Отчет о проделанной работе

Содержание

Термины

Машинное обучение – обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.

API — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.

VK, FB (ВКонтакте, Facebook) — социальные сети, используется публичное API для сбора информации зарегистрированных пользователей.

LEADER-ID – информационная платформа Института развития лидеров. Открывает доступ к информации о системе возможностей направленной на формирование новых компетенций.

TF-IDF (от англ. TF — term frequency, IDF — inverse document frequency) — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален количеству употребления этого слова в документе, и обратно пропорционален частоте употребления слова в других документах коллекции.

1. Описание проекта

Исходя из целей АНО «Агентство стратегических инициатив по продвижению новых проектов» (Агентство), была разработана информационная платформа сопровождения Института развития лидеров. Была разработка концепция такой платформы, которая получила название LEADER–ID.RU. В рамках настоящего договора Исполнитель осуществлял доработку и поддержку платформы.

Институт развития компетенций - стратегическая инициатива Агентства, реализуемая с целью развития и поддержки лидеров – вовлечения активной части гражданского общества в деятельность осуществляемую Агентством, поддержки и сопровождения лидерских проектов, а так же система разнообразных «социальных лифтов», позволяющих достигать нового уровня карьерного, профессионального, личностного и социального развития.

Цель Института развития компетенций:

Создание системы поиска, анализа и предсказания возможных компетенций молодых лидеров на основе данных социальных сетей VK и FB, которые возможно привязать в личном кабинете.

2. Архитектура проекта

Техническая платформа

Имея ввиду специфику проекта, система управления сервисом была спроектирована и самостоятельно написана Исполнителем.

Технологии на базе которых работает сервис:

Python 3.6

Язык Python — универсальный язык программирования, применимый в том числе и в вебе. С технической точки зрения web-приложение на Python — полноценное приложение, загруженное в память, обладающее своим внутренним состоянием, сохраняемым от запроса к запросу. (В отличие от PHP)

Основные нововведения Python 3.6:

- Появилась поддержка форматируемых строковых литералов
- Читаемость чисел можно улучшать при помощи символов подчеркивания, например, таким образом — 1_000_000 или 0x_FF_FF_FF
- В новой версии определен синтаксис аннотаций для переменных
- Добавлена возможность определения асинхронных генераторов

Flask 0.12.2

Фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. Относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности. Поддерживается установка посредством пакетного менеджера PyPI, требует Python 2.6 и выше.

Scikit-learn 0.19.1

Библиотека scikit-learn предоставляет реализацию целого ряда алгоритмов для обучения с учителем (Supervised Learning) и обучения без учителя (Unsupervised Learning) через интерфейс для языка программирования Python.

Scikit-learn построена поверх SciPy (Scientific Python), который должен быть установлен перед использованием scikit-learn. Данный стек включает в себя:

- **NumPy**: расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц
- **SciPy**: открытая библиотека высококачественных научных инструментов для языка программирования Python
- **Matplotlib**: библиотека на языке программирования Python для визуализации данных
- **Pandas**: различные структуры данных и анализ

Одна из основных концепций библиотеки scikit-learn - библиотека с уровнем надежности и поддержки, который необходим для продакшн-систем, а это значит, что большое внимание уделяется вопросам удобства использования, качества кода, документации и оптимизации скорости работы библиотеки.

Redis 4.0

Redis (REmote DIctionary Server) — это *не*реляционная высокопроизводительная СУБД. Redis хранит все данные в памяти, доступ к данным осуществляется по ключу. Опционально копия данных может храниться на диске. Этот подход обеспечивает производительность, в десятки раз превосходящую производительность реляционных СУБД, а также упрощает секционирование (шардинг) данных.

Docker 4.0

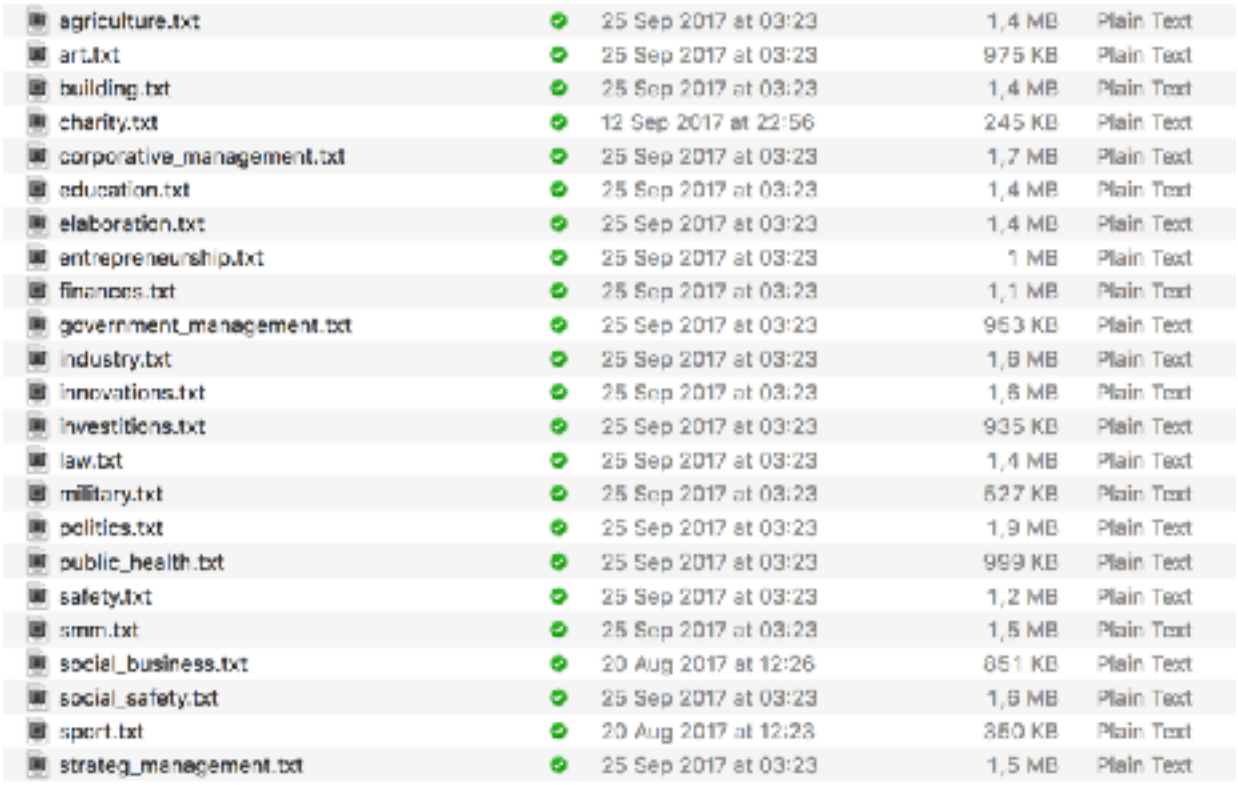
Docker — это открытая платформа для разработки, доставки и эксплуатации приложений. Docker разработан для более быстрого выкладывания ваших приложений. С помощью docker вы можете отделить ваше приложение от вашей инфраструктуры и обращаться с инфраструктурой как управляемым приложением. Docker помогает выкладывать ваш код быстрее, быстрее тестировать, быстрее выкладывать приложения и уменьшить время между написанием кода и запуска кода. Docker делает это с помощью легковесной платформы контейнерной виртуализации, используя процессы и утилиты, которые помогают управлять и выкладывать ваши приложения.

3. Проделанная работа

3.1. Собран корпус из текстов групп ВК и страниц FB

Задача: Составить набор текстов, чтобы в дальнейшем использовать эти данные для обучения с учителем.

Решение: Найдены подходящие сайты и публичные страницы ВКонтакте. Произведен парсинг сайтов и работа с API ВКонтакте для извлечения текстовой информации со стены публичных страниц.



agriculture.txt	25 Sep 2017 at 03:23	1,4 MB	Plain Text
art.txt	25 Sep 2017 at 03:23	975 KB	Plain Text
building.txt	25 Sep 2017 at 03:23	1,4 MB	Plain Text
charity.txt	12 Sep 2017 at 22:56	245 KB	Plain Text
corporative_management.txt	25 Sep 2017 at 03:23	1,7 MB	Plain Text
education.txt	25 Sep 2017 at 03:23	1,4 MB	Plain Text
elaboration.txt	25 Sep 2017 at 03:23	1,4 MB	Plain Text
entrepreneurship.txt	25 Sep 2017 at 03:23	1 MB	Plain Text
finances.txt	25 Sep 2017 at 03:23	1,1 MB	Plain Text
government_management.txt	25 Sep 2017 at 03:23	953 KB	Plain Text
industry.txt	25 Sep 2017 at 03:23	1,8 MB	Plain Text
innovations.txt	25 Sep 2017 at 03:23	1,8 MB	Plain Text
investitions.txt	25 Sep 2017 at 03:23	935 KB	Plain Text
law.txt	25 Sep 2017 at 03:23	1,4 MB	Plain Text
military.txt	25 Sep 2017 at 03:23	527 KB	Plain Text
politics.txt	25 Sep 2017 at 03:23	1,9 MB	Plain Text
public_health.txt	25 Sep 2017 at 03:23	999 KB	Plain Text
safety.txt	25 Sep 2017 at 03:23	1,2 MB	Plain Text
smm.txt	25 Sep 2017 at 03:23	1,5 MB	Plain Text
social_business.txt	20 Aug 2017 at 12:26	851 KB	Plain Text
social_safety.txt	25 Sep 2017 at 03:23	1,6 MB	Plain Text
sport.txt	20 Aug 2017 at 12:23	360 KB	Plain Text
strateg_management.txt	25 Sep 2017 at 03:23	1,5 MB	Plain Text

3.2. Корпус переделан под нужды проекта путем удаления вручную часто встречающихся однотипных страниц.

Задача: Обработать полученный корпус текстов, чтобы с большей точностью предсказывать компетенции пользователей.

Решение: Вручную просмотрен каждый файл и отобраны фрагменты текста, в которых содержится информация по теме.

3.3. Записи объединены с данными пабликов от размеченных людей с LEADER-ID.

Задача: “Помочь” модели догадаться, какие данные относятся к каждому классу путем подмешивания небольшого количества размеченных данных к каждому документу в корпусе.

Решение: Вручную просмотрен каждый файл и вставлены фрагменты текста размеченных данных, в которых содержится информация по теме.

3.4. Создан алгоритм для удаления в фрагменте текста нерелевантных записей и html-подобного кода.

Задача: Очистить от ненужных данных текст, чтобы не переобучать модель.

Решение: Использовался фреймворк scikit-learn для создания и конфигурации векторизатора текстов.

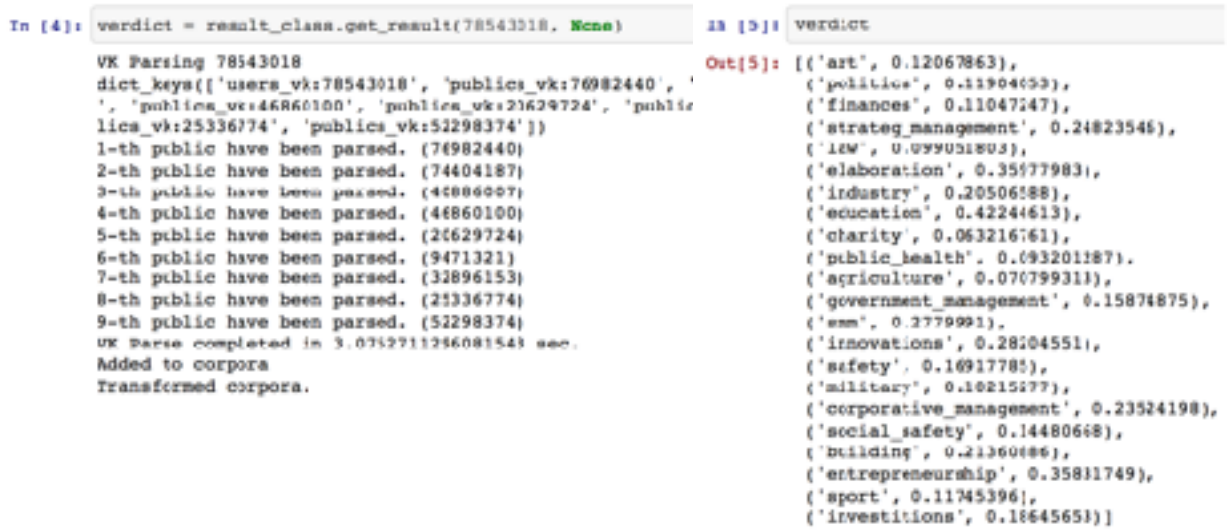
Принцип работы векторизатора:

1. Токенизация текста (избавление от пунктуации и разбитие на “токены” – части текста меньшей длины).
2. Приведение в начальную форму каждого “токена”.
3. Удаление всех токенов, не являющихся существительными, прилагательными, глаголами в н.ф., числительных и слов, длина которых менее 15 символов.
4. Проверка на русские символы с помощью регулярных выражений.
5. Векторизация с помощью TfIdfVectorizer
6. Исключение всех слов из списка 200 самых встречаемых слов, а также 5 самых редко встречаемых слов.
7. Создание спарс-матрицы (разреженную) значений TF-IDF.

3.5. Обучена модель для предсказания 22 категорий по странице VK и FB.

Задача: Обучить модель машинного обучения, чтобы на вход подавался набор текстов, а на выходе получались вероятности каждого класса.

Решение: Использовался фреймворк keras для Feed-Forward нейронной сети.

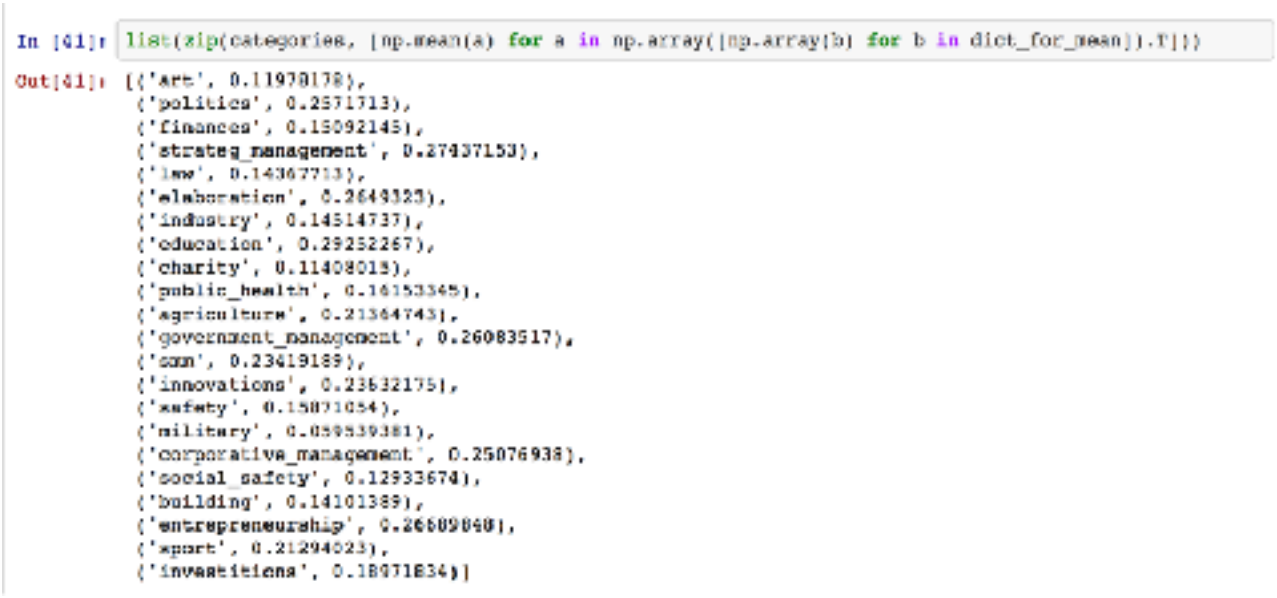


На скриншоте видно, что у данного пользователя однозначно лидируют Образование и Предпринимательство.

3.6. Произведено сравнение с классическими методами машинного обучения и определены пороги, при которых тематики выбираются системой.

Задача: Сравнить качество полученной модели путем с альтернативами, а также определить пороговые значения, по которым сервис будет отбирать 4 наилучших компетенции.

Решение: Сравнение с лог. регрессией и SVM, выгрузка средних значений из распределения ответов каждого класса.



```
In [41]: list(zip(categories, [np.mean(a) for a in np.array([np.array(b) for b in dict_for_mean]).T]))

Out[41]: [('art', 0.11970178),
          ('politics', 0.2571713),
          ('finances', 0.15092145),
          ('strateg_management', 0.27437153),
          ('law', 0.14367713),
          ('elaboration', 0.2648323),
          ('industry', 0.14514737),
          ('education', 0.29252267),
          ('charity', 0.11408013),
          ('public_health', 0.16153345),
          ('agriculture', 0.21364743),
          ('government_management', 0.26083517),
          ('sam', 0.23419189),
          ('innovations', 0.23632175),
          ('safety', 0.15871054),
          ('military', 0.059539381),
          ('corporative_management', 0.25076938),
          ('social_safety', 0.12933674),
          ('building', 0.14101389),
          ('entrepreneurship', 0.26609048),
          ('sport', 0.21294023),
          ('investitions', 0.18971834)]
```

3.7. Написана подпрограмма для загрузки данных с FB пользователей, которые не зарегистрированы в системе.

Facebook, в силу условий приватности данных, запрещает загружать публичные данные со страниц, не являющихся популярными или в странах, где это запрещено (напр. Иран).

Задача: Обойти это ограничение и использовать для страниц в России, где это не запрещено.

Решение: Использована библиотека Selenium, которая позволяет эмулировать поведение браузера.

Принцип работы:

1. Открывается браузер Google Chrome и предоставляется время для ввода логина/пароля в открывшемся сайте facebook.com
2. Скрипт производит поочередное открытие страниц вида *facebook.com/{page}*, где page – уникальный идентификатор страницы.
3. После парсинга данных страница пролистывается вниз, если это возможно, с временным интервалом.

3.8. Запущена база данных – Redis.

Задача: Хранить промежуточные результаты, а также иметь возможность анализировать результаты работы сервиса.

Решение: Запущена и настроена база данных и ее конфигурация, переписан код проекта, чтобы данные записывались в хранилище.

3.9. Написан API и документация к нему.

Задача: Объединить все модули воедино и обеспечить бесперебойную работу сервиса на удаленном сервере по HTTP.

Решение: Использовался Flask и написан API.

Принцип работы API:

1. Загрузка из файла предсказательной модели, векторизатора. Инициализация Redis.
2. Получение данных из запроса от сервера в виде строк.
3. Создание “внутреннего корпуса” текста пользователя, инициализация пустым списком.
4. Получение данных о пользовательских публичных страницах и со стены по токену с помощью открытого API ВК и библиотеки facebook-sdk для FB. Сохранение этих данных в базу данных.
5. Векторизация данных и занесение во внутренний корпус пользователя.
6. Получение вероятностей каждого класса с помощью модели.
7. Выбор 4 наиболее больших вероятности, которые превышают установленные пороги и возвращение этих результатов по протоколу HTTP.

Список методов API:

– GET /

– POST /get_result

```
{
    "name": "Georgiy", (не обязательно)
    "user_vk": VK_TOKEN or VK_ID,
    "user_fb": FB_TOKEN or FB_ID,
    "verbose": True/False (default: False)
}
```

Пример используя Python 3.6:

```
>>> requests.post("http://78.155.197.212:9999/get_result", json={"name": "Georgiy", "user_vk": 134070307}).json()
{'name': 'Georgiy', 'results': ['Юриспруденция', 'Исследования и разработки', 'Благотворительность', 'Инновации и модернизация']}
```

```
>>> requests.post("http://78.155.197.212:9999/get_result", json={"name": "Georgiy", "user_fb": "skf4lsf84fl4j309f..."}).json()
{'name': 'Georgiy', 'results': ['Исследования и разработки', 'Инновации и модернизация']}
```

3.10. Произведена виртуализация всего проекта с помощью Docker.

Задача: Разворачивание проекта на удаленном сервере с установкой всех зависимостей.

Решение: Использовался Docker.

Конфигурация Dockerfile:

1. Смена домашней директории на ~/vk_text_classifier.
2. Выполнение команд, нацеленных на установку зависимостей Python 3.6.
3. Запуск контейнера, содержащего Redis.
4. Копирование конфигурации и резервной копии данных Redis в домашнюю директорию.
5. Открытие портов 9999 для сервера и 6379 для Redis.
6. Старт Redis и асинхронный запуск сервера через Gunicorn.

3.11. Произведено тестирование API.

Задача: Провести нагрузочное тестирование и получить распределение ответов на случайных пользователях.

Решение: Выбрана контрольная группа из сотрудников АСИ, а также по 1000 пользователей ВКонтакте из некоторых регионов России. Получены значения RPS (Requests per Second) – кол-ва запросов в секунду, а также распределения ответов.

Протестировано на следующих регионах: 1. Томская область 2. Татарстан 3. Челябинская область 4. Красноярский край 5. Ульяновская область 6. Новосибирская область 7. Краснодарский край

Тестирование проходило следующим образом:

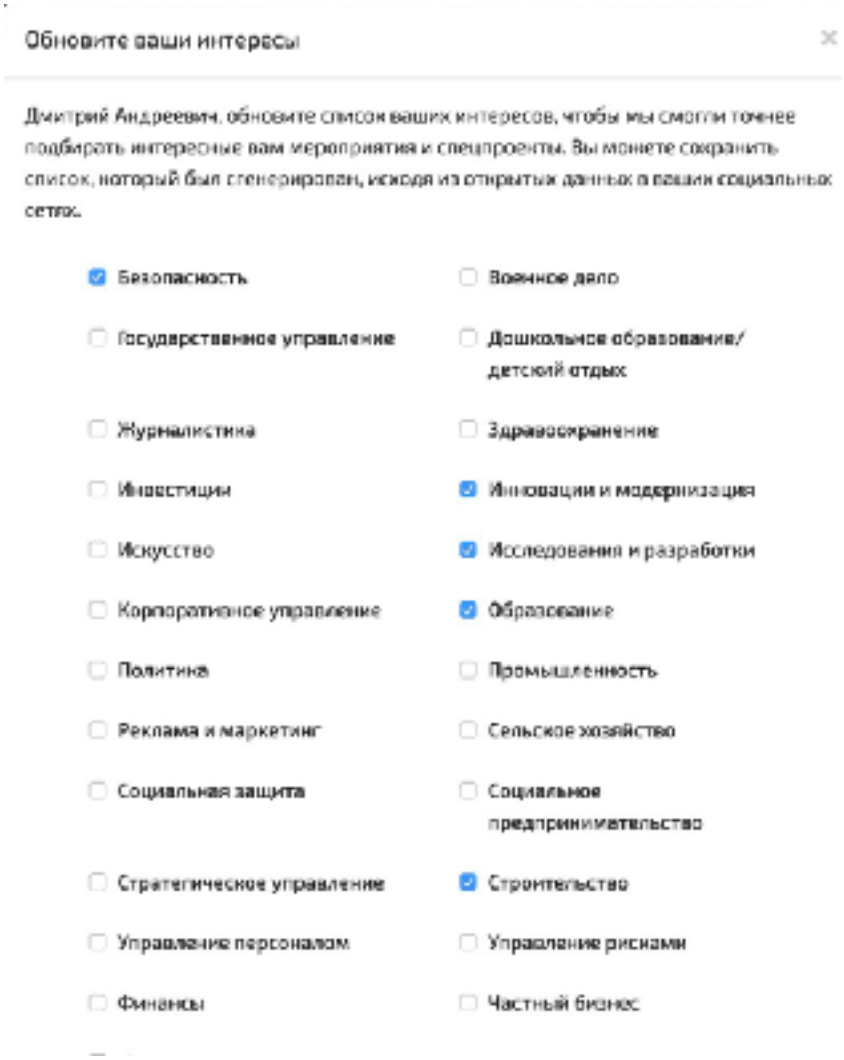
1. Нахождение id случайных пользователей ВКонтакте данных регионов, выбор случайных Facebook token.
2. Применение API и фиксация нагрузочного тестирования
3. Выгрузка данных в таблице csv.
4. Сравнение распределения ответов контрольной выборки и тестируемой.

Результаты нагрузочного тестирования: 0.3 rps (3 секунды на пользователя) для новых пользователей и 10 rps для получения данных из Redis.

3.12. Проведена интеграция сервиса.

Задача: Провести интеграцию на сайте LEADER-ID, чтобы при привязывании страницы VK, FB к аккаунту происходил вызов API и получение результатов в виде всплывающего окна, предлагающего выбрать компетенции при показывающихся предсказанных.

Решение: Произведена интеграция.



4. Системное администрирование и техническая поддержка

На протяжении всего периода действия контракта исполнителем осуществлялось работы по поддержке инфраструктуры проекта в рабочем состоянии.

4.1. Техническая поддержка

Исполнителем осуществлялась техническая поддержка сервиса, в том числе:

- Работы по созданию нового функционала и модернизации существующего
- Исправление найденных проблем
- Профилактические работы
- Консультационные работы

Итоговая структура проекта

vk_text_classifier/
├── Dockerfile — конфигурационный файл для проекта Docker
├── LICENSE — файл лицензии для Github
├── README.md — файл README для Github с описанием работы API
├── api.py — файл с API
├── assets/
│ ├── margins.json — записанные границы ответов
│ ├── vectorizer.p — векторизатор текста
│ └── vk_texts_classifier.h5 — обученная модель
├── docker-compose.yml — конфигурационный файл для docker-compose
├── notebooks/
│ ├── Parser.ipynb — Jupyter Notebook файл для скачивания из FB, если нужно загрузить без API.
│ ├── Task.ipynb — Jupyter Notebook файл с обучением модели и подготовкой данных.
│ ├── config.py — личные токены к ВК и Facebook, на случай загрузки без сторонних токенов.
│ ├── corpora/ — папка с подобранными вручную txt документами для обучения модели
│ ├── labels.p — названия категорий
│ └── social.xlsx — размеченные данные с LEADER-ID
├── redis.conf — конфигурационный файл для Redis
├── requirements.txt — зависимости для Python pip
└── util.py — основные скрипты для работы API