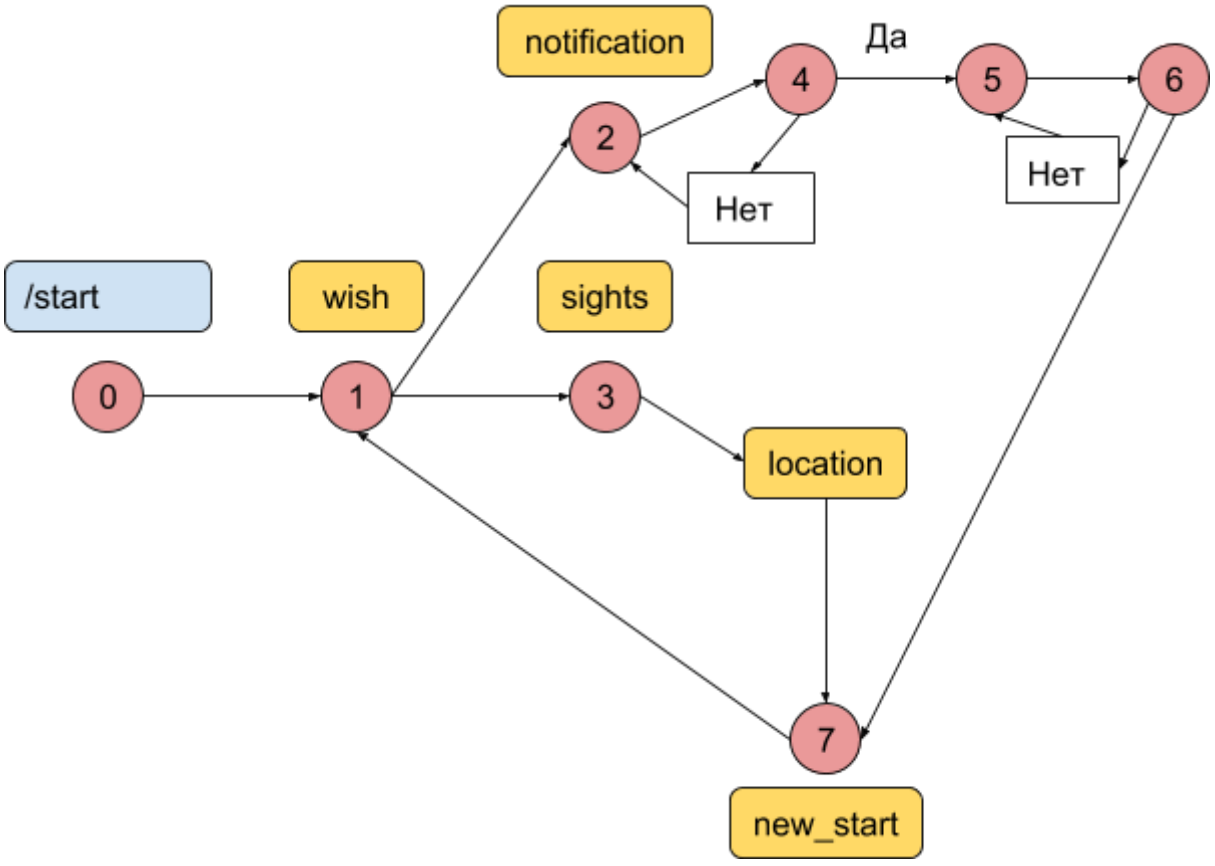
Отчет

Для чего нужен бот:

- Определение достопримечательностей
- Определение местоположения
- Проложение маршрута
- Уведомления о пересадках

Логин:
@pirozhenka_bot

Описание состояний:



0 - начальное состояние. Когда пользователь вводит команду /start, бот спрашивает: "Привет, меня зовут Сережа! Чем могу быть полезен? Я умею включать уведомления и показывать крутые места)". Переходит в состояние 1.

1 - в этом состоянии бот определяет, что хочет пользователь с помощью функции near_request. Данная функция возвращает 1, 2 или 3. Если 2, спрашивает: "Отлично! На какую станцию планируете поехать?" и переходит в состояние 2. Если 1 - спрашивает: "Отлично! Показать места рядом с Вами или около конкретной станции метро?" и переходит в состояние 3. Если 3 - он ничего не понял, и снова возвращает в состояние 0.

2 - В этом состоянии бот определяет название станции, на которую планирует поехать пользователь с помощью функции define_station. Далее спрашивает, верно ли он определил. Переходит в состояние 4.

4 - Если на последний вопрос состояния 2, пользователь ответил: “Да”, бот спрашивает: "Замечательно!Укажите Ваши пожелания по маршруту(пересадки)" и переходит в состояние 5. Если “Нет”, просит указать место прибытия еще раз и возвращается в состояние 2. Если отвечает что-то несвязное, то бот просит написать только да или нет.

5 - В этом состоянии бот определяет станции, на которых пользователь хочет сделать пересадки с помощью функции changes. Спрашивает, правильно ли он понял пользователя и отправляет в состояние 6. Если в качестве пересадок был возвращен пустой список, бот просит указать еще раз.

6 - Если пользователь на последний вопрос состояния 5 отвечает: ”Да”, бот пишет, что уведомления включены и переходит в состояние 7. Если ”Нет”, просит указать еще раз и возвращает в состояние 5. Если пользователь написал что-то несвязное, просит написать еще раз.

3 - В этом состоянии появляется кнопка и бот предлагает пользователю отправить информацию о местоположении, нажав на эту кнопку. Переходит в состояние ‘location’.

‘location’ - В этом состоянии удаляется кнопка, а также с помощью функции near_metrostation определяется ближайшая станция метро. Переходит в состояние 7.

7 - Это псевдоначальное состояние, в котором не нужно снова вводить команду /start. Бот снова спрашивает пользователя: "Чем еще я могу Вам помочь? Напомню: Я умею включать уведомления и показывать крутые места)". Переход в состояние 1.

Обработка состояний

Для состояний используется однофайловая база данных Vedis, в которой хранятся пары значений (key, value), где key - id пользователя, value - состояние пользователя. Это нужно, чтобы при сбоях Telegram, бот не начинал диалог сначала, а начинал с текущего состояния.

В данном проекте обрабатываются:

- номера состояний (нужны для реализации конечного автомата)
- команды /start, /reset
- location (нужен для конечного автомата, а также для того, чтобы принимать от пользователя данные о его местоположении.)

Команда /reset возвращает пользователя в состояние 0, но ему уже не нужно будет вводить команду /start.

Что используют дополнительные функции

near_request - в данной функции для определения пожеланий пользователя используется модель word2vec, которая была скачана с интернета, а также функция wmdistance. Есть два класса, которые содержат несколько фраз, характеризующих пожелания. Когда на вход поступает сообщение пользователя, к нему применяется .lower().split(). А дальше каждое слово проверяется. Если оно принадлежит к стоп словам, то не включается в конечные список. Дальше применяется функция wmdistance, которая применяется по очереди к полученному списку слов и каждому предложению из класса. Эти расстояния суммируются отдельно для каждого класса и сравниваются. Если модуль их разницы отличается меньше, чем на epsilon = 0.001, или одна из сумм равняется бесконечности, тогда возвращается 3. Если нет, тогда выводится номер класса, соответствующий данному суммарному расстоянию.

define_station и **changes** - данные функции используют класс, в котором есть названия всех станций метро. Изначально из входящего сообщения удаляются все стоп - слова, а дальше сравниваются расстояния до названий всех станций. Выбирается наименьшее и возвращается название соответствующей станции.

near_metrostation - данная функция использует файл ‘metro.csv’, в котором есть названия станций метро, их id, номера линий, координаты, а также списки станций, на которые есть переход. Так как на вход подаются координаты пользователя, можно вычислить расстояние до каждой станции с помощью great_circle из geopy.distance.

Проблемы:

1. Яндекс.карты API платное.
2. Не совсем понятно, как периодически запрашивать местоположение, так как вариант с кнопкой не совсем удобный.
3. Из-за того, что API платное, не получилось получать достопримечательности.
4. Уведомления не получилось сделать, потому что они привязаны к определениею местоположения.