УДК 004.896

АЛГОРИТМ РЕКОМЕНДАЦИЙ МЕРОПРИЯТИЙ

Ермолович Д.С. [1], студент гр.053504

Белорусский государственный университет информатики и радиоэлектроники[1]
г. Минск, Республика Беларусь

Боброва Н.Л. – доцент кафедры информатика

Аннотация. В данной научной работе представлен уникальный алгоритм рекомендаций, который эффективно и умно решет задачу рекомендаций мероприятий пользователям, основанных на их предпочтениях. Рекомендательный алгоритм построен на *content-based filtering technique*.

Ключевые слова. Машинное обучение, математика, анализ данных, *natural language processing*.

ВВЕДЕНИЕ

Идея рекомендательной системы заключается в том, чтобы предлагать конкретные развлечения, такие как рестораны, бары, футбольные или хоккейные матчи, лекции и т. д., исходя из предпочтений пользователя.

Это очень важная задача, ведь пользователь приходит в приложение поиска досуга и не знает, что он хочет посетить, но исходя из его предпочтений, алгоритм рекомендаций рекомендует ему то, что ему нравится. Пользователь будет счастлив и останется в приложении.

Сегодня самые популярные приложения в мире, такие как *Netflix*, *TikTok*, *Amazon*, *YouTube*, используют рекомендательные системы. Их успех напрямую зависит от рекомендательных систем. Вот почему применение этой технологии в приложениях поиска досуга будет очень полезным и принесет много денег.

В данной научной работе будет представлен уникальный алгоритм рекомендаций, который эффективно и умно решет задачу рекомендаций пользователю мероприятий основанных на их предпочтениях. Рекомендательный алгоритм построен на *content-based filtering technique*.

Проблемы:

Проблема и сложность разработки заключается в том, что не существует шаблонов к решению данной задачи. Рекомендация музыки происходит по одному алгоритму, рекомендация фильмов по-другому. В данной научной работе рассматривается задача рекомендаций мероприятий пользователем, основанных на их предпочтениях. Был разработан алгоритм рекомендаций, который учитывает предпочтения пользователей, то есть мероприятия, которые он посетил, и время, когда он их посетил, так как мероприятия, которые пользователь посетил давно, нужно рекомендовать с меньшей вероятностью.

1 АЛГОРИТМ

Существует простой подход к решению данной задачи. Берётся последние мероприятия, которые посетил пользователь, потом с помощью нейронной сети она представляется, как вектор, находятся самые близкие к этим векторам вектора и просто самые близкие события рекомендуем

Рассматриваемый алгоритм будет учитывать время и предпочтения пользователя.

Рассмотрим пользователя с *id* 32.

*user*_32: *e*1, *e*2, *e*3, …, *e*n.

*e*1 – последнее событие, которое посетил пользователь.

*e*n – самое старое событие, которое посетил пользователь.

Рассмотрим последние пять событий, которые посетил пользователь.

V = {*e*1, *e*2, *e*3, *e*4, *e*5}.

Найдем для них ближайшие 5 векторов из базы данных, в которой хранятся все активные векторные события.

E = [
[*e*11, *e*12, *e*13, *e*14, *e*15]
[*e*21, *e*22, *e*23, *e*24, *e*25]
[e31, *e*32, *e*33, *e*34, *e*35]
[*e*41, *e*42, *e*43, *e*44, *e*45]
[*e*51, *e*52, *e*53, *e*54, *e*55]
]

$e_{i,j}$ — это расстояние между вектором i, вектор мероприятие, который посетил пользователь, и j, вектор мероприятия, которое сейчас активно. Эта величина больше единицы и была получена с помощью технологии FAISS.

К каждой величине из матрицы E применим функцию, которая даст ближайшим векторам большой вес, а далеким маленький. Функция определяется по формуле

$$relevance\_weight_{i,j} = \frac{num1}{(e_{i,j}+0.0001)^d}, \quad (1)$$

где *num1* – числитель. $e_{i,j}$- расстояние между вектором i и вектором j, d – степень.

После применения этой функции для матрицы E, получилась R-матрица.

R = [
[*rw*11, *rw*12, *rw*13, *rw*14, *rw*15]
[*rw*21, *rw*22, *rw*23, *rw*24, *rw*25]
[*rw*31, *rw*32, *rw*33, *rw*34, *rw*35]
[*rw*41, *rw*42, *rw*43, *rw*44, *rw*45]
[*rw*51, *rw*52, *rw*53, *rw*54, *rw*55]
]

Кроме того, старые события должны быть наказаны.

Штрафное время определяется по формуле

$$time\_weight(k) = \frac{num2}{func(k)}, \quad (2)$$

где *k* – вес времени встречания. Например, если *k* = 1, то событие самое новое. Если *k* = *n*, то событие является самым старым. На практике каждой точке времени задается значение, которое отражает порядок следования. $func(k)$ — это функция, основная цель которой, делать штраф более резким или сглаживать его. Она может быть линейной, логарифмической, степенной, квадратичной и так далее.

T = [
[*tw*1, *tw*1, *tw*1, *tw*1, *tw*1]
[*tw*2, *tw*2, *tw*2, *tw*2, *tw*2]
[*tw*3, *tw*3, *tw*3, *tw*3, *tw*3]
[*tw*4, *tw*4, *tw*4, *tw*4, *tw*4]
[*tw*5, *tw*5, *tw*5, *tw*5, *tw*5]
]

Перемножим матрицы релевантности и времени.

Мы получим матрицу *P*.

Матрица P представляет собой матрицу весов интересов пользователя. Для того, чтобы увеличить вероятность сэмплирования интересных мероприятий для пользователя и уменьшить вероятность сэмплирования неинтересных мероприятий, происходит домножение матрицы на коэффициенты. Если Pij < медиана, то происходит умножение на low_median_coefficient < 1, если Pij >= медиана, то происходит умножение на up_median_coefficient, тем самым интересные мероприятия стали еще интереснее, а неинтересные менее интересными.

Затем происходит процесс преобразования матрицы в вектор и передача его в SoftMax. Получается распределение вероятностей и берется из него выборка.

Softmax определяется по формуле

$$P_i = \frac{e^{\frac{z_i}{T}}}{\sum e^{\frac{z_k}{T}}}, \quad (3)$$

где *T* – температура.

Гиперпараметры:

1 *T*: 1,0.5, 1.5;

2 *K*: *log(k), k*^2, *k*^3, *k*^0.5;

3 Количество выбранных мероприятий.

4 num1, num2
5 low_median_coefficient, up_median_coefficient
6 step, start_pun

Алгоритм рекомендаций является универсальным, это означает, что компания, которая будет им пользоваться, может лично настраивать значения гиперпараметров в зависимости от их целей. Например: если компания хочет, чтобы пользователям рекомендовались только мероприятия, которые ему точно понравятся без привязки ко времени, то low_median_coefficient -> 0, up_median_coefficient -> 100, temperature -> 15, step -> 0. Если же компания хочет, чтобы пользователям рекомендовались только новые мероприятия, тогда step -> 100 и f = x**100. Если же компания хочет, чтобы пользователям рекомендовались балансированные мероприятия, то коэффициенты нужно балансировать.

2 ТЕСТИРОВАНИЕ

2.1 Сбор данных

Первым шагом в тестировании модели системы рекомендаций является сбор данных. Вручную были найден и построен набор данных. Набор данных представляет собой список *json*. Каждый *json* описывает каждое событие. Имеет ключи: категория, заголовок, теги, описание. Было рассмотрено 10 разных категорий: спорт, конференции, прогулки, путешествия и походы, экскурсии, здоровье, книжные вечера, выставки, онлайн-лекции, музыка и танцы. Эти события взяты из белорусских и российских источников.

Некоторых событий было много, некоторых мало, поэтому распределение данных неравномерно. На рисунке 1 мы видим распределение данных.

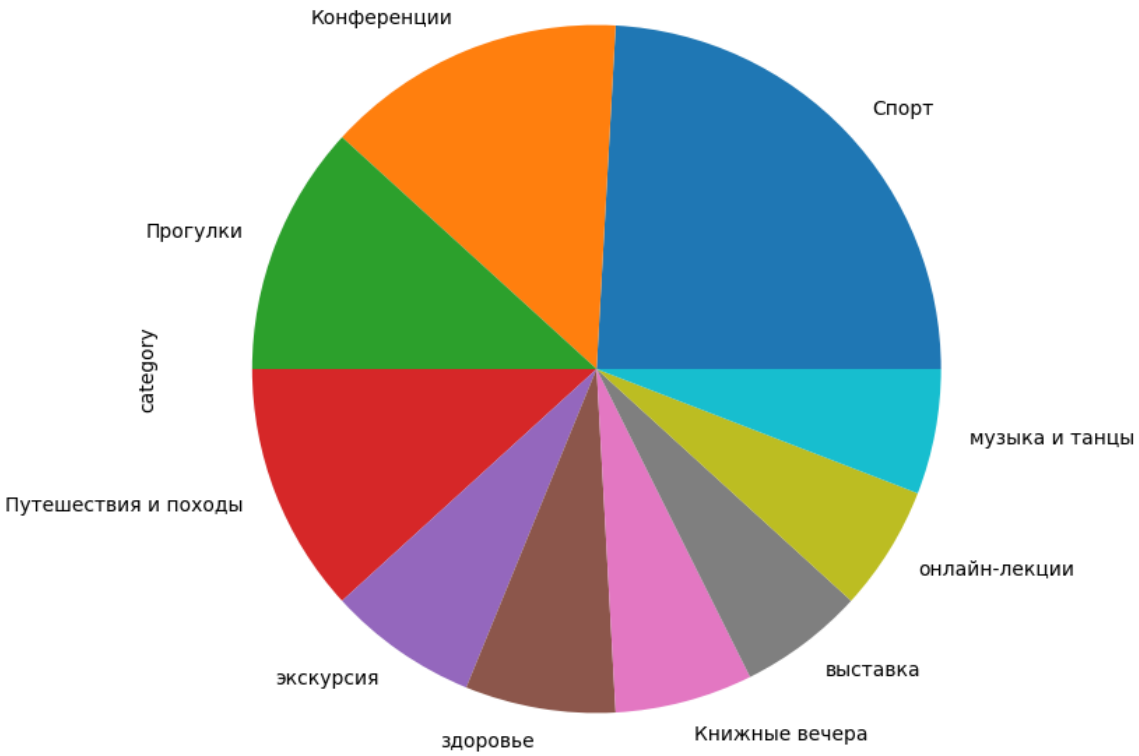

Рисунок 1 – Распределение данных

На рисунке 2 мы видим гистограмму количества токенов.

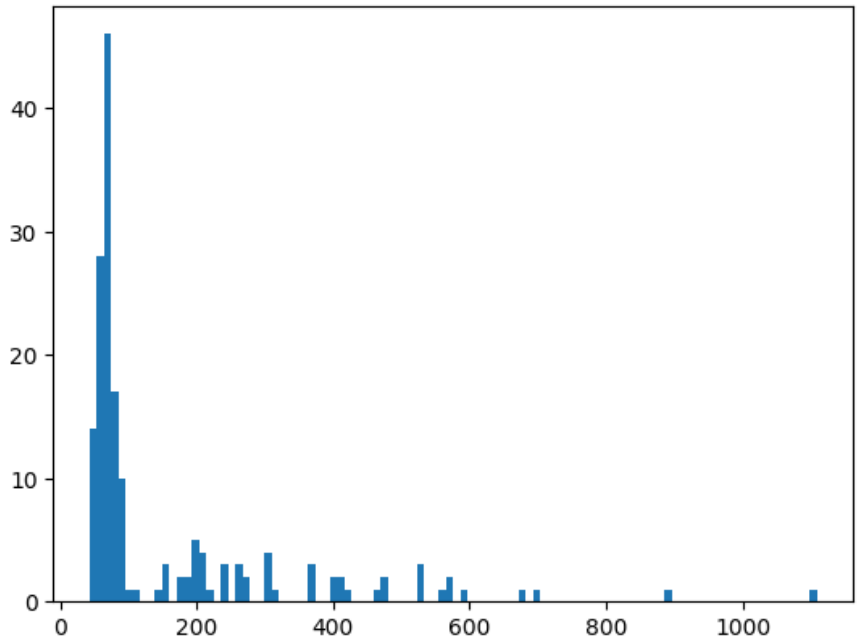

Рисунок 2 – Гистограмма количества токенов

Видно, что количество токенов в описании больше 512, поэтому простой BERT подходит.

Пример данных

" *Category* ": "экскурсия".

" *Title*": "Хатынь — память о трагедии белорусского народа".

"*Tags*": ["Мемориальный комплекс", "Хатынь", "Народная память", "История", "Вторая мировая война", "Трагедия", "Деревня Хатынь", "Памятник", "Минск", "Экскурсия"].

"*Description*": "Мемориальный комплекс «Хатынь» — символ народной памяти, который оставляет неизгладимые впечатления после его посещения. Во времена Второй Мировой войны на месте комплекса находилась деревня, которая была сожжена карательным отрядом. У вас есть возможность проникнуться историей этой трагедии и прочувствовать атмосферу этого места. Что вас ждет. Всего в часе езды от Минска находится Мемориальный комплекс «Хатынь». Это очень атмосферное место, которое пробирает каждого до мурашек. Вы прогуляетесь по комплексу и узнаете все о трагедии в деревне Хатынь во времена Второй мировой войны."

2.2 Модель машинного обучения

В качестве модели была выбранна *ai-forever/sbert_large_nlu_ru*, потому что она подходит для русских текстов, а модели *nlu* универсальны, то есть их можно использовать для вопрос-ответа, распознавание именованых сущностей, классификации и т.д.

sbert_ big_nlu_ru использует токенизатор *WordPiece*.

Модель имеет 25 слоев, а размер скрытого вектора составляет 1024.

Как получить векторы? Отправляются токены и маска в модель машинного обучения и на основе этих данных получается векторный вывод каждого токена. Берем среднее значение этих векторов и получаем вектор результата. Берется вектор со второго слоя с конца.

2.3 Тесты

В тестах использовались такие гиперпараметры: last_k_visited=5, degree_=1, numerator_1_=1, f=sp.sqrt(x), numerator_2_=1, low_median_coefficient=0.54, up_median_coefficient=2, temperature=1.5, step_=0.5, start_pun_=2

Тест 1

На рисунке 3 видно среднее расстояние всех пар внутри одной категории.

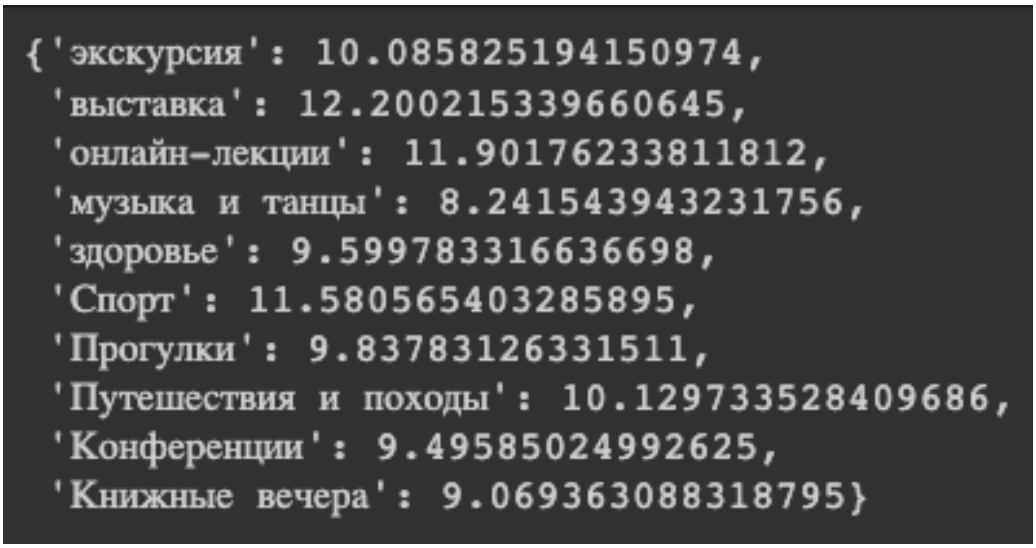

Рисунок 3 – Средние расстояния

На рисунке 4 мы видим среднее расстояние всех пар в разных категориях.

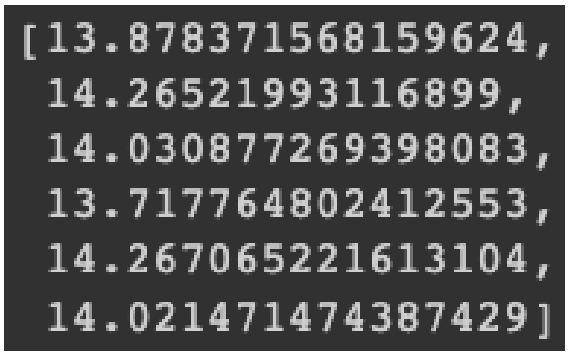

Рисунок 4 – Средние расстояния

Видно, что, расстояния внутри одной категории меньше, чем расстояния внутри разных категорий. Можно сделать вывод, что *Bert* успешно справился с этой задачей.

Тест 2

На рисунке 5 мы видим рекомендации теста 2.

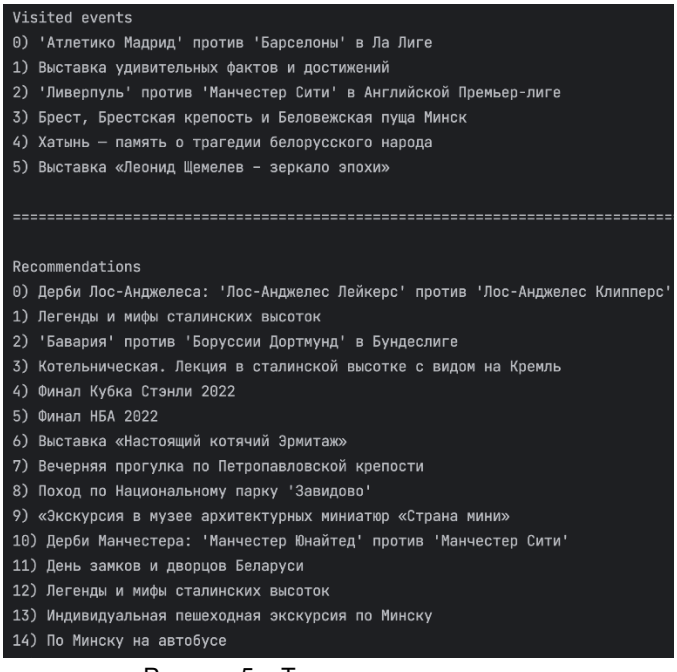

Рисунок 5 – Теплые рекомендации

Можно сделать вывод, что рекомендации хорошие. Человек посещал спортивные мероприятия, выставки, экскурсии и рекомендуемые мероприятия в этих жанрах.

Тест 3

Сложный тест означает рекомендацию в одной категории. Это означает, что, если человек посетил футбольный матч по дерби, можно предположить, что ему нравится футбол среди всех видов спорта или дерби в любом виде спорта. Рассмотрим спорт.

Category = "Спорт".

Title = "Дерби Манчестера: 'Манчестер Юнайтед' против 'Манчестер Сити'".

Description = "Эмоциональное противостояние двух манчестерских команд, 'Манчестер Юнайтед' и 'Манчестер Сити', в одном из самых горячих дерби в Английской Премьер-лиге.".

Tags = "дерби, Манчестер, Манчестер Юнайтед, Манчестер Сити".

На рисунке 6 видны сложные рекомендации, основанные на этой категории.

```
1) Дерби Нью-Йорка: 'Нью-Йорк Рейнджерс' против 'Нью-Йорк Айлендерс'
2) Матч 'Голден Стэйт Уорриорз' против 'Лос-Анджелес Лейкерс'
3) Дерби Лос-Анджелеса: 'Лос-Анджелес Лейкерс' против 'Лос-Анджелес Клипперс'
4) 'Бавария' против 'Боруссии Дортмунд' в Бундеслиге
5) 'Пари Сен-Жермен' против 'Марселя' в Лиге 1
6) Матч 'Чикаго Блэкхокс' против 'Детройт Ред Уингс'
7) Классико: 'Барселона' против 'Реал Мадрид'
8) Финал Кубка Стэнли 2022
9) Сборная Бразилии против Сборной Аргентины
```

Рисунок 6 – Сложная рекомендация

Как видно, результаты очень хорошие. Алгоритм выдал только футбольные рекомендации по футбольным предпочтениям и дерби.

ВЫВОД ПО НАУЧНОЙ РАБОТЕ

В результате научной работы был разработан уникальный алгоритм рекомендаций, построенный на *content-based filtering technique*, который эффективно и умно решет задачу рекомендаций пользователю мероприятий, основанных на их предпочтениях. Были определены гиперпараметры алгоритма и показано, как их использовать. Были проведены тесты, и они показали положительный результат. Все поставленные цели были выполнены.

UDC 004.896

ALGORITHM FOR RECOMMENDATIONS OF EVENTS

Yermalovich D.S. [1]

Belarusian State University of Informatics and Radioelectronics[1]*, Minsk, Republic of Belarus*

Bobrova N.L. – Associate Professor at the Department of Infotmatics

Annotation. This scientific work presents a unique recommendation algorithm that effectively and intelligently solves the problem of recommending activities to the user based on their preferences. The recommendation algorithm is built on a content-based filtering technique.

Keywords. Machine learning, mathematics, data analysis, natural language processing.