УДК 004.942

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА С ПРИМЕНЕНИЕМ APACHE SPARK

Паньков А.В., Хололович И. С.
ГрГУ им. Я. Купалы, Гродно, Беларусь, iryna.khal@gmail.com

Рекомендательные системы – программы, которые пытаются предсказать, какие объекты (фильмы, музыка, книги, новости, веб-сайты) будут интересны пользователю, имея определенную информацию о его профиле [1]. Рекомендательные системы сравнивают однотипные данные от разных людей и вычисляют список рекомендаций для конкретного пользователя. Это удобная альтернатива поисковым алгоритмам, так как позволяют обнаружить объекты, которые не могут быть найдены последними [2].

Виды рекомендательных систем:

- Статистические (statistical approach) – это системы, которые основываются на статистических данных, собранных с пользователей;
- Демографические (demographic recommendation) сравнивают характеристики объекта с характеристиками пользователя;
- Ассоциативные (association rules) строят рекомендации на основе данных о том, какие объекты используются вместе;
- Информационные (content based) – это системы, которые ищут объекты, которые похожие на те, что пользователь уже положительно оценил;
- Коллективные (коллаборативные) (collaborative filtering) – самые распространенные системы, которые для предсказания рейтинга определенного объекта руководствуются оценками других пользователей;
- Гибридные (hybrid) - используют несколько выше упомянутых технологий.

Рассмотрим задачу разработки рекомендательной системы для интернет сервиса крупной сети ресторанов. Сервис предоставляет возможность сделать заказ через Интернет. Около 1 000 000 пользователей в течение последних 6 лет сделали около 2 500 000 заказов. К сервису постоянно присоединяются новые рестораны, регистрируются новые пользователи и, соответственно, объемы данных имеют тенденцию к увеличению.

Необходимо провести анализ заказов и предоставить рекомендации о том, когда отправить пользователю уведомление и какие дополнительные продукты могут его заинтересовать.

Данные хранятся в MongoDB – документоориентированной системе управления базами данных с открытым исходным кодом, не требующей описания схемы таблиц. В заказе отражено кто, когда сделал заказ, наименования продуктов и т.д.

Учитывая объем информации, обоснованно будет обратиться к алгоритмам машинного обучения и анализа данных. В мире, где информации все больше, машинное обучение часто единственный способ как-то ее осмыслить [6]. Для решения поставленной задачи выбран Apache Spark – быстрый и мощный механизм для крупномасштабной обработки данных [3]. Spark предоставляет библиотеку (ML) для работы с алгоритмами машинного обучения.

Однако, прежде всего необходимо настроить интегрирование данных в Spark, т.к. они хранятся не в HDFS (распределенная файловая система Apache Hadoop, которую использует Apache Spark). Данный процесс упростился с переходом на новую версию Apache Spark (2.2.0). Используя mongo-spark connector, импорт данных выглядит следующим образом:

```
spark = SparkSession \
 .builder \
 .appName("ALS") \
 .config("spark.mongodb.input.uri",
      "mongodb://hadoop.master:27017:restaurants.orders") \
 .getOrCreate()
df = spark.read.format("com.mongodb.spark.sql.DefaultSource") \
 .load()
```

После этого идет сложная процедура преобразования данных в Spark DataFrame (распределённая коллекция данных, организованная в виде именованных колонок).

Разработанная гибридная рекомендательная система использует алгоритмы поиска ассоциативных правил и коллаборативной фильтрации.

FP-Growth – алгоритм поиска ассоциативных правил, работает по принципу “разделяй и властвуй”. Он требует два сканирования базы данных. Во время первого сканирования базы FP-Growth вычисляет список часто встречающихся элементов, отсортированных по частоте в порядке убывания. При втором сканировании извлеченные данные сжимаются в FP-дерево. Затем FP-Growth начинает генерировать FP-деревья для каждого элемента, поддержка которого больше ξ (некоторого заданного значения минимальной поддержки) путем рекурсивного построения его условного FP-дерева. Алгоритм рекурсивно выполняет анализ FP-дерева. Проблема поиска часто встречающихся наборов элементов трансформируется в поиск и рекурсивное построение деревьев.

Spark использует параллельный алгоритм FP-Growth [5]. Этот алгоритм основан на новой схеме распределения данных и вычислений, которая фактически устраняет связь между компьютерами и позволяет выразить алгоритм с помощью модели MapReduce.

Пример входных данных для алгоритма FPG отражен в таблице 1. Где в колонке items находится информация о том, в какой день, время и кто сделал заказ. В ходе анализа этих данных можно найти пользователей, которые с некоторой частотой или периодичностью делают заказы.

Items
[Wed, 21:00, 48]
[Thu, 06:00, 57]
[Fri, 06:00, 60]
[Fri, 07:00, 61]
[Fri, 07:00, 61]
[Sat, 05:00, 68]
[Sat, 19:00, 170]

Таблица 1 – Пример входных данных для алгоритма FPG

На первом шаге FPG генерирует часто встречающиеся последовательности, пример которых, представлен в таблице 2. В колонке items уже найденные часто встречающиеся последовательности, freq – количество таких последовательностей в исследуемой выборке заказов.

items	freq
[179, Tue, 06:00]	15
[127, 22:00, Mon]	12
[170, 06:00]	25
[170, Mon]	18
[535, 00:00]	7

Таблица 2 – Часто встречающиеся последовательности

На втором шаге генерируются ассоциативные правила (таблица 3). Ассоциативным правилом называется импликация X=>Y, где X – причина ассоциативного правила, Y – следствие.

Правило X=>Y имеет поддержку ξ, если ξ процентов транзакций из D, содержат и X и Y. Достоверность δ правила показывает, какова вероятность того, что из X следует Y.

antecedent	consequent	confidence
[22:00, Mon]	[127]	0.23529411764705882
[519]	[23:00]	1.0
[Tue, 06:00]	[179]	0.10714285714285714

Таблица 3 – Ассоциативные правила

Таким образом, извлекается информация о том, когда пользователи обычно делают заказы и в назначенное время система может напоминать им об этом. Кроме того можно отправлять различные купоны и информацию о скидках на день раньше (позже) обычного времени заказа и, следовательно, увеличивать их количество.

Но кроме того когда отправить напоминание, можно предложить клиенту дополнительные продукты, которые могут его заинтересовать. Тут начинает работу алгоритм коллаборативной фильтрации.

ALS (Alternative Least Square) – один из самых известных алгоритмов коллаборативной фильтрации. Обучающая выборка для этого алгоритма задается в виде таблицы user, product, rating (таблица 4). В качестве рейтинга используется количество заказов данного продукта. После чего делается обучение модели с помощью ALS [4].

user	item	rating
83	11	1
147	37	1
166	70	1
173	53	1
199	77	2
170	29	6
200	85	1

Таблица 4 – Пример входных данных для алгоритма ALS

После работы алгоритма получаем таблицу с рекомендациями (таблица 5). Где user – идентификатор пользователя, recommendations – массив с идентификаторами продуктов и предполагаемым рейтингом.

user	recommendations
471	[[150,1.2139659], [66,1.0240686]]
463	[[168,2.1046312], [14,1.8773503]]
148	[[27,1.3024328], [168,1.1794108]]
516	[[168,1.1493344], [57,1.0392976]]

Таблица 5 – Пример результатов работы алгоритма ALS

В результате работы двух алгоритмов можно получить прогноз (таблица 6), когда пользователь сделает следующий заказ и что можно ему предложить. Эта информация позволяет увеличивать прибыль ресторанов путем увеличения количества заказов, а также предложением дополнительных продуктов и поддержанием интереса к системе.

antecedent	consequent	confidence	recommendations
[Sat]	[115]	0.19736842105263157	[[168,4.36477], [14,3.658646]]
[Tue, 06:00]	[179]	0.10714285714285714	[[82,6.4727564], [31,4.701923]]
[Mon, 22:00]	[127]	0.23529411764705882	[[58,7.8199296], [108,5.544762]]
[22:00]	[52]	0.0641025641025641	[[27,6.438395], [67,6.437268]]

Таблица 6 – Результат работы рекомендательной системы

В ходе проведенных исследований было установлено, что самым трудоемким и длительным является процесс подготовки данных, алгоритмы же выполняются в допустимые интервалы времени. Для оптимизации работы рекомендательной системы можно предварительно сохранить подготовленные данные на HDFS и периодически обновлять их при появлении новых. Это позволит избежать ненужных вычислений, ускорит время обработки, а так же решит проблему постоянно увеличивающегося объема данных.

Список литературы:

1. Википедия [Электронный ресурс] / Wikimedia Foundation, Inc – Режим доступа: https://ru.wikipedia.org. – Дата доступа: 01.11.2017.
2. Академик [Электронный ресурс] – Режим доступа: https://dic.academic.ru. – Дата доступа: 01.11.2017.
3. Apache Spark – Lightning-fast cluster computing [Электронный ресурс] / The Apache Software Foundation – Режим доступа: http://spark.apache.org. – Дата доступа: 01.11.2017.
4. Yifan Hu, Collaborative Filtering for Implicit Feedback Datasets / Yifan Hu, Yehuda Koren, Chris Volinsky // Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on - 10 February 2009.
5. Haoyuan Li, PFP: Parallel FP-Growth for Query Recommendation / Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, Edward Y. Chang // RecSys '08 Proceedings of the 2008 ACM conference on Recommender systems – Lausanne, Switzerland – October 23 - 25, 2008. – P. 107-114.

6. Новый ум короля: как создаются лучшие системы машинного обучения в мире [Электронный ресурс] / Популярная механика – Режим доступа: https://www.popmech.ru. – Дата доступа: 01.11.2017.