Лабораторная работа № 4.3
Умная система рекомендаций и воспроизведения видео

Теоретическая часть:

Интересно, как Google придумывает фильмы, похожие на те, что вам нравятся? Прочитав этот пост, вы сможете создать для себя одну такую систему рекомендаций.

Оказывается, есть (в основном) три способа создать механизм рекомендаций:

1. Система рекомендаций на основе популярности
2. Система рекомендаций на основе контента
3. Механизм рекомендаций на основе совместной фильтрации

Теперь вы можете подумать: «Это интересно. Но в чем разница между этими системами рекомендаций? ». Позвольте мне помочь вам с этим.

Система рекомендаций на основе популярности:

Пожалуй, это самый простой вид рекомендательного механизма, с которым вы столкнетесь. Список тенденций, который вы видите на YouTube или Netflix, основан на этом алгоритме. Он отслеживает количество просмотров для каждого фильма / видео, а затем перечисляет фильмы на основе просмотров в порядке убывания (от максимального количества просмотров до самого низкого количества просмотров). Довольно просто, но эффективно.

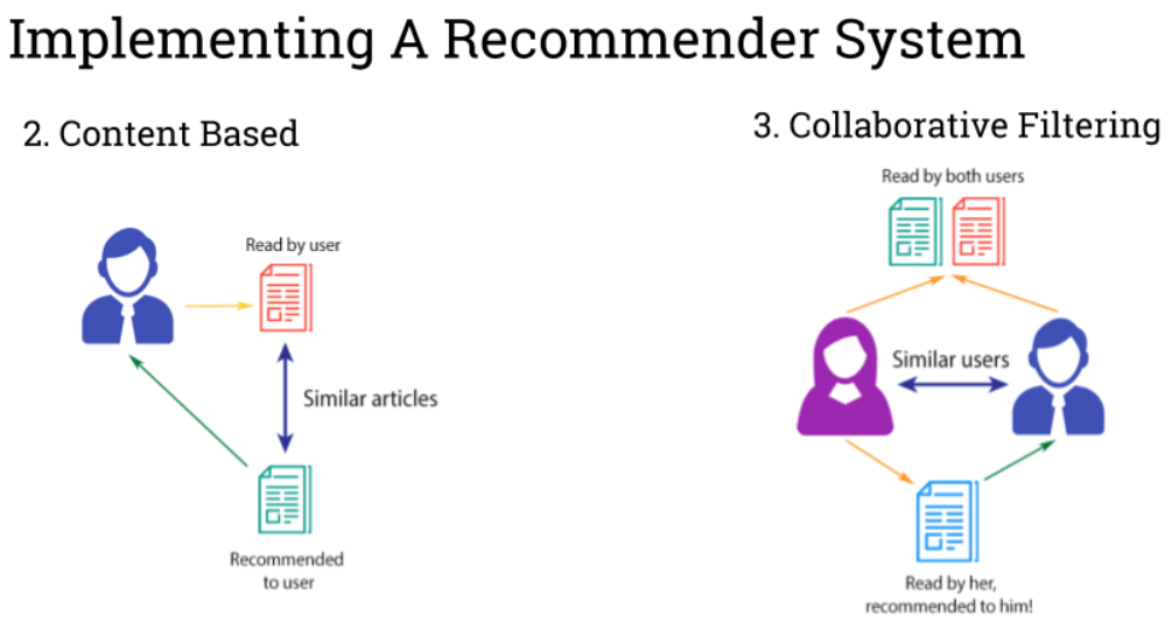
Система рекомендаций на основе содержания:

Этот тип рекомендательных систем принимает в качестве входных данных фильм, который в настоящее время нравится пользователю. Затем он анализирует содержание (сюжет, жанр, актеры, режиссер и т. Д.) Фильма, чтобы найти другие фильмы с похожим содержанием. Затем он ранжирует похожие фильмы по их показателям схожести и рекомендует пользователю наиболее релевантные фильмы.

Система рекомендаций на основе совместной фильтрации:

Этот алгоритм сначала пытается найти похожих пользователей на основе их действий и предпочтений (например, оба пользователя смотрят фильмы одного типа или фильмы, снятые одним и тем же режиссером). Теперь, между этими пользователями (скажем, A и B), если пользователь A видел фильм, который пользователь B еще не видел, то этот фильм рекомендуется пользователю B и наоборот. Другими словами, рекомендации фильтруются на основе взаимодействия схожих пользовательских предпочтений (отсюда и название «Совместная фильтрация»). Типичное применение этого алгоритма

можно увидеть на платформе электронной коммерции Amazon, где вы можете увидеть списки «Покупатели, которые просматривали этот товар, также просматривали» и «Покупатели, которые купили этот товар, также купили».



Другой тип рекомендательной системы может быть создан путем смешивания свойств двух или более типов рекомендательных систем. Этот тип рекомендательных систем известен как гибридная рекомендательная система.

Обнаружение сходства

Мы знаем, что наша система рекомендаций будет основана на содержании. Итак, нам нужно найти фильмы, похожие на данный фильм, а затем рекомендовать эти похожие фильмы пользователю. Логика довольно проста. Верно?

Но ждать…. Как мы можем узнать, какие фильмы похожи на данный фильм в первую очередь? Как мы можем узнать, насколько похожи (или не похожи) два фильма?

Начнем с чего-то простого и понятного.

Предположим, вам даны следующие два текста:

Текст A: Лондон Париж Лондон

Текст B: Париж Париж Лондон

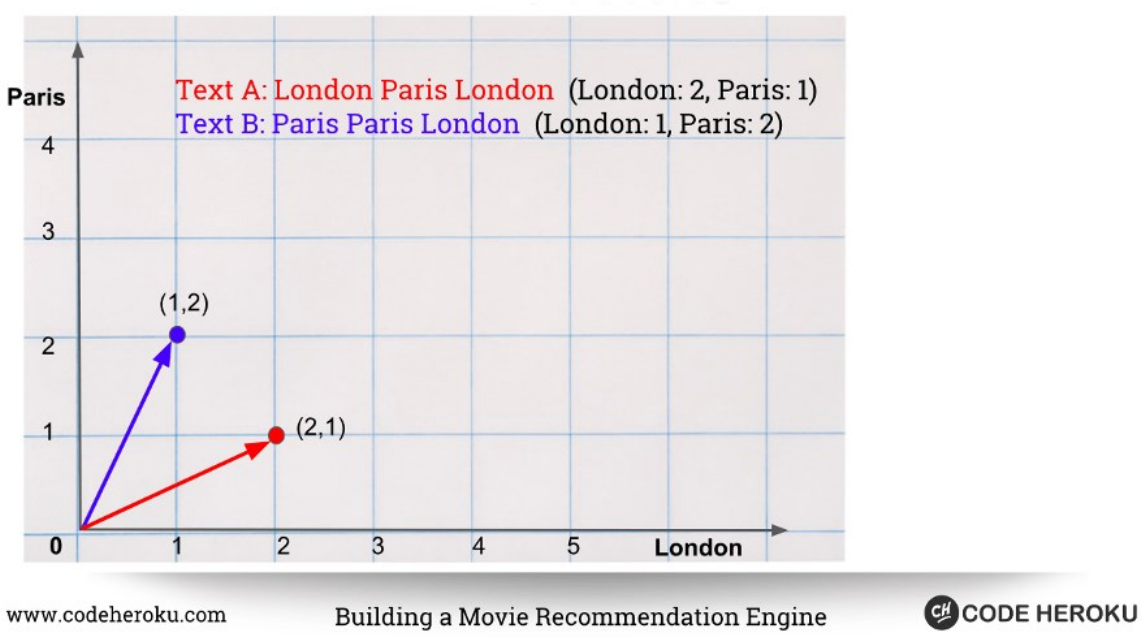
Как бы вы обнаружили сходство между текстом A и текстом B?

Давайте проанализируем эти тексты….

1. Текст A: содержит слово «Лондон» 2 раза и слово «Париж» 1 раз.
2. Текст B: содержит слово «Лондон» 1 раз и слово «Париж» 2 раза.

Теперь, что произойдет, если мы попытаемся представить эти два текста в двухмерной плоскости (с «Лондоном» на оси X и «Парижем» на оси Y)? Давай попробуем это сделать.

Это будет выглядеть так-



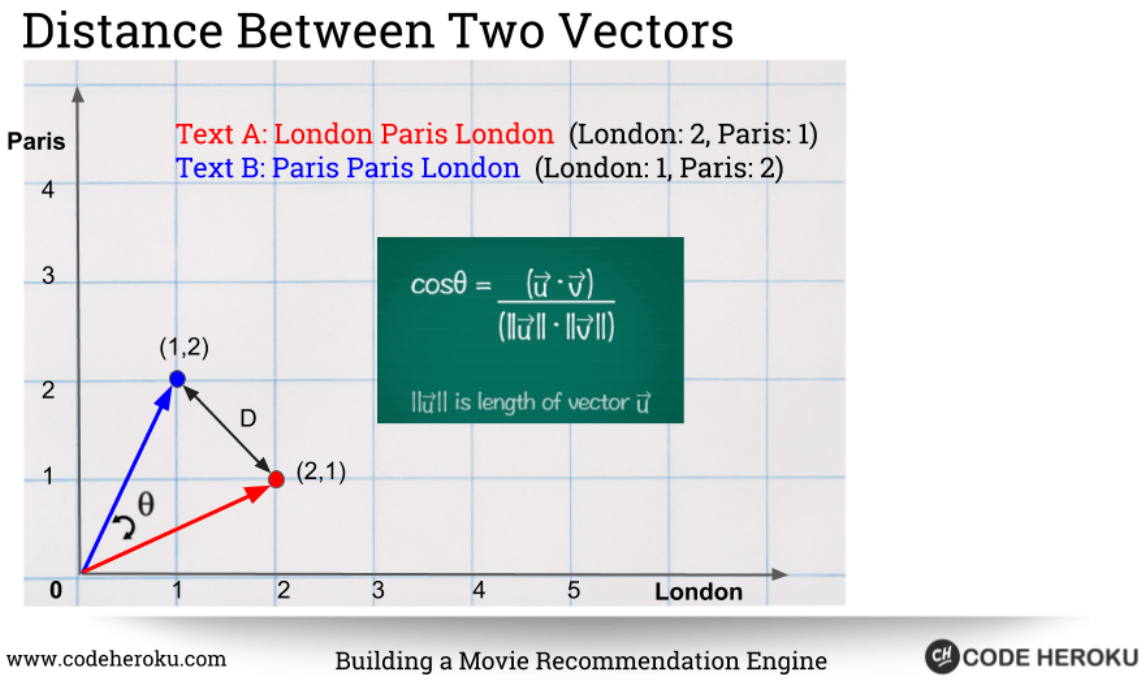
Здесь красный вектор представляет «Текст A», а синий вектор представляет «Текст B».

Теперь мы графически изобразили эти два текста. Итак, теперь можем мы выяснить сходство между этими двумя текстами?

Ответ: «Да, можем». Но как именно?

Эти два текста представлены в виде векторов. Верно? Итак, можно сказать, что два вектора подобны, если расстояние между ними небольшое. Под расстоянием мы понимаем угловое расстояние между двумя векторами, которое обозначается θ (тета). Поразмыслив дальше с точки зрения машинного обучения, мы можем понять, что значение **cos θ** имеет для нас больше смысла, чем значение θ (тета), потому что функция косинуса (или «cos») отобразит значение θ в первом квадранте от 0 до 1 (помните? cos 90 ° = 0 и cos 0 ° = 1).

Из школьной математики мы можем вспомнить, что на самом деле существует формула для определения cos θ между двумя векторами. Смотрите картинку ниже-



Не пугайтесь, нам не нужно реализовывать формулу с нуля для определения cos θ. У нас есть наш друг Scikit Learn, чтобы рассчитать это для нас.

Практическая часть:

Создадим скрипт с использованием модели машинного обучения для рекомендации видео на основе уже просмотренных. Можно воспользоваться библиотекой scikit-learn и pandas для решения данной задачи. Эти библиотеки уже содержат встроенные методы для работы с векторизацией текста и вычисления косинусной схожести. Ниже представлен пример модели на Python, в котором предполагается, что у вас есть список фильмов и их описания:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
# Пример данных об описании фильмов
data = {
    'title': ['Фильм 1', 'Фильм 2', 'Фильм 3'],
    'description': ['Описание фильма 1', 'Описание фильма 2',
'Описание фильма 3']
}
# Создание DataFrame
movies = pd.DataFrame(data)
# Векторизация описания фильмов
```

```
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movies['description'])
# Вычисление косинусной схожести
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
# Функция для получения рекомендаций
def get_recommendations(title, cosine_sim=cosine_sim):
    idx = movies[movies['title'] == title].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1],
reverse=True)
    sim_scores = sim_scores[1:2]
    movie_indices = [i[0] for i in sim_scores]
    return movies['title'].iloc[movie_indices]
# Пример использования
movie_title = input("Введите название фильма: ")
recommendation = get_recommendations(movie_title)
print("Рекомендуемый фильм:", recommendation.values[0])
```

В данном примере предполагается, что у вас уже есть данные о фильмах и их описания в виде DataFrame. Если у вас есть файл с данными, вы можете его загрузить с помощью метода pd.read_csv().

Вам необходимо заменить переменную data на список с названием видео, а в качестве описания ссылкой на youtube

Для воспроизведения видео из YouTube на Raspberry Pi 4 вам понадобится использовать команду omxplayer и URL-адрес видео с YouTube. Важно помнить, что воспроизведение видео из YouTube может потребовать загрузки видео сначала с помощью утилиты youtube-dl, а затем воспроизведения с помощью omxplayer.

YouTube, однако, имеет свои правила и политику, поэтому важно учитывать их в случае использования контента из YouTube.

Ниже приведен код, который показывает, как можно воспроизвести видео из YouTube на Raspberry Pi 4.

```
import os

def play_youtube_video(video_id):
    os.system(f"chromium-browser
https://www.youtube.com/watch?v={video_id}")


def main():
    youtube_video_id = 'Kr-V4IgJFes'  # Замените нафактический
идентификатор видео с YouTube
    play_youtube_video(youtube_video_id)


if __name__ == "__main__":
    main()
```

> Чтобы воспроизвести видео из YouTube, вам необходимо извлечь идентификатор видео из URL-адреса видео на YouTube, например, в URL https://www.youtube.com/watch?v=your_video_id, your_video_id это идентификатор видео.

В качестве ссылки необходимо использовать description с url из датафрейма созданного в предыдущем скрипте.

Таким образом на выходе у вас должна получиться программа, которая по видео запускает вам похожее, которое может вам понравится.

Обратите внимание, что этот примерный код демонстрирует общий подход, и в реальном программном обеспечении потребуется более тщательное управление ошибками, механизмы безопасности и т. д. Помните, что хорошей практикой будет уведомить официальные источники контента или соблюдать правила и условия использования контента при использовании его в вашем приложении.

Усложняем задачу:

1) Добавьте подключение к базе данных Postgres и берите данные о названии и ссылке оттуда
2) Добавьте определение геолокации пользователя и добавьте этот параметр в алгоритм подбора рекомендованного видео
3) Запрашивайте у пользователя помимо названия просмотренного видео еще и возраст и пол. Добавьте эти параметры в модель рекомендации видео

***

4) Создайте визуальный интерфейс веб платформы для проигрывания видео и выводите его в вашей веб форме, вместо встроенного проигрывателя.