ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»»
МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ
ДЕПАРТАМЕНТ КОМПЬЮТЕРНОЙ ИНЖЕНЕРИИ

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

по дисциплине: “Сетевые видеотехнологии”
на тему: “Поиск схожих видеозаписей методом перцептивного хэша”

Выполнили:
Студенты 3 курса, группы БИВ 161:
Титова Екатерина Егоровна
Хасанова Дина Фаридовна

Проверил:
Королев Денис Александрович,
Доцент

_____

г. Москва, 2019 г.

Аннотация

В данной работе мы описываем один из реально действующих алгоритмов поиска схожих видеозаписей в сетевой папке и проводим тестирование на реальной библиотеке видеозаписей. Этот алгоритм оснащен математической базой и реализован на языке программирования Python.

Введение

После исследования методов поиска схожих видеозаписей мы разобрались с понятиями «Нечеткий дубликат видео», «Сигнатура видеозаписи» и ознакомились с существующими методами исследования данной темы в глобальном пространстве. Целью команды было реализовать наиболее подходящий механизм, позволяющий выявлять копии видеосегментов в библиотеке, исходя из условий сдачи. Отталкиваясь от выборки в предыдущем исследовании, вариант с использованием метода рассмотрения видеозаписи как ДНК стал наиболее возможным исходя из условий руководителя проекта. По сравнению с другими подходами, его можно выделить точным. Также данная реализация использует менее затруднительные математические методы для расчета. В нем сравнение между представлениями реализуется только внутри скользящего окна. Предположим, его размерность n. Тогда видеоряды будут сравниваться на основе n-граммной матрицы с заданными параметрами константы «n» и размера шага – сдвига окна

после сравнения. Профилем видео станет узел дерева, а его листьями – видео ДНК. Далее, помимо окон матрицы расстояний, сравниваем способом машинного обучения Neighbor Joining профили, после чего реализуем дерево принятых решений. Так мы получаем огромный поток массивов данных из листьев различных деревьев, представляющие каждые сцены. И для заданного количества одинаковых кадров (со схожим параметром ДНК) приходится также сравнивать с учетом нового параметра – количества допустимых схожих кадров в видеосегменте, каждый с каждым. Даже при выборке ключевых кадров в видеозаписи, во время попытки реализации метода на наших ПК не хватило мощности для обработки и составления деревьев ДНК, что говорит о невозможности реализации на текущих условиях в режиме данного курса.

Обсудив возможные перспективы с руководителем проекта, командой было решено попробовать иной метод, который объединяет некоторые варианты из предыдущего исследования.

Вариант со сравнением путем применения хэширования кадров предлагал Wan-Lei Zhao и его команда в исследовании «Large-scale near-duplicate Web video search: challenge and opportunity» [1]. Основная идея локально-чувствительного хэширования заключается в подборе хэш-функций для высокой вероятности попадания похожих объектов при некоторых изменениях в одну корзину. Данная идея применяется для отображения цветовой гистограммы каждого ключевого кадра на бинарный вектор. Эти диаграммы в последующем сравниваются для похожих кадров. Таким образом, показатели каждого кадра представляют из себя набор дискретных величин. В нашем методе начало применения также берется из хэширования кадров, однако не по основным точкам кадра, а по всем. Далее, производя алгоритмы, мы вспомнили метод с

применением порядковых подписей, в котором видеоролик содержит особые события с набором ключевых кадров. Таким образом, измеряя длины между ключевыми кадрами с помощью гистограмм яркости, записываем их в одномерную последовательность, которую называют сигнатурой или подписью. Заимствуя выделение ключевых кадров и сжатие картинки до матрицы цветов, в нашем методе используем алгоритм «ahash» для записи хэшей и сравниваем известным в данной тематике методом Хэмминга.

Таким образом, наш метод позволяет воспользоваться наиболее эффективными для условий выполнения проекта данными, сочетая методы, исследуемые ранее. Алгоритм «AHash» является самым простым (но при этом эффективным) способом создания хэшей для изображений, который дает небольшое количество сбоев при тестировании. В основной части исследования рассмотрим реализацию данного метода в детальных подробностях.

Основная часть

Метод перцептивного хэширования кадров в данной главе будет рассмотрен подробнее для реализации поиска схожих видеозаписей. Для начала стоит уточнить, что нечеткий дубликат видео — это неполное или частичное совпадение объекта с другим объектом подобного класса. Дубликаты классифицируются на естественные и искусственные. В естественных дубликатах объекты схожи при схожих условиях, а искусственные берут истоки на основе оригинального видео, полностью копируя его. [2] Таким образом, пиратские видео сохраняют угол съемки, порядок изменения кадров, и могут отличаться четкостью и настройками яркости, а также эти версии могут быть сжаты. Наша программа позволяет выявить как искусственные, так и естественные копии, однако в первом случае все значительно проще.

Теперь пойдем постепенно по программе и разберем подробно, как это работает. На вход подается абсолютный путь к папке, в которой хранятся видеозаписи. Важно уточнить, что на данном этапе реализации программа считывает только файлы формата MP4. Таким образом, если такая видеозапись всего одна или их не имеется вообще, сравнение не будет произведено. Если мы нашли видеозаписи для сравнения, создается новая папка с названием “pict”, в которую будут нарезаны основные кадры сцен с помощью FFMPEG для дальнейшей обработки. Далее, пользуясь ретрансляцией кадров — протоколом канального уровня сетевой модели OSI, узнаем FCS (Frame Check Sequence) — проверочную последовательность кадра (служит для обнаружения ошибок и формируется аналогично циклическому коду HDLC) [3]. С ее помощью мы

можем, записав в название файла эту величину, узнать при обращении к файлу время кадра в видеозаписи для записи результата сравнения.

Теперь основной задачей становится формирование перцептивных хэшей и их сравнение для поиска одинаковых и схожих кадров. Понятие «хэширование» означает однозначное и точно известное вычисление набора символов фиксированной длины на основе входных данных произвольной длины. При этом изменение хотя бы одного символа в исходных данных гарантирует, что и полученная фиксированная строка будет иной. Можно сказать, что хэширование это «снятие отпечатка» с большого набора данных [4]. В основе перцептивного хэширования лежат методы, позволяющие получить хэш из изображения и однозначно определить содержимое с помощью символов. Перцептивные хэши — это другая концепция по сравнению с криптографическими хэш-функциями. В криптографии каждый хэш является случайным. Данные, которые используются для генерации хэша, выполняют роль источника случайных чисел, так что одинаковые данные дадут одинаковый результат, а разные данные — разный результат. Если хэши отличаются, значит, данные разные и наоборот [5]. В отличие от них, перцептивные хэши можно сравнивать между собой и делать вывод о степени различия двух наборов данных, то есть информация о схожести данных по ним имеет не бинарную величину. Таким образом, применяя эту методологию, можно определить не только искусственный дубликат видеозаписи, но и естественный.

Есть несколько способов узнать перцептивный хэш из изображения. В основном разделяют AHash и PHash реализации. Первый метод является самым простым, поскольку берется из усредненных значений. Второй же подразумевает работу с дискретным косинусным преобразованием и

производит итерацию для точности результата. С учетом времени для выполнения этой работы, мы выбрали Ahash.

Два стартовых шага при любом способе записей хэшей одинаковые. Сначала изображение уменьшают по размеру (в нашем случае до 64 бит), затем убирают цветовые параметры, оставляя вариант в черно-белом формате. В AHash следующим шагом вычисляется среднее значение для всех 64 битов изображения. Идем пошагово по каждому биту, сравниваем его со средней величиной и присваиваем битовое значение: 0 или 1. Таким образом, если сжимать, растягивать и масштабировать как либо картинку, хэш не изменится. Любые попытки изменить яркость и контраст также не позволят обмануть эту систему - хэш остается неизменным. Поскольку Ahash не закреплен сложными вычислениями, скорость выполнения алгоритма быстрая. Последним шагом для получения хэша является перевод 64 битов в единое значение. Если не менять порядок записи битов, он не имеет значения. В нашем методе преобразование слева направо, сверху вниз. Эта реализация в коде представлена в функции average_hash.

Далее, получив массив хэшей для каждого видеоряда, нужно их сравнить. Один из самых распространенных методов является поиск расстояния Хэмминга — числа позиций, в которых соответствующие символы двух слов одинаковой длины различны. Его используют как раз для определения разницы между несколькими величинами с одной размерностью [6]. В основе алгоритма лежит теорема: “Если для двух строк a и b расстояние HD(a, b) <= k, то если поделить строки a и b на подстроки методом rcut используя фактор сегментации r >= ⌊k/2⌋ + 1 обязательно найдется, по крайней мере, q= r − ⌊k/2⌋ подстрок, когда их расстояние не будет превышать единицу, HD (ai, bi) <= 1.” Выделение подстрок из базовой строки выполняется по следующим принципам:

выбирается значение, названное фактором сегментации, которое удовлетворяет условию r >= ⌊k/2⌋ + 1, где длина r, встречающихся впервые подстрок будет иметь длину ⌊m/r⌋, а последние подстроки ⌈m/r⌉. Где m — это длина строки, ⌊ — округление до ближайшего снизу, а ⌉ округление до ближайшего сверху. В программном коде алгоритм описан в функции hash_distance. После выполнения она выдает значение - то самое расстояние. Если оно принимает нулевое значение,сцены видеозаписи имеют абсолютное сходство, т.е. являются искусственными копиями.

Все алгоритмы, описанные выше, реализованы в процедуре search_similar, которая выполняет поиск схожих видеозаписей. В начале она начинает идти последовательно по всем создавшимся папкам в ходе резки видеозаписей. Как мы писали ранее, наша программа позволяет искать не только искусственные дубликаты, но и естественные. В таком случае, перцептивное хеширование позволяет выдать некоторые сходства. В логическую функцию is_look_alike на вход подаются 2 изображения, которые необходимо сравнить, и параметр tolerance — константа, которая определяет схожесть видеозаписи. Внутри функции это числовое значение сравнивается с расстоянием Хэмминга, и если оно равно ему или меньше, видеозаписи имеют схожие черты. Задать эту величину пользователь может вручную, в зависимости от потребностей применения наших алгоритмов.

В результате выполнения программы если найдены точные или схожие параметры видеозаписей, данные о времени начала и конца кадров с копируемыми элементами записываются в лог отдельного созданного файла-результата. При отсутствии копий лог остается пустым. Затем видеозапись перемещается в папку с изображениями, а название папки

изменяется на название видеозаписи, что дает понять, что работа с этим файлом завершена.

Тестирование

После успешных компиляций программы, тестирование происходит на двух видеозаписях со схожими частями: кусок мультфильма был записан на мобильный телефон, что является пиратской копией файла. До запуска программы в папке содержатся несколько видео MP4 , которые необходимо сравнить и другие файлы (рис.1). video2.mp4 является схожим файлом - записано video1 на мобильный телефон. В папке sample1 находится видеоряд sample1.mp4, в sample2 файл sample2.mp4. Они содержат одинаковые части кадров с файлом video1.mp4.

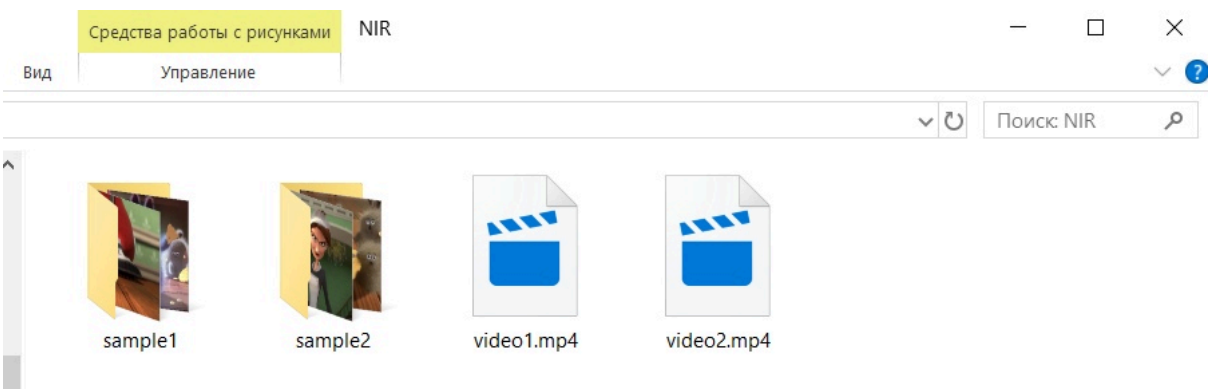

Рисунок 1. Содержимое папки до запуска программы.

При запуске программы запрашивается абсолютный путь к папке, которую нужно проверить на наличие копий (рис.2).

```
In [*]: print('Введите путь к файлам')
        dir_main = input()
        subdir = dir_main + '/pict/'
        main()
```

```
C:/Users/user/Desktop/NIR/samples
Введите путь к файлам
```

Рисунок 2. Запуск программы

После выполнения программы, исходная папка стала выглядеть как показано на рисунке 3. Появились новые папки video1 и video2, в которых содержатся логи с временем копий, если такие нашлись.

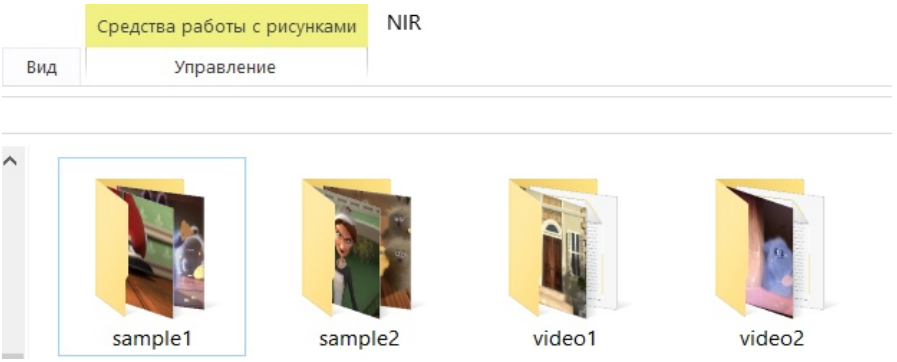

Рисунок 3. Содержимое папки после запуска программы.

Внутри папки video1, в которой по нашим соображениям должны были присутствовать копии, содержатся нарезанные кадры, сам видеофайл и лог (рис. 4).

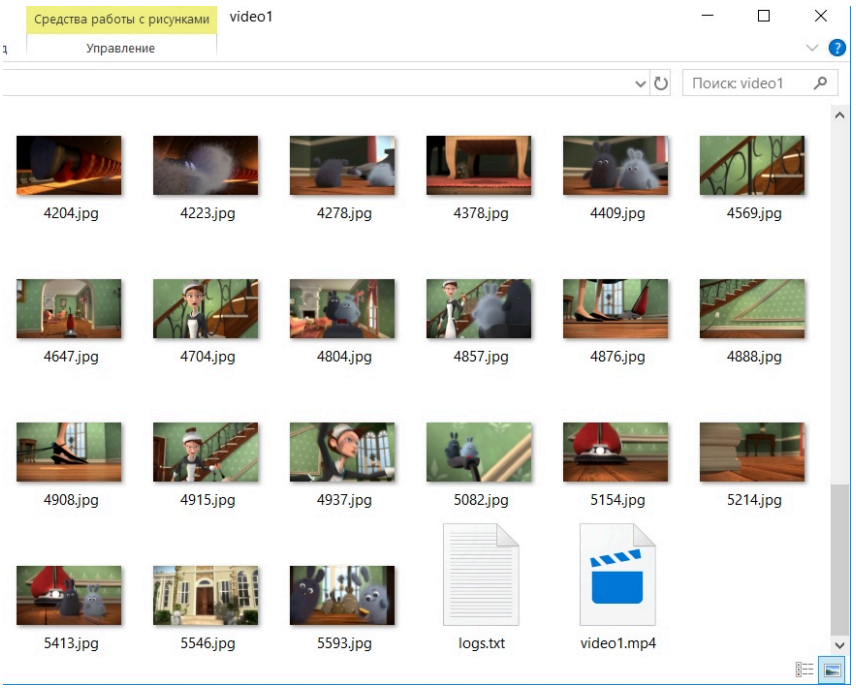

Рисунок 4. Содержимое папки video1 после запуска программы.

В файле logs.txt в формате Схожее видео:_Время в исходном файле_и_Время в сравниваемом файле_(Имя сравниваемого файла) лежат все выявленные копии (рисунок 5).

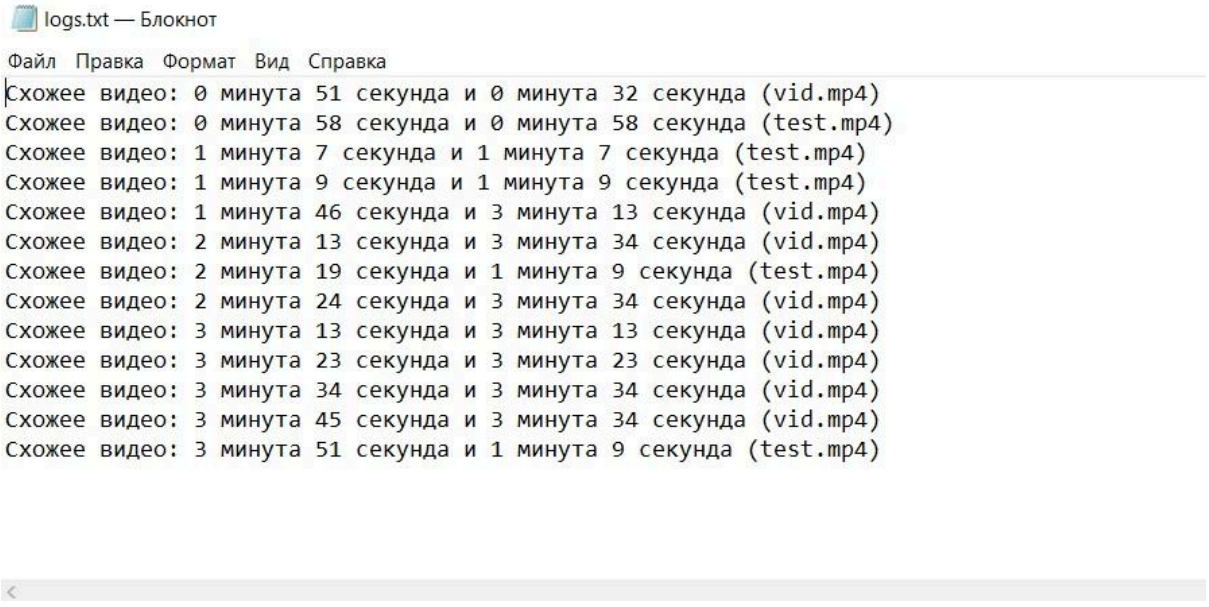

Рисунок 5. Содержимое файла logs.txt

Заключение

В ходе выполнения данной научно-исследовательской работы, на основе предыдущего анализа существующих методов поиска схожих видеозаписей, был разработан способ, позволяющий искать пиратские копии и определять похожие файлы, в зависимости от требований. Команда разобрала математическую составляющую метода, произвела описательную составляющую работы, реализовала программно метод на Python и вставками на FFMPEG и протестировала его. В результате мы получили работающую программу, которая действительно позволят обнаружить копии видеоряда.

В дальнейшем развитии планируется улучшить алгоритмы проекта для применения на хранилище данных видеозаписей НИУ ВШЭ для поиска схожих файлов (ночные съемки, лекции и т.д.) и работы с искусственными копиями.

Приложение. Исходный код

Код опубликован на GitHube Хасановой Дины. Ссылка: https://github.com/dinahas/duplicate_search

Список литературы

1. Wan-Lei Zhao, Song Tan and Chong-Wah Ngo. Large-scale near-duplicate web video search: challenge and opportunity. https://vireo.cs.cityu.edu.hk/papers/icme09-wanlei.pdf

2. Никитин Илья Константинович. Элементы поиска нечетких дубликатов видео. https://www.slideshare.net/w-495/nkp-2015

3. Статья из свободной энциклопедии. Frame relay. https://ru.wikipedia.org/wiki/Frame_relay

4. Сайт “Что это такое?” Хэш. https://chto-eto-takoe.ru/hash

5. Анатолий Ализар. «Выглядит похоже». Как работает перцептивный хэш. https://habr.com/ru/post/120562/

6. Статья из свободной энциклопедии. Расстояние Хэмминга. https://ru.wikipedia.org/wiki/Расстояние_Хэмминга