


[НА ГЛАВНУЮ](#)[О ПРОЕКТЕ](#)

0

Распознавание речи с помощью CMU Sphinx +18

23.09.15 10:39 • chubakur • #267539 • Хабрахабр •  2 •  7694

[Работа со звуком](#)



CMU Sphinx сейчас является крупнейшим проектом по распознаванию человеческой речи. В инструментарий входят следующие программы и библиотеки:

Pocketsphinx — небольшая программа, которая принимает на вход произвольные акустические модели, грамматики и словари, а также звуковой поток(либо звуковой файл, либо сам берет поток с микрофона). На выходе получается распознанный текст.

Написана на C, работает быстро.

Sphinxbase — библиотека необходимая для работы Pocketsphinx

Sphinx4 — гибкая библиотека для распознавания, написана на Java.

Sphinxtrain — программа для обучения акустических моделей.

Для работы со CMU Sphinx важно запомнить несколько определений и понять их отличия.

Акустическая модель — отвечает за сопоставление звуку произнесенной фонемы.

Акустическую модель для русского языка можно скачать на сайте проекта. [Русская акустическая и языковая модели. А также словарь.](#)

Словарь — это файл, в котором написаны сопоставлены лексемы и фонемы (слово и его транскрипция). Например, калькулятор (k au ll k u ll ja t au r). Он необходим для преобразования фонем, распознанных акустической моделью в лексемы.

Грамматика — это формальные правила, которые описывают простые правила построения предложений. Лексемы, полученные на предыдущем шаге пытаются сопоставиться с грамматикой и если удачно, то выводится результат.

Языковая модель — это статистическая модель языка. Она описывает вероятности слов и их комбинаций. Таким образом распознавание лексем — это максимизация правдоподобности распознанной фразы.

Чем сложнее язык, чем обширнее правила и размер словаря, тем хуже точность распознавания. Поэтому, для минимизации ошибки, имеет смысл создания упрощенных правил, которые будут описывать конкретную задачу.

Наш формальный язык

Пусть наша программа будет реагировать на фразы <<ок, компьютер>> или <<ок, калькулятор>> за которой будет идти простое выражение. Например:

“ ок, компьютер сорок семь плюс двадцать один.

Начнем описывать формальную грамматику.

Приветствие или активация (символ | означает или, скобки используются в качестве группирующего оператора):

<greeting> = (ок|окей)(компьютер|калькулятор);

Однозначные числа:

<n1> = (ноль|один|два|три|четыре|пять|шесть|семь|восемь|девять);

Двузначные числа(это числа от 10 до 19, а также 20,30,40,50,60,70,80,90 с

опциональным однозначным):

```
<n2> = (десять|одиннадцать|двенадцать|тринадцать|четырнадцать|пятнадцать|
шестнадцать|семнадцать|восемнадцать|девятнадцать)|(двадцать|тридцать|сорок|
пятьдесят|шестьдесят|семьдесят|восемьдесят|девяносто)[<n1>];
```

Трехзначные числа(100,200,300,400,500,600,700,800,900 плюс опциональное
двузначное или однозначное):

```
<n3> = (сто|двести|триста|четыреста|пятьсот|шестьсот|семьсот|восемьсот|девяносто)[<n2>|<n1>];
```

Многочисленные числа(п тысяч плюс опциональные двузначные или однозначные):

```
<n4> = [<n1>|<n2>|<n3>](тысяча|тысяч|тысячи)[<n3>|<n2>|<n1>];
```

На этом я решил остановиться, но не сложно добавить еще одно правило для миллиона,
миллиарда и прочих больших чисел.

Общее правило для числа:

```
<number> = <n4>|<n3>|<n2>|<n1>;
```

Теперь опишем возможные операции:

```
<operation> = плюс | минус | умножить на | разделить на;
```

Выражение:

```
<expression> = <number> <operation> <number>;
```

И собственно сам запрос, то что мы и хотим распознавать:

```
public <query> = <greeting> <expression>;
```

Файл с грамматикой можно взять с гитхаба [calc.jsgrf](#).

Составление словаря

Теперь составим словарь всех возможных слов. В принципе, можно использовать готовый словарь для русского языка, который скачивается вместе с акустической моделью. Ведь все эти слова наверняка там есть, а грамматика просто не пропустит слово, которого в ней не описано. Однако чем меньше словарь — тем быстрее поиск по нему и зачем нам

хранить в оперативке словарь на 500000 слов, когда мы используем лишь несколько десятков. Но если вам лень составлять свой, то можно просто подключить готовый, я уверен что он будет работать. Сначала запишем все слова, которые будут корректны для нашего языка. Вот возьмем прямо все слова что мы описали в грамматике и запишем их в файл по слову на строку. Получится что-то вроде [этого](#).

Для того чтобы получить транскрипцию для этих слов можно воспользоваться проектом [ru4sphinx](#). Скрипт [dict2transcript.pl](#) принимает на вход 2 аргумента(наш файл со списком слов и выходной файл, куда записать слова и их транскрипции).

Получившийся словарь имеет следующий вид:

спойлер

восемнадцать v ay ss i m n aa c ay tt

восемь v oo ss i mm

восемьдесят v oo ss i mm dd i ss i t

восемьдесят(2) v oo ss i mm ss i t

восемьсот v ay ss i mm s oo t

два d v aa

двадцать d v aa c ay tt

две d vv je

двенадцать d vv i n aa c ay tt

двести d vv je ss tt i

девяносто dd i vv i n oo s t ay

девятнадцать dd i vv i t n aa c ay tt

девять dd je vv i tt

девятьсот dd i vv i c oo t

десять dd je ss i tt

калькулятор k ay ll k u ll ja t ay r

компьютер k a m pp j ju tt i r

минус mm ii n u s

на n aa

ноль n oo ll

один a dd ii n

одиннадцать a dd ii n ay c ay tt
одна a d n aa
ок oo k
окей a kk je j
плюс p ll ju s
пятнадцать pp i t n aa c ay tt
пять pp ja tt
пятьдесят pp i tt dd i ss ja t
пятьдесят(2) pp i ss ja t
пятьсот pp i c oo t
разделить r ay z dd i ll ii tt
семнадцать ss i m n aa c ay tt
семь ss je mm
семьдесят ss je mm dd i ss i t
семьдесят(2) ss je mm ss i t
семьсот ss i mm s oo t
сорок s oo r ay k
сорок(2) s a r oo k
сто s t oo
три t rr ii
тридцать t rr ii c ay tt
тринадцать t rr i n aa c ay tt
триста t rr ii s t ay
тысяч t yy ss i ch
тысяча t yy ss i ch i
тысяча(2) t yy s ay ch i
тысяча(3) t yy sch i
тысячи t yy ss i ch i
тысячи(2) t yy sch i
умножить u m n oo zh y tt
четыре ch i t yy rr i
четыреста ch i t yy rr i s t ay
четырнадцать ch i t yy r n ay c ay tt
шестнадцать sh y s n aa c ay tt

шесть sh oo ss tt
шесть(2) sh ee ss tt
шестьдесят sh y ss tt dd i ss ja t
шестьдесят(2) sh y z dd i ss ja t
шестьдесят(3) sh y ss ja t
шестьсот sh y s oo t
шестьсот(2) sh y s oo t

[И доступен для скачивания.](#)

Проверка работы

Попробуем распознать несколько команд, воспользуемся для этого `pocketsphinx`.
`Pocketsphinx_continuous` принимает множество разных параметров я опишу лишь некоторые из них, они будут необходимы для запуска:

- hmm <путь к акустической модели>. Если вы скачивали модель по приведенной выше ссылке, то акустическая модель будет находится в папке `zero_ru_cont_8k_v3/zero_ru.cd_cont_4000`.
- dict <путь к словарю>
- jsgf <путь к грамматике>
- lm <путь к языковой модели>
- logfn <путь к файлу для логгирования>. По умолчанию лог пишется в `stdout`.
- infile <путь к файлу с голосом>. Надо удостовериться что частота дискретизации файла совпадает с частотой дискретизации модели.
- inmic <yes|no>. Звук в реальном времени с микрофона.
- remove_noise <yes|no>. Фильтрация шумов. По умолчанию `yes`.

```
pocketsphinx_continuous -hmm zero_ru.cd_cont_4000 -dict  
calc_lang/vocabular.dict -jsgf calc_lang/calc.jsgf -inmic yes
```



Вы можете помочь и перевести немного средств на развитие сайта

Назначение перевода

На развитие сайта

Сумма

Перевести

Теги:

[распознавание речи](#), [cmu sphinx](#), [cmusphinx](#), [распознавание голоса](#)



Комментарии (2):



Klukonin

23.09.15 02:50 / [#8586723](#)

А на деле придется учитывать то что числа распознаются отдельно. Если следом за числом не идет действие — числа суммируются до тех пор, пока парсер не встретит математическое действие. И потом производит его с такой же суммой чисел. Как-то так я это вижу.

mbait

24.09.15 04:37 / **#8589169** / +1

Для приветствия существует тип поиска keyword search, читайте справку по параметрам `-kws`` и `-kws_search``. Этот метод распознает лучше чем грамматика.

“ Однако чем меньше словарь — тем быстрее поиск по нему и зачем нам хранить в оперативке словарь на 500000 слов

Словарь весь не грузится — только те слова, что используются в языковой модели.

“ `<n3> = (сто|двести|триста|четыреста|пятьсот|шестьсот|девятьсот)
[<n2>|<n1>];`

Можно переписать оптимальнее:

```
<n34> = ( три | четыре ) ста ; <n59> = ( пять | шесть | семь |  
восемь | девять ) сот ; <n3> = ( сто | двести | <h34> | <h59> ) [  
<n2> | <n1> ] ;
```



2015 ITnan.ru Design by [Styleshout](#).

Права на текст статей, расположенные на сайте, принадлежат их авторам. Источники статей: [Хабрахабр](#) и [Гиктаймс](#).