2. Урбанович, П.П. Защита информации методами криптографии, стеганографии и обфускации/ П.П. Урбанович. – Минск : БГТУ, 2016, – 220 с.

3. Урбанович, П.П. Безопасность данных мобильных приложений/ П.П. Урбанович, А.В. Годун // Информационные технологии: тезисы 82-й науч.-техн. конференции профессорско-преподавательского состава, научных сотрудников и аспирантов (с международным участием), Минск, 1-14 февраля 2018 г. [Электронный ресурс] / отв. за издание И.В. Войтов; УО БГТУ. – Минск: БГТУ, 2018. – С.12.

УДК 004.421.6

студ. А.В. Кравцевич
Науч. рук. ассист. Д.И. Черняк
(кафедра информационных систем и технологий, БГТУ)

СЕРВИС ДЛЯ ОБРАБОТКИ СТАТИСТИЧЕСКИХ ДАННЫХ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

В настоящее время статистические данные являются единственным источником информации, позволяющим судить о популярности интернет-ресурса, либо его определенных фрагментов. Путем сбора и обработки статистики владельцы интернет-ресурсов могут узнать, что больше всего привлекает их аудиторию и выявить наиболее перспективные пути развития ресурса.

Однако крупнейшие существующие системы сбора и обработки статистических данных (такие как Google Аналитика, Яндекс.Метрики и Piwik) полностью сконцентрированы на анализе посещений ресурса, игнорируя его содержание и обратную реакцию пользователей на это содержание. Таким образом, современные системы аналитики не позволяют с точностью судить о мнениях, интересах и пожеланиях посетителей веб-ресурсов.

С целью решения этой проблемы разрабатывается веб-сервис Taskmaster, который предоставляет публичный API, легко интегрируемый сайтами, веб-приложениями и иными сетевыми ресурсами, и позволяет производить автоматизированный анализ текстовых статистических данных, таких как публикации, комментарии и т.д. С его помощью владельцы ресурсов смогут автоматизировано обрабатывать обратную реакцию пользователей, определять наиболее популярные темы, а также оценить вкусы конкретных пользователей. Таким образом, подобный сервис сможет заменить одновременно как систему сбора и анализа статистических данных, так и систему рекомендаций, что значительно облегчит работу разработчиков веб-ресурса.

Основными методами анализа данных, которые позволяют автоматизировано обрабатывать текстовую информацию веб-ресурсов, являются метод тематического моделирования и анализ тональностей.

Метод тематического моделирования позволяет на основе большого массива текстов создать тематическую модель – модель коллекции текстовых документов, которая определяет, к каким темам относится каждый документ и какие слова составляют эти темы. Конкретной реализацией метода тематического моделирования, использованной в данном веб-сервисе, является латентное размещение Дирихле.

Анализ тональностей позволяет выделить эмоциональную составляющую текста, мнение автора о предмете обсуждения. Обрабатывая комментарии с помощью метода анализа тональностей, можно получить их эмоциональную характеристику (являются ли он положительными, нейтральными либо отрицательными) в виде чисел, которые позже гораздо легче поддаются методам сбора и обработки статистики. Метод анализа тональностей в данной работе реализован с помощью нейронной сети, содержащей векторизирующий слой, LSTM-слой, полносвязный слой и слой регрессии.

Готовым продуктом является веб-сервис, основаный на микросервисной архитектуре. Микросервисы представляют собой docker-контейнеры, каждый из которых содержит минималистичное Python-приложение, решающее определенную задачу. Для взаимодействия между собой микросервисы используют RESTAPI интерфейс. Основным инструментом для оркестровки контейнеров был выбран Kubernetes – система с открытым исходным кодом, разработанная Google, и гарантирующая совместный запуск, отказоустойчивость, репликацию и масштабирование контейнеров. Kubernetes-кластер развернут вручную на выделенных виртуальных серверах, арендованных на американском хостинг-провайдере DigitalOcean.

В процессе разработки сервиса для облегчения тестирования, доставки и развертывания кода было решено создать инфраструктуру на основе CI/CD методологии. Благодаря использованию ряда сервисов (GitHub, TravisCI, BetterCodeHub, DockerHub) тестирование, анализ кода, сборка и доставка контейнеров производится автоматически. Такой подход позволяет существенно сократить время разработки и увеличить производительность труда. Так же, в совокупности с использованием микросервисной архитектуры, CI/CD позволяет производить горячую замену кода – обновление приложения без необходимости его перезапуска, что крайне важно для отказоустойчивых высоконагруженных систем.

ЛИТЕРАТУРА

1. W. Richert, L.P. Coelho. Building machine learning systems with Python – «Pact», 2013. – 21с.

УДК 004.421.6

студ. В.А. Озик
Науч. рук. ассист. Д.И. Черняк
(кафедра информационных систем и технологий, БГТУ)

ПРОГРАММНЫЙ ПРОДУКТ ДЛЯ РЕАЛИЗАЦИИ ГОТОВЫХ РЕШЕНИЙ

На сегодняшний день приобретение различных товаров является обыденностью. И люди заезжают после работы в различные магазины, чтобы купить необходимые товары, и зачастую эти товары приобретаются для одной цели: сменить интерьер в комнате, собрать компьютер, приготовить блюдо, одеться соответствующе поре года т. д.

Интернет-магазины упростили эти действия путем заказа товара из любого места. Но, возможность того, что интернет-магазин будет предлагать готовые решения, состоящие из нескольких товаров, не реализовано в полной мере, по этой причине была поставлена цель, разработать программный продукт для размещения товаров в виде готовых решений, позволяющий приобретать различную продукцию или узнать информацию, где ее можно приобрести.

Для достижения моей цели, необходимо решить несколько задач:

1. Спроектировать и разработать базу данных для хранения информации о товарах, готовых решениях, пользователях и их действиях;
2. Спроектировать архитектуру программного продукта: провести иерархическую декомпозицию — сначала разбить систему на крупные функциональные модули/подсистемы, описывающие ее работу в самом общем виде. Затем, полученные модули, проанализировать более детально и, в свою очередь, поделить на под-модули либо на объекты;
3. Реализовать разграничения прав пользователей: будет реализовано несколько ролей, у каждой из которых будет своя задача и возможности;
4. Создать пользовательский интерфейс: для взаимодействия с приложением будет использоваться веб-интерфейс.