ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК: 004.8

Отчет об исследовательском проекте

на тему: Голосовой ассистент Алина

Выполнил:
Студент группы БПМИ193 _____ А.О.Голубь
Подпись И.О.Фамилия
24.05.2021
Дата

Принял:
Руководитель проекта Денис Андреевич Симагин
Имя, Отчество, Фамилия
разработчик / ООО «Яндекс.Технологии»
Должность / Место работы

Дата проверки | Оценка (по 10-ти бальной шкале) | Подпись

Москва 2021

Аннотация

В данной работе рассматриваются современные походы к созданию wake-up word детектора, разрабатывается устройство для голосового управления «умным» домом с использованием современных IoT технологий на основе мини компьютера Raspberry Pi.

Содержание

1 Основные термины и определения

1. Нейронная сеть – математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.
2. Машинное обучение — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач.
3. Raspberry Pi – одноплатный компьютер размером с банковскую карту, изначально разработанный как бюджетная система для обучения информатике, но позже получивший более широкое применение и известность.
4. Text To Speech – Синтез речи – формирование речевого сигнала по печатному тексту
5. Облачные вычисления – модель обеспечения удобного сетевого доступа по требованию к некоторому общему фонду конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам — как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру
6. Wake Up Word Detector – фоновый процесс, который активирует устройство, если услышит ключевую фразу, например, "Алина"
7. Голосовой ассистент – программный агент, который может выполнять задачи для пользователя на основе информации, введенной пользователем или полученной из различных интернет-ресурсов.
8. Датасет – это обработанная и структурированная информация в табличном виде.
9. IoT(internet of things) – концепция сети передачи данных между физическими объектами(«вещами»), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой
10. Функция активации – функция нейрона, определяющая выходной сигнал, который определяется входным сигналом или набором входных сигналов
11. PyTorch – фреймворк машинного обучения для языка Python с открытым исходным кодом, созданный на базе Torch
12. Свёрточная нейронная сеть – специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения.
13. gRPC(Remote Procedure Calls) – это система удалённого вызова процедур с открытым исходным кодом, первоначально разработанная в Google в 2015 году.

2 Введение

Голосовой ассистент – устройство на основе искуственного интеллекта, которое способно понимать человеческую речь и выполнять определенные действия в ответ на голосовую команду. В последнее десятилетие мы видим бурное развитие головых ассистентов, они встроены почти в каждый смартфон. Современные помощники умеют создавать напоминания, списки дел, ставить будильники, таймеры, включать и выключать свет, зачитывать новости, показывать фильмы, воспроизводить музыку, подкасты, вызывать такси, заказывать доставку и даже записывать вас на прием к парикмахеру.

В работе голосовых ассистентов есть множетсво нюансов - начиная с отделения голоса человека от шума телевизора и заканчивая определением местоположения для вызова такси.

Для того, чтобы лучше разобраться в теме, обратимся к истории создания устройст распознавания речи:

- Впервые устройство, отдаленно похожее на современных помощников было, разработано в 1952 году в лаборатории Bell Labs [6] и имело название "Audrey". Это была машина размером с целую комнату, она потребляла уйму энергии и умела распознавать цифры от 0 до 9, сказанные одним человеком – ее создателем.

- На всемирной выставке в 1962 году компания IBM продемонстрировала устройство под назавнием "Shoebox", способное понимать 16 разговорных английских слов. Примерно в то же время был разработан алгоритм динамической трансформации временной шкалы(DWT) [4], который был способен работать со словарным запасом уже в 200 слов. Но все эти системы были основаны на сопоставлении шаблонов, где отдельные слова сопоставляются с сохраненными голосовыми образцами.

- В 2010 году компания Google, благодаря облачным вычислениям и нейронным сетям, смогла добавить полноценное распознование голоса в смартфоны на операционной системе Android. В 2011 году компания Apple представила своего голосового помощника "Siri", а в 2017 году компания Яндекс представил "Алису", полноценно работающую на русском языке.

В настоящее время все умные помощники основаны на нейронных сетей, которые работают не на устройстве пользователя, а в облаках, так как требуют больших вычислительных мощностей. В своем проекте я хочу разобраться в современных методах для распознавания и генерации речи, попрбовать обьединить их вместе на мини-компьютере Raspberry Pi.

Цели и задачи

Цель моего проекта: сделать свое ”умное” устройство на базе Raspberry Pi 3B+, способное эффективно распознавать голосовые команды и обрабатывать их в режиме реального времени.

Для достижения цели были поставлены следующие **задачи:**

1. Изучить основы машинного обучения
2. Изучить современные решения в области распознавания звуков
3. Подготовить датасет
4. Создать свой wake-up word detector
5. Реализовать распознавание и/или генерацию речи, при необходимости, с использованием облачных вычислений
6. Добавить «умные» навыки в голосового ассистента

3 Обзор и сравнительный анализ

В настоящее время существует несколько подходов к распознаванию key word в режиме реального времени на слабом железе.

Один из подходов заключается в использовании сверточных нейронных сетей. Рассмотрим этот подход на примере TC-ResNet [2]. Авторы статьи используют идею skip connections из классической ResNet [3], но вместо 2D сверток используют одномерные временные свертки, которые позволяют увеличить точность и уменьшить размер сети, а значит уменьшить и время работы нейросети на слабых устройствах. Среди плюсов этого подхода стоит отметить очень высокое качество. Однако, при использовании сверточных нейросетей в режиме реального времени возникают ощутимые технические проблемы с необходимостью кэшировать промежуточные вычисления модели для избавления от задержек, что усложняет реализацию. Помимо этого, как выяснилось в процессе тестирования, модель получается достаточно большая и плохо учится.

Второй подход, который часто используется для задачи keyword spotting – рекурентные нейронные сети, а точнее их модификации – LSTM [5] и GRU [1]. Главное преимущество рекурентных нейросетей заключается в том, что модели получаются очень легкими и они очень хорошо работают на потоке данных в реальном времени, так как данные в модель подаются последовательно, а она возвращает предсказание и внутреннее состояние, которое передается в модель на следующем шаге. Таким образом, модель имеет некоторую память и может делать предсказание по совокупности факторов за небольшой временной промежуток. Из недостатков можно отметить то, что качество распознавания получается немного хуже, чем у сверточных аналогов.

4 Подробное описание выбранных методов и алгоритмов

4.1 Модель

В качестве основы для своей архитектуры я выбрал Gated Recurrent Unit(далее GRU [1]) с дополнительными линейными слоями. Исключительная черта модели с GRU, которая и стала основным критерием при выборе, это то, что она была создана для работы на потоке непрерывных данных и хорошо подходит для задачи обработки звука. Дополнительным плюсом является то, что GRU содержит не очень много весов, а поэтому должна хорошо работать на слабом железе.

Остановимся немного подробнее на архитектуре:

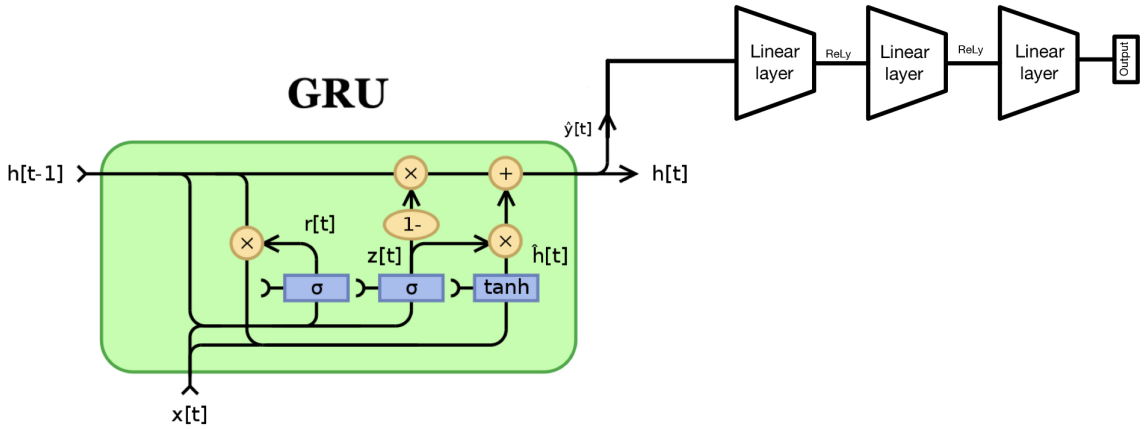

Рисунок 1 – Арихитектура модели

Модель на вход принимает вектор признаков $x[t]$ и свое предыдущее состояние $h[t-1]$. Затем идет блок GRU, который вычисляет вектор предсказаний $\hat{y}_t$ и новое скрытое состояние $h[t]$, которые передается сети на следующей итерации, по следующему правилу:

$$z_t = \sigma\left(W_z x_t + U h_{t-1} + b_z\right)$$

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right)$$

$$\hat{y}_t = \tanh\left(W_h x_t + U_h(r_t * h_{t-1}) + b_h\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t$$

где U и W – матрицы параметров, а $*$ обозначает поэлементное произведение матриц.

4.2 Датасет и обработка данных

Модель мы выбрали, теперь необходимо найти данные, на которых мы будем ее обучать. Мы поступили следующим образом – каждый из участников проекта должен был сделать 5 разных записей:

1. 5 минутная запись в идеальных услових, содержащая только ключевые слова;
2. 5 минтуная запись в шумных условиях(шум дороги/телевизора/...), содержащая только ключевые слова;

3. 5 минутная запись в идеальных услових, содержащая слова, похожие на ключевое('Алиса', 'Долина', ...);
4. 5 минутная запись в шумных услових, содержащая слова, похожие на ключевое;
5. 25 минутная запись случайных разговоров, не содержащая ключевых слов;

После этого датасет необходимо было разметить, то есть поставить метки там, где заканчивалось слово «Алина». Качество автоматической разметки, разметки различными скриптами, оказалось неудовлетворительным, получалось достаточно много ошибок и неправильной разметки. Поэтому весь датасет был размечен вручную.
У нас получился размеченный датасет, содержащий:

- 4245 слов «Алина»
- 9747 других слов и словосочетаний

4.3 Процесс обучения

Перед тем как подавать данные из датасета на вход модели, их необходимо обработать и выделить признаки. Для этого я считаю Мел-частотные кепстральные коэффициенты(mfcc), которые часто используются в качестве характеристики речевых сигналов. Подобные единицы измерения часто применяются при решении задач распознавания, так как они позволяют приблизиться к механизмам человеческого восприятия, которое лидирует среди известных систем распознавания речи. Работает это следующим образом:

- Сначала считаем спектрограмму - то есть зависимость спектральной плотности мощности сигнала от времени, где спектральной плотностью мощности, в свою очередь, называют распределение мощности сигнала в зависимости от частоты.
- Теперь этот спектр необходимо разложить по мел-шкалам - это такие шкалы, которые характеризуют то, насколько человек хорошо отличает соседние частоты в каком-то интервале. Например, разницу между 5000 Гц и 5050 Гц человек почти не услышит, зато он прекрасно отличит 10 Гц от 60 Гц.
- Пусть мы зафиксировали N – количество мел-шкал, на которые хотим разложить спектр, тогда на выходе мы получим N признаков, которые можно отдавать на вход нейронной сети. В моей реализации $N = 40$.

Дальнейший процесс обучения устроем следующим образом:

- В начале каждой эпохи я перемешивал весь датасет, чтобы модель не запомнила порядок правильных ответов;
- В начале каждого шага обнулял внутреннее состояние GRU;
- Выбирал из датасета 5 случайных слов так, чтобы в среднем отношение позитивов к негативам было равно константе $1 : 2$, склеивал их вместе, считал mfcc, нормализовал вектор признаков и подавал эти слова на вход нейросети. Константное отношение позитивов к негативам позволяет равномернее обучать сеть и точнее настраивать сам процесс обучения. Например, это помогает избавиться от ситуации, когда модель получает на вход слишком много негативов подряд и начинает для каждого входа предсказыать, что слово не является ключевым, так как это минимизирует ее функцию потерь в конкретный момент времени. Такой подход сильно уменьшил время обучения и заметно увеличил качество;

- Важно не обнулять внутреннее состояние GRU между словами внутри блока из пяти слов, описанного выше. Это позвляет обучить модель работать с данными переменной длины, что мы и хотим для потокового распознавания;
- Эпоха заканчивается в тот момент, когда модель увидела либо все негативы, либы все позитивы

Весь процесс обучения занимает около 20 эпох

4.4 Оценка качества модели

Для оценки качества модели я использовал метрики precision и recall - точность и полнота.

- Точность – доля документов действительно принадлежащих данному классу относительно всех документов, которые модель отнесла к этому классу.
- Полнота – доля найденных моделью документов в классе относительно всех документов этого класса в тестовой выборке.

На выходе моя модель выдает 2 числа – вероятность того, что слово принадлежит к классу негативов и классу позитивов. Нам интересна только вероятность, что слово принадлежит к классу позитивов. Для достижения наилучшего качества можно менять порог срабатывания – то есть начиная с какого значения будем считать, что слово относится к классу позитивов. Понятно, что метрики полноты и точности зависят друг от друга, поэтому хочется использовать метрику, которая обьединяет в себе информацию о полноте и точности.
Такой метрикой является $F-$мера – гармоническое среднее между полнотой и точностью:

$$F = 2 \cdot \frac{precision \times recall}{precision + recall}$$

Ее и буду максимизировать при поиске оптимального порога срабатывания
Для обученной модели получил следующий график зависимости метрик от порога срабатывания:

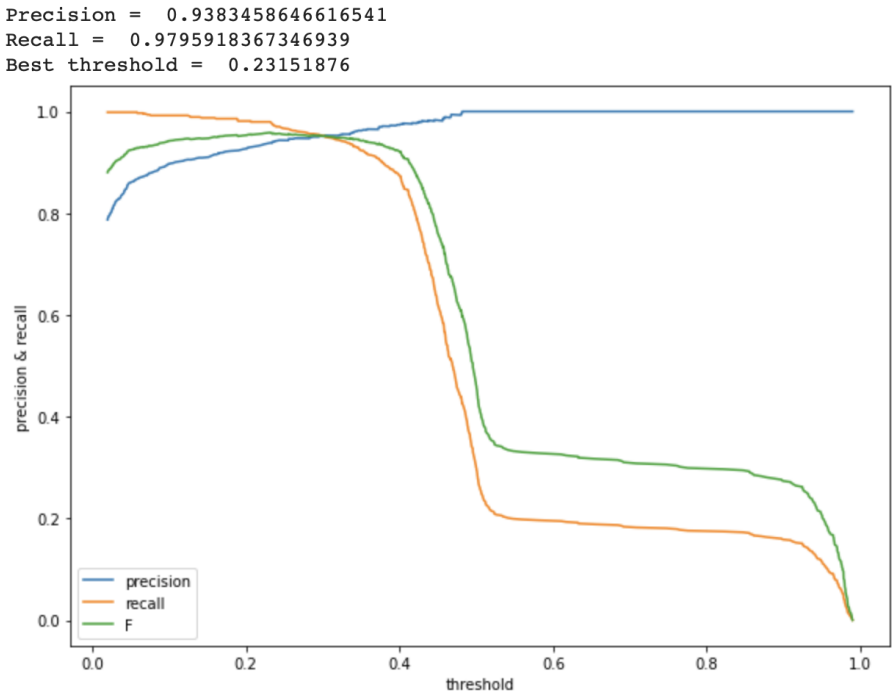

Рисунок 2 – Зависимость метрики от порога срабатывания

4.5 Запуск потокового распознавания на Raspberry Pi и анализ производительности

Для работы с микрофоном была выбрана python библиотека pyaudio, а для обработки звука – библиотека librosa, так как они обладают всеми необходимыми функциями и без проблем запускаются на raspberry pi.

Потоковое распознавание устроено следующим образом:

- Я считываю с микрофона в буфер небольшой фрагмент аудио – $\frac{1}{16}$ секунды;
- Считаю для этого фрагмента mfcc;
- Передаю признаки на вход модели и, после получения положительного результата, переключаю программу в режим «выполенения команды», который будет описан дальше;

Размер буфера выбран из соображений, что он не должен быть слишком большим, иначе будет заметна сильная задержка. При этом уменьшение буфера ведет к увеличению вызовов функции mfcc, которая работает достаточно медленно, а значит растет нагрузка на процессор и он может не справиться с обработкой звука в режиме реального времени.

При размере буфера $\frac{1}{16}$ секунды задержка почти незаметна, а у процессора одно ядро загружено на $50\% - 60\%$, что является удовлетворительным результатом.

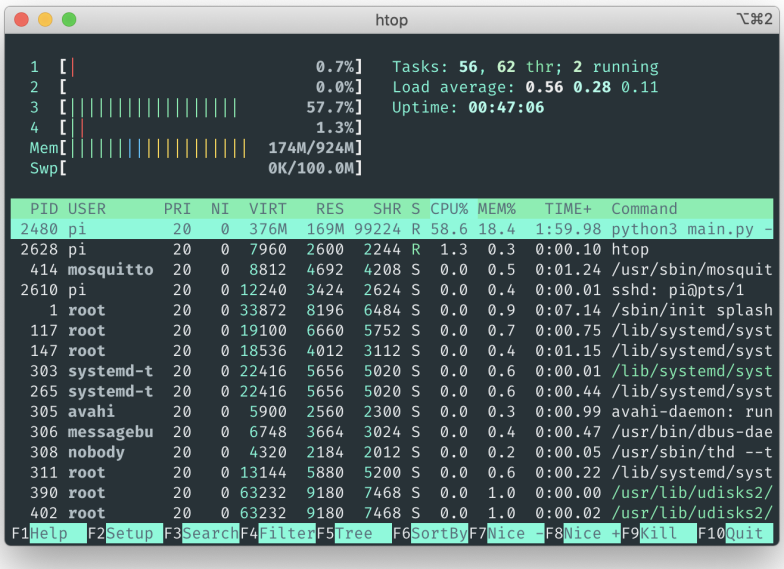

Рисунок 3 – Загруженность процессора при потоковом распознавании аудио

4.6 Платформа для голосового управления умным домом

Следующий этап работы над проектом заключался в реализации функций «умного» дома с голосовым управлением на базе обученного wake up word детектора. Для того, чтобы выполнять команды, необходимо сначала научиться определять, чтобы было сказано после ключевого слов, какую команду произнес пользователь. Для решения этой задачи было решено использовать распознавание голоса на базе облачных вычислений.

4.6.1 Yandex.Cloud

Для распознавания голоса мы выбрали SpeechKit от Яндекс Облака, который неплохо работает для русского языка и имеет несколько различных вариантов распознавания, включая потоковый режим. Итак, после переключения программы в режим «выполение команды», происходит следующее:

- Устанавливается соединение с API SpeechKit посредством gRPC сервера, который запущен на Raspberry Pi, происходит авторизация;
- Звук с микрофона начинает передаваться в облако, а облако в это время обратно пересылает промежуточные результаты распознавания. Промежуточные результаты позволяют проверить, что ключевое слово действительно было сказано, для избавления от ложных срабатываний.
- Предача данных происходит до тех пор, пока одна из сторон не закроет соединение, а это может случиться в нескольких случаях:
  - Облако распознало окончание фразы и отправило итоговый вариант распознавания;
  - Мы определили ложное срабатывание;
- После распознавания команды она передается на дальнейшую обработку для определния необходимого действия, программа обратно переключается в режим «ожидание активации»

4.6.2 ESP-32

В качестве конечного устройства, которое и будет выполнять команды, я выбрал небольшой микроконтроллер ESP32 – его часто применяют для постоения IoT из-за его дешевизны, энергоэффективности, поддержки всех необходимых систем связи – WiFi и Bluetooth, большого количества выходных пинов для подключения различных устройств, мощного процессора относительно других микроконтроллеров и стабильности работы.
Так как микроконтроллер можно подключить к WiFi, то это позвляется избавиться от необходимости тянуть провода от raspberry pi в другую комнату, чтобы сделать там «умную» лампочку или розетку. Беспроводное подключение позвояет достаточно просто расширять и дополнять систему «умного» дома.

4.6.3 MQTT

Для общение между raspberry pi и конечными устройствами по сети выбран современный IoT протокол MQTT [7] или Message Queue Telemetry Transport – легкий, компактный и открытый протокол для обмена данными, который имеет ряд особенностей, делающих его отличным выбором для данного проекта:

- Протокол спроектирован для работы в условиях нестабильной связи, а значит «умное» устройство будет работать даже вдали от роутера;
- Для его работы требуется совсем небольшое количество трафика, а значит «умный» дом не будет мешать работе других устройств в WiFi сети;
- Легкая интеграция новых устройств;

Обмен сообщениями осуществляется между клиентом и брокером, который контролирует и осуществляет доставку сообщений, при этом клиент и брокер не обязаны находиться в одной WiFi сети, а это значит, что если разместить брокера в облаке, то можно будет управлять «умным» домом из любой точки мира со смартфона.

4.6.4 «Умная» лампочка

Для демонстрации работоспособности проекта – на базе ESP32 и управляемого реле была разработана «умная» лампочка с голосовым управлением. MQTT-брокер размещен в облаке, что позвляет управлять лампочкой не только голосом, но и командами через интернет.

5 Выводы и дальнейшее развитие

В результате работы над проектом разработано устройство для голосового управления «умным» домом на основе мини компьютера Raspberry Pi. Были исследованы современные подходы к распознаванию речи для реализации своего wake-up word детектора. Была разработана архитектура и обучена модель для эффективной работы на слабом железе Raspberry Pi. Была реализована «умная» лампочка с голосовым управлением, работающая по WiFi.

Планы по дальнейшему развитию проекта включают в себя добавление в систему «умного» дома новых устройств и датчиков, отказ от облачного распознавания команд и обучения собственной speech-to-text модели для работы на Raspberry Pi.

Список литературы

[1] Justice Amoh and Kofi Odame. An optimized recurrent unit for ultra-low-power keyword spotting, 2019.

[2] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha. Temporal convolution for real-time keyword spotting on mobile devices, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, December 2015.

[4] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980.

[5] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding lstm – a tutorial into long short-term memory recurrent neural networks, 2019.

[6] Wikipedia contributors. Bell labs — Wikipedia, the free encyclopedia, 2021. [Online; accessed 8-February-2021].

[7] Wikipedia contributors. Mqtt — Wikipedia, the free encyclopedia, 2021. [Online; accessed 23-May-2021].

6 Приложение

6.1 Репозиторий проекта

Репозиторий проекта можно найти по ссылке - https://github.com/alekseygolub/voice-control