Министерство науки и высшего образования России
Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

РАБОТА ПРОВЕРЕНА
рецензент

_____( )

_____2021г

ДОПУСТИТЬ К ЗАЩИТЕ
заведующий кафедрой

_____Б.М. Суховилов

_____2021г

Система рекомендаций книг с использованием данных

социальных сетей

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ ЮУрГУ–090403.2021.007 ПЗ ВКР

Руководитель проекта
заведующий кафедрой,
д.т.н., с.н.с.

_____Б.М. Суховилов

_____2021г.

Автор проекта
студент группы ЭУ-221

_____М. М. Каргар

_____2021г.

Нормоконтролёр

_____

_____2021г.

Челябинск 2021

АННОТАЦИЯ

Каргар М.М. Система рекомендаций книг с использованием данных социальных сетей. – Челябинск: ЮУрГУ, ЭиУ; 2021, 101 с. 14 ил., библиогр. список – 20 наименований.

Выпускная квалификационная работа посвящена анализу и внедрению системы рекомендаций с использованием данных социальных сетей.

Целью исследовательской работы является изучение теоретических основ и разработка практических методов системы рекомендаций. В ходе работы были изучены классические и современные методы классификации текстов, дано сравнение эффективности методов и разработан метод multiclass классификации текстов, близкий по эффективности к лучшим практикам современности отрасли.

Также исследовательская работа включает изучение, сравнение и внедрение различных алгоритмов рекомендаций, которые в настоящее время существуют и работают на веб-сайтах электронной коммерции и других веб-сайтах, таких как Youtube, Netflix и других.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

За последние несколько десятилетий, с появлением Youtube, Amazon, Netflix и многих других подобных веб-сервисов, системы рекомендаций занимали все больше и больше места в нашей жизни. От электронной коммерции (предлагая клиентам статьи, которые могут их заинтересовать) до онлайн-рекламы (предлагая пользователям правильный контент, соответствующий их предпочтениям), системы рекомендаций сегодня неизбежны в наших ежедневных онлайн-путешествиях.

В самом общем смысле системы рекомендаций -это алгоритмы, нацеленные на то, чтобы предлагать пользователям релевантные товары (например, фильмы для просмотра, текст для чтения, продукты для покупки или что-то еще, в зависимости от отрасли).

Рекомендательные системы действительно важны в некоторых отраслях, потому что они могут принести огромную прибыль, когда они эффективны, или также могут быть способом значительно выделиться среди конкурентов. В качестве доказательства важности рекомендательных систем мы можем упомянуть, что несколько лет назад Netflix организовал конкурс («Приз Netflix»), целью которого было создание рекомендательной системы, которая работает лучше, чем ее собственный алгоритм, отмеченный наградами. 1 миллион долларов на победу.

В этом проекте и тезисе алгоритмы машинного обучения использовались, чтобы рекомендовать книги пользователям, используя алгоритмы NLP, анализируя информацию пользователей и твиты из твиттера и находя интерес каждого пользователя из этих данных, и на основе этой информации механизм рекомендаций будет рекомендовать книги. для пользователей.

1 ОБЪЯСНЕНИЕ ЭТАПА ПРОЕКТА

Эта система рекомендаций будет содержать 3 основных шага, и у каждого шага есть под шаги.

1. Анализ данных с использованием NLP: на этом этапе с использованием обработки естественного языка я собираю все данные из twitter, очищаю данные, чтобы быть готовыми к анализу, и после того, как я применю 2 модели машинного обучения NLP для интеллектуального анализа текста, чтобы найти интерес пользователя twitter.

2. Применение рекомендательных алгоритмов: на основе информации, полученной на первом этапе, этот тезис охватывает применение двух различных моделей машинного обучения для более эффективного исправления нашей рекомендательной системы.

3. отображение: на завершающем этапе этой работы, с помощью фильтрации контента, он будет отображать книги для пользователей в зависимости от их интересов.

2 АНАЛИЗ ДАННЫХ

2.1 УТИЛИЗАЦИЯ ДАННЫХ

Web-Scrapping позволяет пользователям загружать данные с разных веб-сайтов через Интернет в нашу локальную систему. Это интеллектуальный анализ данных с различных онлайн-порталов с использованием протоколов передачи гипертекста, который использует эти данные в соответствии с требованиями проекта. Многие компании используют это для сбора данных и создания ботов для поисковых систем.

Python имеет большое количество пакетов / модулей, которые могут помочь в процессе парсинга веб-страниц, например, красивый суп, селен. Существует несколько библиотек, которые могут автоматизировать процесс

очистки веб-страниц, например Autoscraper. Все эти библиотеки используют разные API-интерфейсы, с помощью которых можно очищать данные и сохранять их в фреймворке данных на нашем локальном компьютере.

Данные, которые мы собираемся удалить, - этотвиты из твиттера с использованием библиотеки Python Twint.

Twint— это библиотека Python с открытым исходным кодом, которая используется для синтаксического анализа Twitter, что означает, что мы можем использовать twint для получения данных из Twitter, и это тоже без использования Twitter API. Twint имеет несколько функций, которые делают его более удобным и уникальным по сравнению с другими API синтаксического анализа Twitter, а именно:

API Twitter ограничен 3200 (последними) твитами, в то время как twint не имеет ограничений на загрузку твитов, он может загружать почти все твиты.

Легко использовать и очень быстро. Для получения данных не требуется первоначального входа в систему или регистрации.

Twint можно использовать для очистки твитов с использованием различных параметров, таких как хэштеги, имена пользователей, темы и т. Д. Он даже может извлекать из твитов такую информацию, как номер телефона и идентификатор электронной почты.

В этой статье мы рассмотрим twint и посмотрим, какие функции он предлагает для получения данных из Twitter.

2.1.1 РЕАЛИЗАЦИЯ TWINT

Twint использует операторы поиска Twitter, чтобы позволить пользователям получать твиты от определенных пользователей, получать твиты, связанные с конкретными темами, хэштегами и тенденциями, или сортировать конфиденциальную информацию из твитов, например, адреса электронной почты и номера телефонов.

Листинг 1 – «Сбор данных из Twitter»

```
import twint
```

```
import nest_asyncio

net_asyncio.apply()

twint -u username
```

3 MultiClassTextClassification

В коммерческом мире существует множество применений классификации текста. Например, новости обычно группируются по темам; контент или продукты часто помечаются по категориям; пользователей можно разделить на группы в зависимости от того, как они говорят о продукте или бренде в Интернете.

Однако подавляющее большинство онлайн-статей и руководств по классификации текста представляют собой двоичную классификацию текста, такую как фильтрация спама по электронной почте (спам по сравнению с любительской), анализ тональности (положительный или отрицательный). В большинстве случаев наши настоящие проблемы намного сложнее. Таким образом, он использует мультиклассовую классификацию текста, которая представляет собой контролируемый метод обучения, поэтому тексту нужны помеченные данные для обучения модели. для этого будет использоваться другой набор данных. Это набор данных новостной статьи с ручными тегами, который попадает в один из 9 классов: бизнес, путешествия, спорт, стиль и стиль, преступность, образование, наука, религия, политика, технологии, развлечения.

Out[25]:

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
32	CRIME	Rachel Dolezal Faces Felony Charges For Welfar...	Carla Herreria	https://www.huffingtonpost.com/entry/rachel-do...	State prosecutors say almost $84,000 had been ...	2018-05-25
40	CRIME	Man Faces Charges After Pulling Knife, Stun Gu...	Jenna Amatulli	https://www.huffingtonpost.com/entry/man-knife...	"We thought we were going to die," one of the ...	2018-05-25
42	CRIME	2 People Injured In Indiana School Shooting	Marina Fang	https://www.huffingtonpost.com/entry/indiana-m...	A male student, believed to be the suspect, ha...	2018-05-25
126	TRAVEL	14 Ways To Make Family Road Trips Easier, From...	Taylor Pittman	https://www.huffingtonpost.com/entry/family-ro...	Having waterproof covers on the seats is kind ...	2018-05-24
...	...	...	...	...	...	...
200806	STYLE & BEAUTY	Cheryl Tiegs In A Sauna: A Look Back	Sarah Leon	https://www.huffingtonpost.com/entry/cheryl-ti...	"A Look Back" is a daily column that highlight...	2012-01-28
200815	SCIENCE	Russian Cargo Ship Docks At International Spac...		https://www.huffingtonpost.com/entry/russian-c...	Gallery: Space Station's Expedition 30 Mission...	2012-01-28
200816	SCIENCE	Robots Play Catch, Starring Agile Justin And R...	Travis Korte	https://www.huffingtonpost.com/entry/robots-pl...	image 1: throw As Hizook reports, DLR started ...	2012-01-28
200817	SCIENCE	Thomas Edison Voted Most Iconic Inventor In U....		https://www.huffingtonpost.com/entry/thomas-ed...	That doesn't mean Jobs lacks for fans in the w...	2012-01-28
200818	SCIENCE	Aurora Borealis Caused By Huge Solar Storm Shi...		https://www.huffingtonpost.com/entry/aurora-fr...	Aurora borealis can typically only be seen at ...	2012-01-28

28679 rows × 6 columns

Рисунок 1 – первый набор данных для обучения модели

Out[38]:

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...
...	...	...
2220	business	cars pull down us retail figures us retail sal...
2221	politics	kilroy unveils immigration policy ex-chatshow ...
2222	entertainment	rem announce new glasgow concert us band rem h...
2223	politics	how political squabbles snowball it s become c...
2224	sport	souness delight at euro progress boss graeme s...

2225 rows × 2 columns

Рисунок 2 – второй набор данных для обучения модели

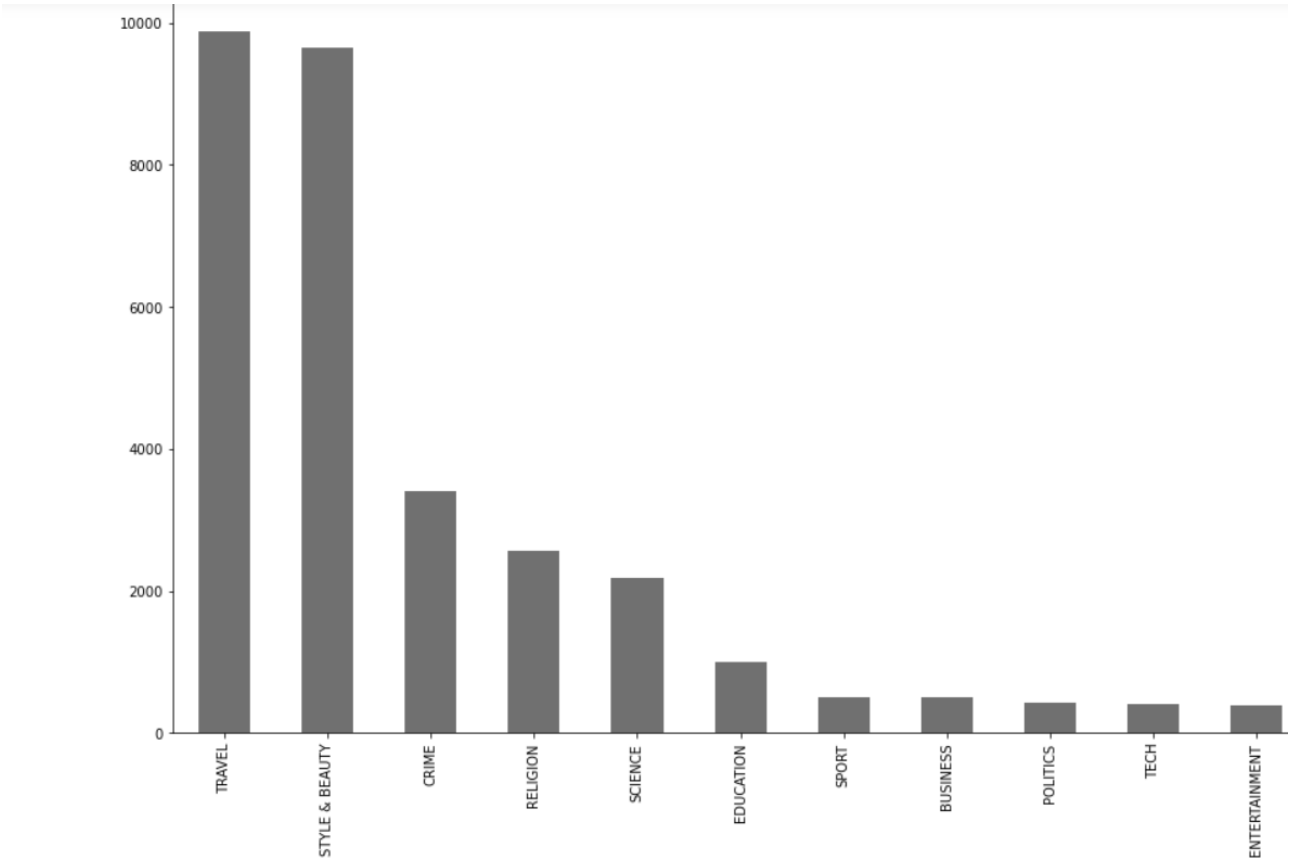


Рисунок 3 – набор данных на основе категории

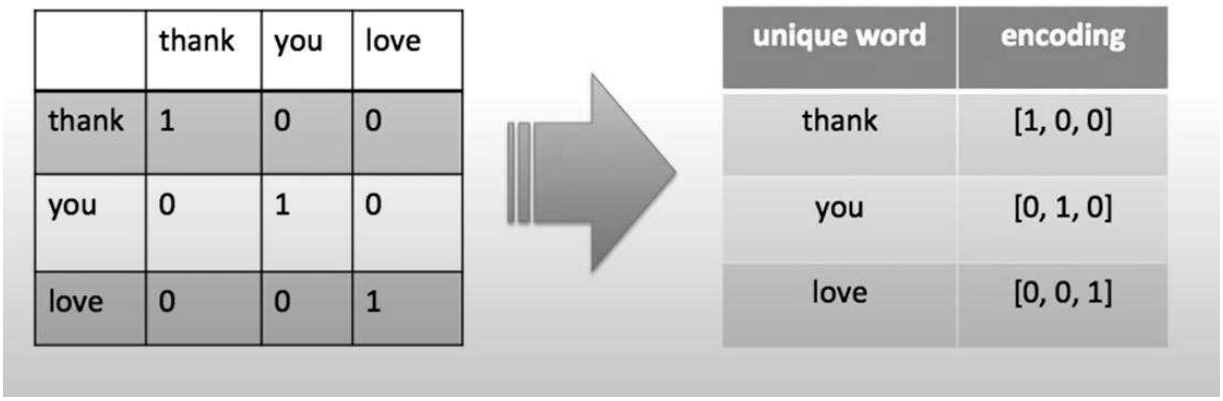
3.1 ПОСТАНОВКА ПРОБЛЕМЫ

Проблема заключается в контролируемой классификации текста, и применение двух разных моделей машинного обучения NLP будет моделью с точностью для системы рекомендаций по проектам.

3.2 МНОГОКЛАССНАЯ КЛАССИФИКАЦИЯ ТЕКСТА ИСПОЛЬЗОВАНИЕМ DOCS&LogisticRegression

Первая важная часть word2Vector - это кодировка

кодировка меняет слова на числа.

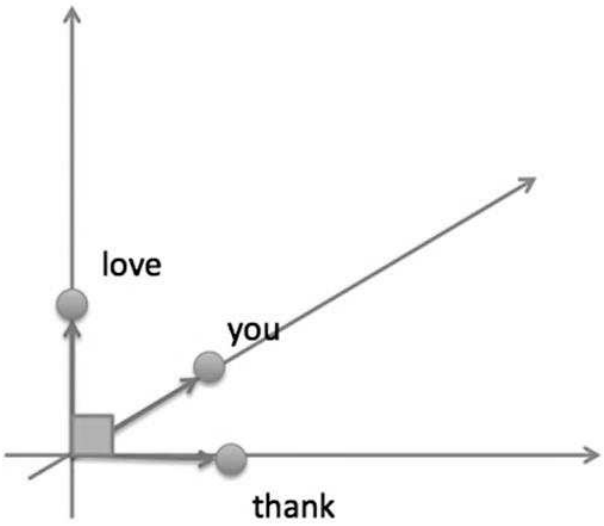


	thank	you	love
thank	1	0	0
you	0	1	0
love	0	0	1

unique word	encoding
thank	[1, 0, 0]
you	[0, 1, 0]
love	[0, 0, 1]

Рисунок4 – Преобразовать текст в числа

Здесь были использованы 3 слова, и каждое слово пронумеровано цифрами, но здесь выполнено горячее кодирование, но нет никакого сходства, как показано на 5 картинке.



unique word	encoding
thank	[1, 0, 0]
you	[0, 1, 0]
love	[0, 0, 1]

Рисунок 5 – wordEmbedding

Чтобы найти сходство, используется встраивание слов, которое представляет собой класс подходов для представления слов и документов с использованием плотного векторного представления. ... Вместо этого при встраивании слова представлены плотными векторами, где вектор — это проекция слова в непрерывное векторное пространство.

unique word	encoding	embedding
king	[1, 0, 0, 0]	[1, 2]
man	[0, 1, 0, 0]	[1, 3]
queen	[0, 0, 1, 0]	[5, 1]
woman	[0, 0, 0, 0]	[5, 3]

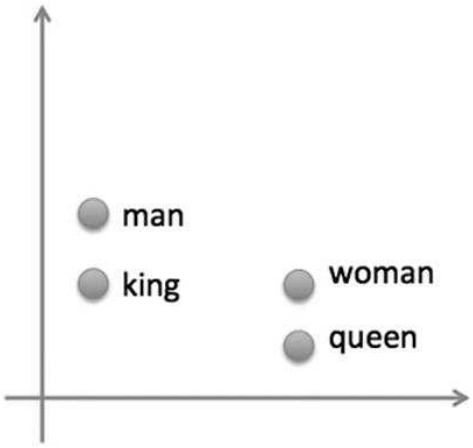


Рисунок6 – вложение слов с графами

один из методов - слово в вектор, который находит сходство у соседей

“king brave man”
“queen beautiful woman”

word	neighbor
king	brave
brave	king
brave	man
man	brave
queen	beautiful
beautiful	queen
beautiful	woman
woman	beautiful

Рисунок 7 – Векторизация соседей

В этой таблице слово было взято в качестве входных данных, а соседи - в качестве цели, чтобы найти сходство между ними.

word	word one hot encoding	neighbor	neighbor one hot encoding
king	[1, 0, 0, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
king	[1, 0, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
queen	[0, 0, 0, 1, 0, 0]	beautiful	[0, 0, 0, 0, 1, 0]
queen	[0, 0, 0, 1, 0, 0]	woman	[0, 0, 0, 0, 0, 1]
beautiful	[0, 0, 0, 0, 1, 0]	queen	[0, 0, 0, 1, 0, 0]
beautiful	[0, 0, 0, 0, 1, 0]	woman	[0, 0, 0, 0, 0, 1]
woman	[0, 0, 0, 0, 0, 1]	queen	[0, 0, 0, 1, 0, 0]
woman	[0, 0, 0, 0, 0, 1]	beautiful	[0, 0, 0, 0, 1, 0]

Рисунок 8 – найти сходство

у нас есть 3 слоя, которые найдут нам сходство

- входные слои
- выходные слои
- скрытые слои

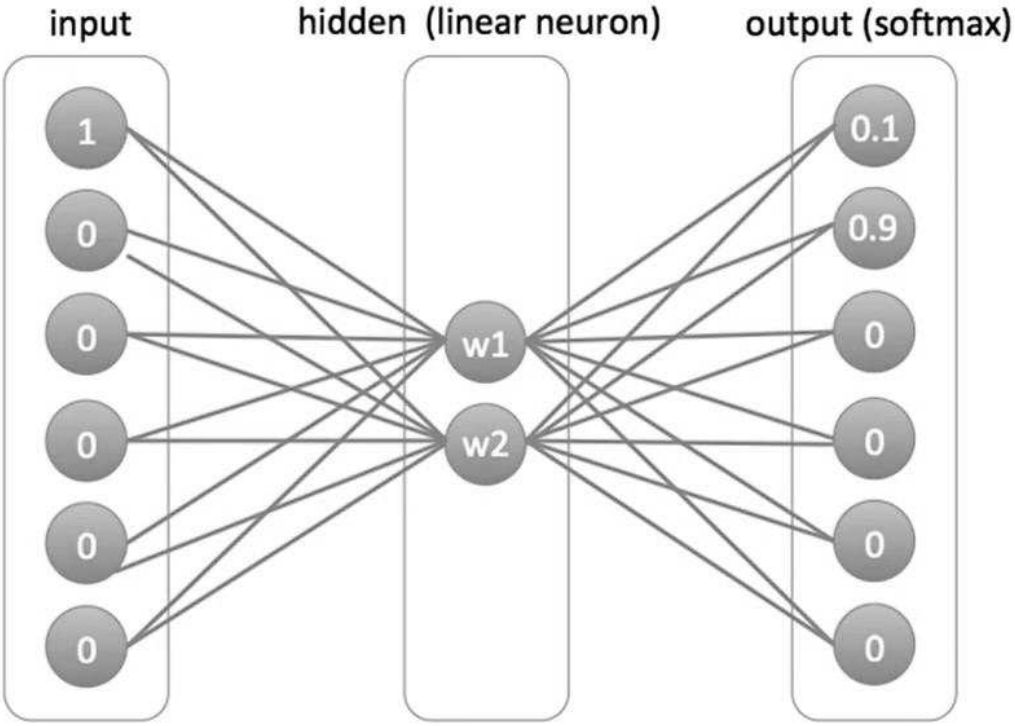


Рисунок9 – три слоя w2v

В качестве примера возьмем человека в качестве входных данных, и получится наиболее похожее значение для человека - это слово храбрый.

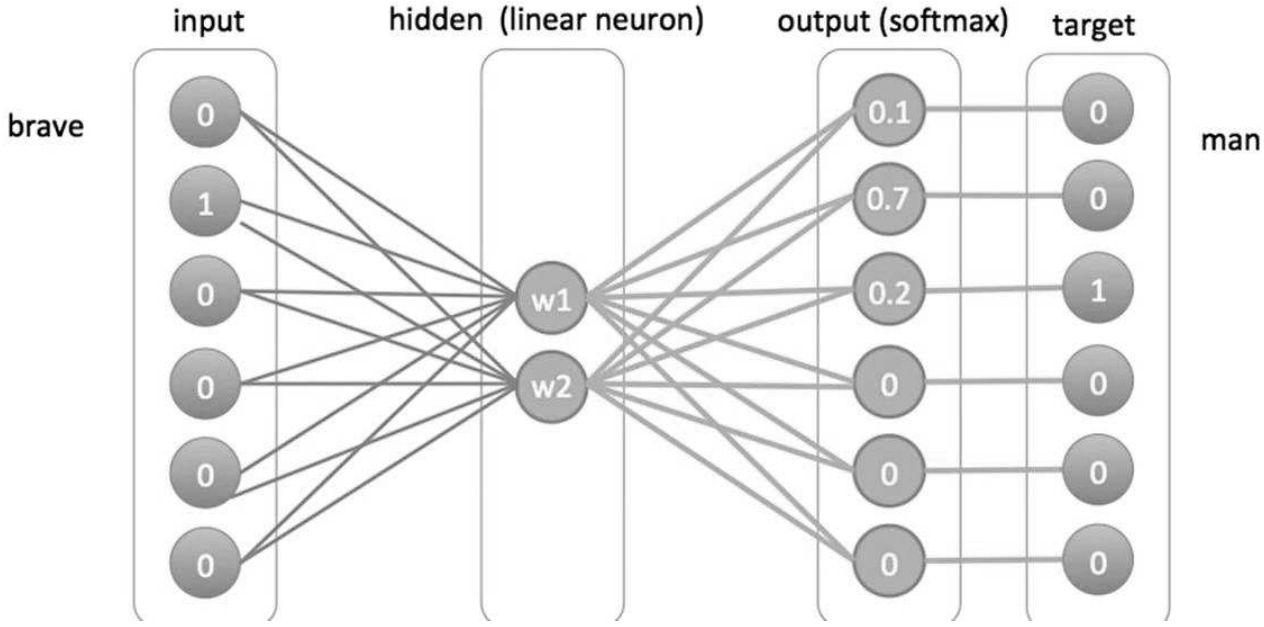
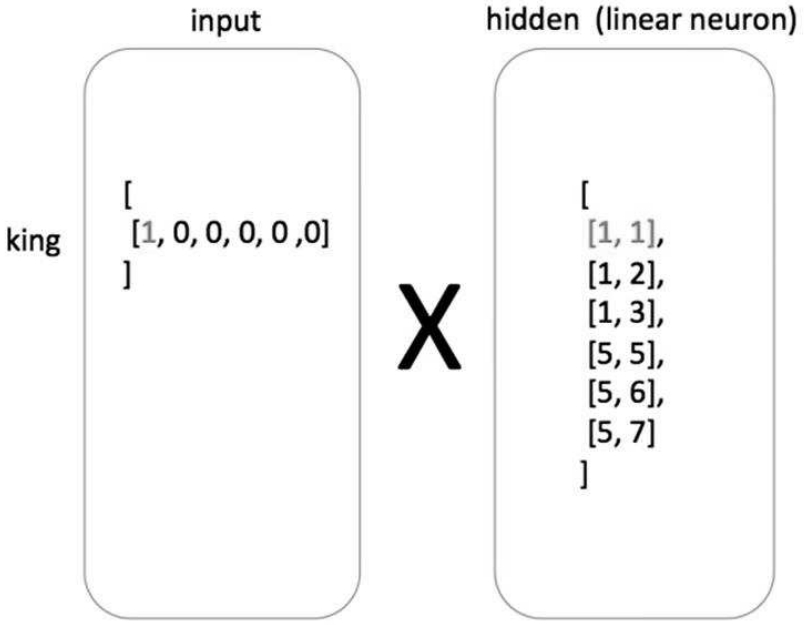


Рисунок10 – пример ввода и вывода

Создает матрицу и воспроизводит все входы со всеми выходами, в результате чего мы получаем ее номер, а затем строим его и назначаем ему точки.



unique word	embedding
king	[1, 1]
brave	[1, 2]
man	[1, 3]
queen	[5, 5]
beautiful	[5, 6]
woman	[5, 7]

Рисунок 11 – Матрица wordToVector

В этой части рассказывается о применении нейронных сетей к тексту, который очистит и подготовит к анализу, чтобы использовать w2v.

И сначала давайте вспомним, что это за текст? он может думать об этом как о последовательности символов, слов или чего-то еще.

Во всех моделях классификации текста текст воспринимается как последовательность чисел, каждое слово представляет собой двоичное число, а число представляет слово и совокупность этих чисел, представляющих сигнал, поэтому нам нужно найти взаимосвязь между этими числами, чтобы найти смысл часового.

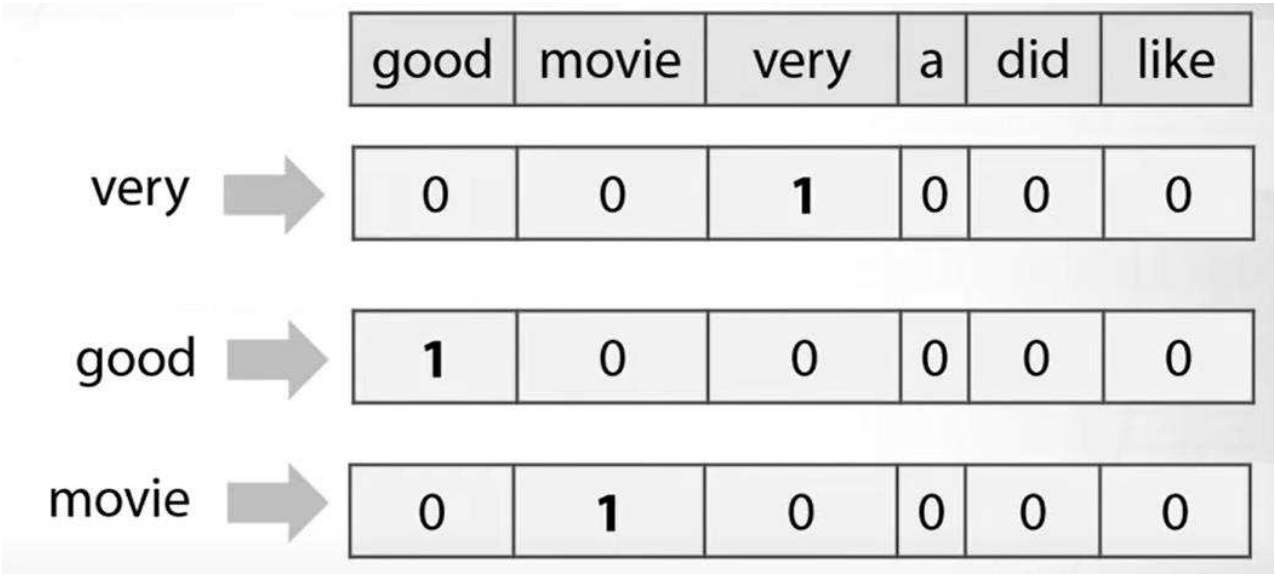


Рисунок 12 – представление слова в таблице

есть все слова и всегда отличное слово, это набор данных слов, есть столбец функций. Этот вектор представляет собой единственное ненулевое значение, которое находится в столбце, соответствующем этому конкретному слову. Итак, в этом примере есть очень, ну и фильм. И в этом случае для реальных проблем есть сотни тысяч столбцов.

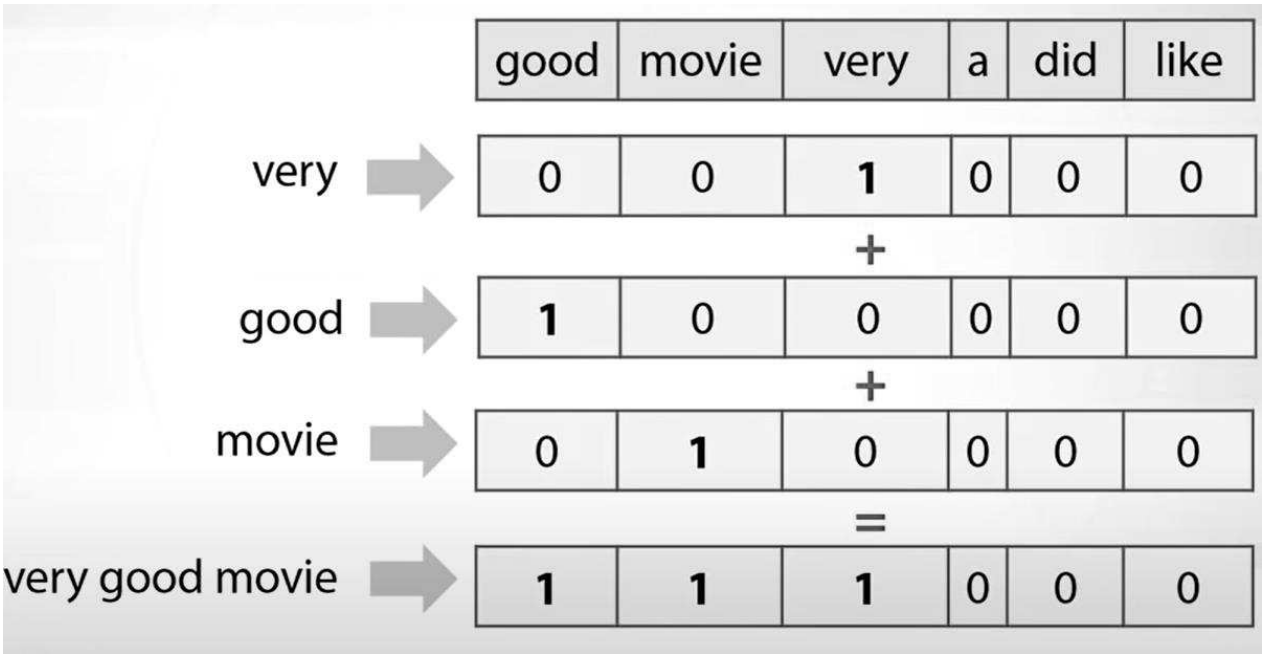


Рисунок13 – Мешки слов Представление

А как попасть на слово "посылка"? Фактически, сюда добавлены все значения и все эти векторы, и можно получить пакет векторизации слов, который теперь соответствует очень хорошему фильму. Итак, было бы хорошо представить пакет слов как сумму разреженных векторов с горячим кодированием, соответствующих каждому конкретному слову.

И примером таких векторов являются вложения word2vec, которые представляют собой предварительно обученные вложения, которые выполняются неконтролируемым образом.

И об этом мы расскажем в следующей части. Но все в этой части нужно знать прямо сейчас, это то, что векторы word2vec имеют хорошее свойство. Слова, имеющие схожий контекст с точки зрения соседних слов, как правило, имеют векторы, которые коллинеарны, фактически указывая примерно в одном направлении. И это очень хорошая собственность, которую мы будем использовать в будущем.

Он может просто вычислить сумму этих векторов, и есть представление на основе вложенности word2vec для всего текста, как в очень хорошем фильме. И вот некоторые из векторов word2vec, которые действительно работают на практике: каждый вектор, описывающий отношения слов, например, если фильм много раз связан с хорошим или плохим или какой-либо другой фразой, это означает, что этот фильм хороший или плохой и основан на их Реальный результат строится, он может быть хорошим или плохим или любым другим результатом.

Это может дать модели отличный базовый дескриптор, базовую функциональность для классификатора, и это может действительно хорошо работать. Другой подход заключается в создании нейронной сети на основе этих привязок и в каждой серии привязок, которая пытается распознать лежащие в основе взаимосвязи в наборе данных посредством процесса, имитирующего работу человеческого мозга.

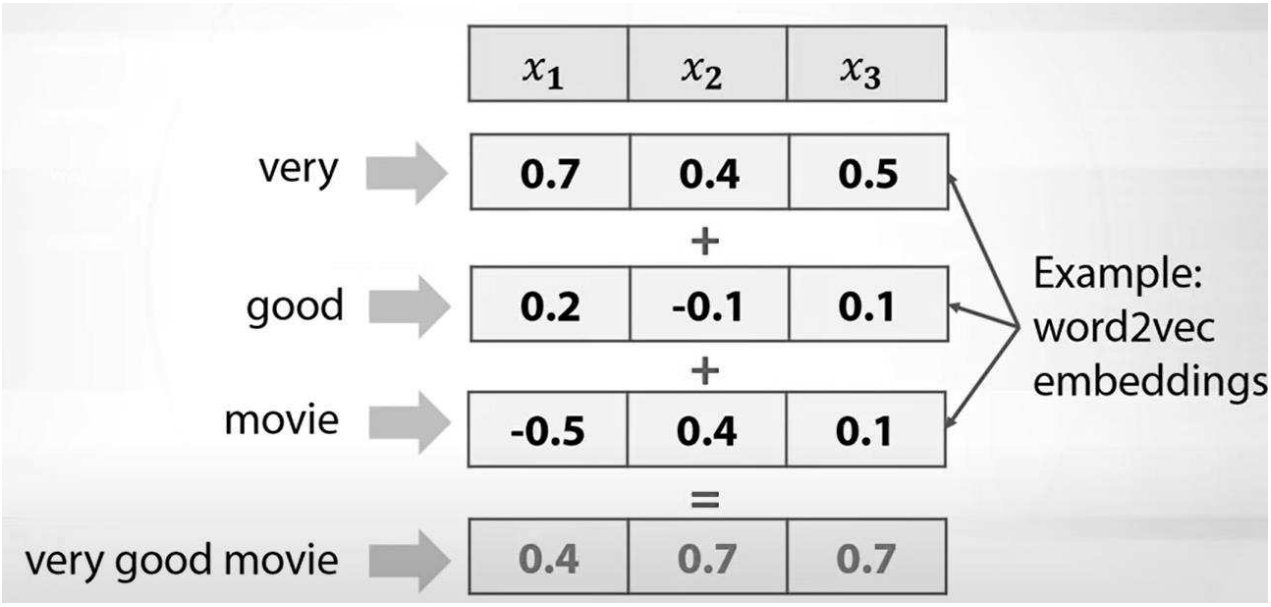


Рисунок 14 – Сумма двух Векторов

Word2Vector использует библиотеку Gensim, которая используется для неконтролируемого моделирования тем и обработки естественного языка. Он предназначен для извлечения семантических тем из документов. Он может обрабатывать большие текстовые коллекции.

Этим он отличается от других программных пакетов машинного обучения, предназначенных для обработки памяти.
Gensim также обеспечивает эффективную многоядерную функциональность. реализация различных алгоритмов для увеличения скорости обработки. Он предоставляет более удобные средства обработки текста, чем другие пакеты, такие как Scikit-learn, R и т. Д.

В этом руководстве будут рассмотрены следующие концепции:

1) Создать корпус из заданного набора данных
2) Создайте матрицу TFIDF в Gensim
3) Создавайте биграммы и триграммы с помощью Gensim
4) Создайте модель Word2Vec с помощью Gensim
5) Создайте модель Doc2Vec с помощью Gensim
6) Создание тематической модели с LDA

7) Создание тематической модели с LSI

8) Вычислить матрицы подобия

9) Обобщать текстовые документы

Прежде чем двигаться дальше, разберитесь, что означают некоторые из приведенных ниже терминов.

- Корпус: собрание текстовых документов.
- Вектор: форма представления текста.
- Модель: алгоритм, используемый для создания представления данных.
- Тематическое моделирование: это инструмент интеллектуального анализа информации, который используется для извлечения семантических тем из документов.
- Тема: повторяющаяся группа слов, часто встречающихся вместе.

Классы несбалансированы, однако наивный классификатор, который предсказывает, что все будет взыскать долги, достигнет точности только более 20%.

Взгляните на несколько описаний продуктов и сопутствующих товаров.

Листинг 2 – «W2V Function»

```
def print_complaint(index):
    example = w2v_df[w2v_df.index == index][['text',
'Category']].values[0]
    if len(example) > 0:

        print(example[0])
print('Category:', example[1])

print_complaint(12)
```

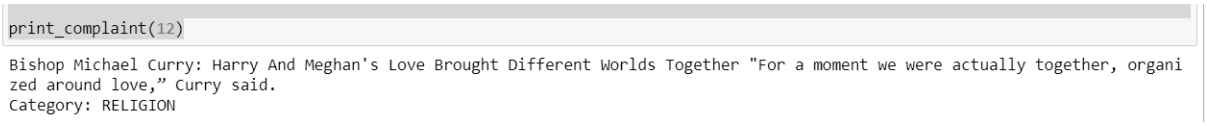


Рисунок 15– Сумма двух векторов

В коде определяется функция для преобразования текста в нижний регистр и удаления знаков препинания / символов из слов и так далее.

Листинг 3 – «Очистка текста с помощью BeautifulSoup»

```
from bs4 import BeautifulSoup
def cleanText(text):

    text = BeautifulSoup(text, "lxml").text
    text = re.sub(r'\|\|\|', r' ', text)
    text = re.sub(r'http\S+', r'<URL>', text)
    text = text.lower()
    text = text.replace('x', '')
    return text

w2v_df['text'] = w2v_df['text'].apply(cleanText)
```

Следующие шаги включают разделение 70/30 на поезд / тест, удаление стоп-слов и разметку текста с помощью токенизатора NLTK. Для первой попытки отметьте каждое повышение соответствующим продуктом.

Листинг 4 – «Функции токанизаторов»

```
import nltk
from nltk.corpus import stopwords
def tokenize_text(text):
    tokens = []
    for sent in nltk.sent_tokenize(text):
        for word in nltk.word_tokenize(sent):
            if len(word) < 2:
```

```
                continue
tokens.append(word.lower())
    return tokens

train_tagged = train.apply(
    lambda r: TaggedDocument(words=tokenize_text(r['text']),
tags=[r.Category]), axis=1)
test_tagged = test.apply(
    lambda r: TaggedDocument(words=tokenize_text(r['text']),
tags=[r.Category]), axis=1)


train_tagged.values[30]
```

```
: train_tagged.values[30]

: TaggedDocument(words=['zooey', 'deschanel', 'golden', 'globes', 'dress', '2013', 'see', 'her', 'red', 'carpet', 'look', 'photo
  s', 'see', 'all', 'of', 'the', 'looks', 'from', 'the', '2013', 'golden', 'globes', 'red', 'carpet', 'photos', 'zooey', 'deschan
  el', 'aka', 'the', 'unofficial', 'queen', 'of', 'quirk', 'takes'], tags=['STYLE & BEAUTY'])
```

Рисунок16 – Tokanizers

3.2.1 НАСТРОЙКА ТРЕНИРОВОЧНЫХ МОДЕЛЕЙ Doc2Vec

Сначала создадим экземпляр модели doc2vec - Распределенный пакет Word (DBOW). В архитектуре word2vec есть два имени алгоритма - непрерывный пакет слов (CBOW) и «скип-грамма» (SG); в архитектуре doc2vec соответствующими алгоритмами являются распределенная память (DM) и распределенный пакет слов (DBOW).

- Если dm = 0, используется распределенный мешок слов (PV-DBOW); если dm = 1, используется «распределенная память» (PV-DM).
- 300-мерные векторы признаков.
- min_count = 2, игнорирует все слова с общей частотой ниже этой.
- negative = 5, указывает, сколько «шумовых слов» нужно нарисовать.
- hs = 0, а отрицательное значение не равно нулю, будет использоваться отрицательная выборка.

- sample = 0, порог для настройки того, какие слова с более высокой частотой случайным образом подвергаются понижающей дискретизации.
- worker = cores, используйте это множество рабочих потоков для обучения модели (= более быстрое обучение на многоядерных машинах).

Листинг 4 – «функция применения модели w2v»

```
import multiprocessing

cores = multiprocessing.cpu_count()

model_dbow = Doc2Vec(dm=0, vector_size=300, negative=5, hs=0,
min_count=2, sample = 0, workers=cores)
model_dbow.build_vocab([x for x in tqdm(train_tagged.values)])



%%time
for epoch in range(30):
model_dbow.train(utils.shuffle([x for x in
tqdm(train_tagged.values)]),
total_examples=len(train_tagged.values), epochs=1)
model_dbow.alpha -= 0.002
model_dbow.min_alpha = model_dbow.alpha



model_dbow = Doc2Vec(dm=0, vector_size=300, negative=5, hs=0,
min_count=2, sample = 0, workers=cores)
model_dbow.build_vocab([x for x in tqdm(train_tagged.values)])




for epoch in range(30):
```

```
model_dbow.train(utils.shuffle([x for x in
tqdm(train_tagged.values)]),
total_examples=len(train_tagged.values), epochs=1)
model_dbow.alpha -= 0.002
model_dbow.min_alpha = model_dbow.alpha
```

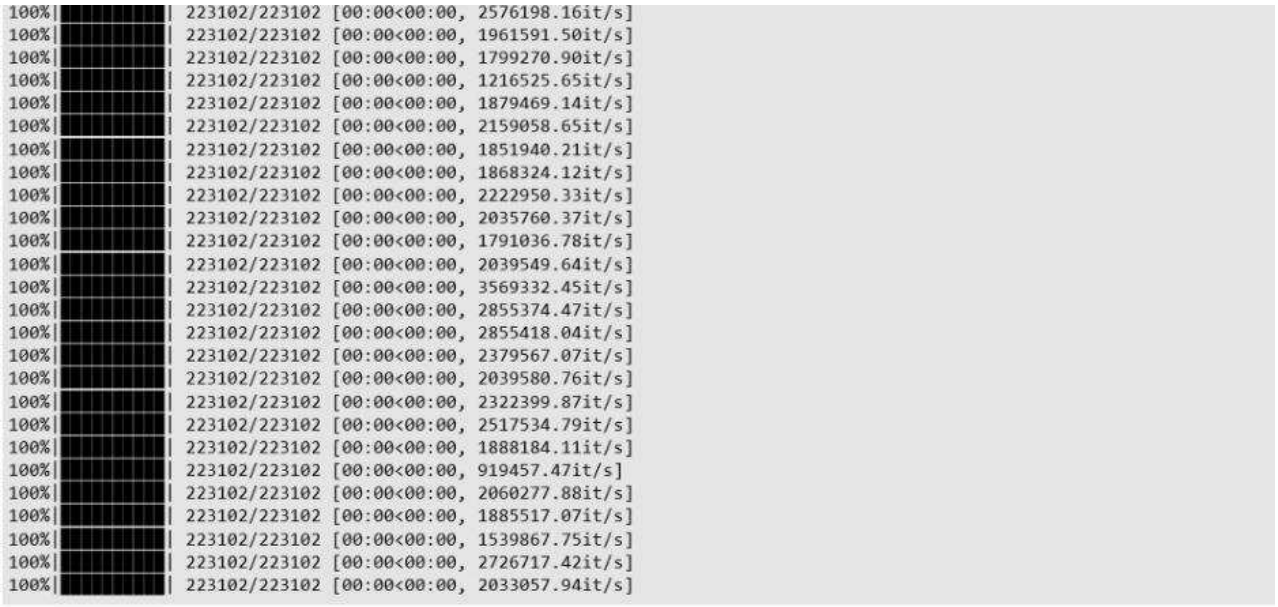


Рисунок 17 – РЕЗУЛЬТАТ КОНФИГУРАЦИИ

Построение окончательного векторного объекта для классификатора

Листинг5 – «точностьмоделитестирования»

```
def vec_for_learning(model, tagged_docs):
sents = tagged_docs.values

    targets, regressors = zip(*[(doc.tags[0],
model.infer_vector(doc.words, steps=20)) for doc in sents])
    return targets, regressorsdefvec_for_learning(model,
tagged_docs):

sents = tagged_docs.values
    targets, regressors = zip(*[(doc.tags[0],
model.infer_vector(doc.words, steps=20)) for doc in sents])
    returntargets, regressors
```

```
y_train, X_train = vec_for_learning(model_dbow, train_tagged)
y_test, X_test = vec_for_learning(model_dbow, test_tagged)

logreg = LogisticRegression(n_jobs=1, C=1e5)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)


from sklearn.metrics import accuracy_score, f1_score

print('Testing accuracy %s' % accuracy_score(y_test, y_pred))
print('Testing F1 score: {}'.format(f1_score(y_test, y_pred,
average='weighted')))
```

```
Testing accuracy 0.6683609437751004
Testing F1 score: 0.651646431211616
```

Рисунок18 – точность модели

3.2.2 РАСПРЕДЕЛЕННАЯ ПАМЯТЬ

Распределенная память (DM) действует как память, которая запоминает то, что отсутствует в текущем контексте, или как тема абзаца. В то время как векторы слов представляют понятие слова, вектор документа предназначен для представления концепции документа. Мы снова создаем экземпляр модели Doc2Vec с размером вектора 300 слов и повторяем обучающий корпус 30 раз.

Листинг6 – «настройкасловарямодели»

```
model_dmm = Doc2Vec(dm=1, dm_mean=1, vector_size=300, window=10,
negative=5, min_count=1, workers=5, alpha=0.065, min_alpha=0.065)
```

```
model_dmm.build_vocab([x for x in tqdm(train_tagged.values)])
```

```
%%time
for epoch in range(30):
model_dmm.train(utils.shuffle([x for x in
tqdm(train_tagged.values)]),
total_examples=len(train_tagged.values), epochs=1)
model_dmm.alpha -= 0.002
model_dmm.min_alpha = model_dmm.alpha
```

```
100%|██████████| 223102/223102 [00:00<00:00, 1578083.26it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1745168.51it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1854244.79it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1893698.00it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1885653.85it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1869836.11it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1869832.37it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2445159.40it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2022787.35it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1165813.40it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2379542.86it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2379585.22it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1784628.78it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1784628.78it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1940508.05it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2039602.98it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1104039.77it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2039647.44it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2039562.97it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2379573.12it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2855418.04it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2379506.56it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2039554.08it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1784632.19it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1751177.79it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1892361.48it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1959722.32it/s]
100%|██████████| 223102/223102 [00:00<00:00, 1649827.85it/s]
100%|██████████| 223102/223102 [00:00<00:00, 2778023.03it/s]
```

Рисунок19 – Result

Листинг7 – «Обучите классификатор логистической регрессии
»

```
y_train, X_train = vec_for_learning(model_dmm, train_tagged)
y_test, X_test = vec_for_learning(model_dmm, test_tagged)

logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

print('Testing accuracy %s' % accuracy_score(y_test, y_pred))
```

```
print('Testing F1 score: {}'.format(f1_score(y_test, y_pred,
average='weighted')))
```

Эта модель представляет обучающий набор doc2vec для обучения, однако в учебнике Gensim для обучения использовался весь набор данных, этот подход использует весь набор данных для обучения классификатора doc2vec для нашей классификации жалоб потребителей, эта модель может обеспечить точность 65%. Вы можете найти этот блокнот здесь, это немного другой подход. результат точности, который мы получили для этой модели, близок к 65

Рисунок20 – Точностьмодели

3.3 КЛАССИФИКАЦИЯ ТЕКСТА С ИСПОЛЬЗОВАНИЕМ TFIDF

Интуитивно, чтобы понять, о чем идет речь, применяем поиск общих слов. Частота включает определение того, сколько раз слово встречается в тексте. Однако в этой статье слово «the» встречается более 200 раз, а слово TF - всего 55 (включая используемый код). Переводные слова используются слишком часто. Это обратное взвешивание называется обратным документом. Вместе TF-IDF фиксирует относительную важность слов в документах или текстовых наборах.

Есть много замечательных статей, посвященных интуиции, лежащей в основе TF-IDF, и не так много статей о том, как получить точные значения для TF-IDF. Основное внимание в этой статье уделяется объединению различных вычислений и описанию того, как вывести каждый шаг с помощью кода Python, чтобы вы могли вывести его из текстов, с которыми вы работаете.

3.3.1 ПОЧЕМУ TF IDF ?

В области обработки естественного языка открытие встраивания слов в 2013 году и языковых моделей в 2018 году изменило ситуацию и привело ко многим захватывающим разработкам в НЛП. Так что кажется немного странным, что здесь в 2020 году так решили написать более 2000 слов о TF IDF, которая была впервые сформулирована в 1970-х годах. Вот 2 причины, по которым TFIDF лучше понимает лежащие в основе вычисления.

- Понимание TF-IDF на один шаг упрощает понимание и интерпретацию результатов алгоритмов, которые вы применяете поверх TF-IDF.
- В прикладном бизнес-контексте проблема классификации текста - одна из распространенных проблем в NLP. В задачах классификации текста алгоритмы должны предсказывать тему на основе заранее определенного набора тем, по которым они обучались. В 2018 году Google выпустил структуру классификации текстов, основанную на экспериментах 450 тыс. С несколькими разными наборами текста. Основываясь на экспериментах с 450K, Google обнаружил, что когда количество образцов / количество слов <1500, TFIDF был лучшим способом представления текста. Если у вас небольшой размер выборки для решения относительно распространенной проблемы, полезно попробовать TFIDF.

3.3.2 CORPUS

Чтобы проиллюстрировать концепцию TF-IDF, нужен корпус. Корпус - это собрание документов. В типичной задаче обработки списка на естественном языке корпус может варьироваться от списка журналов центра обработки вызовов, списка объявлений в социальных сетях до большого собрания исследовательских документов.

Чтобы проиллюстрировать различные шаги, необходимо сделать корпус как можно меньше, чтобы матрицы аккуратно помещались на странице. Я наткнулся на эту цитату / красивое стихотворение персидского поэта 13 века и

суфийского мистика Руми (Джалал ад-Дин Мухаммад Руми), и оно идеально подходит для нашего случая использования. Таким образом, в этом случае он будет использовать это стихотворение как список документов, где каждое предложение будет рассматриваться как документ.

Листинг 7 – «Corpus»

```
corpus =  ["you were born with potential",
"you were born with goodness and trust",
"you were born with ideals and dreams",
"you were born with greatness",
"you were born with wings",
"you are not meant for crawling, so don't",
"you have wings",
"learn to use them and fly"
]
```

3.3.3 ПОИСК КОНЕЧНОГО РЕЗУЛЬТАТА

На этом этапе текст будет разбит на загадочные десятичные дроби. Но, в конце концов, мы пытаемся демистифицировать эти десятичные дроби, понимая вычисления, используемые в TF-IDF. Как сообщалось ранее, его довольно легко получить с помощью пакета sklearn.

Листинг8 – «tfidf vectorizer»

```
#transform the tfidf vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
tf_idf_vect = TfidfVectorizer()
X_train_tf_idf = tf_idf_vect.fit_transform(corpus)
tf_idf_vect = TfidfVectorizer()
X_train_tf_idf = tf_idf_vect.fit_transform(corpus)

terms = tf_idf_vect.get_feature_names()
```

В приведенной ниже матрице каждая строка представляет предложение из вышеприведенного стихотворения. Каждый столбец представляет собой уникальное слово в стихотворении в алфавитном порядке. Как видите, в матрице много нулей. Таким образом, для представления этого используется разреженная матрица с эффективным использованием памяти.

		Vocabulary words																	
	Index & Sentence	and	are	born	crawlin	don	dreams	fly	for	goodne	greatne:	have	ideals	learn	meant	not	potential	so	them
0	you were born with potential	0	0	0.3833	0	0	0	0	0	0	0	0	0	0	0	0	0.682895	0	
1	you were born with goodness and trust	0.3776	0	0.2931	0	0	0	0	0	0.5222	0	0	0	0	0	0	0	0	
2	you were born with ideals and dreams	0.3776	0	0.2931	0	0	0.5222	0	0	0	0	0	0.5222	0	0	0	0	0	
3	you were born with greatness	0	0	0.3833	0	0	0	0	0	0	0.6829	0	0	0	0	0	0	0	
4	you were born with wings	0	0	0.413	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	you are not meant for crawling, so don't	0	0.3727	0	0.3727	0.3727	0	0	0.3727	0	0	0	0	0	0.3727	0.3727	0	0.3727	
6	you have wings	0	0	0	0	0	0	0	0	0	0	0.7252	0	0	0	0	0	0	
7	learn to use them and fly	0.3077	0	0	0	0	0	0.4255	0	0	0	0	0	0.4255	0	0	0	0	0.425

Рисунок 21 – TFIDF

Давайте интерпретируем полученные числа. слова «ты родился» повторяются на протяжении всего стихотворения. Поэтому мы не ожидаем, что эти слова получат высокие оценки TF-IDF. Если вы посмотрите на значения этих трех слов, вы увидите, что чаще всего они находятся в диапазоне от 0,2 до 0,3.

Давайте посмотрим на Документ 0 - вы родились с потенциалом. Слово потенциал выделяется. Если вы посмотрите на различные значения TF-IDF в первой строке матрицы, вы увидите, что слово «потенциальный» имеет наивысшее значение TF-IDF.

Давайте посмотрим на Документ 0 - вы родились с потенциалом. Слово потенциал выделяется. Если вы посмотрите на различные значения TF-IDF в первой строке матрицы, вы увидите, что слово «потенциальный» имеет наивысшее значение TF-IDF.

Цель этой статьи - увидеть, как указанные выше значения TF-IDF могут быть рассчитаны с нуля. Мы используем расчетные значения для слов «крылья» и «потенциал», в частности, чтобы получить те, которые выделены красным в приведенной выше матрице.

модель разделит компоненты, а затем снова соединит их. Мы сделаем это в три этапа:

Шаг 1. Получите значения частоты использования терминов с нуля

Шаг 2. Получите значения частоты обратного документа с нуля

Шаг 3. Агрегируйте два вышеупомянутых значения, используя умножение и нормализацию.

Шаг 1. Рассчитайте значения частоты запросовТермин частота довольно прост. Он рассчитывается как количество раз, когда слова / термины встречаются в документе.
Для предложений «вы родились с потенциалом» и «вы родились с крыльями» ниже приведены значения частоты терминов.

	you	were	born	with	potential
Doc 0	1	1	1	1	1

Total number of words
5

	you	were	born	with	wings
Doc 4	1	1	1	1	1

Total number of words
5

Рисунок 22 – разбиение на числа

Распространяя то же самое на все 8 предложений в стихотворении, мы получаем следующую матрицу подсчета слов. Как и раньше, строки представляют предложения, а столбцы представляют слова стихотворения, упорядоченные в алфавитном порядке. CountVectorizer возвращает разреженную матрицу, которая преобразуется во фрейм данных для простоты визуализации.

	Count of Words																		
	Vocabulary words																		
Index & Sentence	0 and	1 are	2 born	3 crawlin	4 don't	5 dreams	6 fly	7 for	8 goodne	9 greatness	10 have	11 ideals	12 learn	13 meant	14 not	15 potential	16 so	17 them
0 you were born with potential			1													1		
1 you were born with goodness and trust	1		1						1									
2 you were born with ideals and dreams	1		1			1						1						
3 you were born with greatness			1							1								
4 you were born with wings			1															
5 you are not meant for crawling, so don't		1		1	1			1						1	1		1	
6 you have wings											1							
7 learn to use them and fly	1						1						1					1
# Documents with the word	3	1	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Рисунок 23 – разбиение на числа

Вышеуказанное рассчитывается по следующему коду.

Листинг9 – «Функцияизвлеченияиз текста»

```
from sklearn.feature_extraction.text import
CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(corpus)
terms = count_vect.get_feature_names()
```

Выясним, как часто встречаются слова потенциал и крылья.

- Частота термина для слова "потенциал" в документе 0 = 1
- Частота термина для слова «крылья» в Док. 4 = 1.
- Частота термина для слова «крылья» в Док. 6 = 1.

Шаг 2: реверсивная частота документа

Должен быть механизм для уменьшения относительной важности слов, которые слишком часто встречаются во всех документах. Введите периодичность возврата документа. Интуитивно понятно, что если слово присутствует во всех документах, оно может не играть такой большой роли в различении документов. Как и в случае с конечной частотой,

Периодичность документов (срок t) = количество документов со сроком t / общее количество документов = d (t) / n

Частота реверсирования документов = общее количество документов / количество документов со сроком t = n / d (t)

Если слово присутствует во всех документах, оно должно находиться в нижней части диапазона 0–1. Таким образом, логарифмический масштаб интуитивно имеет смысл использовать здесь, поскольку log 1 равен 0.

Частота обратного документа для настроек по умолчанию в векторизаторе TF IDF в sklearn установлена, как показано ниже (настройки по умолчанию имеют smooth_idf = True, что указывает «1» в числителе и знаменателе, как если бы был просмотрен дополнительный документ, каждый отдельно термин в коллекции ровно один раз, что предотвращает нулевое деление).

$$idf(t) = \ln\left(\frac{1+n}{1+df(t)}\right) + 1$$

- n - общее количество документов в наборе документов.
- d (t) - количество документов в наборе документов, содержащих термин.
- Определим индивидуальные значения слов - потенциал и крылья.
- Количество документов = 8
- Количество документов в корпусе, содержащих слово «потенциал» = 1
- Количество документов в корпусе, содержащих слово «крылья» = 2.
- Применяя формулу для обратной частоты документов, получаем

$$idf(potential) = \ln\left(\frac{1+8}{1+1}\right) + 1 = \ln\left(\frac{9}{2}\right) + 1 = 2.504077$$

$$idf(wings) = \ln\left(\frac{1+8}{2+1}\right) + 1 = \ln\left(\frac{9}{3}\right) + 1 = 2.098612$$

Значения IDF для стихотворения можно получить, запустив следующий код Python.

Листинг10 – «изучитьIDF»

```
# exploreidf
# idf_ attribute can be used to extract IDF values
```

```
# transpose the 1D IDF array to convert to a dataframe to make it
easy to visualise
df_idf = idf2df(vectorizer.idf_[:,np.newaxis].T ,terms)
display(HTML(df_idf.to_html()))
```

Значения, как показано ниже, и можно дважды проверить значения из наших расчетов для слов потенциал и крылья. Обратите внимание, что для слова в корпусе есть только одно значение IDF.

Шаг 3: умножение и нормализация

Как следует из названия, TF-IDF представляет собой комбинацию TermFrequency (TF) и InverseDocumentFrequency (IDF), полученную путем умножения двух значений вместе. Затем реализация sklearn применяет нормализацию к продукту между TF и IDF. Давайте подробно рассмотрим каждый из этих шагов.

При умножении двух матриц мы берем поэлементное умножение матрицы частот термина и частоты обратного документа. Рассмотрим первое предложение - «Вы родились с потенциалом». Чтобы найти произведение TF и IDF для этого предложения, оно рассчитывается, как показано ниже.

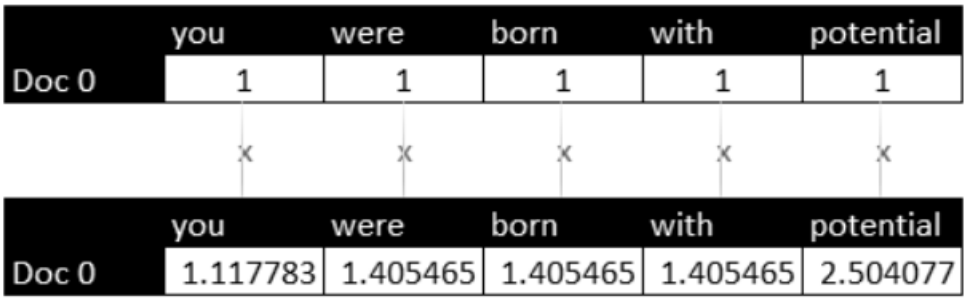


Рисунок24 – Matrix

Листинг 10 – «умножение в шоу в формате массива»

```
df_mul = df_count.mul(df_idf.to_numpy())
display(HTML(df_mul.to_html()))
```

Multiply TF and IDF Vocabulary Words																								
Index & Sentence	and	are	born	crawlin	don	dreams	fly	for	goodne	greatness	have	ideals	learn	meant	not	potential	so	them	to	trust	use	were	wings	with	you
0 you were born with potential	0	0	1.4055	0	0	0	0	0	0	0	0	0	0	0	0	2.504077	0	0	0	0	0	1.4055	0	1.4055	1.1
1 you were born with goodness and trust	1.81093	0	1.4055	0	0	0	0	0	2.5041	0	0	0	0	0	0	0	0	0	0	2.5041	0	1.4055	0	1.4055	1.1
2 you were born with ideals and dreams	1.81093	0	1.4055	0	0	2.5041	0	0	0	0	0	2.5041	0	0	0	0	0	0	0	0	0	1.4055	0	1.4055	1.1
3 you were born with greatness	0	0	1.4055	0	0	0	0	0	0	2.504077	0	0	0	0	0	0	0	0	0	0	0	1.4055	0	1.4055	1.1
4 you were born with wings	0	0	1.4055	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.4055	2.098612	1.4055	1.1
5 you are not meant for crawling, so don't	0	2.5041	0	2.5041	2.50408	0	0	2.50408	0	0	0	0	0	2.50408	2.5041	0	2.50408	0	0	0	0	0	0	0	1.1
6 you have wings	0	0	0	0	0	0	0	0	0	0	2.5041	0	0	0	0	0	0	0	0	0	0	0	2.098612	0	1.1
7 learn to use them and fly	1.81093	0	0	0	0	0	2.5041	0	0	0	0	0	2.5041	0	0	0	0	2.5041	2.5041	0	2.5041	0	0	0	

Рисунок 25 – CorpusMatrix

В повседневной жизни, когда люди хотят нормализовать значения, чтобы их можно было легко сравнивать, они используют проценты или пропорции. Таким образом, мы могли гипотетически вычислить долю значений TF-IDF в разных словах в предложении. Обратите внимание, что и TF, и IDF являются неотрицательными значениями, поскольку минимально возможное значение для TermFrequency и Inverse Document Frequency равно 0. Таким образом, удаление соотношений сторон будет эквивалентно так называемой нормализации L1. В нормализации L1 каждый элемент в векторе (подумайте о разных значениях предложения TF-IDF) делится на сумму абсолютных значений всех элементов. L1 может нормализовать значения в sklearn, но это не настройка по умолчанию.

В sklearn по умолчанию применяется нормализация L2. Самый простой способ подумать о нормализации L2 - это подумать о длине прямой или теореме Пифагора с одним из углов треугольника в начале координат.

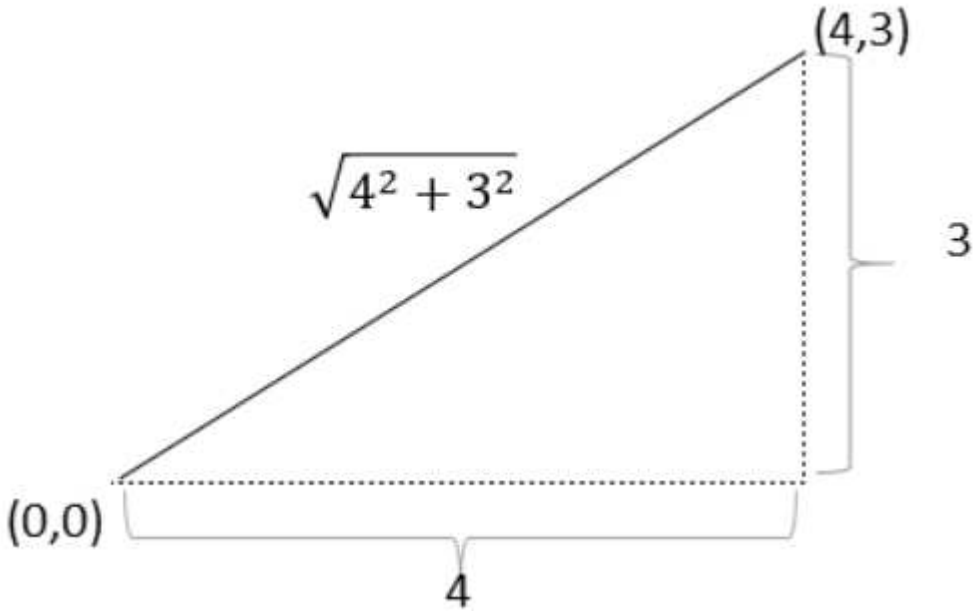


Рисунок 26 – formulas

На диаграмме выше длина линии равна 5. В этом случае линия является одномерным вектором. Когда векторы являются n-мерными, длина вектора аналогична длине линии, но увеличивается до n измерений. Итак, если вектор v состоит из n элементов, длина вектора вычисляется как

$$\|V\| = \sqrt{v_1^2 + v_2^2 + v_3^2 \ \ldots\ldots v_n^2}$$

В нормализации L2, по сути, есть деление вектора на длину вектора. Для более математического объяснения норм L1 и L2.

Чтобы применить норму L2 для каждого предложения, нам нужно вычислить квадратный корень из суммы квадратов произведения TF и IDF.

Это можно сделать на Python, как показано ниже:

Index & Sentence	SQRT (Sum of Squares of the product of TF and IDF)
0 you were born with potential	3.666856427
1 you were born with goodness and trust	4.795383733
2 you were born with ideals and dreams	4.795383733
3 you were born with greatness	3.666856427
4 you were born with wings	3.402882126
5 you are not meant for crawling, so don't	6.71879827
6 you have wings	3.453116387
7 learn to use them and fly	5.884851364

Рисунок27 – Окончательный результат TF-IDF

Наконец, алгоритмы TFIDF готовы к расчету

TF-IDF для слова "потенциал в вас родились с потенциалом" (Doc 0):

2,504077 / 3. 66856427 = 0,682895.

TF-IDF для слова «крылья» у вас родились с крыльями

(Док. 4) = 2,098612 / 3. 402882126 = 0,616716.

TF-IDF для слова «крылья у вас есть крылья»

(Док. 6) = 2,098612 / 3. 452116387 = 0,607744

Это можно рассчитать с помощью следующего кода.

Листинг 10 – «умножение в шоу в формате массива»

```
from sklearn.preprocessing import Normalizer
df_mul.iloc[:,:] = Normalizer(norm='l2').fit_transform(df_mul)
display(HTML(df_mul.to_html()))
```

		Vocabulary words																
	Index & Sentence	and	are	born	crawlin	don	dreams	fly	for	goodne	greatne	have	ideals	learn	meant	not	potential	so
0	you were born with potential	0	0	0.3833	0	0	0	0	0	0	0	0	0	0	0	0	0.682895	0
1	you were born with goodness and trust	0.3776	0	0.2931	0	0	0	0	0	0.5222	0	0	0	0	0	0	0	0
2	you were born with ideals and dreams	0.3776	0	0.2931	0	0	0.5222	0	0	0	0	0	0.5222	0	0	0	0	0
3	you were born with greatness	0	0	0.3833	0	0	0	0	0	0	0.6829	0	0	0	0	0	0	0
4	you were born with wings	0	0	0.413	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	you are not meant for crawling, so don't	0	0.3727	0	0.3727	0.3727	0	0	0.3727	0	0	0	0	0	0.3727	0.3727	0	0.3727
6	you have wings	0	0	0	0	0	0	0	0	0	0	0.7252	0	0	0	0	0	0
7	learn to use them and fly	0.3077	0	0	0	0	0	0.4255	0	0	0	0	0	0.4255	0	0	0	0

Рисунок28 – Final TF-IDF score

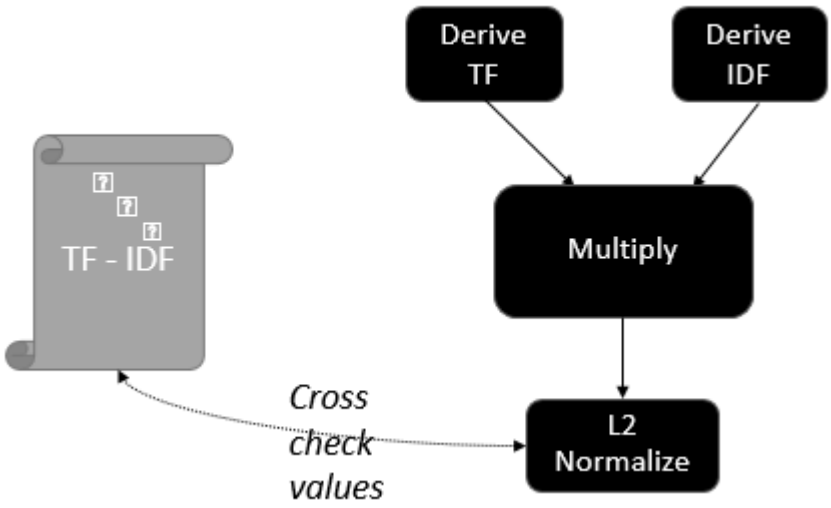


Рисунок 29 – Графическоепредставление TFIDF

3.3.4 ОГРАНИЧЕНИЕ

Основным ограничением TF-IDF является то, что порядок слов, который является важной частью понимания значения предложения, не учитывается в TF-IDF.

Кроме того, длина документа может привести к значительным отклонениям в значениях TFIDF.

swifter предназначен для обучения и тестирования модели

Листинг 12 – «установитьswifter и подать заявку на модель
»

```
stop_words_ = set(stopwords.words('english'))
wn = WordNetLemmatizer()
my_sw = ['make', 'amp',  'news','new' ,'time', 'u','s', 'photos',
'get', 'say']

def black_txt(token):
return  token not in stop_words_ and token not in
list(string.punctuation)  and len(token)>2 and token not in my_sw

def clean_txt(text):
```

```
clean_text = []
    clean_text2 = []
    text = re.sub("'", "",text)
    text=re.sub("(\\d|\\W)+"," ",text)
clean_text = [ wn.lemmatize(word, pos="v") for word in
word_tokenize(text.lower()) if black_txt(word)]
    clean_text2 = [word for word in clean_text if black_txt(word)]
    return " ".join(clean_text2)

def subj_txt(text):
return  TextBlob(text).sentiment[1]

def polarity_txt(text):
    return TextBlob(text).sentiment[0]

def len_text(text):
    if len(text.split())>0:
         return len(set(clean_txt(text).split()))/
len(text.split())
    else:
         return 0


final_dataset['text'] =
final_dataset['text'].swifter.apply(clean_txt)

final_dataset['polarity'] =
final_dataset['text'].swifter.apply(polarity_txt)

final_dataset['subjectivity'] =
final_dataset['text'].swifter.apply(subj_txt)

final_dataset['len'] = final_dataset['text'].swifter.apply(lambda
x: len(x))
```

```
X = final_dataset[['text', 'polarity', 'subjectivity','len']]
y =final_dataset['Category']

encoder = LabelEncoder()
y = encoder.fit_transform(y)

x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, stratify=y)
v = dict(zip(list(y), final_dataset['Category'].to_list()))

text_clf = Pipeline([
...     ('vect', CountVectorizer(analyzer="word",
stop_words="english")),
...     ('tfidf', TfidfTransformer(use_idf=True)),
...     ('clf', MultinomialNB(alpha=.01)),
... ])

text_clf.fit(x_train['text'].to_list(), list(y_train))
```

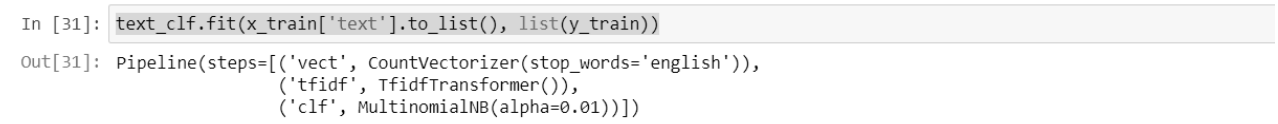


Рисунок30 – Результат кода

Листинг 12 – «Текстовое предсказание»

```
X_TEST = x_test['text'].to_list()

Y_TEST = list(y_test)

predicted = text_clf.predict(X_TEST)

c = 0

for doc, category in zip(X_TEST, predicted):

    if c == 5:break
```

```
print("-"*55)
    print(doc)
    print(v[category])
print("-"*55)

    c = c + 1
```

```
nature adventure travel company smartphone policies think nature adventure travel company official cell phone policies place ev
er bad good experience cell phone users adventure trip
TRAVEL
-------------------------------------------------------
-------------------------------------------------------
hilary duff publicly drag neighbor keep night nightmares
SCIENCE
-------------------------------------------------------
-------------------------------------------------------
john legend chrissy teigen red hot red carpet
TRAVEL
-------------------------------------------------------
-------------------------------------------------------
ultra cheap microscopes could save millions live microscope one quintessential symbols science invoke image researchers white l
ab coat microscopes save live decades help diagnose number deadly diseases many part world short supply
TECH
-------------------------------------------------------
-------------------------------------------------------
like free dive whale filmmaker photographer michaela skovranova family move slovakia sydney jump big
SCIENCE
```

Рисунок31 – Текстовое предсказание

Листинг 13 – «МодельTFIDF Точность»

```
predicted = text_clf.predict(X_TEST)
np.mean(predicted == Y_TEST)
```

```
predicted = text_clf.predict(X_TEST)
np.mean(predicted == Y_TEST)
```

```
0.8549984030661131
```

Рисунок 32 – Модель TFIDF Точность

Как видно, есть хороший прогноз точности, 85 процента для НЛП.

это хороший результат, так что продолжим с этим результатом.

3.4 ПРИМЕНЕНИЕ МОДЕЛИ К ДАННЫМ TWITTER

В этом разделе собираемся применить модель к данным Twitter.

Листинг 14 – «модель к данным Twitter»

```
df_tweets = pd.DataFrame(columns =
["id","conversation_id","created_at","date","time","timezone","use
r_id","username","name","place","tweet",

"language","mentions","urls","photos","replies_count","hashtags","
cashtags","link",

"retweet","quote_url","video","thumbnail","near","geo","source","u
ser_rt_id",

"user_rt","retweet_id","reply_to","retweet_date","translate","tran
s_src","trans_dest"])


for i in range(0, len(data)):
currentItem = data[i]
df_tweets.loc[i] =
[data[i]["id"],data[i]["conversation_id"],data[i]["created_at"],da
ta[i]["date"],data[i]["time"],data[i]["timezone"],data[i]["user_id
"],

data[i]["username"],data[i]["name"],data[i]["place"],data[i]["twee
t"],data[i]["language"],data[i]["mentions"],data[i]["urls"],

data[i]["photos"],data[i]["replies_count"],data[i]["hashtags"],dat
a[i]["cashtags"],data[i]["link"],

data[i]["retweet"],data[i]["quote_url"],data[i]["video"],data[i]["
thumbnail"],data[i]["near"],data[i]["geo"],data[i]["source"],
data[i]["user_rt_id"],data[i]["user_rt"],data[i]["retweet_id"],dat
a[i]["reply_to"],data[i]["retweet_date"],data[i]["translate"],data
[i]["trans_src"],

                data[i]["trans_dest"]]
df_tweets[['username', 'tweet']]
```

Out[45]:

	username	tweet
0	billgates	The amount of cement China has consumed is a s...
1	billgates	Cement is responsible for 6% of carbon emissio...
2	billgates	Who installed the most concrete? Check your a...
3	billgates	The best way to prevent new variants from emer...
4	billgates	It's encouraging to see innovation and clean e...
...	...	...
18655	kapilsharmak9	@feedbacks give your feed backs
18656	kapilsharmak9	my new show's promo... comedy nights with kapil..
18657	kapilsharmak9	COMEDY NIGHTS WITH KAPIL, STARRING DHARMENDRA,...
18658	kapilsharmak9	working on my new show comedy nights with kapi...
18659	kapilsharmak9	hi frns.its my first tweet..earlier i had a ac...

Рисунок 33 – pandasmodel

3.4.1 ОЧИСТКА ДАННЫХ

Интеллектуальный анализ данных — это процесс извлечения ценной информации из данных, которая может использоваться для принятия бизнес-решений и стратегии. Но прежде, чем можно будет начать интеллектуальный анализ данных, важно потратить время на их очистку. Очистка данных - это процесс подготовки необработанных данных для анализа путем удаления неверных данных, организации необработанных данных и заполнения нулевых значений. В итоге очистка данных подготавливает данные к процессу интеллектуального анализа данных, когда наиболее ценная информация может быть извлечена из набора данных.

Способность понимать и исправлять качество ваших данных является обязательным условием для проведения точного окончательного анализа. Данные необходимо подготовить, чтобы выявить важные закономерности. Интеллектуальный анализ данных считается исследовательским; Очистка данных при интеллектуальном анализе данных дает пользователю возможность обнаруживать неточные или неполные данные до проведения бизнес-анализа и

анализа. В большинстве случаев очистка данных при интеллектуальном анализе данных может быть трудоемким процессом и обычно требует ИТ-ресурсов для помощи на начальном этапе оценки ваших данных. Поскольку очистка данных перед интеллектуальным анализом данных занимает очень много времени, это создает дилемму для аналитиков данных: у вас не хватает персонала или времени для очистки данных. Но без надлежащего качества данных ваш окончательный анализ может потерять точность или вы потенциально можете прийти к неверному выводу.

Современная очистка данных для интеллектуального анализа данных с помощью автоматизированных инструментов визуального профилирования в Trifacta Wrangler экономит время и деньги, предлагая превосходные результаты по сравнению с методами ручного профилирования. По оценкам Forrester, до 80% времени большинства аналитиков тратится на подготовку данных. Trifacta помогает компаниям или организациям немедленно сократить время, затрачиваемое на очистку данных при интеллектуальном анализе данных. После этого предприятия и организации могут обмениваться лучшими и последовательными результатами из централизованного хранилища - независимо от уровня пользователя и операционной системы.

Очистка данных в интеллектуальном анализе данных имеет неизмеримое значение при работе с большими данными. Trifacta помогает компаниям любого размера максимально увеличить эту ценность, включая исключительную визуализацию в обработку данных, инструменты и методы на всех этапах любого проекта миграции данных.

Есть несколько шагов, чтобы очистить набор данных новостей, чтобы удалить ненужный контент и выделить ключевые атрибуты, подходящие для модели машинного обучения.

В тексте заголовка есть несколько знаков препинания. Пунктуация часто не нужна, потому что она не добавляет ценности или смысла модели НЛП. В «строковой» библиотеке 32 знака препинания. Знаки препинания:

Чтобы удалить знаки препинания в нашем наборе данных, давайте создадим функцию и применим ее к набору данных:

Листинг 15 – «remove_punctuation»

```
def remove_punctuation(text):
no_punct=[words for words in text if words not in
string.punctation]
words_wo_punct=''.join(no_punct)
    return words_wo_punct
news['title_wo_punct']=news['title'].apply(lambda x:
remove_punctuation(x))
news.head()
```

Токенизация— это процесс разделения строк на список слов. использовать регулярные выражения или регулярное выражение для разделения. Регулярное выражение можно использовать для описания шаблона поиска.

Листинг 15 – «токенизация и разделение модели»

```
deftokenize(text):
split=re.split("\W+",text)
    return split
news['title_wo_punct_split']=news['title_wo_punct'].apply(lambda
x: tokenize(x.lower()))
news.head()
```

Есть список слов без знаков препинания. Уберем стоп-слова. Стоп-слова - это не относящиеся к делу слова, которые не помогут идентифицировать текст как настоящий или поддельный. Можно использовать библиотеку "nltk" для стоп-слов и некоторых стоп-слов в этой библиотеке:

Стемминг и лемматизация- это процесс сведения слова к его корневой форме. Основная цель - уменьшить количество вариаций одного и того же слова, тем самым уменьшив корпус слов, которые мы включаем в модель. Разница между выделением корней и лемматизацией заключается в том, что при построении

корней слова обрезаются без учета контекста слова. Принимая во внимание, что лемматизация учитывает контекст слова и сокращает слово до его корневой формы на основе определения словаря. Стемминг - более быстрый процесс по сравнению с леммантизацией. Следовательно, это компромисс между скоростью и точностью.

Рассмотрим, например, слово «вера». Различные варианты веры могут быть верой, верой, верой и верой.

Другие шаги очистки могут быть выполнены на основе данных. Некоторые из них перечислены ниже,

- Удалить URL
- Удалить HTML-теги
- Удалить смайлики
- Удалить числа

Листинг 16 – «Очистка текста twitter»

```
def clearText(text):
    text = text.lower()
    emoji = re.compile("["
                           u"\U0001F600-\U0001FFFF"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols &
pictographs
                           u"\U0001F680-\U0001F6FF"  # transport &
map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
"]+", flags=re.UNICODE)
    text = emoji.sub(r'', text)
    text = re.sub(r'#[A-Za-z0-9]+','',text)
    text = re.sub(r'()+','',text)
```

```
    text = re.sub(r'RT[\s]+','',text)
    text = re.sub(r'@[A-Za-z0-9]+','',text)
    text = re.sub(r'https?:\/\/\S+','',text)
    text = re.sub(r"it's", "It is", text)
    text = re.sub(r"i'm", "i am", text)
    text = re.sub(r"he's", "he is", text)
    text = re.sub(r"she's", "she is", text)
    text = re.sub(r"that's", "that is", text)
    text = re.sub(r"what's", "what is", text)
    text = re.sub(r"where's", "where is", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"don't", "do not", text)
    text = re.sub(r"did't", "did not", text)
    text = re.sub(r"can't", "can not", text)


    returntext


df_tweets['clean_tweet'] = df_tweets['tweet'].apply(clearText)

predicted = text_clf.predict(df_tweets['clean_tweet'])

c=0
result = []
for x in predicted:
    print(df_tweets['clean_tweet'].iloc[x])
result.append(v[x])

print('----------')
```

```
print(v[x])


print('----------')
```

```
after you finish your pancakes this morning, come ask me anything on  at 11:15 pacific time:
----------
TRAVEL
----------
the technological transformation we need to address climate change can create good, safe jobs and build a more equitable, pro
sperous economy. to make that happen, we need to think big:
----------
SCIENCE
----------
after you finish your pancakes this morning, come ask me anything on  at 11:15 pacific time:
----------
TRAVEL
----------
in this video, i answered some really good questions, including one about two really important numbers. come ask me some ques
tions of your own on  at 11:15:
----------
TECH
----------
the technological transformation we need to address climate change can create good, safe jobs and build a more equitable, pro
sperous economy. to make that happen, we need to think big:
```

Рисунок34 – предсказывать твиты в твиттере

Листинг 17 – «просмотр с категорией твита»

```
df_tweets['tweet-category'] = my_result

df_tweets[['username','tweet','clean_tweet','tweet-category']]
```

Out[53]:

	username	tweet	clean_tweet	tweet-category
0	billgates	The amount of cement China has consumed is a s...	the amount of cement china has consumed is a s...	TRAVEL
1	billgates	Cement is responsible for 6% of carbon emissio...	cement is responsible for 6% of carbon emissio...	SCIENCE
2	billgates	Who installed the most concrete? Check your a...	who installed the most concrete? check your a...	TRAVEL
3	billgates	The best way to prevent new variants from emer...	the best way to prevent new variants from emer...	TECH
4	billgates	It's encouraging to see innovation and clean e...	It is encouraging to see innovation and clean ...	SCIENCE
...	...	...	...	...
18655	kapilsharmak9	@feedbacks give your feed backs	give your feed backs	TRAVEL
18656	kapilsharmak9	my new show's promo... comedy nights with kapil..	my new show's promo... comedy nights with kapil..	ENTERTAINMENT
18657	kapilsharmak9	COMEDY NIGHTS WITH KAPIL, STARRING DHARMENDRA,...	comedy nights with kapil, starring dharmendra,...	ENTERTAINMENT
18658	kapilsharmak9	working on my new show comedy nights with kapi...	working on my new show comedy nights with kapi...	ENTERTAINMENT
18659	kapilsharmak9	hi frns.its my first tweet..earlier i had a ac...	hi frns.its my first tweet..earlier i had a ac...	STYLE & BEAUTY

Рисунок35 – просмотр с категорией твита

Теперь есть твиты с каждой их категорией, и теперь можно применить алгоритм оценки.

Листинг 18 – «твиты с категорией в форме DataFrame»

```
sdd = df_tweets.groupby('username')['tweet-
category'].value_counts()
```

```
resu = sdd.groupby('username').head(5)

result = resu.to_json(orient='split')
parsed = json.loads(result)
parsed

{'name': 'tweet-category',
 'index': [['billgates', 'TRAVEL'],
  ['billgates', 'ENTERTAINMENT'],
  ['billgates', 'SCIENCE'],
  ['billgates', 'EDUCATION'],
  ['billgates', 'STYLE & BEAUTY'],
  ['chrisrock', 'ENTERTAINMENT'],
  ['chrisrock', 'TRAVEL'],
  ['chrisrock', 'STYLE & BEAUTY'],
  ['chrisrock', 'CRIME'],
  ['chrisrock', 'RELIGION'],
  ['debruynekev', 'ENTERTAINMENT'],
  ['debruynekev', 'TRAVEL'],
  ['debruynekev', 'STYLE & BEAUTY'],
  ['debruynekev', 'SPORT'],
  ['debruynekev', 'BUSINESS'],
  ['kapilsharmak9', 'ENTERTAINMENT'],
  ['kapilsharmak9', 'TRAVEL'],
  ['kapilsharmak9', 'STYLE & BEAUTY'],
  ['kapilsharmak9', 'RELIGION'],
  ['kapilsharmak9', 'CRIME'],
  ['vp45', 'ENTERTAINMENT'],
  ['vp45', 'TRAVEL'],
  ['vp45', 'RELIGION'],
  ['vp45', 'EDUCATION'],
  ['vp45', 'BUSINESS']],
 'data': [1046,
  945,
```

```
  411,
  338,
  144,
  1570,
  172,
  173,
  94,
  57,
  32,
  5440,
  766,
  332,
  217,
  64,
  1672,
  1467,
  1186,
  304,
  269]}

result_4[['Name','Category']] =
pd.DataFrame(result_4.name.tolist(), index= result_4.index)
```

Out[66]:

	name	data	Name	Category
0	[billgates, TRAVEL]	1046	billgates	TRAVEL
1	[billgates, ENTERTAINMENT]	945	billgates	ENTERTAINMENT
2	[billgates, SCIENCE]	411	billgates	SCIENCE
3	[billgates, EDUCATION]	338	billgates	EDUCATION
4	[billgates, STYLE & BEAUTY]	144	billgates	STYLE & BEAUTY
5	[chrisrock, ENTERTAINMENT]	1570	chrisrock	ENTERTAINMENT
6	[chrisrock, TRAVEL]	172	chrisrock	TRAVEL
7	[chrisrock, STYLE & BEAUTY]	135	chrisrock	STYLE & BEAUTY
8	[chrisrock, CRIME]	38	chrisrock	CRIME
9	[chrisrock, RELIGION]	13	chrisrock	RELIGION
10	[debruynekev, ENTERTAINMENT]	735	debruynekev	ENTERTAINMENT
11	[debruynekev, TRAVEL]	173	debruynekev	TRAVEL
12	[debruynekev, STYLE & BEAUTY]	94	debruynekev	STYLE & BEAUTY
13	[debruynekev, SPORT]	57	debruynekev	SPORT
14	[debruynekev, BUSINESS]	32	debruynekev	BUSINESS
15	[kapilsharmak9, ENTERTAINMENT]	5440	kapilsharmak9	ENTERTAINMENT
16	[kapilsharmak9, TRAVEL]	766	kapilsharmak9	TRAVEL
17	[kapilsharmak9, STYLE & BEAUTY]	332	kapilsharmak9	STYLE & BEAUTY
18	[kapilsharmak9, RELIGION]	217	kapilsharmak9	RELIGION
19	[kapilsharmak9, CRIME]	64	kapilsharmak9	CRIME
20	[vp45, ENTERTAINMENT]	1672	vp45	ENTERTAINMENT

Рисунок36 – твиты с категорией в форме DataFrame

Листинг 19 – «Твиты с категорией с рейтингом»

```
list = []
for i in range(int(len(result5)/5)):
    for y in range(5, 0, -1):
list.append(y)
result5['rate'] = list
users = result5['Name'].unique()
users
result5
```

	data	Name	Category	rate
0	1046	billgates	TRAVEL	5
1	945	billgates	ENTERTAINMENT	4
2	411	billgates	SCIENCE	3
3	338	billgates	EDUCATION	2
4	144	billgates	STYLE & BEAUTY	1
5	1570	chrisrock	ENTERTAINMENT	5
6	172	chrisrock	TRAVEL	4
7	135	chrisrock	STYLE & BEAUTY	3
8	38	chrisrock	CRIME	2
9	13	chrisrock	RELIGION	1
10	735	debruynekev	ENTERTAINMENT	5
11	173	debruynekev	TRAVEL	4
12	94	debruynekev	STYLE & BEAUTY	3
13	57	debruynekev	SPORT	2
14	32	debruynekev	BUSINESS	1
15	5440	kapilsharmak9	ENTERTAINMENT	5
16	766	kapilsharmak9	TRAVEL	4
17	332	kapilsharmak9	STYLE & BEAUTY	3
18	217	kapilsharmak9	RELIGION	2
19	64	kapilsharmak9	CRIME	1
20	1672	vp45	ENTERTAINMENT	5

Рисунок37 – Твиты с категорией с рейтингом

РЕКОМЕНДАЦИЯ ДВИГАТЕЛИ

В этом разделе я опишу, как работает система рекомендаций, все типы системы рекомендаций и какая из них подойдет для моей системы или механизма рекомендаций.

3.5 ВСТУПЛЕНИЕ

Рекомендательные системы можно использовать в сценариях, когда многие пользователи взаимодействуют со многими элементами.

За последние несколько десятилетий, с появлением Youtube, Amazon, Netflix и многих других подобных веб-сервисов, системы рекомендаций занимали все больше и больше места в нашей жизни.

От электронной коммерции (предлагая клиентам статьи, которые могут их заинтересовать) до онлайн-рекламы (предлагая пользователям правильный контент, соответствующий их предпочтениям), системы рекомендаций сегодня неизбежны в наших ежедневных онлайн-путешествиях.

В самом общем смысле системы рекомендаций - это алгоритмы, нацеленные на то, чтобы предлагать пользователям релевантные товары (например, фильмы для просмотра, текст для чтения, продукты для покупки или что-то еще, в зависимости от отрасли).

Рекомендательные системы действительно важны в некоторых отраслях, потому что они могут принести огромную прибыль, когда они эффективны, или также могут быть способом значительно выделиться среди конкурентов. В качестве доказательства важности рекомендательных систем мы можем упомянуть, что несколько лет назад Netflix организовал конкурс («Приз Netflix»), целью которого было создание рекомендательной системы, которая работала бы лучше, чем ее собственный алгоритм, удостоенный награды. 1 миллион долларов на победу.

Компании электронной коммерции и розничной торговли используют возможности данных и увеличивают продажи, внедряя системы рекомендаций на своих веб-сайтах. Сценарии использования этих систем в последние годы неуклонно росли, и сейчас самое время глубже погрузиться в эту удивительную технику машинного обучения.

3.6 ЧТО ТАКОЕ СИСТЕМА РЕКОМЕНДАЦИЙ?

Рекомендательные системы разработаны, чтобы предугадывать интересы пользователей и рекомендовать продукты, которые их больше всего интересуют. Это одни из самых мощных систем машинного обучения, которые интернет-магазины используют для увеличения продаж.

Данные, необходимые для рекомендательных систем, поступают из явных оценок пользователей после просмотра фильма или прослушивания

песни, неявных поисков и историй покупок или других знаний о самих пользователях / продуктах.

Такие сайты, как Spotify, YouTube или Netflix, используют эти данные, чтобы предлагать плейлисты, так называемые ежедневные миксы или давать рекомендации по видео, соответственно.

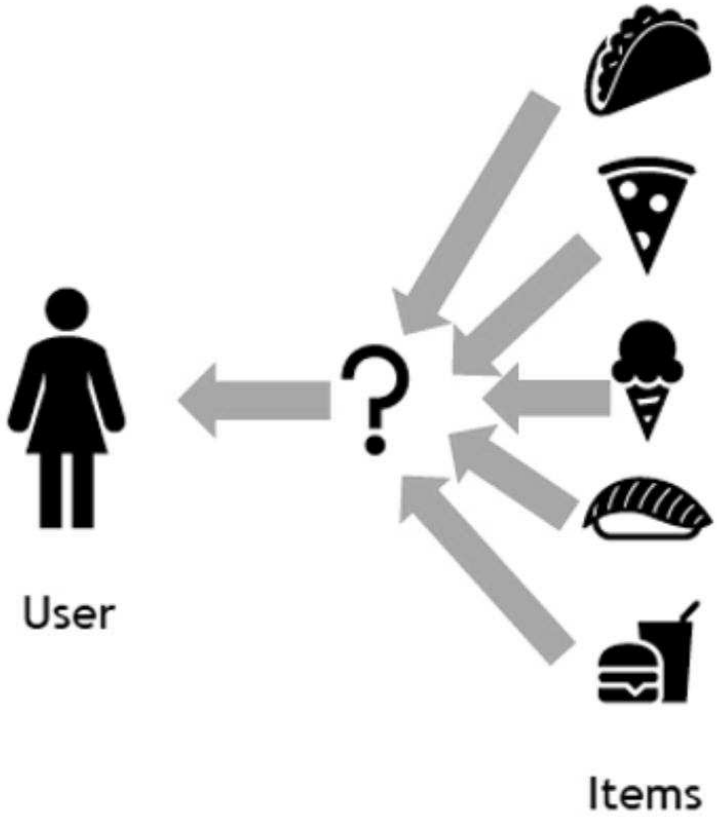


Рисунок38 – Пример рекомендательной системы

3.7 ЗАЧЕМ НАМ НУЖНЫ РЕКОМЕНДАЦИОННЫЕ СИСТЕМЫ

Компании, использующие реферальные системы, сосредоточены на увеличении продаж за счет персонализированных предложений и лучшего обслуживания клиентов.

Рекомендации обычно ускоряют поиск и облегчают пользователям доступ к интересующему их контенту, а также удивляют их предложениями, которые они никогда бы не искали.

Более того, компании могут привлекать и удерживать клиентов, рассылая электронные письма со ссылками на новые предложения, которые служат

интересам получателей, или предложения фильмов и телешоу, соответствующие их профилям.

Пользователь начинает чувствовать, что его знают и понимают, и он с большей вероятностью купит дополнительные продукты или потребит больше контента. Зная, чего хочет пользователь, компания получает конкурентное преимущество, и угроза потери клиента в пользу конкурента снижается.

Обеспечение этой дополнительной ценности для пользователей путем включения рекомендаций в системы и продукты является привлекательным. Это также позволяет компаниям опережать своих конкурентов и в конечном итоге увеличивать свои доходы.

3.8 КАК РАБОТАЕТ СИСТЕМА РЕКОМЕНДАЦИЙ

Рекомендательные системы работают с двумя типами информации:

- Характерная информация. Это информация об элементах (ключевые слова, категории и т. Д.) И пользователях (предпочтения, профили и т. Д.).
- Взаимодействие пользователя с предметом. Это такая информация, как рейтинги, количество покупок, лайков и т. Д.

Исходя из этого, в рекомендательных системах используются три алгоритма:

- Контентные системы, использующие характерную информацию.
- Системы совместной фильтрации, основанные на взаимодействии настраиваемых элементов.
- Гибридные системы, сочетающие в себе оба типа информации, чтобы избежать проблем, возникающих при работе только с одним типом.
- Далее мы немного углубимся в контент-ориентированные и совместные системы фильтрации и посмотрим, чем они отличаются.

3.9 ВИДЫ СИСТЕМЫ РЕКОМЕНДАЦИЙ

3.9.1 DEMOGRAPHICСИСТЕМА ДЕМОГРАФИЧЕСКИХ:

Система DemographicRecommender генерирует рекомендации на основе пользователя демографические признаки.

Он классифицирует пользователей на основе их атрибутов и рекомендует фильмы, используя их демографические данные [4]. В отличие от совместная фильтрация и система рекомендаций на основе контента, ее легко реализовать и не требует оценок пользователей.

Предлагаемая структура состоит из 3 этапов: источник данных, вычисление подобия.

На этапе ввода данных собираются демографические данные пользователей и рейтинги пользователей, которые являются два основных источника ввода. На этапе вычисления сходства используются демографические данные пользователей и разбиения на группы, имеющие одинаковые демографические особенности.

Эти перегородки впоследствии используются для расчета сходства пользователей на основе оценок, присвоенных пользователей фильмов, тем самым формируя еще один подкластер похожих пользователей, используя эффективный алгоритм кластеризации.

Каждый пользователь может принадлежать к одному из "n" кластеры, и эти кластеры содержат тех же пользователей, что и другие кластеры, что дает больше точности. Наконец, этап рекомендаций рекомендует фильмы целевому пользователю на основе мнений похожих пользователей, т. е. того, к какому кластеру принадлежит пользователь, который вычисляется на основе демографических данных пользователей, собранных из профиля пользователя и рейтингов пользователей.

Преимущество предлагаемой системы в том, что она сокращает время генерациирекомендации путем создания подмножеств пользователей.

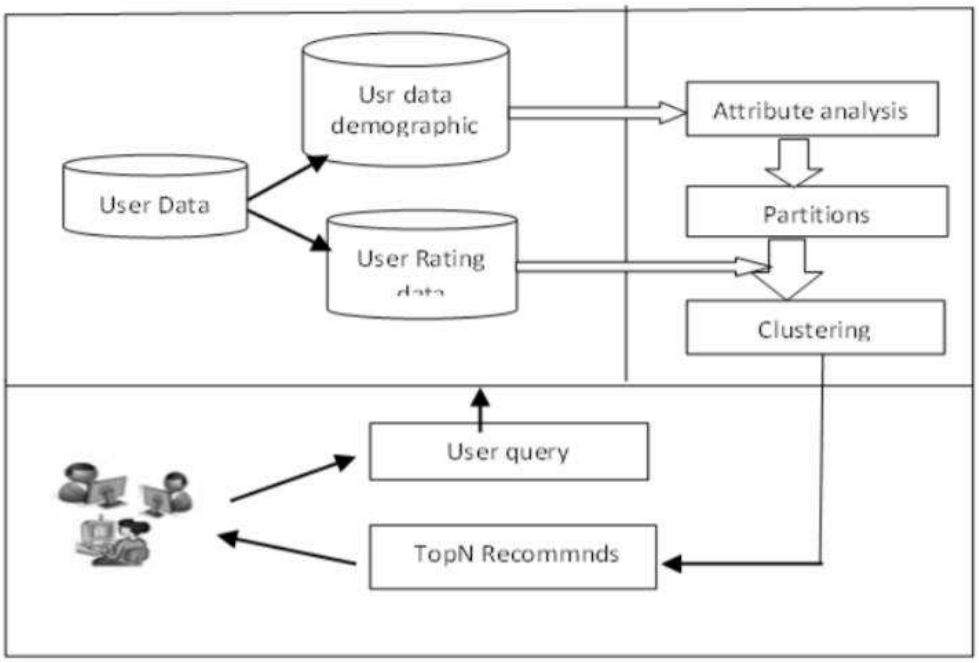


Рисунок39 – Структура Предлагаемой системы

Евклидово расстояние используется для расчета расстояния между пользователями, чтобы найти похожих пользователей на основе рейтинговой матрицы и алгоритма K-средних, формирующих кластер похожих пользователей.

Расстояние между пользователями x1и x2 рассчитывается ниже:

$$r_2(x^1, x^2) = \text{Dist}(x^1, x^2) = \sqrt{(x_1^1 - x_1^2)^2 + (x_2^1 - x_2^2)^2 + \cdots + (x_D^1 - x_D^2)^2} = \sqrt{\sum_{d=1}^{D}(x_d^1 - x_d^2)^2}$$

вычисление сходства между двумя пользователями x и y путем выполнения следующих вычислений

Его рекомендуют Книги, взвешивая оценку «каждого критика» с вероятностью пользователя. Общий балл для каждого фильма получается путем сложения этих взвешенных баллов. Если u - пользователь и у нас есть C

критиков, то оценочная оценка дается пользователю фильма z, scu (z) получается следующим образом:

$$SC_u(Z) = \frac{1}{\sum_{c=1}^{C} sim(X^u, X^c)} \sum_{c=1}^{C} sim(X^u, X^C).SC_c(Z)$$

Предлагаемый алгоритм:

Входные данные: количество кластеров K, демографические данные пользователей, матрица рейтингов пользовательских элементов;

Вывод: Topn рекомендаций целевому пользователю.

Начинать:

1. Рассмотрим набор пользователей = {U1, U2, U3,…. Гм}
2. Рассмотрим набор Movie = {M1, M2, M3… .Mn}
3. Рассмотрим рейтинговую матрицу UmXMn.
4. Извлеките любые два атрибута из пользовательского набора Um. Пусть это будет возрастная группа и пол.

5.Разделите пользователей на разделы, рассмотрев все возможные комбинации значений выбранных.

атрибуты, то есть {M-M}, {M-F}, {S-M}, {S-F}. Пусть это будет PU = {PU1, PU2, PU3, PU4}

где каждый пользователь может принадлежать к любому из разделов PUi.

6. для каждого PUi, где i = от 1 до 4

CU = {CU1, CU2,…CUk}

i. Select ‘k’ users randomly as the initial cluster centre

CU={CU1,CU2,…CUk}

II. Для каждого пользователя Um∈PUi

a. Для каждого центра кластера CUj∈CU

б) Вычислить sim (Um, CUj) на основе рейтинга фильма пользователя.

iii. Обновите центры кластеризации.

iv. Повторяйте шаг 6.ii до тех пор, пока центры кластеризации не останутся без изменений.

7. конец для.

8. Поместите активного пользователя Ua в наиболее похожий кластер, сгенерированный из раздела списки.

9. Сгенерируйте прогнозы и порекомендуйте лучшие фильмы целевому пользователю.

10. Конец.

3.9.2 UTILITY BASEDСИСТЕМА РЕКОМЕНДАЦИЙ:

Принимать решения нелегко, и по мере того, как становится доступно все больше и больше вариантов, клиенты сталкиваются с нашими ограничениями, когда дело доходит до выбора. К счастью, компании предоставляют реферальные системы, чтобы помочь клиентам сделать выбор. Они предлагают похожие продукты, связывают это с предыдущим поведением или показывают нам, что выбрали другие клиенты. Но реферальные системы помогают не только клиентам. Собирая все эти в основном персонализированные данные, он также может предоставить компании более подробную информацию о своих клиентах. Эта информация может быть использована для составления личных предложений или корректировки цен в зависимости от популярности товаров.

Шольц и его коллеги исследовали эту возможность, проверив, могут ли системы рекомендаций на основе коммунальных услуг измерить готовность потребителей платить (WTP) за продукт (Scholzetal. 2015). В этом посте мы обсудим их статью и кратко опишем их исследования.

В настоящее время доступны модели принятия решений, которые измеряют предпочтения клиентов и WTP, но они дороги и требуют больших усилий со стороны клиентов, чтобы определить их предпочтения. Но утверждается, что рекомендательные системы на основе коммунальных услуг решают обе эти проблемы, потому что клиенты хотят и мотивированы использовать системы рекомендаций, одновременно предоставляя информацию

компании. Это делает их более надежными, менее дорогостоящими для клиентов и менее дорогостоящими для компаний.

Для этого исследования они использовали рекомендательные системы на основе утилит. Этот тип системы требует активного ввода от клиента о предпочтении полезности для одного атрибута (SAU) и может вычислить полезность клиента для всех атрибутов в общий коэффициент полезности, называемый многоатрибутной полезностью (MAU). ... Фактор может быть линейным или нелинейным, оба из которых проверены в этом исследовании. Использование этой системы вызывает споры, потому что исследователи заявили, что усилия клиентов должны быть минимальными. Однако в документе утверждается, что большую часть этого активного вклада, о котором спрашивали респонденты, можно получить из онлайн-источников в реальной жизни.

Переменная WTP измеряется на уровне продукта. Респондентам было предложено заполнить WTP из-за безразличия к конкретному продукту. Этот WTP разделен на несколько уровней:
Floor WTP: это максимальная цена, по которой покупатель обязательно купит товар (100% вероятность покупки).

Безразличие WTP: это точка безразличия между покупкой или отказом от покупки продукта (вероятность покупки 50%).
Потолок WTP: самая высокая цена из всех, но самая низкая цена, по которой покупатель определенно не купит продукт (вероятность покупки 0%).
Затем уровень GP был рассчитан для различных продуктов на основе порогового значения полезности продуктов. Это было рассчитано с учетом различных функций SAU и безразличия WTP, на которое ответили респонденты.

В ходе исследования респондентам рассказали о цифровых фотоаппаратах. Им приходилось выполнять разные задания. Сначала им объяснили, какие атрибуты добавляются к продукту, а затем их попросили

добавить веса к различным атрибутам в зависимости от их предпочтений. Это было решено в рекомендациях, которые были показаны респондентам в следующем задании. Респондентов спрашивали, сколько они готовы платить за рекомендованную камеру (WTP - лучшая рекомендация). Последняя задача, которую они выполнили, заключалась в том, чтобы также включить своего GP в десятку лучших продуктов.

Результаты показывают, что взаимосвязь между атрибутами и полезностью воспринимается как экспоненциальная функция. Это указывает на то, что клиенты думают о компромиссах при оценке атрибутов. Например, размер атрибута - это компромисс между хрупкостью полезности и простотой транспортировки. Большая разница с WTP заключается в том, что он лучше измеряется линейной функцией. Но оба метода лечения занижали реальную стоимость. Интересно, что, согласно объявленному государственному предприятию, выручка продолжала расти.

Это важный вывод, сделанный из данного исследования. Он показывает, что системы рекомендаций могут генерировать персонализированные предложения для клиентов в зависимости от их готовности платить за продукт, который лучше всего соответствует их общей стоимости. Рекомендательные системы способны собирать информацию о предпочтениях клиентов в большом масштабе, не требуя при этом больших усилий со стороны клиентов. Компании могут и должны использовать эту информацию для получения большего дохода. Они могут делать это с помощью специальных предложений для клиентов, основанных на их предпочтениях, или продавая свою информацию производителям продуктов, чтобы они могли разрабатывать продукты на основе предпочтений клиентов.

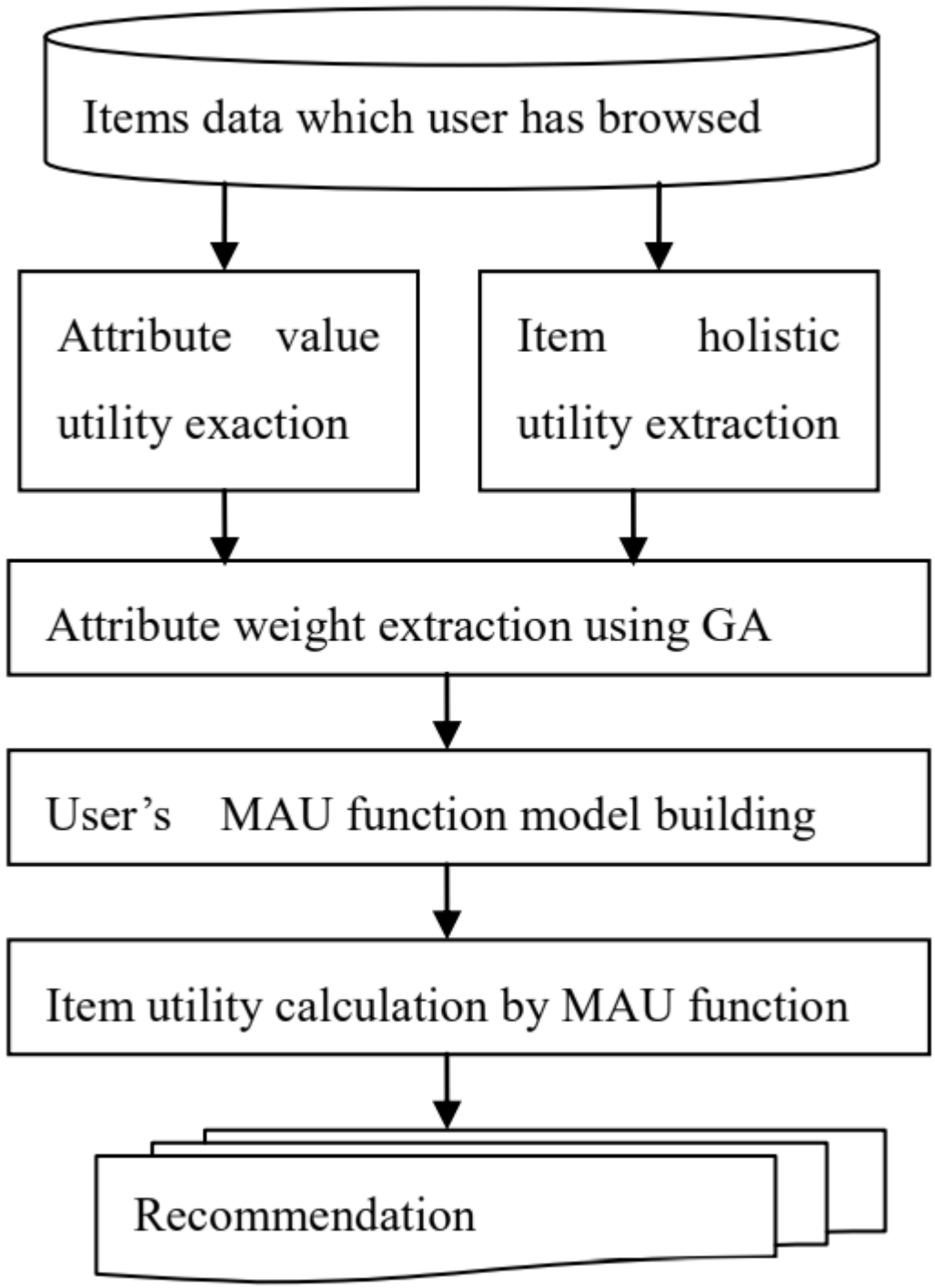


Рисунок40 – Таблица рекомендательной системы на основе утилит

3.9.3 KNOWLEDGE BASED СИСТЕМА РЕКОМЕНДАЦИЙ:

Этот тип рекомендательной системы пытается предложить объекты на основе предположений о потребностях и предпочтениях пользователя.

Рекомендации, основанные на знаниях, работают на функциональных знаниях: они знают, как конкретный элемент отвечает потребностям конкретного пользователя, и поэтому могут рассуждать о взаимосвязи между потребностью и возможной рекомендацией.

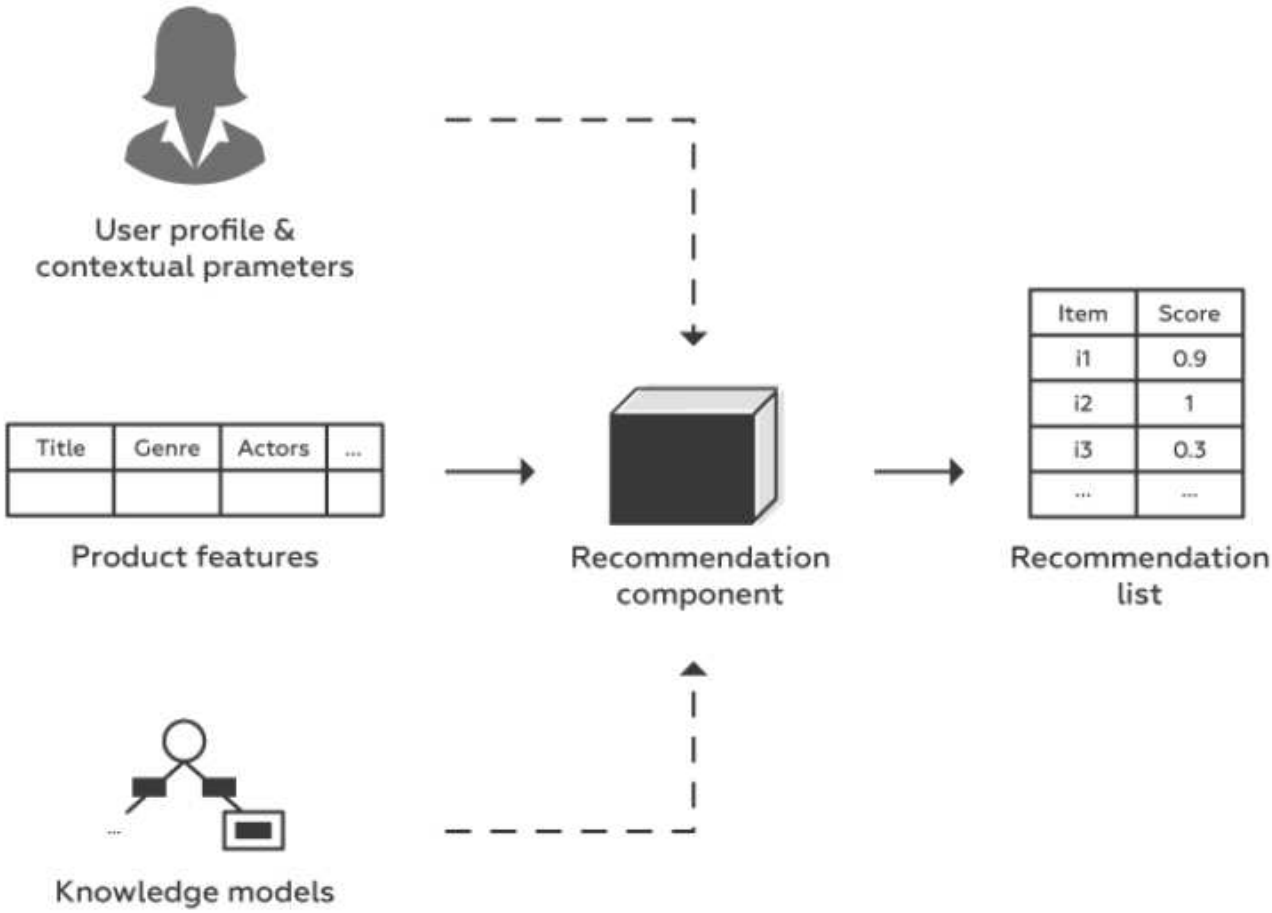


Рисунок41 – Модель рекомендаций, основанная на знаниях

3.9.4 Collaborativefiltering

Алгоритм совместной фильтрации (CF) используется для прогнозирования предпочтений пользователя при выборе на основе известных им рейтингов элементов. Как один из наиболее ценных алгоритмов, используемых в рекомендательных системах, CF доказал свою эффективность при решении проблем информационной перегрузки. Типичные вычисления подобия в большинстве случаев неэффективны, что страдает из-за разреженности данных

и проблем с низким качеством прогнозирования во время обучения, что приводит к неадекватному обучению.

Для решения этой проблемы предлагается рекомендательный алгоритм совместной фильтрации. Традиционные алгоритмы ориентированы только на оценки пользователей и не учитывают изменения интереса пользователей и достоверности рейтинговых данных, которые серьезно повлияли на качество рекомендаций системы. В этой статье мы более подробно рассмотрим различные типы рекомендательных систем CF и соответствующие методы. Ключевые слова - рекомендательные системы; совместная фильтрация; на основе памяти; модельно-ориентированный; мера сходства.

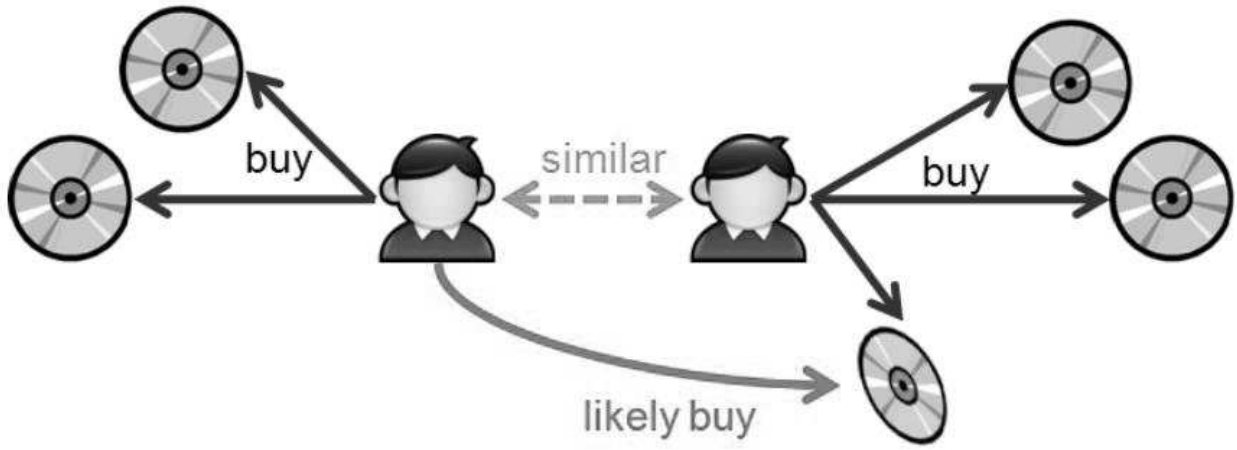


Рисунок42 – Пример совместной фильтрации

В современном мире пользователи сталкиваются с проблемой выбора подходящих товаров или услуг в огромном потоке информации. Разнообразные системы рекомендаций помогают пользователям выбрать правильный продукт или услугу. С момента появления CollaborativeFiltering (CF) в приложениях электронной коммерции для генерации персонализированных рекомендаций для пользователей алгоритм постоянно развивается и эффективно используется во многих практических областях.

Он основан на предположении, что похожие пользователи имеют схожие предпочтения, поэтому цель модели комментатора - изучить функцию, которая предсказывает полезность сходства для каждого пользователя. Матрица полезности обычно очень разреженная, огромная и содержит отдаленные значения.

В то время как исследования поднимают все больше и больше вопросов о некоторых фундаментальных проблемах систем совместной фильтрации, таких как разреженность данных, холодный старт, масштабируемость и т. Д., Все больше исследований направлено на поиск решений этих проблем.

В широком смысле совместную фильтрацию (CF) можно разделить на две основные ветви: основанную на памяти и основанную на модели.

Теперь исследователи оценили подход на основе модели с высокой точностью предсказания, что привело к получению более точных рекомендаций, поэтому он более эффективен, чем подход на основе памяти, который измеряет сходство между пользователями элемента.
Подходы CF обычно предназначены для доставки продуктов потенциальным клиентам, поэтому точность этого алгоритма имеет решающее значение. Следовательно, существует потребность в эффективных и точных методах рекомендаций, которые предоставят пользователям актуальные и надежные рекомендации. Эта статья направлена на изучение различных типов CF и связанных с ними методов.

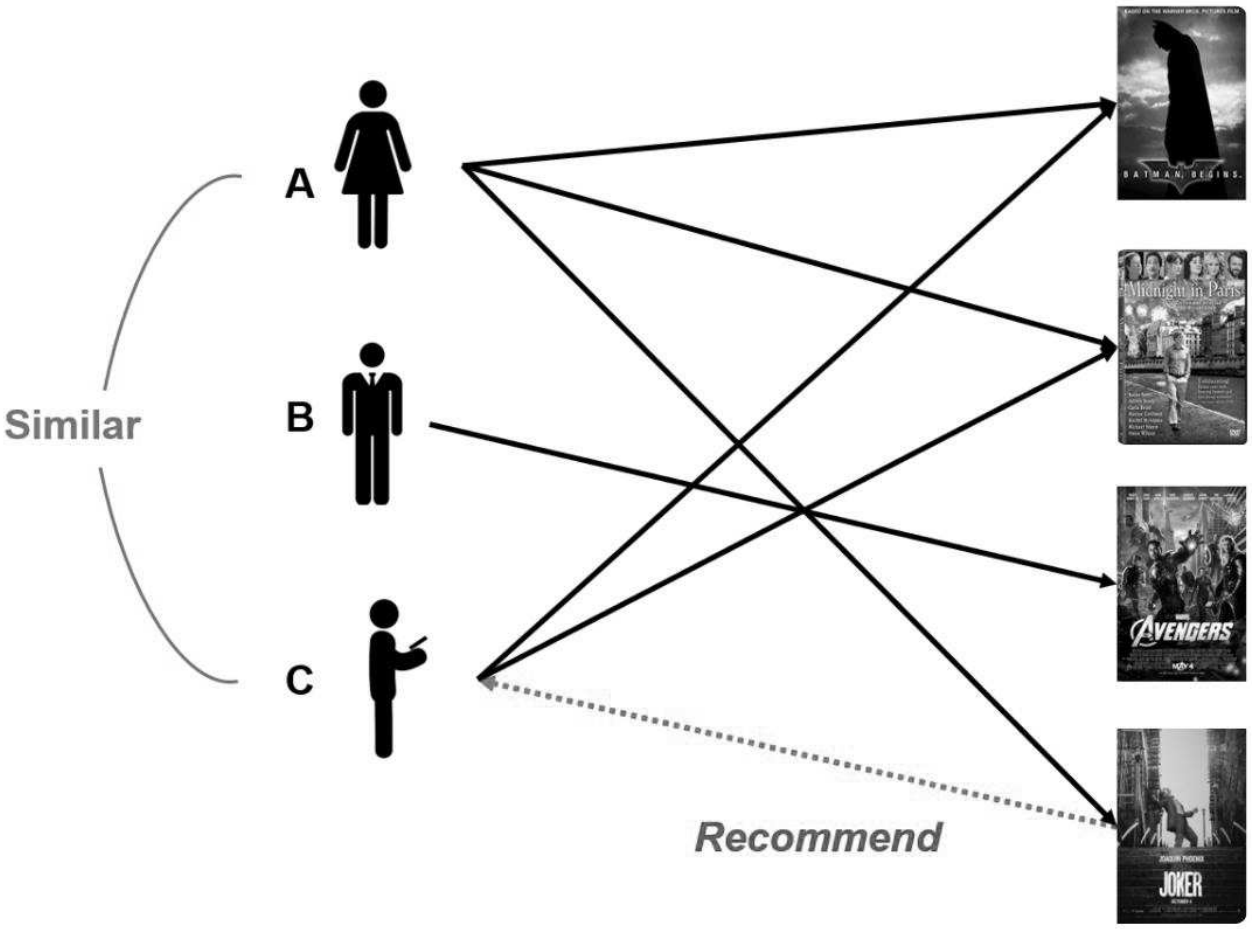


Рисунок43 – поток совместной фильтрации

Совместная фильтрация - это подход к автоматическому прогнозированию (фильтрации) интересов пользователя путем сбора данных об интересах многих связанных пользователей и этот алгоритм обрабатывается в матрице оценок пользователей.

Матрица «пользователь-элемент» обычно описывается как матрица оценок Rmnm × n, где строка представляет m пользователей, а столбец представляет количество пользователей.

Элемент matrixrij обозначает рейтинг, присвоенный пользователю, поэтому матрица "пользователь-элемент" представляет собой матрицу клиентов по сравнению с продуктами, компоненты которых представляют собой явные оценки клиентов по продуктам (от пользователя к элементу).

Исходя из первоначальных предположений, что элементы, которые нравились другим пользователям с аналогичными предпочтениями в прошлом, рекомендуются целевому пользователю, таким образом CF предположил, что те пользователи, которые соглашались в прошлом, как правило, соглашаются и в будущем.

Каждая методология CF имеет возможность использовать прошлые оценки пользователей, чтобы рекомендовать новый контент, который понравится отдельному пользователю.

Настоящее предположение хорошо основано на концепции сходства между пользователями или между продуктами, при этом сходство выражается как функция согласия между прошлыми рейтингами или предпочтениями. Два основных варианта подхода CF можно разделить на пользовательский и элементный. Кроме того, рекомендации по совместной фильтрации можно разделить на две группы: на основе памяти и на основе модели.

Методы, основанные на памяти, обычно являются эвристическими, поэтому прогнозы оценки зависят от всего набора элементов, ранее оцененных пользователями. Эти методы требуют, чтобы все рейтинги, элементы и пользователи сохранялись в памяти. Методы, основанные на моделях, используют групповой выбор рейтингов для изучения модели, которая затем используется для создания рейтинговых прогнозов.

Эти методы регулярно переводят эффективные рейтинговые шаблоны в автономный режим благодаря взаимодействию онлайн-пользователя с системой совместных рекомендаций через некоторый веб-интерфейс.

теперь мы применяем подходы коллаборативной фильтрации

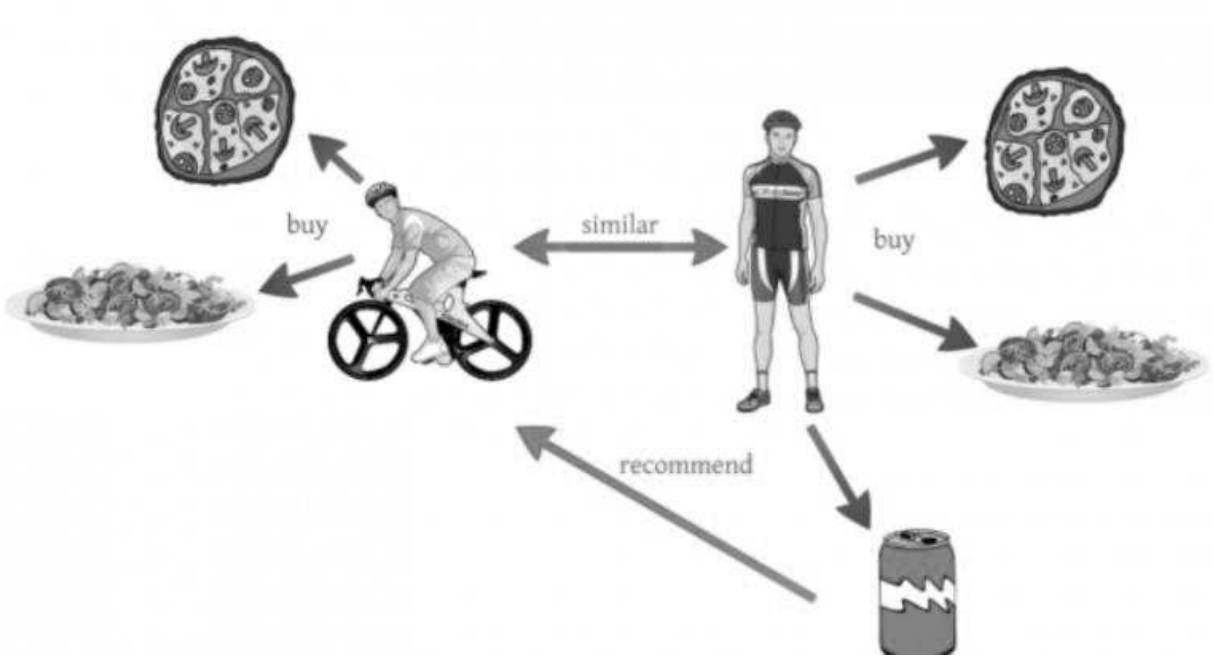


Рисунок44 – Пример совместной фильтрации в реальном мире

ModelBasedapproach: Алгоритмы на основе памяти, также известные как алгоритмы на основе соседей, работают со всей базой данных оценок, собранных поставщиком, и широко используются на многих крупных коммерческих сайтах, таких как Amazon и т. Д. Подходы к совместной фильтрации на основе памяти можно разделить на следующие: Далее: есть два основных раздела: фильтрация настраиваемых элементов и фильтрация по элементам.

Фильтрация по элементу пользователя берет пользователя, находит пользователей, похожих на них, на основе схожести оценок, и рекомендует элементы, которые нравятся похожим пользователям. Напротив, фильтрация элементов элементов будет брать элемент, находить пользователей, которым понравился элемент, и находить другие элементы, которые также понравились этим пользователям или аналогичным пользователям.

Он принимает элементы и выводит другие элементы в качестве рекомендаций.

Для вычисления сходства между двумя элементами используется несколько методов, однако в этой статье основное внимание уделяется только косинусному сходству и коэффициенту корреляции Пирсона.

1.Cosinesimilarity: Здесь два пользователя рассматриваются как два вектора в трехмерном пространстве пользователя. Сходство между ними измеряется путем вычисления косинуса угла между этими двумя векторами. Формально в матрице оценок m × n сходство между пользователями u и v, обозначенное sim (u, v).

Теперь давайте обсудим одну из наиболее часто используемых мер подобия - косинусное сходство. Если бы я дал вам векторы U = (3,4) и V = (1,1), вы могли бы легко сказать мне, насколько далеко друг от друга находятся эти два вектора (опять же, используя евклидово расстояние или другую метрику). Теперьважныйвопрос: насколькопохожиэтидвавектора?

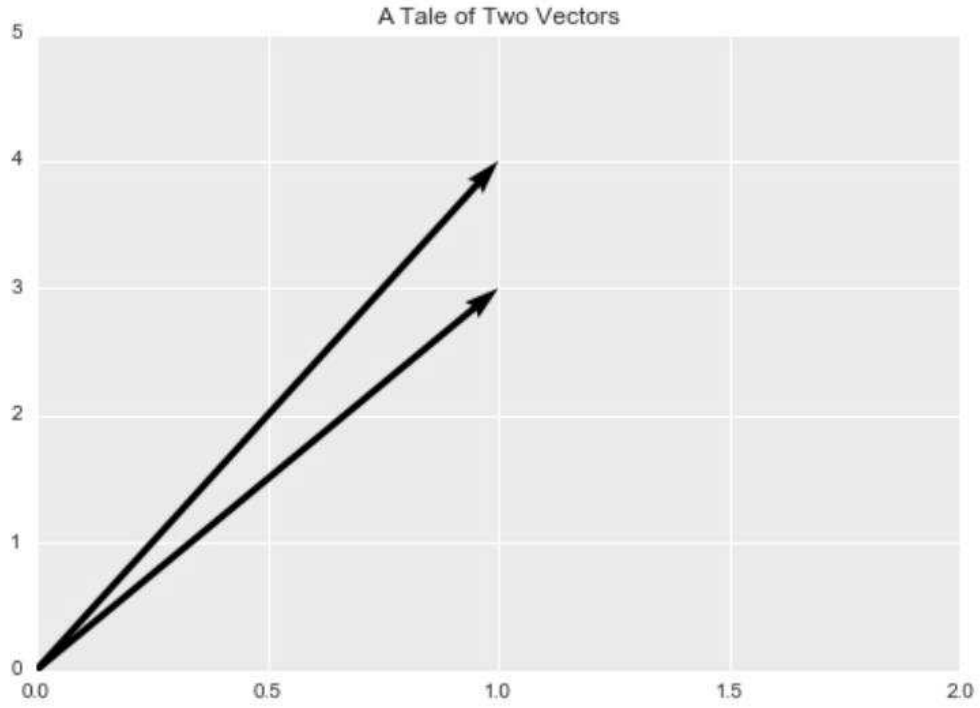


Рисунок 45 – График косинусного сходства

Опять же, схоже, нужна метрика «анти-расстояния», которая находится между 0 и 1. Если вы помните из тригонометрии, диапазон функции косинуса изменяется от -1 до 1. Следуетвспомнитьнекоторыеважныесвойствакосинуса:

1. Cosine (0 °) = 1
2. Cosine (90 °) = 0

3. Cosine (180 °) = -1

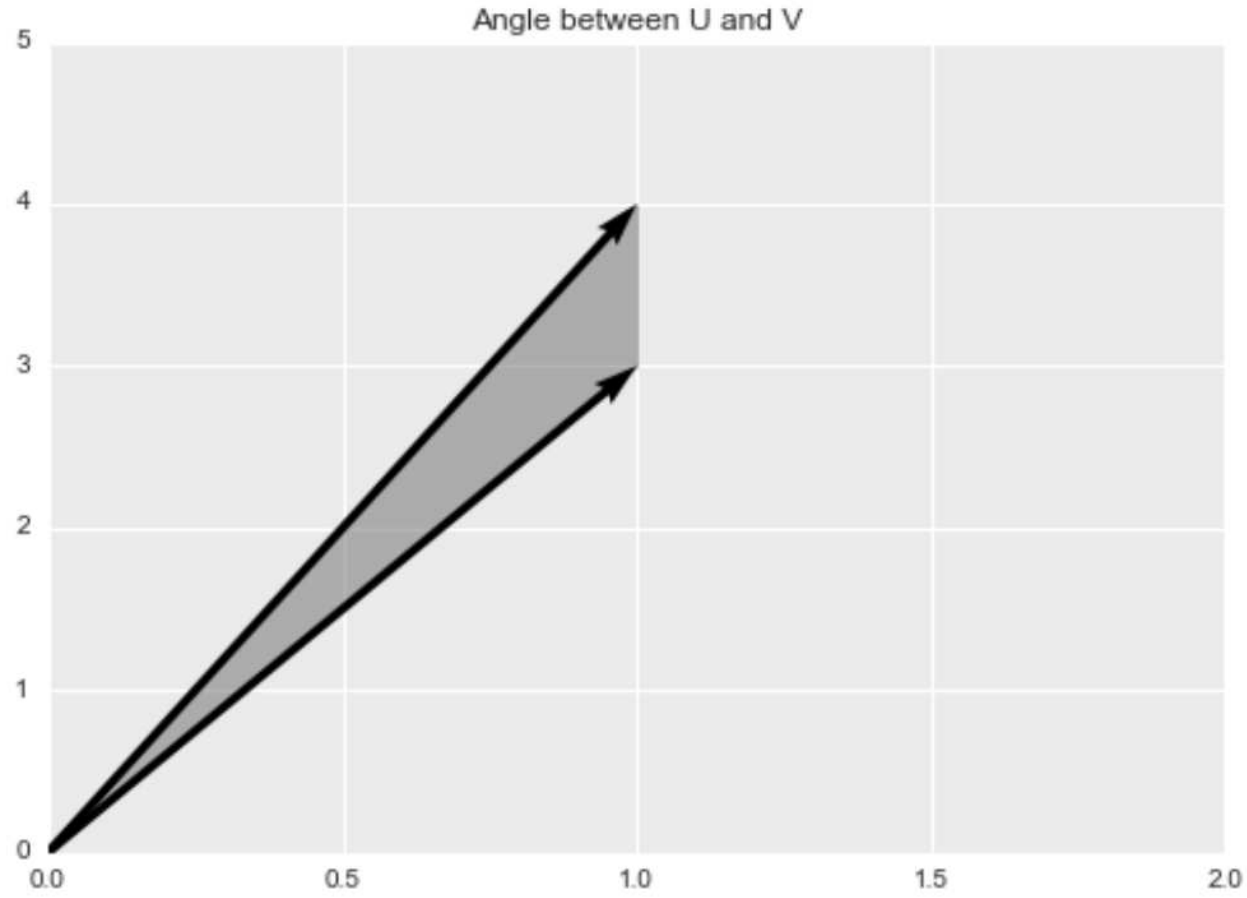


Рисунок46 – Косинусное подобие площадей V и U

Используя косинусное сходство, можно обнаружить, что два вектора достигают максимального сходства, когда угол между ними равен 0 ° (они ориентированы в одном направлении), они имеют нулевое сходство, когда угол между ними составляет 90 ° (они ортогональны друг другу.). , и имеют сходство -1, когда угол между ними равен 180 ° (они ориентированы в диаметрально противоположных направлениях). Ниже приведен пример

ортогональных векторов. Угол между ними составляет 90 °, поэтому косинусное подобие равно 0.

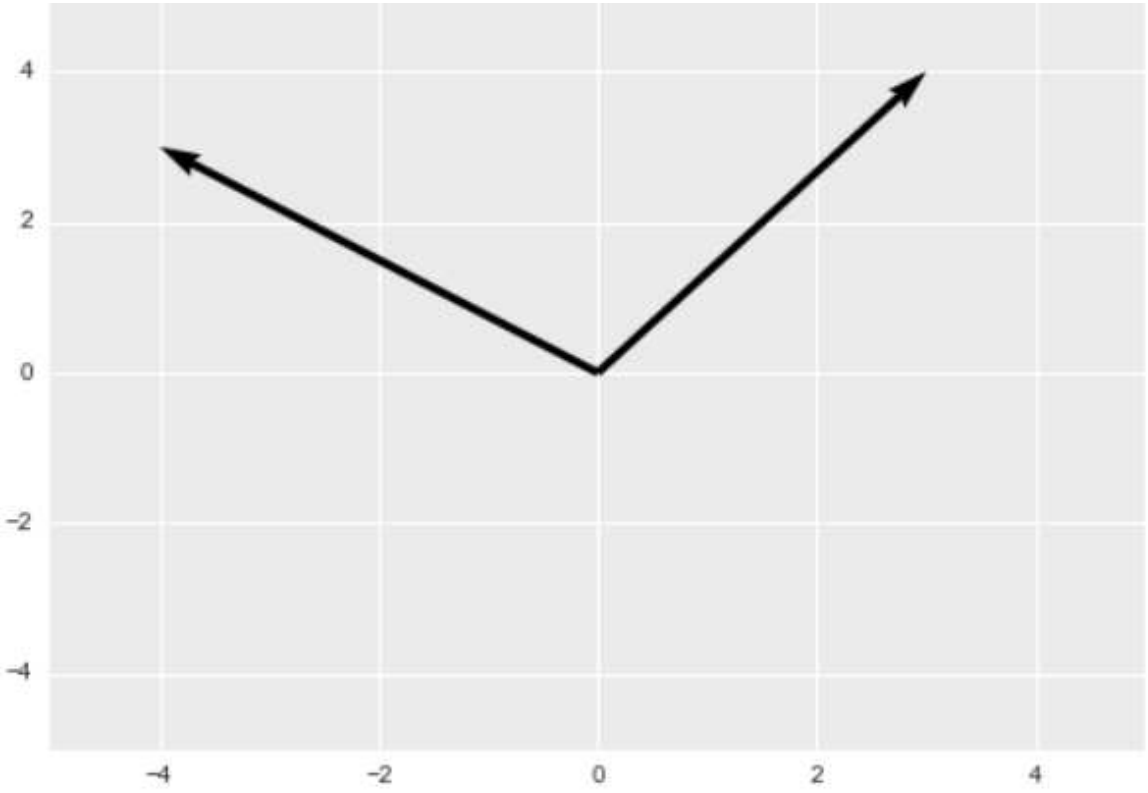


Рисунок 47 – Orthogonal

2. Correlationsimilarity: В этом случае сходство между двумя пользователями u и vis измеряется путем вычисления искажения корреляции Пирсона, v. Чтобы гарантировать точность вычисления корреляции, мы сначала выделяем случаи с одинаковой оценкой (т. Е. Случаи, когда элементы оцениваются u и v). Пусть набор элементов, которые оба имеют рейтинги byu и v, обозначен как Iuv, тогда корреляционное сходство дается как;

$$Sim\ (u,\ v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)) \,.(r_{vi} - \bar{r}_u)}{\sqrt{\sum_{i \in I_{uv}}^{n} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}}^{n} (r_{vi} - \bar{r}_v)^2}}$$

Где sim (u, v) представляет собой сходство между пользователем u и пользователем v, Iuv (Iuv = I (u) ∩I (v)) представляет элемент, установленный одновременно пользователем u, а пользователь v, rui и rvi являются оценками элемента irated. пользователем uи v соответственно. ¯ru и ¯rv представляют собой средние баллы пользователей u и v для оцениваемых элементов соответственно.

В системах совместной фильтрации создание выходного интерфейса с точки зрения прогнозирования является наиболее важным шагом (Sarwaretal., 2001). После того, как мы отделим набор ближайших соседей на основе показателей сходства, следующим шагом будет изучение рейтингов целевых пользователей, поэтому мы используем эту технику для получения прогнозов. Здесь прогноз для элемента ifor пользователя u обозначается Pui, который задается;

$$Puser-based(ui)=\bar{}ru+Pv\in Nisim(u,v))\cdot(rvi-\bar{}rv)Pv\in Ni|sim(u,v)|$$

Здесь ¯rv представляет собой средний балл соседа v для его оцениваемых элементов, а sim (u, v) означает сходство между пользователем u и его соседом v. Ni указывает на набор N самых близких соседей целевого пользователя u.

Другие меры сходства включают коэффициент корреляции Спирмена, скорректированное косинусное сходство , Среднеквадратичная разница (MSD), коэффициент Жаккара и подобие на основе условной вероятности. Среди этих мер сходства многие эксперименты показали, что коэффициент корреляции Пирсона может лучше, чем другие методы, отражать сходство пользователей или элементов.

MemoryBased:

1)PearsonCorrelation: Коэффициент корреляции Пирсона (PCC) для поиска корреляций между пользователями в подходе, который стал популярным в CF на основе памяти. Однако метод PCC может быть неточным, когда данные

немногочисленны, поскольку пропущенные оценки затрудняют поиск корреляций между пользователями. Это приводит к высокому / низкому сходству и, следовательно.

Отношения между пользователями можно определить как:

$$S\,(x,\,y)^{PCC} = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x))\;\;(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}},$$

где S (x, y) PCC- это сходство между пользователями x и y, Ixy представляет собой набор элементов, которые имеют рейтинг как x, так и y, и обозначают средние оценки пользователей x и y, соответственно, a rx, i обозначает значение рейтинга, присвоенное элементу i пользователем x.

2)Ограниченная корреляция Пирсона: Рекомендация RINGO была разработана для предоставления пользователям рекомендаций по музыкальным альбомам и исполнителям. В рамках RINGO пользователи предоставляют отзывы по номинальной шкале от одного («сильная неприязнь») до семи («сильная симпатия») с нейтральным значением («ни нравится, ни не нравится») в середине шкалы. Основываясь на увеличивающемся числе пользователей RINGO, Шраддхананд и Мэй [5] предложили подход с ограниченной корреляцией Пирсона (CPCC) для замены средних рейтинговых переменных, используемых в подходах PCC, на медианное значение шкалы положительных и отрицательных оценок. Корреляция рассчитывается как:

$$S\,(x,\,y)^{CPCC} = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_m))\;\;(r_{y,i} - \bar{r}_m)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_m)^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_m)^2}},$$

3)CosineMethod:Метод косинуса - это модель векторного пространства, которая применяет подход линейной алгебры для определения отношений между парами пользователей [6] как векторов, при этом сходства пользователей вычисляются как косинусное расстояние между каждой парой векторов рейтинга. Эта корреляция определяется как:

$$S\ (x,\ y)^{Cosine} = \frac{\sum_{i \in I_{xy}} (r_{x,i})\ \ (r_{y,i})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i})^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i})^2}},$$

4)New heuristic similarity measure (proximity–significance–singularity)

Проанализировал недостатки подходов «Принципы в шаблонах» и предложил улучшенную версию, названную новой эвристической мерой сходства (NHSM). Модель NHSM учитывает три фактора рейтинга пользователей - близость, значимость и особенность (PSS) - и объединяет местную контекстную информацию об этих рейтингах с глобальными предпочтениями оценок пользователей, чтобы решить проблему холодного старта . Тем не менее, NHSM рассматривает только элементы с одинаковым рейтингом при определении отношений между пользователями. Мера определяется как:

$$\mathbf{S\ (x,\ y)^{PSS}} = \sum_{i \in I} \boldsymbol{PSS}\left(r_{x,i}, r_{y,i}\right),$$

где PSS (rx, i, ry, i) - это значение PSS пользователей x и y, которое рассчитывается как:

$$\boldsymbol{PSS}\left(r_{x,i}, r_{y,i}\right) = \boldsymbol{Proximity}\left(r_{x,i}, r_{y,i}\right) * \boldsymbol{Significance}\left(r_{x,i}, r_{y,i}\right) * \boldsymbol{Singularity}\left(r_{x,i}, r_{y,i}\right).$$

Индивидуальные аспекты представлены:

$$\boldsymbol{Proximity}\left(r_{x,i}, r_{y,i}\right) = 1 - \frac{1}{1+exp(-|r_{x,i}-r_{y,i}|)},$$

$$\boldsymbol{Significance}\left(r_{x,i}, r_{y,i}\right) = \frac{1}{1+exp(-|r_{x,i}-r_{med}| * |r_{y,i}-r_{med}|)},$$

$$\boldsymbol{Singularity}\left(r_{x,i}, r_{y,i}\right) = 1 - \frac{1}{1+exp\left(-\left|\frac{r_{x,i}-r_{y,i}}{2}-\mu_i\right|\right)},$$

сочетание показателей PSS с показателем Жаккара для решения проблемы небольшой доли общих рейтингов. Эта так называемая мера JPSS определяется как

$$\mathbf{S\ (x,\ y)^{JPSS} = S\ (x,\ y)^{Jacc} * S\ (x,\ y)^{PSS}},$$

Чтобы учесть случаи, возникающие, когда разные пользователи предоставляют разные рейтинговые предпочтения (например, высокие оценки одни и низкие оценки другими), они также разработали меру предпочтений пользователей, основанную на средней оценке и стандартной дисперсии:

$$\boldsymbol{S(x,y)^{URP}} = 1 - \frac{1}{1+exp(-|\mu_x-\mu_y| * |\alpha_x-\alpha_y|)},$$

5)Multi-attributedecision-makingmethod:Как упоминалось в предыдущем разделе, большинство исследований по повышению точности CF были сосредоточены на улучшении меры сходства, даже несмотря на то, что модель прогнозируемой оценки имеет аналогичное значение.

В CF на основе памяти после определения местоположения соседей целевого пользователя система собирает их элементы и прогнозирует рейтинговые оценки, которые целевой пользователь применит к ним; затем элементы

ранжируются и рекомендуются в соответствии с этими прогнозируемыми баллами. Ясно, что алгоритм прогнозирования играет важную роль в этом процессе.

В качестве замены прогнозу мы предлагаем использовать TOPSIS в качестве полезного метода MADM для оценки и ранжирования элементов.

Многочисленные альтернативы, с которыми люди сталкиваются в Интернете, могут затруднить процесс принятия решений, особенно когда их просят оценить или выбрать лучшую альтернативу из набора доступных элементов. Как правило, для оценки наборов альтернатив используются несколько критериев.

Например, основными критериями при покупке автомобиля являются стоимость, безопасность, комфорт и расход топлива. Принятие решений по нескольким критериям (MCDM), один из наиболее известных подходов к выбору альтернатив, может применяться, когда необходимо учитывать предпочтения лица, принимающего решения.

В литературе проблемы MCDM делятся на два основных подхода: многоцелевое принятие решений (MODM) и принятие решений с несколькими атрибутами (MADM).

Проблемы MADM отличаются от проблем MODM количеством заранее определенных альтернативных решений. В MADM проблемы принятия решений подвергаются ряду критериев решения, чтобы произвести ранжирование множества альтернатив в соответствии с их атрибутами.

Это в первую очередь включает сбор информации и ее оценку в сравнении с дополнительной информацией, предоставленной лицом, принимающим решения, в результате чего получается матрица решений, которая используется для определения окончательного ранжирования альтернатив.

Описание нескольких методов MADM, включая TOPSIS. Первоначально представленный Юном и Хванем, TOPSIS представляет собой практический

метод ранжирования и выбора нескольких определяемых извне альтернатив с использованием мер расстояния.

Основными преимуществами TOPSIS являются его способность быстро определять лучшую альтернативу и сопоставимые или превосходящие по производительности процессы простого аддитивного взвешивания и аналитической иерархии соответственно. От лиц, принимающих решения, требуется ограниченное количество простых входных данных (т. Е. Весов, связанных с соответствующими критериями), а выходные данные процесса легко понять.

Основополагающий принцип TOPSIS заключается в том, что лучшая альтернатива — это то, что ближе всего к идеальному решению и дальше всего от отрицательного идеального решения.

Методика TOPSIS реализуется в несколько вычислительных шагов, которые описаны ниже:

- определить варианты решения;
- определить критерии (атрибуты), относящиеся к проблеме решения;
- построить матрицу решений, содержащую m альтернатив, связанных с n атрибутами (или критериями);
- нормализовать исходные оценки для построения матрицы оценок приоритета или нормализованного решения. Баллы в нормализованной матрице должны быть преобразованы в нормализованную шкалу;
- построить взвешенную нормализованную матрицу решений, в которой каждому атрибуту присваивается определенный вес, чтобы отразить, насколько он важен для общего решения;
- определить идеальные и отрицательно-идеальные решения;
- вычислить меру разделения как n-мерное евклидово расстояние между альтернативами;

- рассчитать относительную близость каждой альтернативы к идеальному решению;

- создать рейтинг альтернатив, основанный на максимизации показателей относительной близости на предыдущем шаге.

Эти шаги будут объяснены более подробно в следующем разделе.

6) Proposed memory-based CF using TOPSIS:

Предлагаемый метод предполагает применение TOPSIS для рекомендации наборов элементов, которые могут быть интересны пользователю.

Это реализуется в несколько основных этапов, как показано в архитектуре.

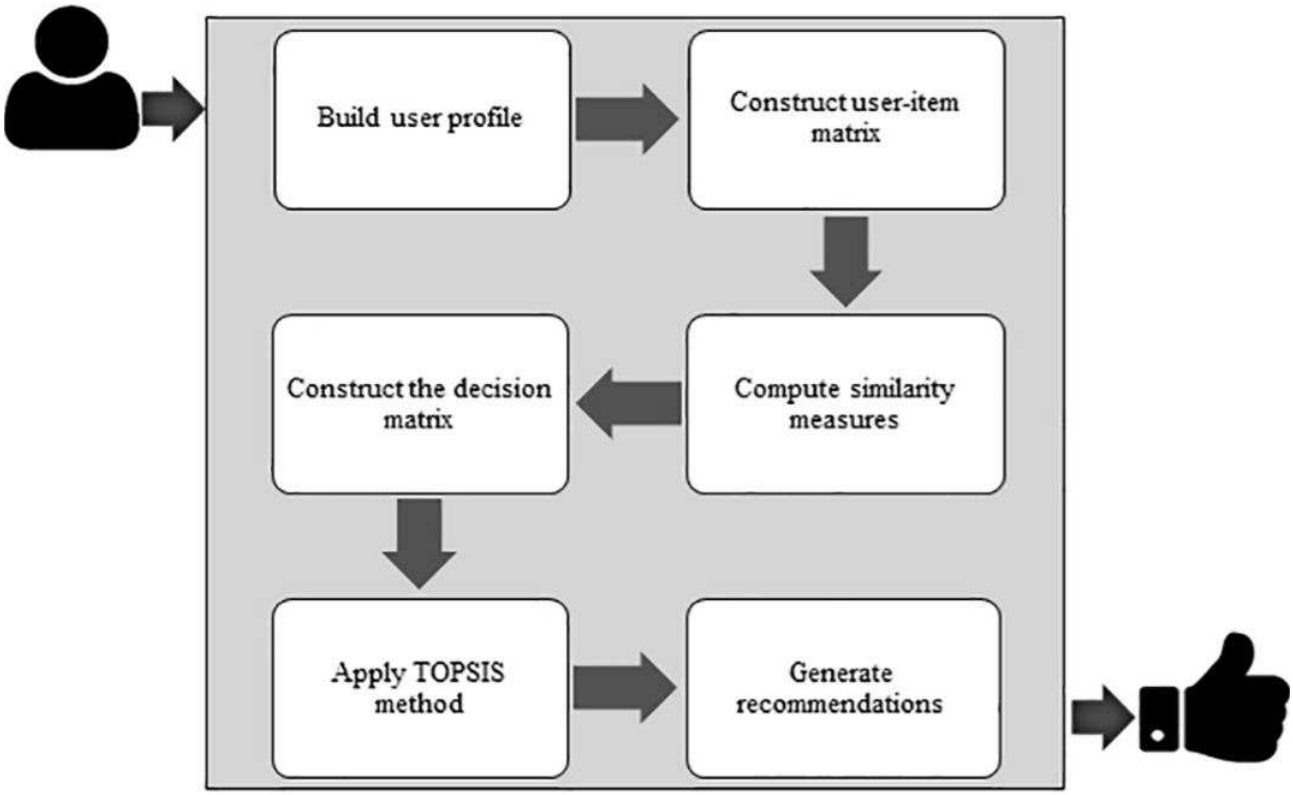


Рисунок48 – Архитектура предлагаемого метода КФ на основе памяти.

Фазы предлагаемого метода резюмируются следующим образом:

i. Создание профиля пользователя: система собирает отзывы от целевого пользователя для создания его / ее профиля предпочтений. Такие

предпочтения обычно связаны со шкалой значений, представляющей степень предпочтения пользователя для элемента, например, от одной до пяти звезд или от одного до десяти баллов. Таким образом, пользователь x оценивает фильм a с оценкой «пять» и фильм b с оценкой «три», поэтому может быть замечено, что он предпочитает a над b.

ii. Построение матрицы пользователь-элемент: данные, относящиеся к пользователям и элементам в системе, вводятся в матрицу элементов-пользователей в виде набора числовых оценок.

iii. Вычислить меры сходства: сходства между пользователями рассчитываются с использованием нескольких общих методов определения базовых показателей CF (например, PCC, CPCC, SPCC, Cos, MSD, JMSD, NHSM). После этого топ-K пользователей с наиболее сильной корреляцией с точки зрения сходства с целевым пользователем используются для формирования его / ее окружения.

iv. Постройте матрицу решений: атрибуты K-ближайших пользователей собираются и используются для заполнения матрицы альтернатив, включающей элементы, которые были оценены этими пользователями, но еще не выбраны целевым пользователем. Пунктам, которые не были оценены, присваиваются значения на основе голосования по умолчанию.

v. Примените метод TOPSIS: затем метод TOPSIS применяется для оценки и ранжирования всех альтернативных элементов. Как обсуждается в следующем разделе, TOPSIS определяет лучшую альтернативу как альтернативу с наименьшим и наибольшим расстоянием от идеального и отрицательно-идеального решений соответственно. TOPSIS позволяет быстро определить лучшую альтернативу, прост в реализации, требует лишь ограниченного количества входных данных от лиц, принимающих решения, и дает легко понятные выходные данные. Единственными входными параметрами являются значения веса, связанные с критериями. Этот основной этап процесса будет подробно объяснен в следующем подразделе.

vi. Создание рекомендаций: как описано выше, вывод, производимый TOPSIS, представляет собой список отсортированных альтернатив (элементов-кандидатов), ранжированных в соответствии с измерением важности на основе нескольких критериев (K-соседей). На заключительном этапе процесса рекомендации элементы Top-M выбираются и представляются целевому пользователю в виде набора предложений элементов.

В предлагаемом методе метод TOPSIS используется вместо рейтингов прогнозов для оценки и ранжирования элементов-кандидатов и создания отсортированного списка рекомендаций по элементам с точки зрения их прогнозируемых предпочтений.

Важным входом в эту процедуру является список K-соседей и их элементы, рейтинги и веса сходства по отношению к целевому пользователю. TOPSIS преобразует эту задачу выбора и ранжирования в матрицу решений X с m альтернативами (строками) и n критериями (столбцами), соответствующими элементам-кандидатам и K-соседям, соответственно.

В X каждая запись xi, j представляет численный результат j-й альтернативы по i-му критерию, то есть значение рейтинга, примененное пользователем i к элементу j.

Чтобы избежать деления на ноль во время выполнения, недостающие оценки представлены средним значением для каждого пользователя. Поскольку нельзя предположить, что критерии имеют одинаковую важность, набор весовых параметров, предоставляемых лицом, принимающим решение, связан с критериями. Затем эти веса сравниваются с весами соседей, принимающих решение, для получения набора K-соседей.

Прежде чем подробно изучить функционирование TOPSIS, мы определим наборы, используемые в анализе:

i. A - это набор элементов-кандидатов, представляющих альтернативы A = {a1, a2,…, aj,…, am-1, am}, где j = 1,2,…, m и m - общее количество элементов-кандидатов.

ii. C - это множество соседей, представляющих различные критерии C = {c1, c2,…, ci,…, cn − 1, cn}, где i = 1,2,…, n и n обозначает количество критериев (K- соседи).

iii. X - множество рейтингов X = {xj, i | j = 1,…, m; i = 1,…, n}, где xj, i - значение рейтинга j-й альтернативы / элемента по i-му критерию / соседнему пользователю.

iv. W - это набор весов W = {w1, w2,…, wi,…, wn − 1, wn | i = 1,2,…, n}, где wi - вес i-го критерия / соседа (т. Е. значение сходства между i-м соседом и целевым пользователем).

Матрица решений X, содержащая m альтернатив, связанных с n критериями, представлена в Фото.

			Neighbors						
X =			c_1	c_2	⋯	c_i	⋯	c_{n-1}	c_n
	Candidate Items	a_1	$x_{1,1}$	$x_{1,2}$	⋯	$x_{1,i}$	⋯	$x_{1,n-1}$	$x_{1,n}$
		a_2	$x_{2,1}$	$x_{2,2}$	⋯	$x_{2,i}$	⋯	$x_{2,n-1}$	$x_{2,n}$
		⋮	⋮	⋮	⋱	⋮	⋱	⋮	⋮
		a_j	$x_{j,1}$	$x_{j,2}$	⋯	$x_{j,i}$	⋯	$x_{j,n-1}$	$x_{j,n}$
		⋮	⋮	⋮	⋱	⋮	⋱	⋮	⋮
		a_{m-1}	$x_{m-1,1}$	$x_{m-1,2}$	⋯	$x_{m-1,i}$	⋯	$x_{m-1,n-1}$	$x_{n-1,n}$
		a_m	$x_{m,1}$	$x_{m,2}$	⋯	$x_{m,i}$	$x_{m,1}$	$x_{m,n-1}$	$x_{n,n}$

Рисунок49– Матрица концептуальных решений X.

Шаги в методе TOPSIS описаны ниже.

Шаг 1. Постройте нормализованную матрицу решений.

Некоторые пользователи предпочитают ставить высокие оценки даже тем элементам, которые им не очень нравятся, тогда как другие ставят низкие оценки тем элементам, которые им нравятся. Чтобы учесть и скорректировать такие рейтинговые диспропорции и нарушения, необходимо нормализовать матрицу решений. Это можно сделать с помощью распределительной нормализации, при которой значения рейтинга в каждом столбце делятся на

квадратный корень из суммы каждой возведенной в квадрат альтернативы в столбце. Таким образом, элементы rj, i нормализованной решающей матрицы R задаются следующим образом:

$$r_{j,I} = \frac{x_{j,i}}{\sqrt{\sum_{j=1}^{m} {x_{j,i}}^2}}, \; j = 1 \ldots m; \; i=1 \ldots n,$$

R =			Neighbors						
			c_1	C_2	$\cdots$	c_i	$\cdots$	c_{n-1}	c_n
	Candidate Items	a_1	$r_{1,1}$	$r_{1,2}$	$\cdots$	$r_{1,i}$	$\cdots$	$r_{1,n-1}$	$r_{1,n}$
		a_2	$r_{2,1}$	$r_{2,2}$	$\cdots$	$r_{2,i}$	$\cdots$	$r_{2,n-1}$	$r_{2,n}$
		$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
		a_j	$r_{j,1}$	$r_{j,2}$	$\cdots$	$r_{j,i}$	$\cdots$	$r_{j,n-1}$	$r_{j,n}$
		$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
		a_{m-1}	$r_{m-1,1}$	$r_{m-1,2}$	$\cdots$	$r_{m-1,i}$	$\cdots$	$r_{m-1,n-1}$	$r_{n-1,n}$
		a_m	$r_{m.1}$	$r_{m.2}$	$\cdots$	$r_{m.i}$	$r_{m.1}$	$r_{m.n-1}$	$r_{n.n}$

Рисунок50 – Концептуальная нормализованная матрица решений R.

Шаг 2: Постройте взвешенную нормализованную матрицу решений

Чтобы принять во внимание веса W, предоставленные лицом, принимающим решение, взвешенная нормализованная матрица решений V задается путем умножения нормированных значений rj, i на их соответствующие веса wi. В предлагаемом методе веса подобия целевого пользователя по отношению к его / ее соседям используются для разработки весовых критериев пользователя. Например, для целевого пользователя u, у которого есть k соседей (с n критериями), веса подобия su = {su, 1, su, 2,…, su, i,…, su, n − 1, su, n | , i = 1,2,…, n}, где si, k обозначает значение подобия между u и i-м соседом, используются для заполнения набора весов wi. Затем взвешенная нормализованная решающая матрица V получается следующим образом:

$V_{j,I} = r_{j,I} * w_{i,j} = 1 \ldots m; \; i=1 \ldots n.$

V =			Neighbors						
			c_1	c_2	$\cdots$	c_i	$\cdots$	c_{n-1}	c_n
	Candidate Items	a_1	$v_{1,1}$	$v_{1,2}$	$\cdots$	$v_{1,i}$	$\cdots$	$v_{1,n-1}$	$v_{1,n}$
		a_2	$v_{2,1}$	$v_{2,2}$	$\cdots$	$v_{2,i}$	$\cdots$	$v_{2,n-1}$	$v_{2,n}$
		$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
		a_j	$v_{j,1}$	$v_{j,2}$	$\cdots$	$v_{j,i}$	$\cdots$	$v_{j,n-1}$	$v_{j,n}$
		$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
		a_{m-1}	$v_{m-1,1}$	$v_{m-1,2}$	$\cdots$	$v_{m-1,i}$	$\cdots$	$v_{m-1,n-1}$	$v_{n-1,n}$
		a_m	$v_{m,1}$	$v_{m,2}$	$\cdots$	$v_{m,i}$	$v_{m,1}$	$v_{m,n-1}$	$v_{n,n}$

Рисунок51 – Концептуальная нормализованная матрица решений R.

Шаг 3. Определите положительные и отрицательные идеальные решения

Затем определяются лучшие и худшие варианты оценки для каждого критерия в нормализованной матрице решений V и используются для представления идеального и отрицательно-идеального решений соответственно.

Для набора положительных атрибутов или критериев I1, связанных с выгодой (больше - лучше), и набора отрицательных атрибутов или критериев I2, связанных с затратами (меньше - лучше), положительные и отрицательные идеальные решения могут быть определены следующим образом:

$A^* = \{v_1^*, \ldots, v_i^*, \ldots, v_n^*\}$, where $v_i^* = \{\max(v_{j,i}) \; if \; c_i \in I_1; \min(v_{j,i}) \; if \; c_i \in I_2\}$,

Шаг 4: Рассчитайте меру разделения

Расстояние от каждой альтернативы до идеальных и отрицательно-идеальных решений для всех альтернатив можно рассчитать с помощью измерения евклидова расстояния. Расстояние каждой альтернативы от идеала определяется следующим образом:

$S_j^* = \sqrt[2]{\sum_{i=1}^{n}(v_{j,i} - v_j^*)^2}$, j=1, 2, …,m.

Similarly, the distance of each alternative from the negative-ideal is given by:

$S_j^1 = \sqrt[2]{\sum_{i=1}^{n}(v_{j,i} - v_j^1)^2}$, j=1, 2, …, m.

V′ =			S^*	S'
	Candidate items	a_1	S_1^*	S_1'
		a_2	S_2^*	S_2'
		⋮	⋮	⋮
		a_j	S_j^*	S_j'
		⋮	⋮	⋮
		a_{m-1}	S_{m-1}^*	S_{m-1}'
		a_m	S_m^*	S_m'

Рисунок 14 – Conceptual separation matrix V′.

Шаг 5. Рассчитайте относительную близость к идеальному решению.

Степень близости каждой альтернативы идеальному решению A * рассчитывается как

$C_j^* = S_j^1 / (S_j^* + S_j^1)$, $0 < C_j^* < 1, j = 1, 2, \ldots, m$.

Рейтинг относительной близости колеблется от нуля до единицы; эти крайности представляют собой, соответственно, наименее и наиболее предпочтительные альтернативы. Чтобы уточнить, если расстояние альтернативы aj от идеального решения A * меньше, чем его расстояние от отрицательного идеала A ′, то C * j будет ближе к единице, чем к нулю, и наоборот, как показано

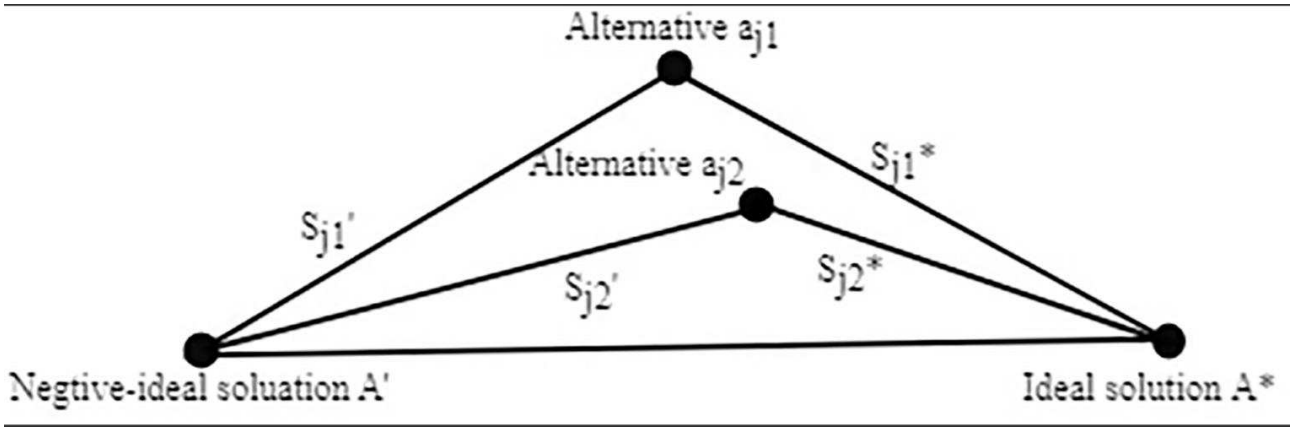


Рисунок 14 – Conceptual separation matrix V′.

7) SVD

Разложение по сингулярным значениям (SVD), метод линейной алгебры, который обычно используется в качестве метода уменьшения размерности в машинном обучении.

SVD - это метод матричной факторизации, который уменьшает количество функций набора данных за счет уменьшения размерности пространства с N-измерения до K-измерения (где K<N).

В контексте рекомендательной системы SVD используется как метод совместной фильтрации. Он использует матричную структуру, где каждая строка представляет пользователя, а каждый столбец представляет элемент. Элементами этой матрицы являются рейтинги, присваиваемые пользователям.

$$A= USV^T$$

Где A - матрица полезности.

U - ортогональная левая матрица сингулярностей, которая коэффициентов представляет собой взаимосвязь между пользователями и скрытыми факторами.

S - диагональная матрица, которая описывает силу каждого скрытого фактора

V - диагональная правая сингулярная матрица arxn, которая указывает сходство между предметами и скрытыми факторами. Скрытыми факторами здесь являются характеристики предметов, например, жанр музыки.
SVD уменьшает размерность матрицы полезности A, извлекая ее скрытые факторы. Он отображает каждого пользователя и каждый элемент в r-мерное скрытое пространство. Это сопоставление облегчает четкое представление отношений между пользователями и элементами.

Пусть каждый элемент представлен вектором xi, а каждый пользователь представлен вектором yu. Ожидаемая оценка пользователем элемента может быть выражена как:

$$\gamma_{ui} = x_i^T \cdot y_u$$

Вот форма факторизации в разложении по сингулярным числам. Xi и yu могут быть получены таким образом, чтобы разница квадратов ошибок между их скалярным произведением и ожидаемой оценкой в матрице пользовательских элементов была минимальной. Это можно выразить как:

$$\text{Min (x, y)} \sum_{(u,i)\in K}(r_{ui} - x_i^T \cdot y_u)^2$$

Чтобы модель хорошо обобщалась и не превышала обучающие данные, к приведенной выше формуле добавляется член регуляризации в качестве штрафа.

$$\text{Min (x, y)} \sum_{(u,i)\in K}(r_{ui} - x_i^T \cdot y_u)^2 + \gamma(\|x_i\|^2 + \|y_u\|^2)$$

Чтобы уменьшить ошибку между значением, предсказанным моделью, и фактическим значением, алгоритм использует член смещения. Пусть для пары пользователь-элемент (u, i) μ- это средняя оценка всех элементов, bi - средняя оценка элемента i минус μ, a bu - средняя оценка пользователя u минус μ,

окончательное уравнение после добавления член регуляризации и смещение можно представить как:

$$Min\ (\text{x, y, } b_i, b_u\ ) \sum_{(u,i)\in K}(r_{ui} - x_i^T . y_u - \mu - b_i - b_u)^2 + \gamma(\|x_i\|^2 + \|y_u\|^2 + b_i^2 + b_u^2)$$

Ниже представлена реализация разложения по сингулярным значениям (SVD) на основе совместной фильтрации в задаче рекомендации фильмов. Эта задача реализована на Python. Для простоты использовались книги 1MDataset. Этот набор данных был выбран, потому что он не требует предварительной обработки, поскольку основное внимание в этой статье уделяется SVD и рекомендательным системам.

	data	Name	Category	rate
0	917	billgates	TRAVEL	5
1	805	billgates	ENTERTAINMENT	4
2	387	billgates	SCIENCE	3
3	316	billgates	WOMEN	2
4	262	billgates	EDUCATION	1
5	1558	chrisrock	ENTERTAINMENT	5
6	167	chrisrock	TRAVEL	4
7	123	chrisrock	STYLE & BEAUTY	3
8	34	chrisrock	CRIME	2
9	29	chrisrock	WOMEN	1
10	732	debruynekev	ENTERTAINMENT	5
11	167	debruynekev	TRAVEL	4
12	78	debruynekev	STYLE & BEAUTY	3
13	53	debruynekev	SPORT	2
14	32	debruynekev	BUSINESS	1
15	5313	kapilsharmak9	ENTERTAINMENT	5
16	713	kapilsharmak9	TRAVEL	4
17	320	kapilsharmak9	STYLE & BEAUTY	3
18	252	kapilsharmak9	RELIGION	2
19	129	kapilsharmak9	WOMEN	1
20	1329	vp45	ENTERTAINMENT	5

Рисунок52– данные с рейтингом ′.
Листинг20 – «Матрица нашего фрейма данных»

```
R_df = result5.pivot(index = 'Name', columns ='Category', values =
'rate').fillna(0)
R_df
```

Category Name	BUSINESS	CRIME	EDUCATION	ENTERTAINMENT	MONEY	RELIGION	SCIENCE	SPORT	STYLE & BEAUTY	TRAVEL	WOMEN
billgates	0.0	0.0	1.0	4.0	0.0	0.0	3.0	0.0	0.0	5.0	2.0
chrisrock	0.0	2.0	0.0	5.0	0.0	0.0	0.0	0.0	3.0	4.0	1.0
debruynekev	1.0	0.0	0.0	5.0	0.0	0.0	0.0	2.0	3.0	4.0	0.0
kapilsharmak9	0.0	0.0	0.0	5.0	0.0	2.0	0.0	0.0	3.0	4.0	1.0
vp45	0.0	0.0	0.0	5.0	1.0	3.0	0.0	0.0	0.0	4.0	2.0

Рисунок53 – Матрица нашего фрейма данных ′.

Листинг20 – «Внедрение метода SVD для поиска похожих пользователей»

```
R = R_df.to_numpy()
user_ratings_mean = np.mean(R, axis = 1)
R_demeaned = R - user_ratings_mean.reshape(-1, 1)
```

```
f = R_df.to_numpy()
f
```

```
array([[0., 0., 1., 4., 0., 0., 3., 0., 0., 5., 2.],
       [0., 2., 0., 5., 0., 0., 0., 0., 3., 4., 1.],
       [1., 0., 0., 5., 0., 0., 0., 2., 3., 4., 0.],
       [0., 0., 0., 5., 0., 2., 0., 0., 3., 4., 1.],
       [0., 0., 0., 5., 1., 3., 0., 0., 0., 4., 2.]])
```

Внедрение метода SVD для поиска похожих пользователей

Листинг21 – «Внедрение метода SVD для поиска похожих пользователей»

```
from scipy.sparse.linalg import svds
U, sigma, Vt = svds(R_demeaned, k = 3)

sigma = np.diag(sigma)

all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) +
user_ratings_mean.reshape(-1, 1)
preds_df = pd.DataFrame(all_user_predicted_ratings, columns =
R_df.columns)

preds_df
```

In [120]: `preds_df`

Out[120]:

Category	BUSINESS	CRIME	EDUCATION	ENTERTAINMENT	MONEY	RELIGION	SCIENCE	SPORT	STYLE & BEAUTY	TRAVEL	WOMEN
0	-0.001019	0.136172	0.973023	4.039831	0.043630	-0.063800	2.934494	0.005674	-0.086977	4.995646	2.023328
1	0.396372	0.710396	0.057230	4.967291	-0.079240	0.251177	0.201114	0.807456	2.995412	4.100386	0.592407
2	0.536154	0.876303	0.060827	4.848949	-0.114122	0.006888	0.074213	1.018173	3.419728	3.902033	0.370854
3	0.167517	0.398229	-0.166301	5.268374	0.274827	1.657770	-0.376415	0.396278	2.380451	4.013348	0.985921
4	-0.132692	-0.093491	0.086906	4.853820	0.854855	3.165578	0.189590	-0.300948	0.345425	3.982585	2.048372

Рисунок54 – SVDРезультат.

4 КАРТИРОВАНИЕ КНИГ ДЛЯ ПОЛЬЗОВАТЕЛЕЙ

4.1 Content Based Filtering

Рекомендательные системы - это активные системы фильтрации информации, которые персонализируют информацию, поступающую к пользователю, в зависимости от его интересов, актуальности информации и т. Д. Рекомендательные системы широко используются для рекомендации фильмов, статей, ресторанов, мест для посещения, предметов для покупки и т. Д. .

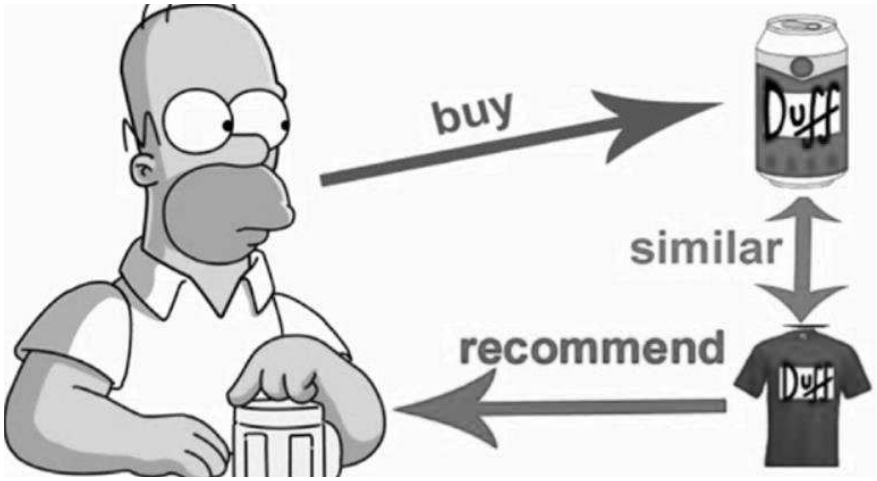


Рисунок 14 – ContentBasedRecomendation.

Системы рекомендаций, основанные на содержании, рекомендуют элементы пользователю, используя сходство элементов. Эта система рекомендаций рекомендует продукты или предметы на основе их описания или характеристик. Он определяет сходство между продуктами на основе их описания.

Он также учитывает предыдущую историю пользователя, чтобы порекомендовать аналогичный продукт.

Пример. Если пользователю нравится роман Сидни Шелдона «Расскажи мне свои мечты», то система рекомендаций рекомендует пользователю прочитать

другие романы Сидни Шелдона или рекомендует роман в жанре «научно-популярная литература». (Романы Сидни Шелдона относятся к жанру научно-популярной литературы).

Как я уже упоминал выше, мы используем данные goodreads.com и не ведем историю чтения пользователей. Следовательно, мы использовали простую систему рекомендаций, основанную на содержании. Мы собираемся построить две рекомендательные системы, используя название книги и описание книги.

Нам нужно найти книги, похожие на данную книгу, а затем рекомендовать эти похожие книги пользователю. Как определить, похожа данная книга или нет? Чтобы найти это, использовалась мера сходства.

Алгоритм TF * IDF используется для взвешивания ключевого слова в любом документе и присвоения важности этому ключевому слову в зависимости от того, сколько раз оно встречается в документе. Проще говоря, чем выше показатель TF * IDF (вес), тем реже и важнее термин, и наоборот. Математически [не волнуйтесь, это просто :)],
Каждому слову или термину соответствует соответствующий показатель TF и IDF. Произведение оценок термина TF и IDF называется весом TF * IDF этого термина.
TF (частота термина) слова - это количество раз, которое оно встречается в документе. Зная это, вы сможете увидеть, используете ли вы термин слишком часто или слишком редко.

$$W_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

$tf_{x,y}$= Frequency of x in y

df_x= number of documents containing x

Term x within document y N = total number of documents

В Pythonscikit-learn предоставляет вам предварительно созданный векторизаторTF-IDF, который пословно вычисляет оценку TF-IDF для описания каждого документа.

Здесь tfidf_matrix- это матрица, содержащая каждое слово и его показатель TF-IDF для каждого документа или элемента в данном случае. Кроме того, стоп-слова - это просто слова, не добавляющие существенной ценности нашей системе, такие как «an», «is», «the», и поэтому система игнорирует их.

Теперь у нас есть представление о каждом элементе с точки зрения его описания.

Далее нам нужно рассчитать релевантность или сходство одного документа с другим.

4.2 ВЕКТОРНО-ПРОСТРАНСТВЕННАЯ МОДЕЛЬ

В этой модели каждый элемент хранится как вектор его атрибутов (которые также являются векторами) в n-мерном пространстве, а углы между векторами вычисляются для определения сходства между векторами.

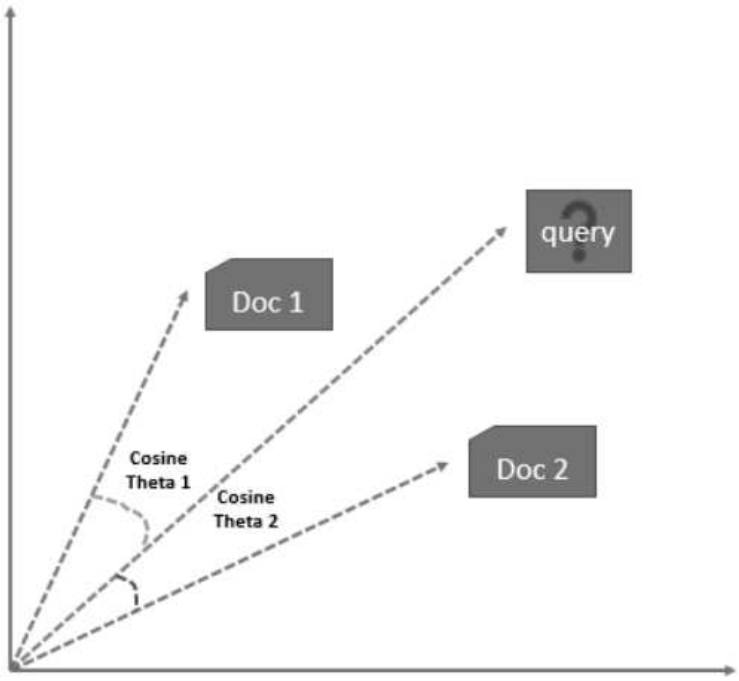


Рисунок 54 – Вектор на основе документа.

Метод вычисления симпатий / антипатий / оценок пользователя рассчитывается путем взятия косинуса угла между вектором профиля пользователя (Ui) и вектором документа; или, в нашем случае, угол между двумя векторами документа.

Конечная причина использования косинуса заключается в том, что значение косинуса будет увеличиваться с уменьшением угла между векторами, что означает большее сходство.

Векторы нормализуются по длине, после чего они становятся векторами длины

1.Calculating Cosine Similarity:

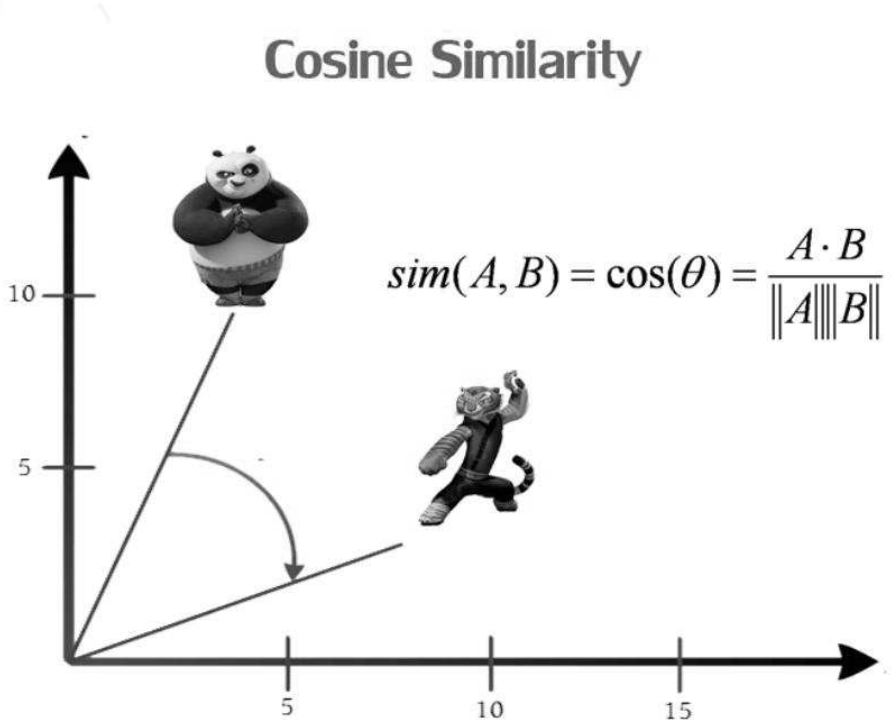


Рисунок 14 – CalculatingCosineSimilarity:

для работы над своим проектом я использую тензорный поток и нахожу пользовательский контент, чтобы рекомендовать книги

TensorFlow - это сквозная платформа с открытым исходным кодом для создания приложений машинного обучения. Это символьная математическая библиотека, которая использует поток данных и дифференцируемое программирование для выполнения различных задач, направленных на обучение и логический вывод глубоких нейронных сетей.
Он позволяет разработчикам создавать приложения для машинного обучения с использованием различных инструментов, библиотек и ресурсов сообщества.
В настоящее время самой известной в мире библиотекой глубокого обучения является TensorFlow от Google. Продукт Google использует машинное обучение во всех своих продуктах, чтобы улучшить поисковую систему, перевод, добавление субтитров к изображениям или рекомендации.

Приведу конкретный пример: пользователи Google могут получить более быстрый и точный поиск с помощью ИИ. Если пользователь вводит ключевое слово в строке поиска, Google дает рекомендацию о том, какое слово может быть следующим.

TensorFlow позволяет создавать графики и структуры потоков данных, чтобы определять, как данные перемещаются по графику, принимая входные данные в виде многомерного массива, называемого Tensor.

Это позволяет вам построить блок-схему операций, которые могут быть выполнены с этими входами, которая идет на одном конце и приходит на другом конце в качестве вывода.

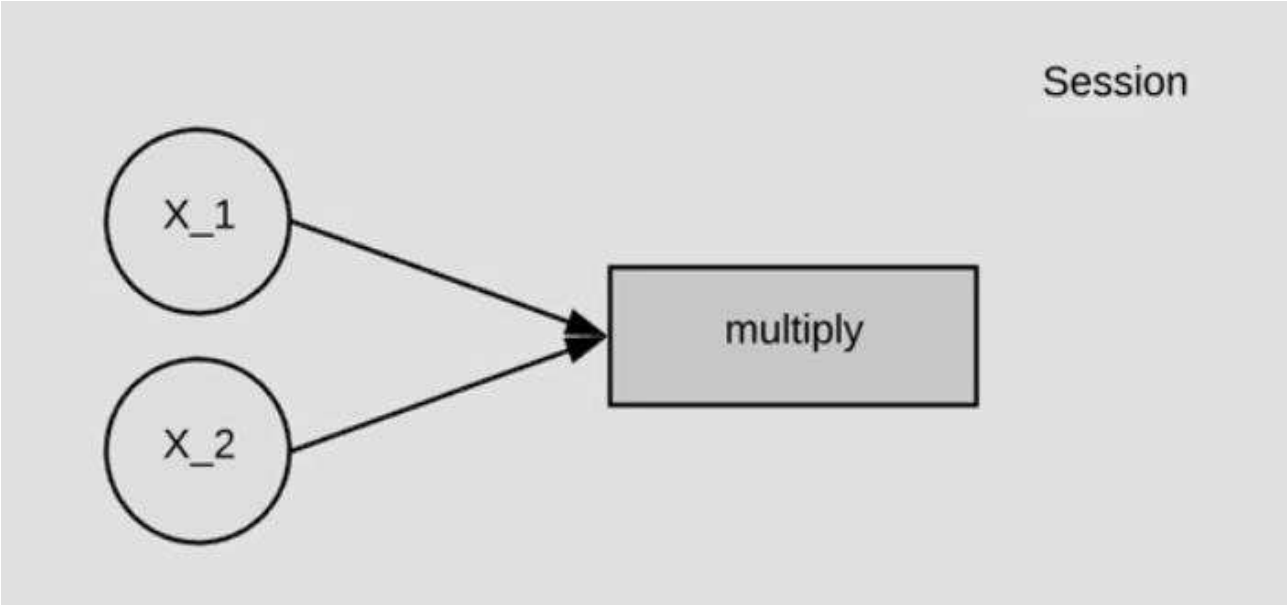


Рисунок55 – TensorFlow

Листинг 21 – «TensorFlow»

```
arr = np.array(preds_df)

users_category = tf.convert_to_tensor(
arr, dtype=tf.float32
)

users_category
```

```
<tf.Tensor: shape=(5, 9), dtype=float32, numpy=
array([[ 0.18158263, -0.2186105 ,  2.0251153 ,  4.075247  , -
0.09407976,
2.8680902 ,  0.26262087,  0.91147614,  4.9885583 ],
[ 0.3563526 ,  1.0379374 , -0.06133255,  5.0657134 ,  1.0771668 ,
        -0.15403107,  0.67709833,  3.0374937 ,  3.963601  ],
[ 0.28823665,  1.0058769 , -0.06753055,  4.7279534 ,  0.29560593,
         0.48775706,  0.9253909 ,  3.287963  ,  4.0487466 ],
[ 0.39723694,  0.94743514,  0.14328231,  5.230062  ,  1.5851232 ,
        -0.37229326,  0.4453818 ,  2.6376667 ,  3.9861052 ],
[ 0.7243516 ,  0.28170577,  1.9514662 ,  4.8780613 ,  3.1674478 ,
         0.21010749, -0.38345248,  0.15425509,  4.016057  ]],
dtype=float32)>
```

в нем описываются показатели всех пользователей, которые мы вычисляем, бросают алгоритмы машинного обучения и то, как это работает, и все результаты отображаются в массиве

в качестве примера мы собираемся использовать постоянные данные для тестирования

Листинг 21 – «books»

```
book = ['CRIME book', 'ENTERTAINMENT book', 'TRAVEL book',
'RELIGION bookbook', 'SCIENCEbook',
       'EDUCATION book', 'STYLE & BEAUTY book', 'TECH book',
'BUSINESS book', 'SPORTbook',
       'POLITICS book']
Book

category_books = tf.constant([
    [1, 1, 0, 1, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 1],
    [0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 1, 0, 1],
    [0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0]],dtype=tf.float32)

num_users = len(result5['Name'].unique())
num_books = len(book)

users_books = tf.matmul(users_category,category_books)

users_books

<tf.Tensor: shape=(5, 9), dtype=float32, numpy=
```

```
array([[0.18158263, 1.9880874 , 0.        , 3.0496728 , 0.1685411
,
4.075247  , 0.35012373, 0.        , 0.5987859 ],
       [0.3563526 , 1.3329574 , 0.        , 0.20232153, 1.7542651
,
5.0657134 , 2.1106176 , 0.        , 5.152598  ],
       [0.28823665, 1.226583  , 0.        , 0.7759937 , 1.2209969
,
4.7279534 , 1.5092335 , 0.        , 4.5894456 ],
       [0.39723694, 1.4879544 , 0.        , 0.02494368, 2.030505
,
5.230062  , 2.427742  , 0.        , 5.170225  ],
       [0.7243516 , 2.9575236 , 0.        , 0.9344591 , 2.7839954
,
4.8780613 , 3.508347  , 0.        , 3.6034088 ]], dtype=float32)>
```

```
top_users_features = tf.nn.top_k(users_books, 5)[1]
top_users_feature
```

```
<tf.Tensor: shape=(5, 5), dtype=int32, numpy=
array([[5, 3, 1, 8, 6],
       [8, 5, 6, 4, 1],
       [5, 8, 6, 1, 4],
       [5, 8, 6, 4, 1],
       [5, 8, 6, 1, 4]])>
```

```
for i in range(num_users):
feature_names = [book[int(index)] for index in
top_users_features[i]]
print('{}: {}'.format(users[i],feature_names))
```

И это конечный результат,
Рекомендуйте книги каждому пользователю на основе его сообщений и интересов.

```
billgates: ['EDUCATION book', 'RELIGION bookbook', 'ENTERTAINMENT
book', 'BUSINESS book', 'STYLE & BEAUTY book']
chrisrock: ['BUSINESS book', 'EDUCATION book', 'STYLE & BEAUTY
book', 'SCIENCEbook', 'ENTERTAINMENT book']
debruynekev: ['EDUCATION book', 'BUSINESS book', 'STYLE & BEAUTY
book', 'ENTERTAINMENT book', 'SCIENCEbook']
kapilsharmak9: ['EDUCATION book', 'BUSINESS book', 'STYLE & BEAUTY
book', 'SCIENCEbook', 'ENTERTAINMENT book']
```

```
vp45: ['EDUCATION book', 'BUSINESS book', 'STYLE & BEAUTY book',
'ENTERTAINMENT book', 'SCIENCEbook']
```

ЗАКЛЮЧЕНИЕ

В этой дипломной работе применяются алгоритмы машинного обучения, чтобы рекомендовать книги пользователям, с использованием алгоритмов НЛП, она классифицирует Tweets Твиттера по заранее определенным классам, таким как бизнес, политика, спорт, развлечения, технологии и другие классы.

система проанализировала и получила информацию об интересах каждого отдельного пользователя, и на основе этой информации была проведена исследовательская работа по применению различных алгоритмов рекомендаций, таких как совместная фильтрация, алгоритмы на основе контента и другие, чтобы рекомендовать книги пользователям.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Kamla Nehru Institute of Technology, Sultanpur | KNIT · Computer Science & Engineering.

2. PijushKanti Dutta PramanikNational Institute of Technology, Durgapur, India · Computer Science & Engineering, PhD (pursuing), M.Tech., M.Phil., MBA (IT), MCA, MIT, C. 3-7

3. Taylor. The Principles of Scientific Management / Frederick Winslow Taylor. – Harper & Brothers in New York, 1911. – 144 c.

4. Fayol. General and industrial management; translated from the French edition (Dunod) by Constance Storrs, with a foreword by L. Urwick. / Henri Fayol. – London: Pitman, 1967. – 110 c.

5.Said A. SalloumUniversity of Sharjah | US · Research Institute of Sciences & EngineeringResearcher in Computer Science, C. 8–12

6. Clark. Application of a Technique for Research and Development Program Evaluation. / Malcolm, D. G., J. H. Roseboom, C. E. Clark, W. Fazar // Operations Research, Vol. 7, No. 5, September–October 1959. – C. 600–650

7. Royce. Managing the Development of Large Software Systems / Winston Walker Royce // Proceedings of IEEE WESCON, №26 (August 1970). – C. 1–9

8.http://www.everyspec.com/DoD/DoD-STD/DOD-STD-2167A_8470/ (дата обращения: 09.06.2021)

9.https://towardsdatascience.com/recommender-system-singular-value-decomposition-svd-truncated-svd-97096338f361

10.http://rstudiopubsstatic.s3.amazonaws.com

11. A Rational Design Process: How And Why To Fake It // https://www.cs.tufts.edu/~nr/cs257/archive/david-parnas/fake-it.pdf (датаобращения: 06/09/2021)

12. Оучи Уильям. Методы организации производства. Японский и американский подходы / Уильям Оучи — М.: Экономика, 1984. – 184 с.

13.http://rstudiopubsstatic.s3.amazonaws.com/335300_11d40bf12d8940f78d9661b3c63150dc.html#applying-svd-to-toy-matrix-a

14. Evanish. How to Structure (and get the most out of) Customer Development Interviews / Evanish Jason // https://jasonevanish.com/2012/01/18/how-to-structure-and-get-the-most-out-of-customer-development-interviews/, 2012 (датаобращения: 06/09/2021)