Архитектура системы кластеризации интернет-пользователей на основе данных о переходах (clickstream)

М.В. Гладких, mg6625@gmail.com

Воронежский Государственный Университет

Аннотация. *Качество взаимодействия с пользователями, понимание их проблем и потребностей - важное конкурентное преимущество в электронной коммерции. Предлагается вариант архитектуры системы кластеризации интернет-пользователей на основе анализа поведения пользователей. Описаны компоненты системы и порядок их взаимодействия. Дан обзор подходов, применимых для анализа потока кликов.*

Ключевые слова: *архитектура рекомендательной системы, поток кликов, анализ поведения интернет-пользователей, clickstream analysis.*

Введение

В настоящее время рынок товаров и услуг все больше переходит в онлайн. Вследствие чего обостряется конкуренция за потребителя. Успех продукта в электронной коммерции во многом зависит от качества взаимодействия с пользователями и понимания их проблем и потребностей.

Особенностью ведения бизнеса в интернете является сведение к минимуму личного контакта с потребителем, либо его отсутствие. С другой стороны, каждый пользователь, использующий любой интернет-ресурс, совершает на нем определенные действия. К таким действиям относятся переходы по страницам сайта, прокрутка страницы, клики, листание и другие.

Сбор и анализ действий, генерируемых посетителями, дает возможность составить более точное представление о предпочтениях и проблемах посетителей и извлечь максимум выгоды из этой информации.

Важным типом действия посетителя является переход по страницам сайта. Последовательность переходов конкретного посетителя составляет поток переходов, в англоязычной среде такой тип данных называется поток кликов или clickstream.

Проблема современных интернет-ресурсов — это сложность персонализации контента под каждого пользователя. Оценка предпочтений и создание рекомендаций является сложной задачей.

Для решения данной задачи предлагается кластеризация посетителей, используя данные об их переходах в рамках отдельного интернет-ресурса. Отнесение посетителя к определенной группе позволит персонализировать контент ресурса для этого посетителя.

В настоящей статье описывается архитектура системы, с помощью которой возможно выполнять полный цикл действий по определению типа посетителя. К указанным действиям относятся следующие:

1. Идентификация пользователя.
2. Сбор и запись данных о его переходах и другой технической информации.
3. Выявление основных типов (групп) посетителей с использованием алгоритмов кластерного анализа машинного обучения.
4. Предсказание того, к какому типу относится конкретный посетитель после сбора достаточного объема данных.

Анализ существующих решений для сбора данных

В настоящее время на рынке не представлены программные продукты с аналогичным функционалом в полном объеме. Однако существуют несколько сервисов, предоставляющих возможность собирать и обрабатывать данные о поведении пользователей. Нижеуказанные сервисы можно сравнивать с отдельными компонентами проектируемой системы, а именно модулем сбора данных.

Проанализированы три наиболее функциональные в настоящее время сервисы:

1. Яндекс.Метрика (metrika.yandex.ru).
– Позволяет собирать данные о любых действиях пользователя на веб-страницах, в том числе кликах, движения мышью, прокрутке и других.
– Данные можно визуализировать в виде «Тепловой карты кликов».
– Отсутствие программного интерфейса для получения необходимых для целей работы системы адаптации данных.
– Недолгий срок хранения данных в сервисе.
– Отсутствует привязка к действиям конкретного пользователя.
2. Google Analytics (analytics.google.com).
– Позволяет собирать данные только по переходам по гиперссылкам.
– Не собирает данные о «холостых» кликах (когда пользователь попадает по области веб-страницы, для которой не предусмотрено перехода).
3. Hotjar (www.hotjar.com).

– Позволяет собирать данные о любых действиях пользователя на страницах.
– Данные можно визуализировать в виде «Тепловой карты кликов».
– Предоставляет данные в виде таблицы.
– К минусам относится недостаточный набор данных.

При изучении указанных сервисов сделан вывод об отсутствии готового источника данных, необходимых для работы системы. Кроме того, использование готового решения ограничит возможности для экспериментов в ходе разработки и совершенствования системы в целом.

Требования к системе

К системе предъявляются следующие требования:

– легкая интеграция с любым веб-сайтом, открываемом в браузере (далее - хост), независимо от технологии, на основе которой разработан хост, в том числе одностраничными приложениями (SPA).
– Отсутствие существенного влияния на производительность и скорость загрузки хоста.
– Отсутствие влияния на работу сторонних сервисов, используемых хостом (аналитики, рекламные сети и т.п.).
– Осуществление передачи данных между компонентами системы наиболее оптимальным способом.

Структура системы

По своей сути система — это клиент-серверное веб-приложение. Она состоит из двух компонентов: клиента и сервера. Клиент работает в браузере и запускается в момент загрузки веб-страницы. Сервер обрабатывает запросы клиента и работает постоянно. Каждый компонент может включать независимые модули.

Учитывая предъявляемые требования, система имеет следующую структуру.

1. Клиент:
– модуль сбора данных,
2. Сервер:
– модуль обработки, хранения и преобразования данных,
– модуль обучения,
3. База данных.

Характеристика клиентской части системы

Модуль сбора данных представляет собой скрипт, написанный на языке программирования JavaScript. Основная задача модуля – отслеживание переходов пользователя, подготовка и передача данных серверу.

Функционал идентификации пользователя реализован с использованием сторонней библиотеки Browser Fingerprint 3. Техника, используемая библиотекой, позволяет осуществлять анонимную идентификацию браузера. Код библиотеки опрашивает браузер пользователя на предмет всех специфичных и уникальных настроек и данных для этого браузера, системы и устройства пользователя. Данные объединяются в массив строк, затем подаются на вход функции хеширования. Функция возвращает хеш общего назначения.

Полученный хеш передается с каждой порцией данных на сервер. Если хеш не удалось получить по какой-либо причине, модуль не инициализируется, данные серверу не передаются.

Модуль отправляет данные серверу несколько раз в течение секции. Первая отправка происходит сразу после загрузки страницы. Для работы модуля не требуется ожидание загрузки и рендеринг контента на странице, все необходимые данные доступны в момент инициализации. Для отправки данных используется встроенный метод *fetch()*. Формат передачи данных – JSON.

Данные, отправляемые серверу, представлены в таблице.

Таблица

Данные о переходе, передаваемые от клиента серверу

Имя параметра	**Описание**
visitorId	Хеш-сумма, получаемая на основе оборудования посетителя
screenWidth	Ширина экрана устройства посетителя
orientation	Положение экрана на устройстве посетителя
timeStamp	Временная метка
lang	Язык, установленный на устройстве посетителя
platform	Операционная система на устройстве посетителя
userAgent	Идентификатор браузера посетителя [1]
pageUri	Адрес веб-страницы, на которую совершен переход

Далее модуль устанавливает наблюдение за полем *href* глобального объекта *location*, в котором хранится текущий адрес веб-страницы. При каждом изменении адреса модуль собирает набор данных из таблицы 1, включая новый адрес, и записывает их в кэш. Накопленные данные

передаются серверу каждые 10 секунд. После отправки данных кэш очищается.

Чтобы избежать потери собранных, но еще не отправленных данных, в случае закрытия вкладки браузера или перезагрузки страницы, модуль устанавливает наблюдение за браузерным событием *beforeunload*. Событие срабатывает перед закрытием окна или вкладки браузера или переходе на другую веб-страницу.

После срабатывания события *beforeunload* данные из кэша отправляются серверу браузерным методом *navigator.sendBeacon()*.

Для взаимодействия с хостом модуль может опционально принимать две функции обратного вызова в качества параметров. Первая функция срабатывает в случае успешного предсказания типа пользователя. Вторая функция запускается в случае возникновения ошибки.

Получив тип пользователя, приложение хоста может использовать эти данные для персонализации контента.

Модуль получает тип пользователя в ответе на первый запрос при условии, что данных по текущему юзеру достаточно и его удалось отнести к какой-либо группе.

Характеристика серверной части системы

Серверная часть системы представляет собой Node.JS приложение, написанное с использованием фрейморка для построения программных интерфейсов - Express.JS. Логически сервер разделен на модули, но технически это единое приложение.

Модуль обработки, хранения и преобразования данных получает от клиента данные о поведении пользователя, разделяет и группирует их по признакам, необходимым для оптимального хранения, и записывает в базу данных.

Главной сущностью при добавлении данных является переход (*transition)*. Включает данные, которые передает клиент. Модуль может принимать один или несколько переходов за один запрос. При каждом новом запросе модуль проверяет, достаточно ли данных, и обращается к модулю обучения.

Модуль обучения использует библиотеку машинного обучения TensorFlowJS. Включает два метода программного интерфейса:

- метод кластеризации посетителей,
- метод предсказания типа посетителя.

После накопления дополнительных данных сервер запускает задачу дообучения.

Работа модуля обучения разделена на две части.

1. Кластеризация посетителей. Задача данной части – выявить группы пользователей из всего массива собранных данных. Метод возвращает список определенных групп.
2. Предсказание типа посетителя. Задача данной части – определить, к какой группе относится текущий посетитель. Метод запускается, если по текущему пользователю накоплен необходимый набор данных. В случае удачного предсказания метод возвращает тип пользователя.

Характеристики каждой выявленной группы устанавливаются опытным путем и не рассматриваются в данной статье.

База данных содержит две основные таблицы:

– visitors, хранит данные пользователей, а также определенный тип пользователя в качестве кэша.
– transitions, хранит данные, переданные клиентской частью, имеет связь с таблицей visitors.

Кроме того, база содержит таблицу errors, в которую записываются ошибки.

Диаграмма базы данных показана на рисунке.

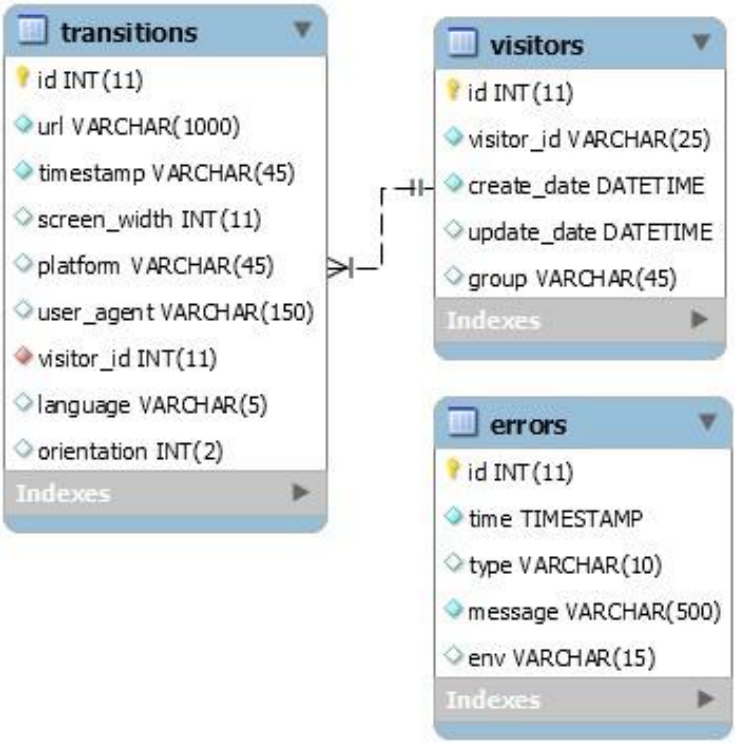

Рисунок. Диаграмма базы данных

Обзор подходов, применимых для анализа потока кликов

Для анализа применяются различные подходы и алгоритмы. Анализ существующих исследований показал, что наиболее эффективными подходами являются:

- цепь Маркова (Markov Chains) [2]. Представляет собой дискретную последовательность состояний, каждое из которых берётся из дискретного пространства состояний (конечного или бесконечного), удовлетворяющее марковскому свойству. Цепи Маркова лучше всего работают с последовательными данными, поток кликов является таким типом данных.
- Алгоритм cSPADE (Sequential Pattern Discovery using Equivalence classes) [3]. Заключается в моделировании данных потока переходов как вероятностей перехода в их представлении в виде последовательных шаблонов. Затем

определяются тенденции, которые происходят наименьшее количество раз. На каждой итерации алгоритм вычисляет частоту последовательностей только с элементом, затем вычисляет частоту последовательностей с двумя элементами и так далее.

- Алгоритм Метод k-средних (K-Means Clustering) [4]. Алгоритм стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. На каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на итерации не происходит изменения внутрикластерного расстояния.
- CUP (Clickstream pattern mining Using Pseudo-IDList) [5]. Алгоритм основан на новой структуре данных Pseudo-IDList. На основе этой структуры предлагается алгоритм CUP (поиск паттернов Clickstream с использованием Pseudo-IDList). Предложенный подход показывает эффективность и требует меньших вычислительных ресурсов.

Выбор нужного подхода либо их комбинации является предметом отдельных исследований.

Заключение

Системы кластеризации интернет-пользователей на основе данных о переходах имеют много вариантов практического применения в электронной коммерции. В данной статье рассмотрен вариант архитектуры такой системы.

Изучены сервисы, предоставляющие возможность сбора и анализа данных о поведении пользователя. В результате установлено, что готовые решения для сбора данных для целей реализации системы отсутствуют.

Определены требования к функционалу системы, форматы и типы собираемых, передаваемых и хранимых данных. На основе указанных форматов спроектирована структура базы данных.

Рассмотрены исследования в области анализа потока кликов. Определены алгоритмы, применяемые для такого анализа.

Список литературы

1. Hypertext Transfer Protocol -- HTTP/1.1 [Электронный ресурс]: спецификация – Режим доступа: tools.ietf.org/html/rfc2616#section-14.43.

2. Scholz, Michael. R Package clickstream - Analyzing Clickstream Data with Markov Chains / Journal of statistical software / Forthcoming, – 2017.

3. Anubhuti Singh, Sandhya Tarar. Clickstream Data Analysis Using Data Mining in R / International Journal of Science and Research (IJSR) / ISSN: 2319-7064, 2018.

4. Makris, Christos & Tsirakis, Nikos / A Model for Clustering Clickstream Data: researchgate.net/publication/255031096, – 2006.

5. M. Huynh, Huy & Nguyen, Loan & Vo, Bay & Yun, Unil & Oplatková, Zuzana & Hong, Tzung-Pei / Efficient algorithms for mining clickstream patterns using pseudo-IDLists / Future Generation Computer Systems, – 2020.