check == myfilter:

mass = []

for m in range(0,len(array\_list)):

if m%3==0:

mass.append([str(i+check2), str(pytesseract.image\_to\_string(array\_list[i], 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(), str(pytesseract.image\_to\_string(array\_list[i], lang = "ukr")).replace('#', '').replace('%', '').replace('\*', '').replace('!', '').replace('@', '').replace('/', '').replace('|','').replace('\n', '').replace('\x0c', '').replace('.', '').replace(',', '').replace("'",'"').replace('(', '').replace(')', '').upper()])

for j in range(0,int(len(mass))):

for l in mass:

if l[1] == '\x0c' or l[1] == ' \n\x0c' or l[1] == ' ' or l[1] == '':

mass.remove(l)

for k in mass:

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

print(str(int(k[0])/number\_of\_frames//60) + ' min ' + str(round(int(k[0])/number\_of\_frames%60,1)) + ' sec\t' + str(k[1]))

array\_list = []

answer\_mass = []

mass = []

check2 += check

check = 0

from joblib import Parallel, delayed

import multiprocessing

from PIL import Image

import imagehash

myfilter = 10\*3

check = 0

check2 = 0

array\_list = []

output = []

input\_container = av.open(file)

input\_packets = input\_container.demux()

del input\_container

hash0 = imagehash.average\_hash(Image.open(road + 'data/splash\_screen/' + '6' + ".jpeg"))

hash3 = imagehash.average\_hash(Image.open(road + 'data/splash\_screen/' + '7' + ".jpeg"))

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:

frame = raw\_frame.reformat(720,396, 'gray')

splash = frame.to\_nd\_array()[:300,]

splash = Image.fromarray(splash)

splash = splash.convert("RGB")

hash1 = imagehash.average\_hash(splash)

zastavka = 0

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

output.append([str(check2+check), 'ЗАСТАВКА', 'ЗАСТАВКА'])

zastavka = 1

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

check +=1

if zastavka == 0:

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 = []

print(check2/30)

delete = []

for index in range(0,len(output)):

if len(output[index]) == 0:

delete.append(index)

elif len(output[index][1]) <=5:

delete.append(index)

for j in range(0, len(delete)):

output.pop(delete[j]-j)

checkmymass = []

for index in output:

checkmymass.append(index[1])

delete = []

for index2 in range(0,len(output)):

if checkmymass.count(output[index2][1]) < 5:

delete.append(index2)

for j in range(0, len(delete)):

output.pop(delete[j]-j)

del checkmymass

last\_output = [output[0]]

for index3 in range(1,len(output)):

if jellyfish.jaro\_distance(output[index3][1],last\_output[-1][1]) < 0.8:

last\_output.append(output[index3])

zastavka\_mass = []

for index in last\_output:

if index[1] == 'ЗАСТАВКА':

zastavka\_mass.append(index[0])

print(zastavka\_mass)

zastavka\_mass2 = []

for index in range(0,len(zastavka\_mass)-1):

zastavka\_mass2.append((int(zastavka\_mass[index+1])-int(zastavka\_mass[index]))/30)

print(zastavka\_mass2)

delete\_mass = []

for index in range(0,len(zastavka\_mass2)):

if zastavka\_mass2[index] < 23:

delete\_mass.append(zastavka\_mass[index+1])

print(delete\_mass)

delete = []

for jindex in delete\_mass:

for index in range(0,len(last\_output)):

if jindex == last\_output[index][0]:

delete.append(index)

print(delete)

delete = sorted(list(set(delete)))

for j in range(0, len(delete)):

try:

last\_output.pop(delete[j]-j)

except: pass

#заставка

delete = []

for index in range(0,len(output)-4):

if output[index][1] == 'ЗАСТАВКА' and output[index+2][1] == 'ЗАСТАВКА':

delete.append(index+1)

delete.append(index+2)

if output[index][1] == 'ЗАСТАВКА' and output[index+3][1] == 'ЗАСТАВКА':

delete.append(index+1)

delete.append(index+2)

delete.append(index+3)

if output[index][1] == 'ЗАСТАВКА' and output[index+4][1] == 'ЗАСТАВКА':

delete.append(index+1)

delete.append(index+2)

delete.append(index+3)

delete.append(index+4)

for j in range(0, len(delete)):

try:

output.pop(delete[j]-j)

except: pass

first\_part = []

second\_part = []

zastavochka = 0

for i in last\_output:

if i[1] == 'ЗАСТАВКА':

zastavochka += 1

if zastavochka == 1:

first\_part.append(i)

elif zastavochka == 2:

first\_part.append(i)

zastavochka = 0

else:

second\_part.append(i)

delete = []

for i in range(0, len(second\_part)-1):

if jellyfish.jaro\_distance(second\_part[i][1],second\_part[i+1][1]) > 0.75:

delete.append(i+1)

delete = list(set(delete))

for j in range(0, len(delete)):

try:

second\_part.pop(delete[j]-j)

except: pass

delete = []

for i in range(0,len(first\_part)):

first\_part[i].append('')

if first\_part[i][1] == 'ЗАСТАВКА':

first\_part[i].append('ЗАСТАВКА')

else:

first\_part[i].append('АНОНС')

news\_anchor\_mass = []

for mm in leading:

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

news\_anchor\_mass += mm

first\_part[i][4] = 'ПРИВІТАННЯ'

first\_part[i][1] = 'ПРИВІТАННЯ'

first\_part[i][2] = 'ПРИВІТАННЯ'

news\_anchor\_name = ' '

try:

for i in news\_anchor\_mass:

news\_anchor\_name += str(i)

news\_anchor\_name += ' '

except:

news\_anchor\_name = ' '

for pp in range(0,len(second\_part)):

second\_part[pp].append([])

for index in range(0,len(second\_part)):

delete = []

for m in range(0,len(second\_part[index][3])):

if jellyfish.jaro\_distance(second\_part[index][3][m],second\_part[index][2]) > 0.75 :

delete.append(m)

for j in range(0, len(delete)):

try:

second\_part[index][3].pop(delete[j]-j)

except: pass

for index\_\_ in range(0,5):

delete = []

for index in range(0, len(second\_part)-2):

if jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+2][1]) > 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+1][1]) < 0.75:

delete.append(index+1)

try:

for names\_index in second\_part[index+1][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

second\_part[index][3].append(second\_part[index+1][2])

except: pass

for index in range(0, len(second\_part)-5):

if jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+5][1]) > 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+1][1]) < 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+2][1]) < 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+3][1]) < 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+4][1]) < 0.75:

delete.append(index+1)

delete.append(index+2)

delete.append(index+3)

delete.append(index+4)

try:

for names\_index in second\_part[index+1][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+2][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+3][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+4][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+5][3]:

second\_part[index][3].append(names\_index)

except: pass

for index in range(0, len(second\_part)-4):

if jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+4][1]) > 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+1][1]) < 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+2][1]) < 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+3][1]) < 0.75:

delete.append(index+1)

delete.append(index+2)

delete.append(index+3)

try:

for names\_index in second\_part[index+1][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+2][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+3][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+4][3]:

second\_part[index][3].append(names\_index)

except: pass

for index in range(0, len(second\_part)-3):

if jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+3][1]) > 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+1][1]) < 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+2][1]) < 0.75:

delete.append(index+1)

delete.append(index+2)

try:

for names\_index in second\_part[index+1][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+2][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

for names\_index in second\_part[index+3][3]:

second\_part[index][3].append(names\_index)

except: pass

for index in range(0, len(second\_part)-2):

if jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+2][1]) > 0.75 and jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+1][1]) < 0.75:

delete.append(index+1)

try:

for names\_index in second\_part[index+1][3]:

second\_part[index][3].append(names\_index)

except: pass

try:

second\_part[index][3].append(second\_part[index+1][2])

except: pass

for index\_ in range(0,20):

for index in range(0, len(second\_part)-1):

if jellyfish.jaro\_distance(second\_part[index][1],second\_part[index+1][1]) > 0.75 or second\_part[index][1] in second\_part[index+1][1] or second\_part[index+1][1] in second\_part[index][1]:

delete.append(index+1)

try:

second\_part[index][3].append(second\_part[index+1][2])

except: pass

try:

for names\_index in second\_part[index+1][3]:

second\_part[index][3].append(names\_index)

except: pass

delete = sorted(list(set(delete)))

for j in range(0, len(delete)):

try:

second\_part.pop(delete[j]-j)

except: pass

for index in range(0,len(second\_part)):

second\_part[index][3] = list(set(second\_part[index][3]))

for i in range(0,len(second\_part)):

if jellyfish.jaro\_distance(second\_part[i][1],u'ЗЕСОРМУАМА') > 0.75 or jellyfish.jaro\_distance(second\_part[i][1],u'ОТОВ«НОВИННАГРУПАУКРАЇНА»') > 0.75:

second\_part[i][1] = 'ПРОЩАННЯ'

second\_part[i][2] = 'ПРОЩАННЯ'

delete = []

for i in range(0,len(second\_part)):

if second\_part[i][1] == 'ПРОЩАННЯ':

first\_part.append(second\_part[i])

delete.append(i)

for j in range(0, len(delete)):

try:

second\_part.pop(delete[j]-j)

except: pass

for i in range(0, len(first\_part)):

if first\_part[i][1] == 'ПРОЩАННЯ':

first\_part[i].append('')

first\_part[i].append('ПРОЩАННЯ')

for index in range(0,len(second\_part)):

delete = []

for index2 in range(0,len(second\_part[index][3])):

if jellyfish.jaro\_distance(second\_part[index][2], second\_part[index][3][index2]) > 0.75:

delete.append(index2)

for j in range(0, len(delete)):

try:

second\_part[index][3].pop(delete[j]-j)

except: pass

for i in range(0,len(second\_part)):

try:

second\_part[i][3] = list(set(second\_part[i][3]))

except:

pass

first\_part.sort(key=lambda i: int(i[0]))

delete =[]

for i in range(0, len(second\_part)):

if len(second\_part[i]) == 3:

second\_part[i].append('')

if len(second\_part[i]) == 4:

second\_part[i].append('НОВИНИ')

input = (first\_part+second\_part)

input.sort(key=lambda i: int(i[0]))

while 1:

if input[-1][1] == 'ЗАСТАВКА' or input[-1][1] == 'ПРОЩАННЯ':

break

else:

input.pop(-1)

while input[0][1] != 'ЗАСТАВКА':

input.pop(0)

delete = []

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

if jellyfish.jaro\_distance(input[i][1],input[i+1][1]) > 0.75:

delete.append(i+1)

delete = sorted(list(set(delete)))

for j in range(0, len(delete)):

try:

input.pop(delete[j]-j)

except: pass

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

if jellyfish.jaro\_distance(input[i][1],u'ПРОЩАННЯ') > 0.95:

input[i].pop(3)

input.append([str(int(input[-1][0])+150),'РЕКЛАМА','РЕКЛАМА','','РЕКЛАМА'])

input.append(['107970','dueto','dueto','','dueto'])

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

delete = []

for index2 in range(0, len(input[index][3])):

if jellyfish.jaro\_distance(input[index][3][index2],input[index][1]) > 0.75:

delete.append(input[index][3][index2])

for j in range(0, len(delete)):

try:

input[index][3].remove(delete[j])

print('//')

except: pass

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

if input[index][1] == 'ДИВІТЬСЯДАЛІ':

input[index][1] = 'АНОНСЗІСЛІВВЕДУЩОГО'

input[index][2] = 'АНОНС ЗІ СЛІВ ВЕДУЩОГО'

input[index][4] = 'АНОНС'

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

delete = []

try:

for index2 in range(0,len(input[index])):

if len(input[index][3][index2]) > 20:

delete.append(index2)

except:pass

delete = sorted(list(set(delete)))

for j in range(0, len(delete)):

try:

input[index][3].pop(delete[j]-j)

except: pass

delete = []

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

if jellyfish.jaro\_distance(input[index][1], 'ЯМАМОВАБМАЙ')>0.75:

delete.append(index)

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

if jellyfish.jaro\_distance(input[index][1], '"ЯМАМОВАЩИА')>0.75:

delete.append(index)

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

if jellyfish.jaro\_distance(input[index][1], 'ЖРЯМАМОВАРРОО')>0.75:

delete.append(index)

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

if jellyfish.jaro\_distance(input[index][1], '"ЯМАМОВАПЕ«ОС')>0.75:

delete.append(index)

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

if jellyfish.jaro\_distance(input[index][1], 'ЯМАМОВАРРТЄСС')>0.75:

delete.append(index)

delete = sorted(list(set(delete)))

print(delete)

for j in range(0, len(delete)):

try:

input.pop(delete[j]-j)

except: pass

delete = []

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

if len(input[index][1]) > 40:

print(len(input[index][1]))

delete.append(index)

try:

for index\_\_ in input[index][3]:

input[index-1][3].append(index\_\_)

except: pass

delete = sorted(list(set(delete)))

print(delete)

for j in range(0, len(delete)):

try:

input.pop(delete[j]-j)

except: pass

#сдвиг

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

if input[index][4] == 'НОВИНИ':

input[index][0] = str(int(input[index][0])-60)

from google\_trans\_new import google\_translator

from spacy import displacy

from nltk.stem import PorterStemmer, WordNetLemmatizer

from nltk.corpus import wordnet

from nltk.sentiment.vader import SentimentIntensityAnalyzer

from textblob import TextBlob

from textblob.sentiments import NaiveBayesAnalyzer

import math

import speech\_recognition as sr

import subprocess

import spacy

import en\_core\_web\_sm

import jellyfish

import openpyxl

import datetime

import os

def compare\_stemmer\_and\_lemmatizer(stemmer, lemmatizer, word, pos):

return lemmatizer.lemmatize(word, pos)

programm = 'Сегодня 19'

channel = name.split('\_')[0]

data = str(name.split('\_')[2][-2:])+'.'+str(name.split('\_')[2][4:6])+'.'+str(name.split('\_')[2][:4])

number\_of\_frames = 30

day = datetime.date(int(data.split(".")[2]), int(data.split(".")[1]), int(data.split(".")[0])).weekday()+1

week = datetime.date(int(data.split(".")[2]), int(data.split(".")[1]), int(data.split(".")[0])).isocalendar()[1]

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

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

k = 2

while sheet['A'+str(k)].value != None:

k+=1

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

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

hours1 = str(int((int(input[i][0]))/number\_of\_frames/60/60/60))

minutes1 = str(int(int(input[i][0])/number\_of\_frames)//60)

seconds1 = str(int(int(input[i][0])/number\_of\_frames%60))

if len(minutes1) == 1:

minutes1 = '0' + str(minutes1)

if len(seconds1) == 1:

seconds1 = '0' + str(seconds1)

start\_time = str(hours1)+':'+ str(minutes1)+':'+str(seconds1)

hours2 = str(int((int(input[i+1][0]))/number\_of\_frames/60/60/60))

minutes2 = str(int(int(input[i+1][0])/number\_of\_frames)//60)

seconds2 = str(int(int(input[i+1][0])/number\_of\_frames%60))

if len(minutes2) == 1:

minutes2 = '0' + str(minutes2)

if len(seconds2) == 1:

seconds2 = '0' + str(seconds2)

end\_time = str(hours2)+':'+ str(minutes2)+':'+str(seconds2)

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

#writing time in excel

writing\_hours1 = str(int((int(input[i][0])+(multiplication\_hours\*60+multiplication\_minutes)\*60\*number\_of\_frames)/number\_of\_frames//3600))

writing\_minutes1 = str(int(int(int(input[i][0])+multiplication\_minutes\*60\*number\_of\_frames)/number\_of\_frames/60)%60)

writing\_seconds1 = str(int(int(int(input[i][0]))/number\_of\_frames%60))

if len(writing\_minutes1) == 1:

writing\_minutes1 = '0' + str(writing\_minutes1)

if len(writing\_seconds1) == 1:

writing\_seconds1 = '0' + str(writing\_seconds1)

if writing\_minutes1 == '60':

writing\_minutes1 = '00'

if writing\_seconds1 == '60':

writing\_seconds1 = '00'

writing\_time = str(writing\_hours1)+':'+ str(writing\_minutes1)+':'+str(writing\_seconds1)

writing\_hours2 = str(int((int(input[i+1][0])+(multiplication\_hours\*60+multiplication\_minutes)\*60\*number\_of\_frames)/number\_of\_frames//3600))

writing\_minutes2 = str(int(int(int(input[i+1][0])+multiplication\_minutes\*60\*number\_of\_frames)/number\_of\_frames/60)%60)

writing\_seconds2 = str(int(int(int(input[i+1][0]))/number\_of\_frames%60))

if len(writing\_minutes2) == 1:

writing\_minutes2 = '0' + str(writing\_minutes2)

if len(writing\_seconds2) == 1:

writing\_seconds2 = '0' + str(writing\_seconds2)

if writing\_minutes2 == '60':

writing\_minutes2 = '00'

if writing\_seconds2 == '60':

writing\_seconds2 = '00'

writing\_time2 = str(writing\_hours2)+':'+ str(writing\_minutes2)+':'+str(writing\_seconds2)

sheet['F'+str(i+k)] = writing\_time

sheet['G'+str(i+k)] = writing\_time2

if len(str(int(duration/number\_of\_frames//60))) == 1:

var1 = '0'+str(int(duration/number\_of\_frames//60))

else:

var1 = str(int(duration/number\_of\_frames//60))

if len(str(round(int(duration/number\_of\_frames%60),1))) == 1:

var2 = '0'+str(round(int(duration/number\_of\_frames%60),1))

else:

var2 = str(round(int(duration/number\_of\_frames%60),1))

sheet['H'+str(i+k)] = '00:'+ var1 + ':' + var2

sheet['I'+str(i+k)] = input[i][2]

if input[i][4] != 'НОВИНИ':

try:

sheet['J'+str(i+k)] = input[i][4]

except: pass

else:

starttimeforcut = str(int(hours1))+':'+ str(minutes1)+':'+str(seconds1)

print(start\_time)

print('0:'+var1+":"+var2)

command = "ffmpeg -i "+file+" -ss "+start\_time+" -t "+'0:'+var1+":"+var2+" -q:a 0 -map a video/sample"+str(i)+".wav"

subprocess.call(command, shell=True)

english\_text = ''

text = ''

qq = ''

r = sr.Recognizer()

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

shift = 35

myduration = shift

myoffset = 0

try:

while 1:

with harvard as source:

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

audio = r.record(source, duration = myduration, offset = myoffset)

myoffset += shift

while 1:

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

try:

print('qq'+'\n'+str(qq))

translator = google\_translator()

mm = translator.translate(qq,lang\_tgt='en')

print('mm'+'\n'+str(mm))

if len(mm) != '' and len(qq) != '':

break

else:

mm = ''

qq=''

except Exception as e: print(e)

text += qq

text += ' '

english\_text += mm

english\_text += ' '

except Exception as e: print(e)

print(text)

print(english\_text)

sheet['J'+str(i+k)] = text

delimiter = 0

rus\_answer = ''

while rus\_answer == '':

try:

delimiter += 1

lemmatizer = WordNetLemmatizer()

stemmer = PorterStemmer()

answer = []

nlp = en\_core\_web\_sm.load()

doc = nlp(english\_text)

delimiter2 = 0

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

print('-----LowPrior-----')

print(LowPrior)

print('-----LowPrior-----')

newtext = english\_text.lower().replace(',', '').split(' ')

newtext2 = []

for ii in newtext:

newtext2.append(compare\_stemmer\_and\_lemmatizer(stemmer, lemmatizer, word = ii, pos = wordnet.VERB))

for l in range(0, len(dictionary[0])):

for ii in dictionary[0][l]:

for j in newtext2:

if jellyfish.jaro\_distance(str(ii).lower(), str(j).lower()) > 0.95:

answer.append(str(ii).lower())

print('-----answer-----')

answer = list(set(answer))

print(answer)

print('-----answer-----')

clean\_result = []

copyanswer = []

answer = list(set(answer))

for ii in answer:

copyanswer.append(ii)

for ii in answer:

for j in copyanswer:

if ii != j and ii in j:

copyanswer.remove(j)

copyanswer = list(set(copyanswer))

for index in LowPrior:

copyanswer.append(str(index))

copyanswer = list(set(copyanswer))

print('-----copyanswer-----')

print(copyanswer)

print('-----copyanswer-----')

text13 = ''

checkforcomma = len(copyanswer)

for m in range(0,len(copyanswer)):

text13 += copyanswer[m]

if checkforcomma - m >1:

text13 += ', '

print('-----text13-----')

print(text13)

print('-----text13-----')

translator = google\_translator()

rus\_answer = ''

delimeter3 = 0

if len(text13)>400:

text13 = text13[0:400]

while 1:

print(delimeter3)

rus\_answer = ''

try:

rus\_answer = translator.translate(str(text13).lower(), lang\_src='en',lang\_tgt='uk')

except: pass

if len(rus\_answer)!= 0 or delimeter3 == 10:

break

delimeter3 += 1

print('-----rus\_answer-----')

print(rus\_answer)

print('-----rus\_answer-----')

sheet['N'+str(i+k)] = rus\_answer

if rus\_answer != '':

break

if delimiter == 10:

break

except Exception as e: print(e)

jornalists = ''

checkforcomma = len(input[i][3])

for m in range(0,len(input[i][3])):

if len(input[i][3][m]) < 20:

jornalists += input[i][3][m]

if checkforcomma - m >1:

jornalists += ', '

sheet['L'+str(i+k)] = jornalists

text14 = ''

for tt in english\_text.split():

text14 += tt

text14 +=' '

print('FFFFF')

print(text14)

sentence = ''

textt = text14.split()[0:25]

for index in textt:

sentence += index

sentence += ' '

print('sentence')

print(sentence)

print('sentence')

data14 = ''

data15 = ''

data15\_mass = []

try:

sid = SentimentIntensityAnalyzer()

ss = sid.polarity\_scores(sentence)

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

text15 = str(text14.split()[0])

try:

for index in range(1,len(text14.split())):

text15 += text14.split()[index]

text15 += ' '

if index % 25 == 0:

data15\_mass.append(text15)

text15 = ''

except:

if len(text15) >= 5:

data15\_mass.append(text15)

print('data15\_mass')

print(data15\_mass)

for sentence2 in data15\_mass:

sid = SentimentIntensityAnalyzer()

ss = sid.polarity\_scores(sentence2)

data15 += 'neg:' + str(ss['neg']) + ' pos:' + str(ss['pos'])

data15 += '/n'

print('data14')

print(data14)

print('data15')

print(data15.split('/n'))

neg\_index = 0

pos\_index = 0

try:

for index in data15.split('/n'):

index\_mass = index.split(' ')

print('////')

print(index\_mass)

neg\_index += float(index\_mass[0][4:])

pos\_index += float(index\_mass[2][4:])

print('neg\_index')

print('pos\_index')

print('////')

except Exception as e: print(e)

data15 = 'neg:' + str(round(neg\_index,3)) + ' pos:' + str(round(pos\_index,3))

print(data15)

sheet['O'+str(i+k)] = data15

sheet['Q'+str(i+k)] = str(round(neg\_index,3))

sheet['R'+str(i+k)] = str(round(pos\_index,3))

sheet['S'+str(i+k)] = str(round(-neg\_index,3))

except Exception as e: print(e)

try:

sheet['K'+str(i+k)] = input[i][4]

except Exception as e: print(e)

if input[i][4] == 'НОВИНИ':

sheet['M'+str(i+k)] = news\_anchor\_name

sheet['P'+str(i+k)] = 'in process...'

sheet['T'+str(i+k)] = i+1

try:

os.remove(path+'video/'+'sample'+str(i)+'.wav')

except Exception as e: print(e)

xfile.save('my.xlsx')

import xlsxwriter

import datetime

import plotly.graph\_objects as go

import matplotlib.pyplot as plt

import numpy as np

from openpyxl import Workbook

from openpyxl.drawing.image import Image

from openpyxl import load\_workbook

import plotly.graph\_objects as go

from openpyxl import load\_workbook

xfile.save('my.xlsx')

wb = load\_workbook('Report\_Time\_Bands\_{\_Рейтинг\_Украина\_}\_20210218\_131006.xlsx',data\_only=True)

sheet = wb.get\_sheet\_by\_name(data)

mass = []

try:

for i in range(4,3903):

try:

mass.append(sheet.cell(i, 4).value)

except:mass.append('')

except:pass

wb = load\_workbook('my.xlsx')

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

index\_from\_my\_table = []

i=2

while sheet.cell(i, 20).value != None:

try:

index\_from\_my\_table.append(sheet.cell(i, 20).value)

except:

index\_from\_my\_table.append('')

i+=1

rannge = len(index\_from\_my\_table)

while index\_from\_my\_table[-1] != 1:

index\_from\_my\_table = index\_from\_my\_table[:-1]

rannge -= len(index\_from\_my\_table)

mas1 = []

try:

for i in range(len(index\_from\_my\_table)+1,len(index\_from\_my\_table)+rannge+2):

try:

mas1.append(sheet.cell(i, 15).value.split(' ')[0:2])

except:

mas1.append('')

except:

pass

negativ = []

positiv = []

for index in mas1:

if index == '':

negativ.append(0.0)

positiv.append(0.0)

else:

negativ.append(round(float(index[0][4:]), 2))

positiv.append(round(float(index[1][4:]), 2))

print(negativ)

print(positiv)

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

negativ[i] = -negativ[i]

lineplotmass = []

for index in range(0,len(negativ)):

lineplotmass.append(negativ[index]+positiv[index])

lineplotmass = list(np.around(lineplotmass, decimals = 2))

from openpyxl import load\_workbook

wb = load\_workbook('my.xlsx')

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

duration = []

try:

for i in range(len(index\_from\_my\_table)+1,len(index\_from\_my\_table)+rannge+2):

try:

minute = int(sheet.cell(i, 8).value[3:].split(':')[0])\*60

seconds = int(sheet.cell(i, 8).value[3:].split(':')[1])

duration.append((minute+seconds))

except: duration.append(0)

except:pass

lineplotwithdurationmass = []

for index in range(0,len(duration)):

for index2 in range(0,duration[index]):

try:

lineplotwithdurationmass.append(lineplotmass[index])

except:pass

for index in range(0,100):

lineplotwithdurationmass.append(0)

x = np.arange(len(lineplotwithdurationmass))

fig = go.Figure()

for index\_\_ in range(0,300):

lineplotwithdurationmass.insert(0, 0)

lineplotwithdurationmass.append(0)

x = [i/60 for i in range(1,len(lineplotwithdurationmass)+1)]

import pandas as pd

df = pd.DataFrame({'data':data,'index': x[0:len(mass)], 'mood': lineplotwithdurationmass[0:len(mass)], 'rating': mass, 'program': programm})

import openpyxl

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

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

k = 2

while sheet['A'+str(k)].value != None:

k+=1

kk = 0

while True:

try:

sheet['E'+str(k+kk)] = df['rating'][kk]

sheet['A'+str(k+kk)] = k+kk-1

sheet['B'+str(k+kk)] = data

sheet['C'+str(k+kk)] = df['index'][kk]

sheet['D'+str(k+kk)] = df['mood'][kk]

sheet['F'+str(k+kk)] = df['program'][kk]

kk += 1

except: break

xfile.save('index.xlsx')