

Программное средство для распознавания речи

CMU Sphinx

Введение

CMU Sphinx — проект университета Карнеги-Меллона по распознаванию человеческой речи. Проект развивается под лицензией BSD, может использоваться в коммерческих и некоммерческих проектах. Исходные коды находятся на [github](#) и доступны для скачивания и дополнения любому человеку. Основными инструментами проекта являются:

- Pocketsphinx — небольшая быстрая программа, обрабатывающая звук, акустические модели, грамматики и словари. Работает под множеством платформ, к примеру Android, iOS, Windows Phone,
- библиотека Sphinxbase, необходимая для работы Pocketsphinx ,
- Sphinx4 — гибкая библиотека для распознавания,
- Sphinxtrain — программа для обучения акустическим моделям.

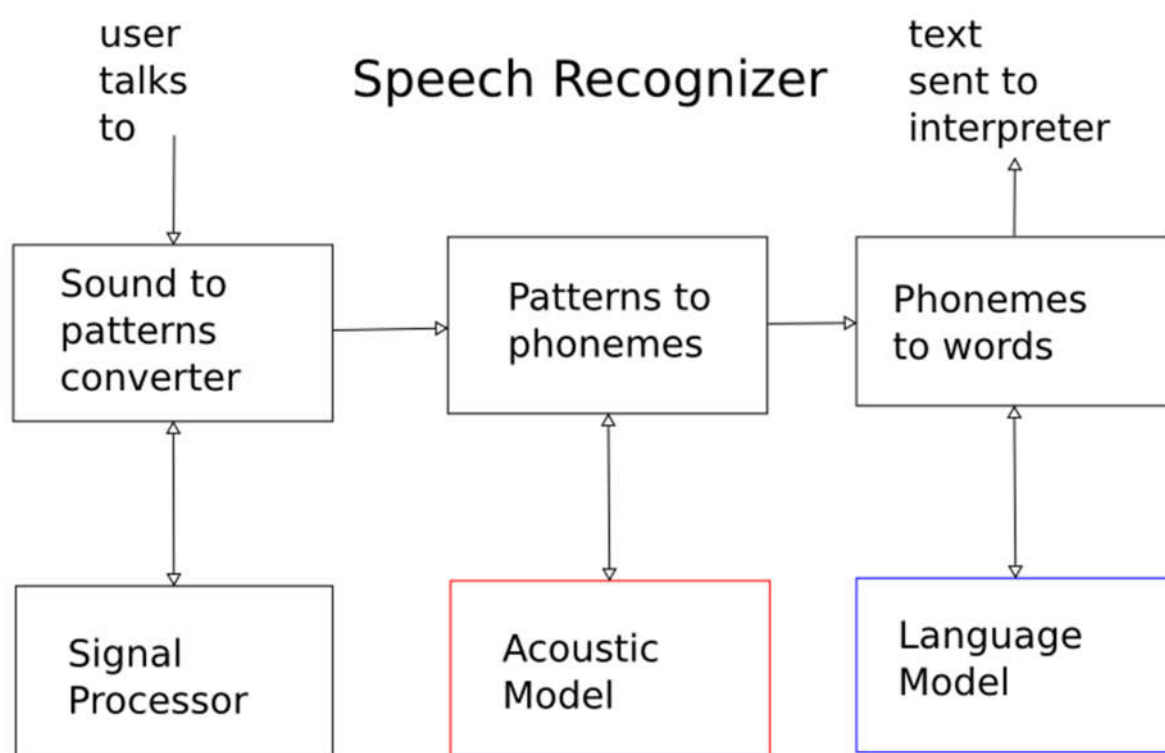
Для дальнейшего понимания материала нужно рассмотреть некоторые термины:

- Акустическая модель — отвечает за сопоставление звуку произнесенной фонемы. Позволяет оценить распознавание речевого сегмента с точки зрения схожести на звуковом уровне. Для каждого звука изначально строится сложная статистическая модель, которая описывает произнесение этого звука в речи.

Акустическая модель описывает плотность распределения вероятностей акустических классов (слов, звуков) на заданном участке речевого сигнала,

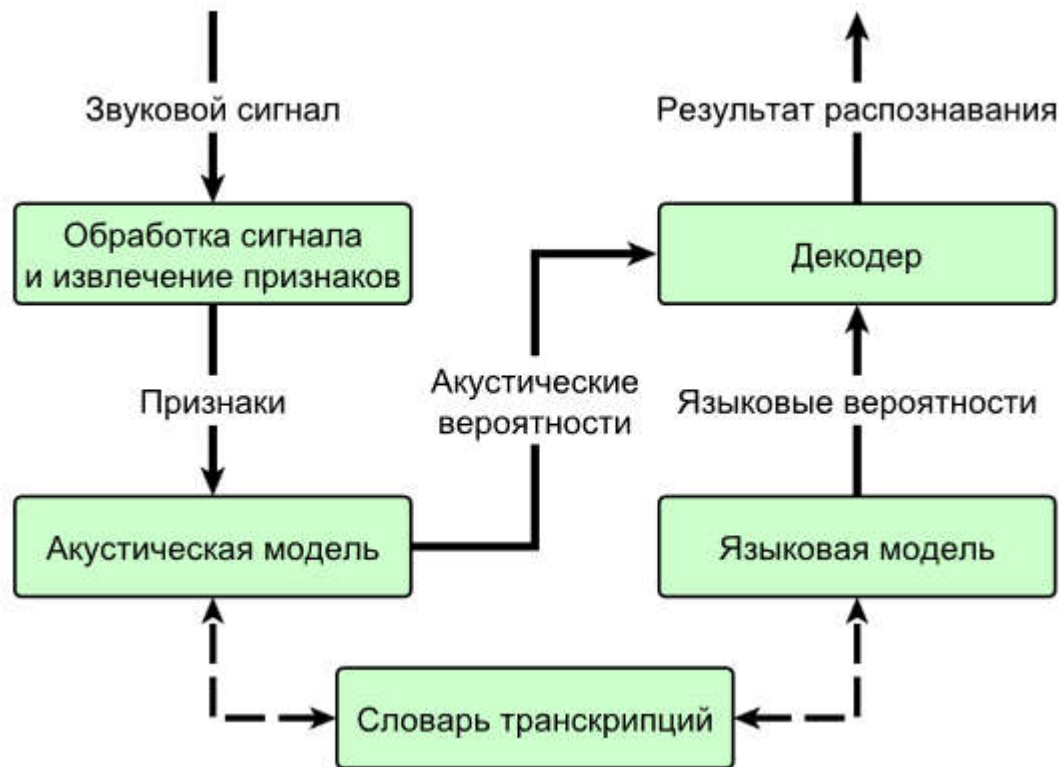
- Словарь — это файл, в котором сопоставлены лексемы и фонемы (слово и его транскрипция). Необходим для преобразования фонем, распознанных акустической моделью, в лексемы ,
- Грамматика — формальные правила, которые описывают простые правила построения предложений. Лексемы, полученные на предыдущем шаге, пытаются сопоставиться с грамматикой,
- Языковая модель — это статистическая модель языка. Она описывает вероятности слов и их комбинаций. Таким образом распознавание лексем — это максимизация правдоподобности распознанной фразы.

Из определения терминов, можно заметить простейшую связь:



Как это работает

Рассмотрим общий случай программного распознавания звуков.



Слайд 8 Теоретическая схема распознавания речи

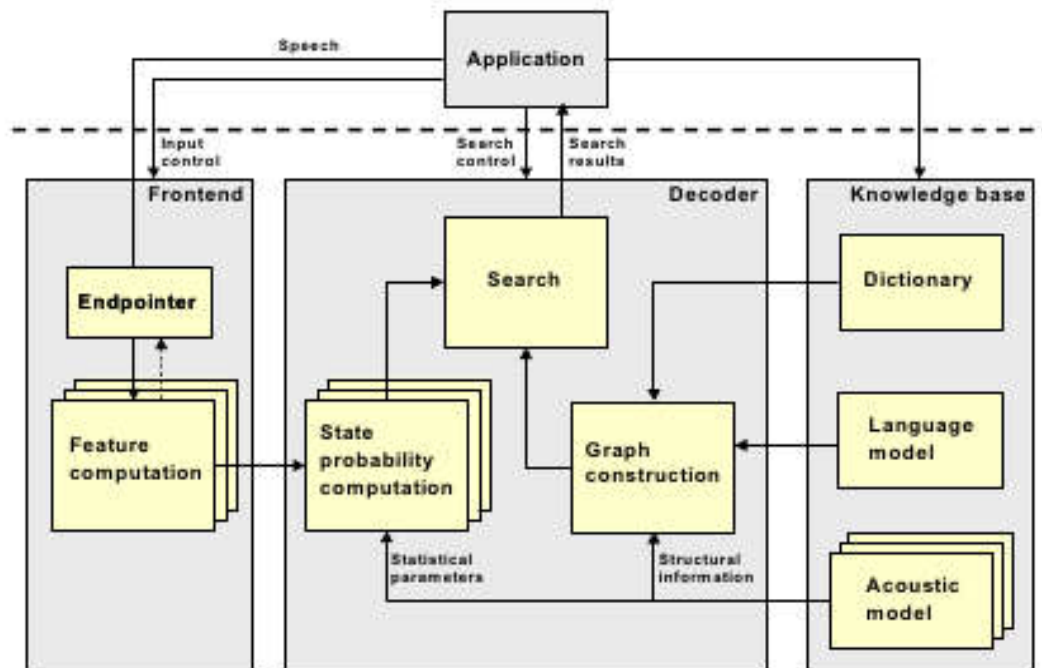
Звуковой сигнал попадает в модуль предварительной обработки сигнала. На этом этапе сигнал делится на мелкие части (фреймы), очищается от шумов, при необходимости сигнал переводится из временной области в частотную (спектр). На выходе появляются векторы информационных признаков.

«Чистый» звуковой сигнал и его признаки попадают в модуль акустической модели, где определяется, какой именно звук был произнесен. Выход – фонемы и вероятности.

С помощью словаря устанавливается связь между последовательностями фонем, описываемых в акустической модели, и словами, описываемыми в языковой модели.

Модуль декодера анализирует вероятности, полученные от модулей акустической модели и языковой модели, и преобразует их в последовательность слов – распознанную речь.

Архитектура CMU Sphinx в общих чертах устроена так же:



Слайд 9 Архитектура Sphinx

Блок Frontend обрабатывает входные данные, готовит для дальнейшей работы.

База знаний содержит информацию необходимую для декодера. Эта информация включает в себя акустическую модель и модель языка. База знаний также может получить ответ от декодера, что позволяет базе знаний динамически изменяться себя на основе результатов поиска. Эти модификации могут включать в себя переключение акустических моделей и/или языка модели, а также обновлять параметры, такие как среднее и дисперсия преобразования для акустических моделей.

Декодер выполняет большую часть работы. Он считывает данные с Frontend, сопоставляет их с данными из базы знаний и данными от приложения

и выполняет поиск наиболее вероятных последовательностей слов, которые могли бы быть представлены рядом особенностей.

Декодер содержит модуль построения графа конструкций, модуль поиска и модуль вычисления вероятности состояния.

Модуль построения графа конструкций по информации из базы знаний строит граф языка. Модуль вычисления вероятности состояния на основании «чистых» данных от блока Frontend и информации от акустической системы считает вероятности выходных состояний. Модуль поиска строит выходное предложение, идя по графу, выбирая состояния на основе вероятностей выходных состояний.

Другие проекты для распознавания речи

Для выявления особенностей CMU SPHINX нужно сравнить его с другими известными проектами по распознаванию речи.

Другие проекты

- Kaldi
- HTK
- Julius
- ISIP
- Google Speech Recognition API
- Yandex Speech Kit

Слайд 10 Другие проекты

Первые четыре проекта выпускаются под лицензией BSD, но решения от Google и Yandex – проприетарные, без открытого исходного кода и их нельзя использовать в коммерческих продуктах. В дальнейшем будем сравнивать первые четыре проекта.

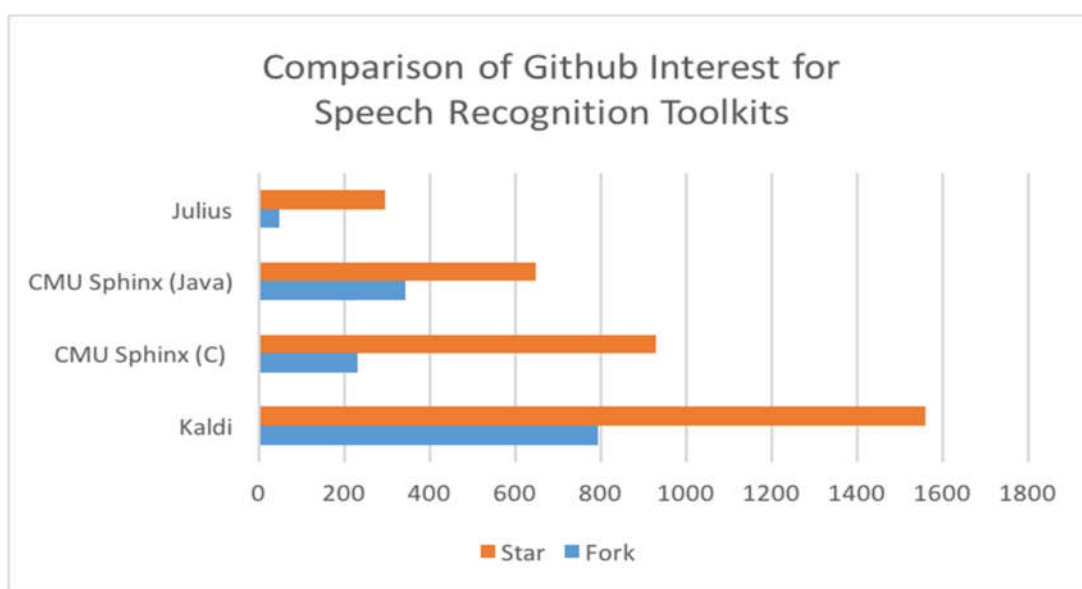
Сравнение будет проводиться по четырем критериям: доступные разработчику языки программирования, наличие людей, использующих программное средство, наличие примеров и обучающих статей для новичков и состояние разработки на данный момент (активное или нет).

Toolkit	Programming languages	Development activity	Tutorials and examples	Community	Trained models
CMU Sphinx	Java, C, Python, others	+++	+++	+++	English plus 10 other languages
Kaldi	C++, Python	+++	++	+++	Subset of English
HTK	C, Python	++	+++	++	None
Julius	C, Python	++	++	+	Japanese
ISIP	C++	++	++	+	Digits only

Слайд 11 Сравнение

Можно заметить, что из всех проектов CMU Sphinx позволяет использовать для разработки большинство популярных языков, имеет активное сообщество разработчиков, а также уже имеется большое количество примеров и обучающих статей.

По активности на github проекты распределены следующим образом:



Слайд 12 Сравнение

Kaldi, как более новый проект, развивается активнее. CMU Sphinx развивается чуть менее активно.

Применение

Предполагаемые сферы применения:

- Автоответчик-секретарь для дома или в компании,
- Соцопросы,
- Приём заказов и автоматизация служб доставки,
- Информационная поддержка, справочная служба,
- Первая линия техподдержки/call центр,
- Обеспечение безопасности, автоматизация анализа телефонных разговоров и прослушки,
- Умный дом, наблюдение за престарелыми (не всегда хватает сил доползти до трубки телефона, но может хватить на крик «Помогите!» - распознав который программа сможет вызвать помощь и связаться с близкими).

Наиболее интересно применение в умном доме, ведь умный дом – это будущее, которое может получить массовое распространение в ближайшее время.

Применение в «умном доме»

- Умный дом — система домашних устройств, способных выполнять действия и решать определенные повседневные задачи по командам человека без его участия



Приложения с использованием Sphinx могут реализовывать систему управления умным домом с голосовым интерфейсом.

Применение в «умном доме»

- Приложения с использованием Sphinx могут реализовывать систему управления умным домом с голосовым интерфейсом



Слайд 14 Роль Сфинкс в умном доме

Подобно тому, как Тони Старк общается с JARVIS, люди смогут отдавать команды системе своего дома с помощью приложений, разработанных с использованием CMU Sphinx.

У CMU Sphinx есть два подхода для распознавания речи: использование пользовательской грамматики и использование языковой модели (обычно созданной специалистами и описывающей большую часть естественного языка).

Стоит рассмотреть оба способа:

1) Применение грамматики

Применение в «умном доме» с использованием собственной грамматики

- Качаем акустическую модель с репозитория проекта CMU Sphinx
- В файл словаря добавляем слова, которые входят в команды
- В файле грамматики описываем предложения, которые будут распознаваться
- ???
- PROFIT!

Слайд 15 Применение грамматики

Особенности использования собственной грамматики

Плюсы:

- Выдает хорошие результаты на малом количестве команд
- Разработчик может создавать удобные ему конструкции

Минусы:

- Говорить нужно членораздельно
- Предложения должны строго соответствовать грамматике

Слайд 16 Особенности использования грамматики

Грамматика для CMU Sphinx создается в формате JSFG (Java Speech Grammar Format):

Пример грамматики в формате JSFG

```
grammar exampleGrammar;  
public <exampleGrammar> = (<object><action>);  
<object> = (Компьютер | Свет | Чайник);  
<action> = (Включить | Выключить);
```

Слайд 17 JSFG

2) Применение языковой модели

Применение в «умном доме» с использованием языковой модели

- Качаем акустическую и языковую модели с репозитория проекта CMU Sphinx
- В файл словаря добавляем слова, которые входят в команды
- ???
- PROFIT!

Слайд 18 Использование ЯМ

Особенности использования языковой модели

Плюсы:

- Предложения могут строиться с ошибками
- Пользователь не ограничен малым набором предложений

Минусы:

- На задачах, где не требуется большее количество команд, распознает фразы хуже, чем приложение с использованием грамматики

Слайд 19 Особенности ЯМ

Вывод: использование языковой модели позволяет распознать больше предложений, но в системах с ограниченным числом команд, как в умном доме, разумнее использовать пользовательскую грамматику. Скорость и качество распознавания будет отличаться в лучшую сторону.

Заключение

Заключение

- Sphinx — мощное средство для задач распознавания речи в оффлайне
- Open-source, постоянно развивается. Активно используется в коммерческих и некоммерческих проектах (Chatter, Misterhouse, Nightingale)
- Можно использовать на разных платформах (Rocketsphinx) и писать приложения на разных языках программирования
- Не вызывает особых сложностей в использовании

Источники

- Мединников И.П. «Методы, алгоритмы и ПС распознавания русской телефонной спонтанной речи», 2016
- Федяев О. И., Бакаленко В. С., Савкова Д. Г. «Речевой интерфейс для интеллектуализации ввода исходного кода программ», 2015
- <https://habrahabr.ru/post/237589/>
- <https://habrahabr.ru/post/167479/>
- <http://cmusphinx.sourceforge.net>
- <https://svds.com/open-source-toolkits-speech-recognition/>