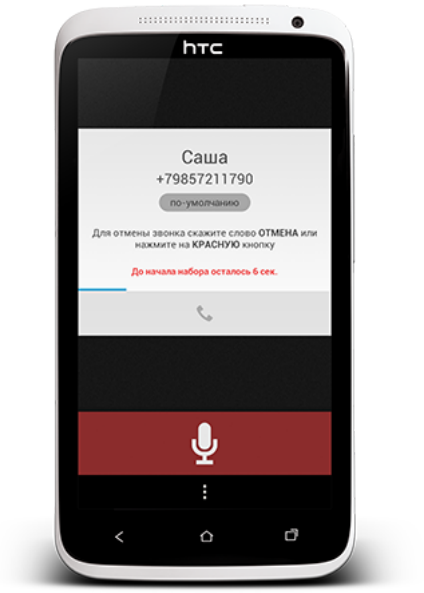
30 декабря 2013 в 19:02

Pocketsphinx — распознавание речи в реальном проекте

в черновиках

Android (https://habr.com/hub/android/), Разработка под Android (https://habr.com/hub/android_dev/), Разработка (https://habr.com/hub/development/)

Содержание



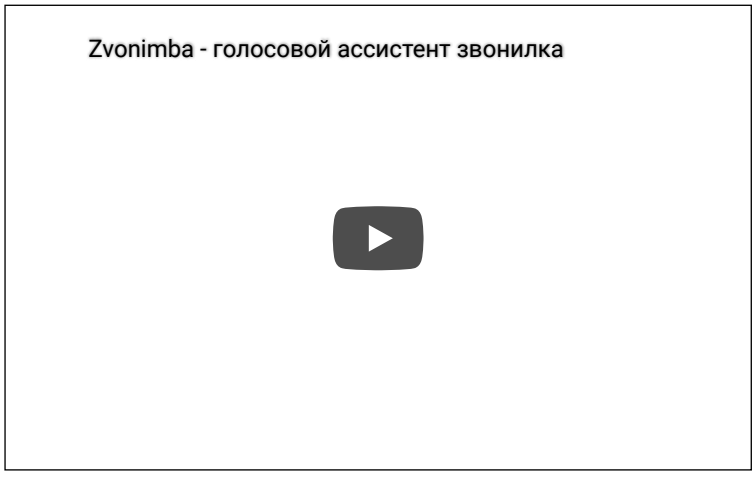
Некоторое время назад я начал большой эксперимент по использованию открытой технологии распознавания речи Pocketsphinx в одном очень интересном проекте под Android. Его целью было создание голосового ассистента-звонилки на русском языке с применением датчиков смартфона в качестве способов активации микрофона.

За короткое время эксперимент перерос в настоящий продукт под названием Zvonimba (https://play.google.com/store/apps/details?id=com.zvonimba), которым уже пользуется немалое количество человек. В этой статье я хочу рассказать, как удалось прикрутить Pocketsphinx для распознавания русской речи на смартфоне в оффлайне и какие трудности при этом возникали.

Я не буду подробно рассказывать о теории распознавания речи — об этом много и хорошо написано в интернете (например, на том же сайте Pocketsphinx (http://cmusphinx.sourceforge.net/wiki/tutorialconcepts)). В этой статье я покажу, как удалось применить pocketsphinx в реальном приложении под Android.

Концепция

Как и у любого другого приложения, у *Звонимбы* есть своя цель и концепция. А именно — дать возможность совершать звонки, не прикасаясь к экрану и не используя гарнитур. Идея в том, чтобы пользователь мог достать телефон из кармана, *поднести его к уху* и произнести имя контакта, после чего программа должна предложить позвонить кому требовалось.



Сенсоры

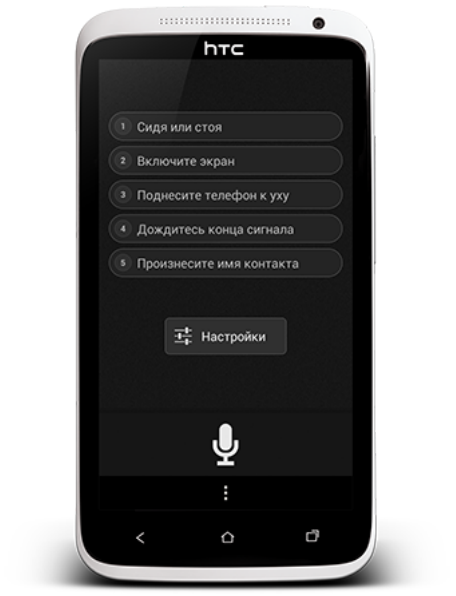
Датчики современных смартфонов позволяют отследить положение аппарата в пространстве, что дает возможность активировать любую программу в тот момент, когда человек подносит телефон к уху. Для этого был написан довольно сложный алгоритм, который получает данные сразу от трех сенсоров — акселерометра, приближения и магнитного поля. Детали его реализации не столь важны в этой статье, главное, что он активирует приложение в нужный момент, когда пользователь располагает телефон именно в том положении, при котором мы обычно совершаем звонки.

Активируясь, *Zvonimba* включает микрофон и пользователь может произнести имя контакта (плюс тип номера телефона, если хочет), а далее программа голосом повторяет это имя и предлагает либо подождать несколько секунд до начала набора, либо отменить набор, сказав команду “Отмена” (также для отмены можно выйти из программы или выключить экран). Естественность процесса позволяет, не имея гарнитуры, не отвлекаться на экран, выбирая контакт из списка.

И, конечно, самое сложное во всем процессе с точки зрения реализации, — это **распознавание русских имен и фамилий в оффлайне** на огромном разнообразии Android-устройств.

Как распознается речь и почему нельзя использовать Google ASR

Вкратце: движок распознавания речи (ASR), используя аккустическую модель, получает на вход данные с микрофона и на выходе возвращает гипотезы о том, что говорит пользователь в виде текста и некоторого цифрового эквивалента точности (confidence). Существует несколько подходов — статистическая языковая модель, ограниченная грамматика запросов и комбинация этих вариантов.



Google ASR предоставляет распознавание речи на основе большой статистической модели языка, что позволяет распознавать речь в так называемом “режиме диктовки”. Наиболее хорошо этот способ подходит для поисковых запросов и диктовки текста. О том, как это работает, подробно рассказано в одном из постов Яндекса про SpeechKit (/habr/post/198556/).
Для такого распознавания требуются большие вычислительные мощности, поэтому они производятся в облаке. *Google* предоставляет также “уменьшенную копию” такого рода модели для распознавания речи в оффлайне на *JellyBean*, благодаря чему можно худо-бедно набирать смс-ки, но для распознавания всего многообразия русских имен и фамилий такой подход не годится.

Для этого лучше подойдет второй метод — распознавание на основе ограниченного словаря произношений и грамматики запросов. При этом движок распознавания как бы “знает заранее”, что может сказать пользователь и ничего кроме этого распознать не может. Если ваша задача описывает вполне конкретные запросы, то такой подход будет наиболее подходящим. Он позволит повысить точность распознавания именно для конкретного набора слов и произвести это прямо на устройстве без подключения к сети.

Pocketsphinx под Android

Pocketsphinx — система распознавания слитной речи — разрабатывается инженерами из Университета Карнеги-Мелон и компилируется из исходников в библиотеку под ARM (http://cmusphinx.sourceforge.net/wiki/tutorialandroid) с помощью Android NDK. При этом можно сгенерировать JNI-обертки для использования нативных методов из кода на *Java*.

Положив скомпилированную библиотеку в libs/armeabi проекта, можно затем загрузить ее из кода таким образом:

```
static {
    System.loadLibrary("pocketsphinx_jni");
}
```

Основными классами для инициализации и работы с распознаванием речи являются **Config** и **Decoder**. С помощью первого указываются параметры распознавания, а также полные пути до языковой модели, словаря произношений и файла грамматики запросов в формате JSGF (Java Speech Grammar Format) (http://www.w3.org/TR/jsgf/). Decoder предоставляет конструктор, принимающий в качестве аргумента эту конфигурацию, и инициализирует собственно интерфейс для распознавания речи.

Для этого можно воспользоваться методами *startUtt*, *processRaw*, *getHyp* и *endUtt*. Первый и последний соответственно начинают и заканчивают сессию, а processRaw и *getHyp* собственно распознают речь и возвращают гипотезы.

processRaw принимает на входе байтовые фреймы — данные с микрофона, а *getHyp* возвращает гипотезу **Hypothesis**, содержащую предположение о том, что произнес пользователь. Таким образом, организовав в одном потоке сбор данных с микрофона, и передавая их в очередь на обработку декодером, можно получить результаты распознавания. Для определения окончания говорения можно использовать различные таймеры — например, разницу во времени между текущей и предыдущей гипотезой (в случае, если они одинаковы) и общее время говорения (для русских имен из контактной книги и типа телефона это время можно вычислить).

После окончания этого процесса мы получаем строку текста, из которой можно выделить имя контакта и (опционально) тип телефона, либо одну из команд (см описание грамматики ниже).

Расчет confidence после распознавания производится по следующей схеме

```
int bestScore = hyp.getBest_score();
long nmsec = endTimestamp - startTimestamp;
float confidence = ((float) bestScore) / (float) nmsec;
```

Что затем позволяет оценить, насколько правильным оказались результаты. *Хорошие значения лежат в области от -3 до 0.*

Грамматика запросов

Как было сказано ранее, движку *Pocketsphinx* можно предложить свою грамматику в формате *JSGF*, которая описывает все возможные фразы. Для задачи распознавания имен контактов и некоторых команд (типа “Отмена” или “Перезвонить”) используется такой вариант:

```
public <command> = [<garbage>] [<garbage>] [/15/ <name> [<type> | <garbage>] | /10/ <cancel> | /5/ <redial>] [<garbage>];
```

Квадратные скобки обозначают, что элемент является *необязательным* и может отсутствовать в речи, вертикальная черта разделяет *варианты*, а угловые скобки — это ссылки на *именованые элементы*, которые должны быть описаны в грамматике ниже.
Также вы можете заметить числа, указанные перед некоторыми элементами в списке альтернатив. Это *веса* — относительные величины, указывающие вероятность той или иной фразы. Из приведенного примера — имя и тип телефона ожидается чаще, чем команда “Отмена” или “Перезвонить”.

Garbage — это то, что позволяет нам отфильтровать некоторые окружающие шумы, воспринимаемые системой как out-of-vocabulary. В качестве “мусора” я использовал все гласные русского алфавита. Таким образом, фраза пользователя может начинаться и заканчиваться несколькими нераспознанными звуками, т.к. следуя из задачи, пользователь начинает произносить имя практически сразу же после сигнала и нам заранее известно примерное время распознавания. Этот подход помогает решить некоторые проблемы, связанные с шумоподавлением.

Немного о подавлении шумов

Словарь произношений

Он содержит соответствие между каждым словом, участвующим в распознавании, и его транскрипцией, записанной в специальном формате. Есть международный фонетический алфавит IPA

(http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0
и его ASCII версия *WorldBet*. Но используемый формат зависит от самой языковой модели и в нашем случае (я использовал русскую модель voxforge (http://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/Russian%20Voxforge/)) он несколько иной. В нем используется набор ASCII символов для всех согласных, мягких согласных, гласных и ударных гласных по следующей схеме: мягкие согласные записываются как двойной вариант твердого аналога, а ударная гласная — как двойной безударной.

Таким образом для всех типов телефонов и большинства самых распространенных русских имен можно вручную составить такой словарь, например:

```
домашний  d ay m aa sh n i j
мобильный  m ay bb ii ll n y j
сотовый  s oo t ay v y j
рабочий  r a b oo ch i j
отмена  ay t mm ee n a
отмени  ay t mm e nn ii
отменить  ay t mm e nn ii tt
перезвони  pp e rr e z v ay nn ii
перезвонить  pp e rr e z v ay nn ii tt
александр  a ll i k s aa n d a r
владимир  v l a dd ii mm i r
олег  a ll ee k
ольга  oo ll g a
```

Конечно, невозможно вручную записать транскрипции для абсолютно всех вариантов фамилий, а также всевозможных сокращений. Поэтому остальные варианты генерируются на ходу с применением правил произношения русского языка (например, популярное окончание мужских фамилий -OB превращается в глухую ay f и т.д.). Минус такой генерации в том, что в некоторых случаях она ошибается и вдобавок невозможно проставить правильные ударения, что понижает точность результатов распознавания.

Сгенерировав файл, содержащий транскрипции для всех имен из имеющийся контактной книги пользователя, можно передать полный его путь в объект класса *Config*, также указав пути до файла грамматики и директории, содержащей данные языковой модели. После чего инициализировать Decoder и начать процесс распознавания в нужный момент.

Как получить данные с микрофона устройства

Для этого используется стандартный класс Android SDK AudioRecord (http://developer.android.com/reference/android/media/AudioRecord.html). В его конструкторе нужно указать тип источника звука (стандартный микрофон, распознавание речи, камкодер и т.п.), частота дискретизации (зависит от модели, чаще всего это 8 или 16 kHz), формат аудио-данных и размерность буфера (также зависит от модели, чем меньше тем лучше). Стоит отметить, что тип источника звука напрямую влияет на качество распознавания, т.к. операционной системой задействуются некоторые возможности по оптимизации выходных данных для каждого конкретного типа (подробнее об этом написано в самой документации по классу *AudioRecord*).

В заключение

Pocketsphinx предоставляет возможность использовать речь во многих проектах для мобильных платформ, обеспечивая приемлемое качество распознавания и относительную простоту внедрения.

Конечно, многие проблемы остаются нерешенными. В частности в моем случае, пользователи зачастую используют непроизносимые имена для некоторых контактов (например, аббревиатуры или однобуквенные сокращения), что влияет на качество. Для таких случаев есть

функциональность создания псевдонимов для каждого контакта.
Но это тоже далеко не все. Многообразие устройств на платформе Android порождает зачастую разное поведение на различных девайсах — от скорости распознавания до качества звука с микрофона и влияния посторонних шумов.

Тем не менее, продемонстрированный подход позволил реализовать реальный продукт, помогающий совершать звонки без прикосновений к экрану. Попробовать его в деле можно, скачав на GooglePlay (https://play.google.com/store/apps/details?id=com.zvonimba)

Надеюсь, статья и *Zvonimba* вам понравились, и вы почерпнули для себя полезные знания в области применения современных технологий распознавания речи с открытым кодом.

С удовольствием отвечу на ваши вопросы!

android (https://habr.com/search/?q=%5Bandroid%5D), распознавание речи (https://habr.com/search/?q=%5B%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5+%D1%80%D0%B5%D1%87%D0%B8%5D), андроид (https://habr.com/search/?q=%5B%D0%B0%D0%BD%D0%B4%D1%80%D0%BE%D0%B8%D0%B4%5D), java (https://habr.com/search/?q=%5Bjava%5D), разработка (https://habr.com/search/?q=%5B%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%5D)

morfeusys (https://habr.com/users/morfeusys/) сканировать

комментарии (0) (/rss/post/habr:207878/)