Рекомендательная машина для формирования индивидуальной образовательной траектории

В. С. Иванов
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
mrcitron@mail.ru

***Аннотация*. В докладе обоснована актуальность применения масштабируемой рекомендательной машины в связи с ростом вариативности учебного процесса. Выполнен обзор и анализ готовых решений, на основе которых было принято решение в проектировании собственного решения. Представлена концепция и архитектура предлагаемого решения.**

Ключевые слова: рекомендательная машина; учебный процесс; динамические модули; Apache

I. Введение

На данный момент рекомендательные системы распространены во многих сферах деятельности. Наиболее широкое применение данных систем наблюдается в сфере электронной коммерции, в результате чего с помощью определенных алгоритмов пользователю предоставляются рекомендации по товарам на основе информации о нём или о его действиях, совершенных ранее.

При этом в последнее время рекомендательные системы всё больше стали применяться в сфере обучения. Они используются для предоставления рекомендаций по выбору программ обучения на этапе поступления, для выбора различных дополнительных курсов, для выбора учебных материалов и т. д. [1]. Можно привести ещё множество примеров, где машинное обучение позволяет создать рекомендательную машину на основе объективных данных и поддержать принятие решений пользователем (абитуриентом, студентом, преподавателем).

Вопросов требующих рекомендаций увеличивается с ростом вариативности учебного процесса. В связи с этим необходима рекомендательная машина, которая может гибко расширяться новыми рекомендательными сервисами и масштабироваться.

В докладе рассматривается проектирование универсального фреймворка, стандартизующего разработку таких сервисов. Для этого был составлен набор требований к рекомендательной машине:

- Унифицированный интерфейс подключения сервисов.
- Поддержка различных моделей и движков машинного обучения.
- Возможность подключения к множеству источников данных.
- Масштабируемость вычислительных ресурсов.
- Поддержка проекта (актуальность последнего релиза).

Предполагается, что решение упростит разработку сервисов и интеграцию разработанных сервисов с уже существующими в организации информационными системами.

II. Существующие решения

A. Apache PredictionIO

Apache PredictionIO [2] – рекомендательный сервис, ориентированный на подключение различных сервисов и использующий в качестве среды машинного обучения Spark [3] и OpenNLP. Позволяет реализовывать свои модели.

У данного решения можно выделить следующие достоинства:

- Открытый исходный код.
- Гибкость инфраструктуры за счёт выбора установки всего пакета программ или установки только нужных его частей.
- Наличие библиотеки шаблонов для разработки собственных движков.
- REST API для интеграции приложений.

Одним из недостатков в данном проекте является ограничение возможности по использованию готовых библиотек и использованию других технологий: обучение только на Spark, все шаблоны выполнены либо на Java, либо на Scala.

Также существенным недостатком является то, что развитие платформы было прекращено в 2016 году. В 2020 году Apache PredictionIO был перемещён в Apache Attic и получил статус «удалён».

B. Harness

Проект является серверов машинного обучения на основе микросервисов. Предоставляет API для подключения модулей и реализует методы для обучения и предсказания.

Harness является REST-сервером с небольшим количеством CRUD-запросов на сущности, связанные с обучением и предсказанием, и на пользователей системы. А также, включающий в себя набор помощников для работы со Spark, MongoDB и HDFS.

Данное решение идейно подходит под поставленную задачу, однако, имеет ряд недостатков:

- Ограничивается реализацией на Scala.
- Интеграция со Spark ограничивается наличием нескольких классов в ядре, при этом не поддерживаются версии 3.X.

- Добавление, удаление и изменение моделей требуют пересборки проекта, что в рамках требуемой концепции является недостатком.
- Проект ориентирован на работу с Big Data.
- Отсутствует документация по созданию и подключению модулей.

C. Выводы

Были рассмотрены аналогичные решения компаний Apache и ActionML:

- Apache PredictionIO.
- Harness.

ТАБЛИЦА I СРАВНЕНИЕ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Критерий	PredicitonIO	Harness
Подключение	+	+
Поддерживаемые движки	Spark, OpenNLP	Реализованные на Scala
Источники	+	-
Масштабируемость	+	+
Последний релиз	2016	2021

Оба решения имеют схожую концепцию с той, которая заложена в поставленной задаче, и обладают своими достоинствами. Однако, на основе недостатков, связанных с ограничениями используемых технологий, методов разворачивания и поддержкой систем, было принято решение в проектировании собственного решения.

III. КОНЦЕПЦИЯ

Идея – требуется спроектировать унифицированную рекомендательную машину с унифицированным REST API.

Машина должна выполнять две функции:

- Выдавать рекомендацию.
- Выполнять дообучение/переобучение.

Функция рекомендации должна принимать в себя вектор некоторых данных, например, профиль абитуриента, и выдавать рекомендации, связанные с учебным процессом.

Для формулирования рекомендаций в системе должна быть некоторая модель (нейронная сеть, деревья или другие модели машинного обучения), полученная в результате обучения с помощью выбранного алгоритма для поставленной задачи [4, 5].

При этом обучение может выполняться как в рамках рекомендательной машины, так и вне её. Соответственно, во втором случае модель должна быть портирована.

Одна и та же модель может по-разному работать на разных движках (ML Engine), например, Apache Spark, Tensorflow [6], sklearn [7], DL4J [8] и т. д. Поэтому, предполагается, что не будет жесткой привязки к определенной библиотеке машинного обучения.

Для интеграции унифицированных REST API и разных движков машинного обучения в системе должен присутствовать промежуточный слой (ядро или центральный сервер рекомендательной машины).

Центральный сервер должен содержать ряд адаптеров для взаимодействия с тем или иным движком и для вызова той или иной модели. Изменения в самих движках при этом планируются на уровне конфигурации.

После появления новых данных в системах, из которых они получаются, потребуется дообучение существующих моделей.

Однако, в отличие от функции предсказания, передача данных через саму функцию является ресурсозатратной, а в некоторых случаях – невозможной. Соответственно, информация для переобучения должна браться из других систем напрямую.

Т. к. разрабатываемая рекомендательная машина должна являться универсальной, то и интерфейс взаимодействия с разными системами должны быть унифицирован.

IV. АРХИТЕКТУРА

В архитектуре универсальной рекомендательной машины было выделено 4 слоя:

- Слой представления.
- Слой ядра.
- Слой доступа к данным.
- Слой модели.

Архитектура приведена на рис. 1 и представлена в виде слоёв по аналогии с [9].

A. Слой представления

Слой представления содержит сторонние приложения, которые будут обращаться к разрабатываемой машине через REST API.

B. Слой ядра

Слой ядра содержит центральный сервер, реализующий бизнес-логику рекомендательной машины. Выделение данного слоя позволяет добиться масштабируемости вычислительных ресурсов, доступ к которым настраивается в конфиг-файлах.

Данный слой содержит 2 основных модуля:

- Relearn – модуль для выполнения обучения. Использование осуществляется с помощью POST запроса на endpoint */api/relearn/<service_name>*.
- Predict – модуль для получения рекомендаций. Использование осуществляется с помощью POST запроса на endpoint */api/predict/<service_name>*.

Оба модуля используют класс ModuleManager. ModuleManager – класс, предназначенный для динамического подключения встраиваемых сервисов по их названию. Соответственно, для выбора нужного сервиса в момент обращения используется параметр *service_name*, по которому выполняется нужный модуль.

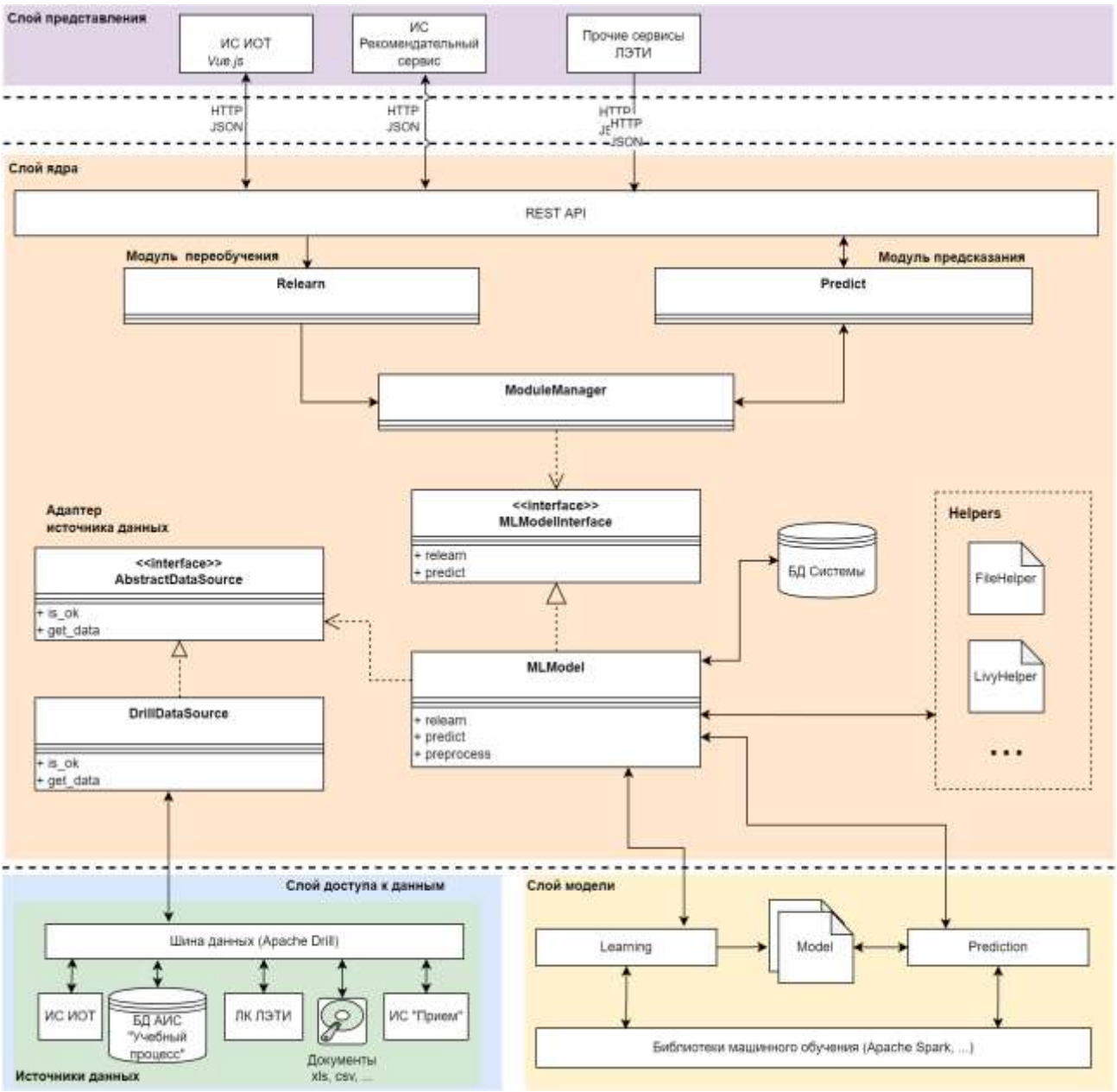

Рис. 1. Архитектура рекомендательной машины

Встраиваемые сервисы должны соответствовать интерфейсу MLModelInterface. Т. е. должны реализовывать 2 метода: relearn и predict. Описанное решение гарантирует единый интерфейс подключения сервисов.

Данные методы могут включать в себя:

- Обращение к источникам данных.
- Предобработку данных.

Предобработка данных для каждого отдельного встраиваемого сервиса является индивидуальной. Однако для упрощения процесса разработки был выделен модуль адаптеров Helpers, который содержит классы реализующие методы для последующего их переиспользования в других.

- Использование базы данных системы.

Для обработки категориальных признаков может потребоваться механизм установления соответствия их кодировки в процессе обучения и составления рекомендаций. При обучении кодировку можно сохранять в базу данных и использовать сохраненные значения при предсказании.

C. Слой доступа к данным

Слой доступа к данным реализован в виде интеграционной шины для решения проблемы неоднородности при подключении множества доступных источников данных. Для обеспечения единого интерфейса для доступа к этим источникам выбран Apache Drill [10]. Drill предоставляет SQL-подобный язык для создания запросов к разноплановым источникам, в т.ч. к нереляционным.

D. Взаимодействие со слоем модели

За счёт вынесения работы с моделями машинного обучения от основной бизнес-логики рекомендательной

машины достигается поддержка спектра движков машинного обучения, с помощью которых могут быть реализованы модели.

Прямая работа с Apache Spark подразумевает консольные варианты использования, что не подходит для данной рекомендательной машины. Для решения данной проблемы был выполнен поиск и анализ существующих решений для возможности удаленного доступа и осуществления взаимодействия с Apache Spark через REST API. В результате поиска был выбран Apache Livy. Это обёртка над Apache Spark, обеспечивающая взаимодействие серверов приложений с кластером Apache Spark через REST API.

Передачи данных с центрального ядра на сервер, реализующий слой модели, будет реализована в виде файлового сервера, реализующего отправку и получение файлов между упомянутыми серверами.

Данное решение обусловлено тем, что:

- с одной стороны, Livy имеет ограничение на размер входных данных при обращении через REST API, поэтому данные отправляются предварительно, а запросе к Livy указываются пути для доступа к отправленным файлам;
- с другой стороны, предоставляет результат выполнения операций Apache Spark в виде лог-файлов, имеющих ограничение по размеру, и при большом количестве результирующих данных использование стандартного механизма (запись в лог-файлы и обращение через REST API к ним) влечет потерю части данных.

V. Заключение

В докладе рассмотрена задача проектирования универсального фреймворка, стандартизирующего разработку рекомендательных сервисов, направленных на формирование индивидуальной образовательной траектории входе учебного процесса.

Данное решение имеет следующие достоинства:

- для сервисов – унифицированный интерфейс получения информации из различных систем организации с предоставлением обезличенных данных;
- для систем – единый интерфейс получения рекомендаций;
- для разработчиков сервисов – упрощение разработки за счёт реализации части функциональности в рамках фреймворка.

Данное решение рассматривалось в контексте учебного процесса, однако, это не является ограничением, в результате чего система может использоваться и на различных предприятиях.

В качестве дальнейшего развития предложенного решения может разработка функциональности, позволяющей не только обращаться к функциям обучения и предсказания, но и экспериментировать с параметрами для используемых моделей машинного обучения, что поможет улучшить их точность и упростить процесс разработки сервисов.

Список литературы

[1] Rivera A.C., Tapia-Leon M., Lujan-Mora S. Recommendation systems in education: A systematic mapping study //International Conference on Information Technology & Systems. Springer, Cham, 2018. С. 937-947.

[2] Chan S. et al. PredictionIO: a distributed machine learning server for practical software development //Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013. С. 2493-2496.

[3] Meng X. et al. Mllib: Machine learning in apache spark //The Journal of Machine Learning Research. 2016. Т. 17. №. 1. С. 1235-1241.

[4] Рогачевский Н. В. ОСНОВЫ ПОСТРОЕНИЯ МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ //Ответственный редактор. 2021. С. 118.

[5] Chentsov D.A., Belyaev S.A. Monte Carlo Tree Search Modification for Computer Games //2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). IEEE, 2020. С. 252-255.

[6] Shukla N., Fricklas K. Machine learning with TensorFlow. Greenwich: Manning, 2018.

[7] Garreta R., Moncecchi G. Learning scikit-learn: machine learning in python. Packt Publishing Ltd, 2013.

[8] Patterson J., Gibson A. Deep learning: A practitioner's approach. "O'Reilly Media, Inc.", 2017.

[9] Беляев С.А., Черепкова Ю.С. Архитектура среды моделирования для проведения экспериментов с интеллектуальными агентами //Программные продукты, системы и алгоритмы. 2017. №. 3. С. 4-4.

[10] Givre, C., & Rogers, P. Learning Apache drill: Query and analyze distributed data sources with SQL. "O'Reilly Media, Inc.", 2018.