Отчет

Задачей преддипломной практической работы стоит создание пользовательского веб-сервиса для построения маршрутов. В качестве реализации данной задачи необходимо позволить пользователю выбрать на карте две произвольные точки, которые в свою очередь будут являться координатами начала и конца маршрута. Кроме того, пользователь должен иметь возможность увидеть маршрут между этими пунктами, который был подобран нейронной сетью.

Построенная нейронная сеть заранее прогнозирует временные ряды для размеченных путей на городской транспортной сети. Исходя из данных прогнозов путем поиска на графе, состоящем из точек, принадлежащих прямоугольной области карты, в которой содержатся обе точки маршрута, строится маршрут, оптимальный по среднему спрогнозированному времени.

Выбранный для реализации язык программирования – Python. Для удобства взаимодействия в качестве сервера используется CGI-скрипт. Для визуализации подобранных маршрутов применяется Python библиотека “folium”, посредством которой создаются .html файлы с картой (OpenStreetMap) и произвольным содержимым. Данная библиотека требует работы с географическими координатами для построения маршрутов, следовательно, необходимо получать точные данные широты и долготы для указанных пользователем точек.

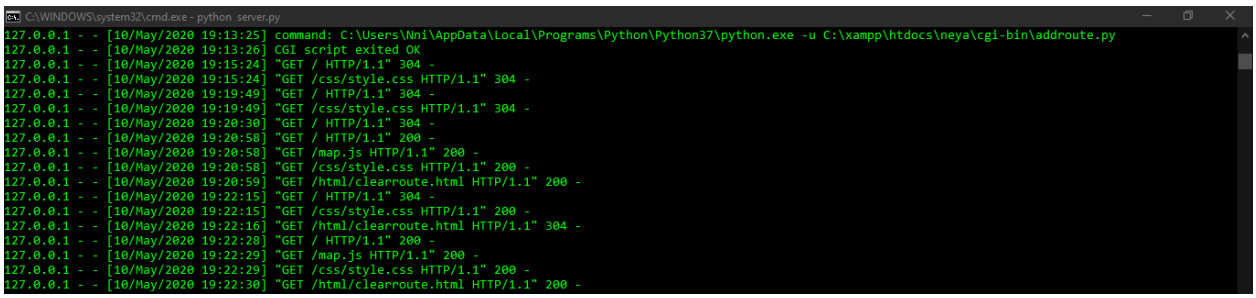

Рисунок 1 - лог сервера CGI.

Трудности в реализации данного этапа сервиса заключаются в использовании геокодера. Бесплатных и при этом удобных сервисов, с помощью которых можно реализовать онлайн геокодирование как таковых нет. Однако, среди существующих был использован API Яндекс карт, который предоставляет зарегистрированному разработчику возможность использовать скриптовый интерфейс Яндекс карт для встраивания в сторонний веб-сервис (конечно, у бесплатной версии имеется определенный ряд ограничений, невыполнение любого из которых влечет к блокировке ключа доступа для разработчика).

Таким образом, с помощью API Яндекс карт имеется возможность использования геокодера и самих Яндекс карт, встроенных в сервис для указания точек маршрута. На части клиента работает JavaScript, в котором при указании пользователем обеих конечных точек маршрута происходит следующее:

- отправляется запрос геокодеру для получения координат точек,

- полученных координаты через ajax запрос методом GET происходит обращение к .py файлу, который загружает координаты в базу данных (база данных стоит на PostgreSQL),

- полученные координаты также записываются в соответствующие четыре <input> на странице сервиса для наглядности установки и изменения самих данных.

Благодаря использованию JavaScript сам .html документ остается относительно небольшим и простым в понимании. Преимущества JavaScript заключаются в том, что его поддерживает любой современный интернет браузер, а также тот факт, что огромное количество действий и функций можно выполнить на стороне клиента без обновления самой просматриваемой страницы (в том числе запросы к серверу).

Рисунок 2 - код страницы веб сервиса.

Все основные функциональные элементы выполняются через JavaScript функции, находящиеся в файле map.js (там же содержится ajax запрос на сохранение географических координат в базе данных).

Для полноценного функционирования веб-сервиса в качестве многопользовательского сайта необходимо использовать авторизацию клиентов сервиса. В соответствии с регистрационным кодом пользователя на сервере сохраняются следующие данные:

- координатные данные точек последнего запрошенного маршрута (или ничего, если клиент не использовал функцию геокодирования ни разу),

- сгенерированный .html документ, содержащий последний пользовательский маршрут (или ничего, если клиент не строил маршрут с помощью нейронной сети),

- файлы логов сессии каждого пользователя (кроме данных самих запросов).

На странице сервиса представлено разделение Яндекс карт и OpenStreetMap карт для наглядности разницы в алгоритмах расчета пути (если разница в самом маршруте будет присутствовать).

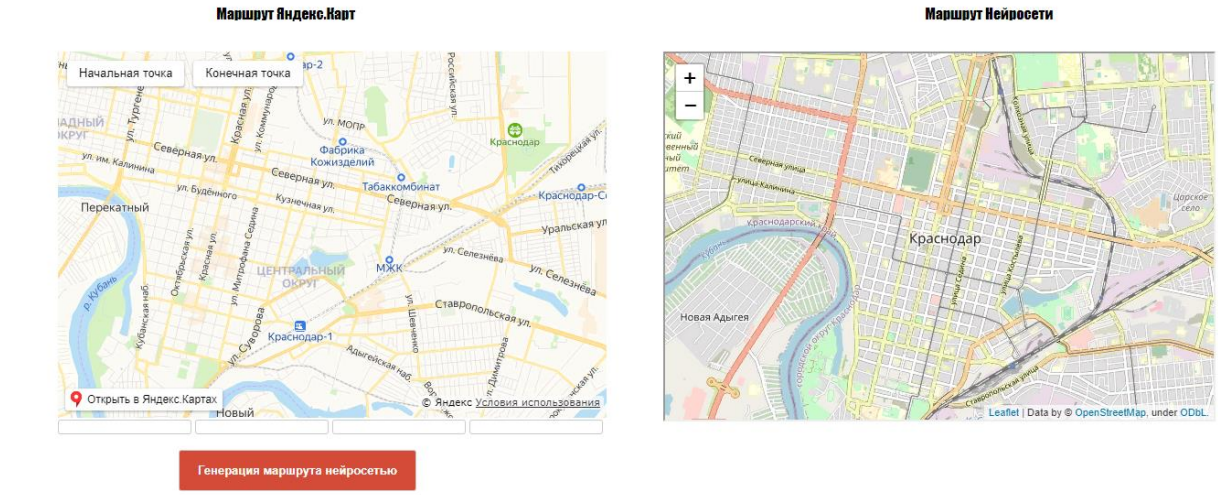

Рисунок 3 - index.html (главная) страница сайта.

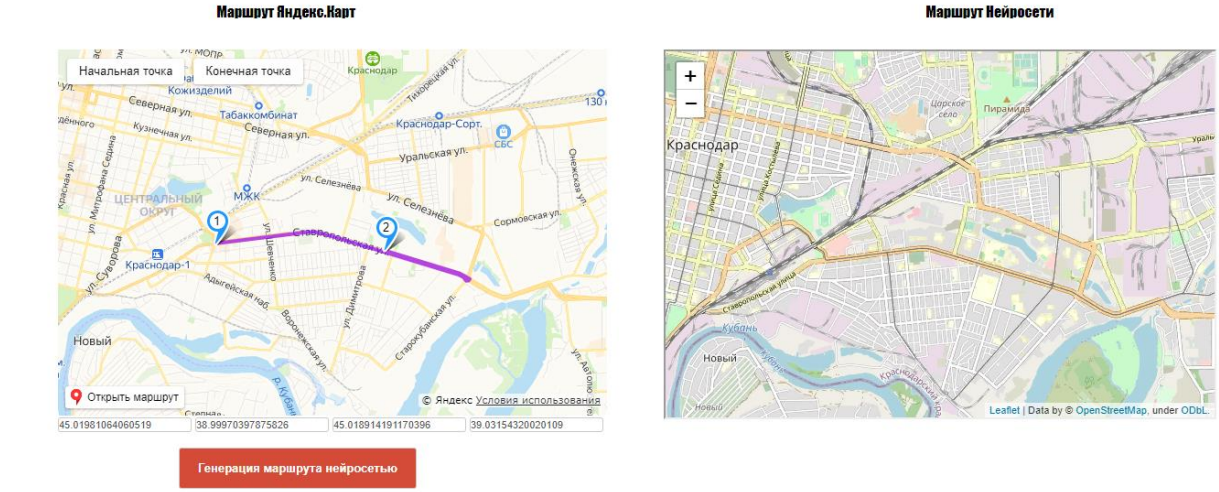

Рисунок 4 - пользователь указал путевые точки маршрута.

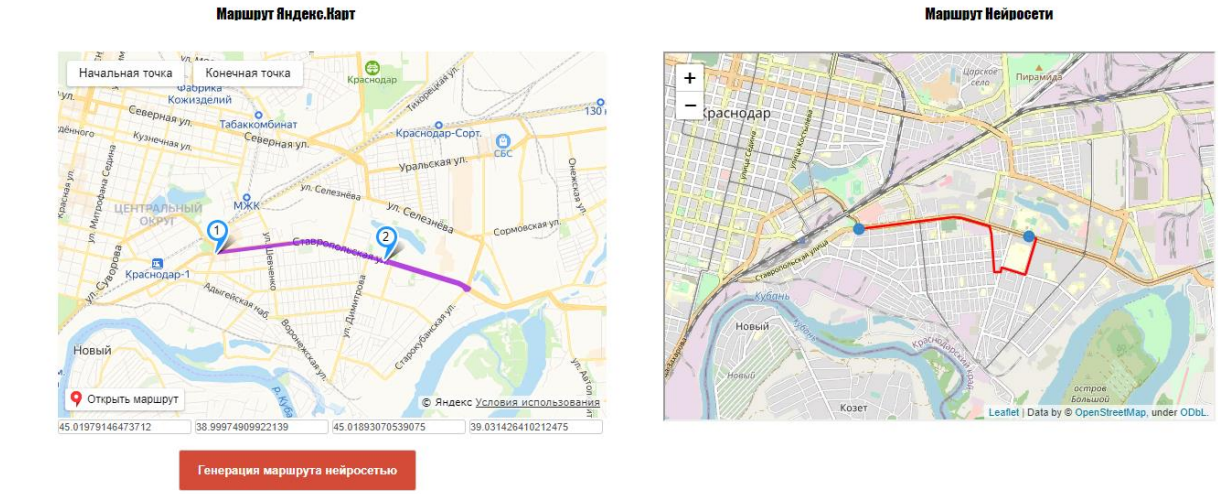

Рисунок 5 - пользователь запросил генерацию маршрута нейросетью.

Как видно на рисунках 4 и 5 маршрут, который ищет пользователь может отличаться, поскольку в случае с прогнозируемыми рядами нейронной сеть получаемые на выходе данные являются ассоциативно-прогнозирующими. Таким образом простроенные маршруты должны отличаться по времени прохождения, при том определить какой из них будет оптимальнее практически невозможно, поскольку один и тот же пользователь проезжает один и тот же участок пути с определенной разницей. Чтобы достоверно убедиться в преимуществе одного пути прохождения точек перед другим, требуется совершить несколько сотен сравнительных испытаний (что не является возможным без подходящего оборудования и тем более в условиях карантинного режима).

Документ (.html) с пользовательским маршрутом строится с помощью заранее прописанных Python функций генерации пустой карты и добавлении на нее маршрута в виде последовательных точек, соединенных прямыми линиями.

```
folium.Map(location = [45.035470, 38.975313],
      zoom_start = 13,
      tiles = "OpenStreetMap")
```

В модульном файле lib.py хранятся основные функции и константы для работы области генерации карт. Благодаря возможности CGI скриптов работать с .py файлами, использование ajax запросов сводится к обращению методом GET к конкретным файлам на сервере. Пример ajax запроса:

```
$.ajax({
      method: "GET",
      url: "cgi-bin/createuserroute.py”,
      data: {
      user_id: session_id,
      route_data: "last_encoding"
      },
})
```

```
        fill_opacity = 0.9).add_to(route_map)

def AddRouteToDB(filename, connection, routename, version):
    file = open(filename, "r")
    lines = file.readlines()
    jsonFile = json.loads(lines[0])
    coordinates = jsonFile["coordinates"]

    with closing(connection.cursor()) as cursor:
        cursor.execute("INSERT INTO diplom.routes(routename, routetypeid, version) "
                       +"VALUES('"+routename+"',4,"+str(version)+")")
        connection.commit()
        cursor.execute("SELECT id FROM diplom.routes WHERE routename = '"+routename+"' AND version = "+str(version)+"")
        for row in cursor:
            routeid = row[0]
        pointnumber = 1
        pointid = 0
        for point in coordinates:
            cursor.execute("SELECT id, lat, lon FROM diplom.points "
                +"WHERE ABS(lat - "+str(point[1])+") <= "+str(LAT_10M)+" "
                +"AND ABS(lon - "+str(point[0])+") <= "+str(LON_10M)+" "
                +"AND ismain = FALSE")
            ids = []; lats = []; lons = []
            for row in cursor:
                ids += [row[0]]; lats += [row[1]]; lons += [row[2]]
            if(len(ids) == 0):
                cursor.execute("INSERT INTO diplom.points(lon, lat, ismain) VALUES("+str(point[0])+","+str(point[1])+",false)")
                connection.commit()
                cursor.execute("SELECT id FROM diplom.points WHERE lon = "+str(point[0])+" AND lat = "+str(point[1])+"")
                for row in cursor:
                    pointid = row[0]
                cursor.execute("INSERT INTO diplom.routepoints(routeid, pointid, pointnumber) "
                    +"VALUES("+str(routeid)+","+str(pointid)+","+str(pointnumber)+")")
                connection.commit()
                pointnumber += 1
            elif(len(ids) == 1):
                cursor.execute("INSERT INTO diplom.routepoints(routeid, pointid, pointnumber) "
                    +"VALUES("+str(routeid)+","+str(ids[0])+","+str(pointnumber)+")")
                connection.commit()
                pointnumber += 1
            else:
                nearest_point_id = ids[0]
```

Рисунок 6 - (lib.py) файл с основными функциями и константами для работы с географическими данными и генерации карт для сервиса.

После первичного тестирования приложения возникает необходимость проверить данные на соответствие реальным. Однако, в условиях введенного карантинного режима испытать сгенерированные маршруты не представляется возможным. Аналогичная проблема возникла на этапе предварительной разметки городской транспортной сети, фиксации путей и сборе реальных данных для обучающей выборки. Данных, собранных до введения карантина посредством прямого замера времени прохода участков маршрута, недостаточно, чтобы эффективно обучить нейронную сеть для прогнозирования временных рядов ожидаемого среднего времени прохождения маршрута (и отдельных участков пути).

Исходя из текущих ограничений большая часть обучающих выборок составлена исходя из предполагаемого времени, рассчитываемого сервисом Google карт и Яндекс карт. На основе исторических данных о дорожной ситуации собранных с этих двух сервисов берется ожидаемое значение для каждого размеченного участка пути. Разделение сезонности нагрузки на транспортную сеть, а также дифференитности нагрузки по дням недели идет следующим образом:

- месяц в году (1-12),
- неделя в году (1-53),

- день недели (1-7),

- час суток (0-23),

- минута в часе (0-59).

По варируемости сезонной нагрузке на определенные участки городских дорог можно судить при наличии большого количества выборочных данных, причем за продолжительное время (больше двух лет). Поскольку даже для одного размеченного пути на транспортной сети разметить такое количество данных вручную займет огромное время, то для решения данной проблемы следует иметь отдельный сервис, занимающийся сбором реальных данных (или данных, приближенных к реальным). Для этого хорошо подошла бы система обратной связи с пользователем, в случае, когда тот действительно прошел построенный маршрут (или участок пути).

Что касается нейросетевой системы прогнозирования временных рядов для участков транспортной сети, то здесь необходимо проводить все вычисления заранее, желательно за сутки и раньше, чтобы в случае необходимости, данные можно было откорректировать вручную или выполнить процесс прогнозирования снова. Однако, в случае постоянного расширения транспортной сети и охватываемых районов, то количество путей для прогнозирования будет пропорционально расти. Из этого следуют затруднения в вычислениях, поскольку количество необходимых данных для прогнозирования может превышать вычислительную мощность одного сервера. В таком случае необходимо сделать что-либо из следующего:

А) увеличить количество серверов и распределить между ними расчёты,

Б) сменить целевой язык программирования на один из более быстрых, поддерживающих распараллеливание вычислений по ядрам (лучший вариант это C++), однако в таком случае придется воссоздавать нейросетевой фреймворк и проводить распараллеливание блоков вычислений.

Первый подход весьма затратный поскольку необходимо иметь не только производительный, но и устойчивый к отказам (от огромного количества запросов) сервер. Второй подход потребует меньше материальных ресурсов, однако намного более затратный по времени реализации. Помимо

смены языка, присутствует возможность смены основы сервера с CGI скрипта на аналогичный HTTP протокол, но с существенным приростом в скорости.

Сама нейронная сеть реализована с помощью библиотек “keras” и “tensorflow”. Учитывая специфику задачи, архитектура модели – компиляция нейронной сети прямого распространения и разряженного автокодировщика.

Рисунок 7 - архитектура нейросети прямого распространения.

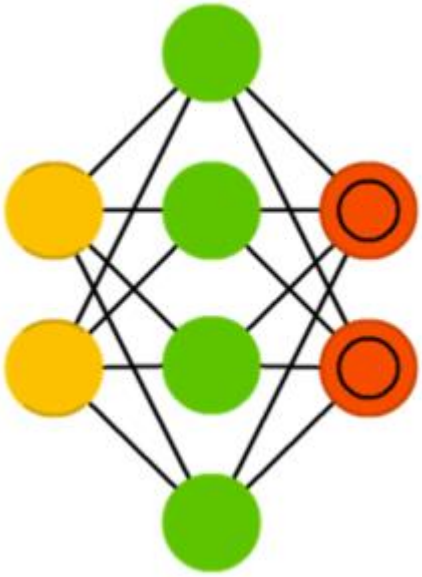

Рисунок 8 - архитектура разряженного автокодировщика.

Данная система содержит два скрытых слоя (в определенных случаях больше), а размер входного слоя обычно превышает выходной. Размер первого скрытого слоя превышает входной слой не менее чем в 1.25 раза, а второй (и дальнейшие) скрытый слой имеет нейронов столько, чтобы выполнялось правило:

$$size(h_1) > size(h_1) > size(o),$$

где h – скрытый слой, $_{1,2}$ – индекс скрытого слоя, o – выходной слой, size(x) – функция, возвращающая размерность слоя нейронной сети.

В общем случае правило выглядит так:

$$size(h_1) > size(h_2) \geq \cdots \geq size(h_n) > size(o),$$

где h – скрытый слой, $_{1,2\ldots n}$ – индекс скрытого слоя, o – выходной слой, size(x) – функция, возвращающая размерность слоя нейронной сети.

Благодаря не чрезмерной разрежённости первого скрытого слоя, нейронная сеть принимает в расчёт как более общие обобщения данных, так и более частные (мелкие факторы ассоциации). Общий алгоритм прохода нейронной сети можно описать пунктами:

- подача интерпретированных данных на вход сети,

- гиперболизированной взвешивание сигналов (функция relu отсеивает все отрицательные значения), как следствие, первый скрытый слой сети обычно имеет на порядок больше активных сигналов, чем входной слой, однако, учитывая скорость роста степени двойки, растет и количество вариантов (не считая машинных дробных вариаций) наборов на скрытом слое,

- первый скрытый слой начинает представлять из себя преобразованный набор входных данных, у которых отсеиваются менее ценные общности и добавляются неявные ассоциации,

- последующие слои выступают аналогом сверточных слоев, где от большего количества данных переходят к меньшему, все ближе подходя к временному ряду,

- на выходной слой поступают сигналы, анализ которых напрямую говорит о спрогнозированном среднем времени прохождения конкретного участка маршрута (при условии, что нейросеть была обучена достаточно хорошо).

Таким образом, результатами преддипломной практической работы являются создание прототипа пользовательского веб-сервиса для визуализации спрогнозированных и построенных маршрутов на транспортной сети (определенной ее области) города Краснодар, анализ взаимодействия частей сервера и клиента, а также необходимые средства, компоненты и сервисы для преобразования прототипа данной программы в полноценный клиентский сервис для построения и сравнения маршрутов. В рамках практической работы все поставленные цели были выполнены в полной мере.