Распознавание речи в Intel Perceptual Computing SDK

Содержание лекции

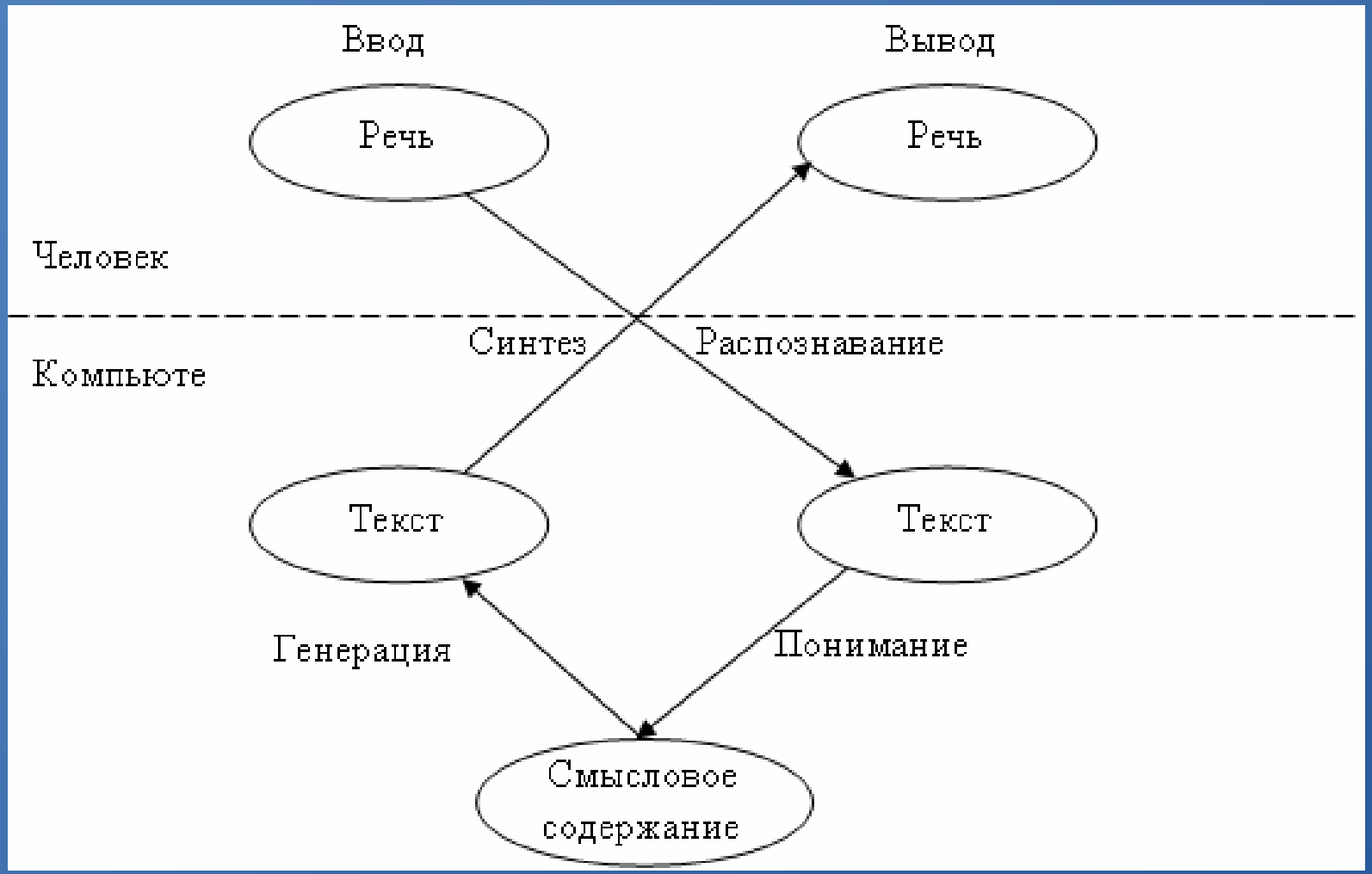
- введение
- особенности речевого взаимодействия с компьютером
- модуль распознавания речи: основные возможности
- программирование распознавания речи, класс UtilPipeline
- процесс программирования распознавания речи
- режим голосового управления
- синтез речи

Введение

Представители Intel показали ноутбук с тестовой версией системы распознавания речи Dragon от одного из лидеров этого рынка — компании Nuance.

В ходе демонстрации программа безошибочно распознавала короткие запросы на естественном английском языке, выполняя поиск в Google, запуская проигрывание музыки определенного жанра (по нечеткому запросу вроде «включи мне какой-нибудь рок»), публикуя сообщение в сервисе микроблогов Twitter — и все это без прикосновения пользователя к клавиатуре.

Схема речевого взаимодействия



Основные направления использования голосовых интерфейсов:

задачи, в которых главным образом требуется распознавание речи:

- простые команды и управление;
- простой ввод данных (по телефону);
- диктант;

задачи, в которых кроме распознавания речи требуется понимание текста (интерактивный разговор):

- информационные киоски;
- диалоговая обработка запросов;
- интеллектуальные агенты.

Основные сложности, возникающие при автоматическом распознавании речи:

- эффект коартикуляции;
- необходимость настраивать систему автоматического распознавания речи на каждого оратора отдельно;
- свободная речь (слова-паразиты и слова, не включенные в словарь);
- необходимость создания модели естественного языка;
- устойчивость к шумам.

Характеристики систем автоматического распознавания речи

- режим речи:

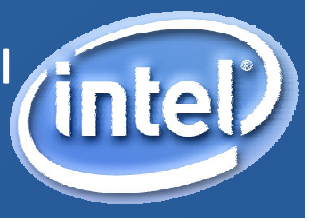
может варьироваться от раздельного произношения слов до непрерывной речи

- стиль речи:

варьируется от чтения текста до спонтанной речи;

- подстройка:

зависимость от оратора – пользователь до работы с системой должен предоставить образцы своей речи для настройки, с другой стороны независимость от оратора не предполагает никаких настроек до использования системы



Характеристики систем автоматического распознавания речи

- словарь:

набор слов может варьироваться от небольшого объема (< 20 слов) до огромного (> 50000 слов);

- языковая модель:

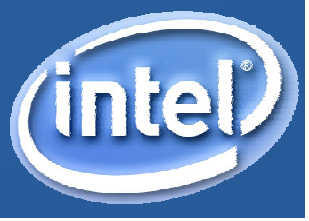
самая простая языковая модель может быть определена как сеть с конечным числом состояний, более сложная, но при этом более близкая к естественному языку модель, описывается в терминах контекстно-зависимых грамматик;

Характеристики систем автоматического распознавания речи

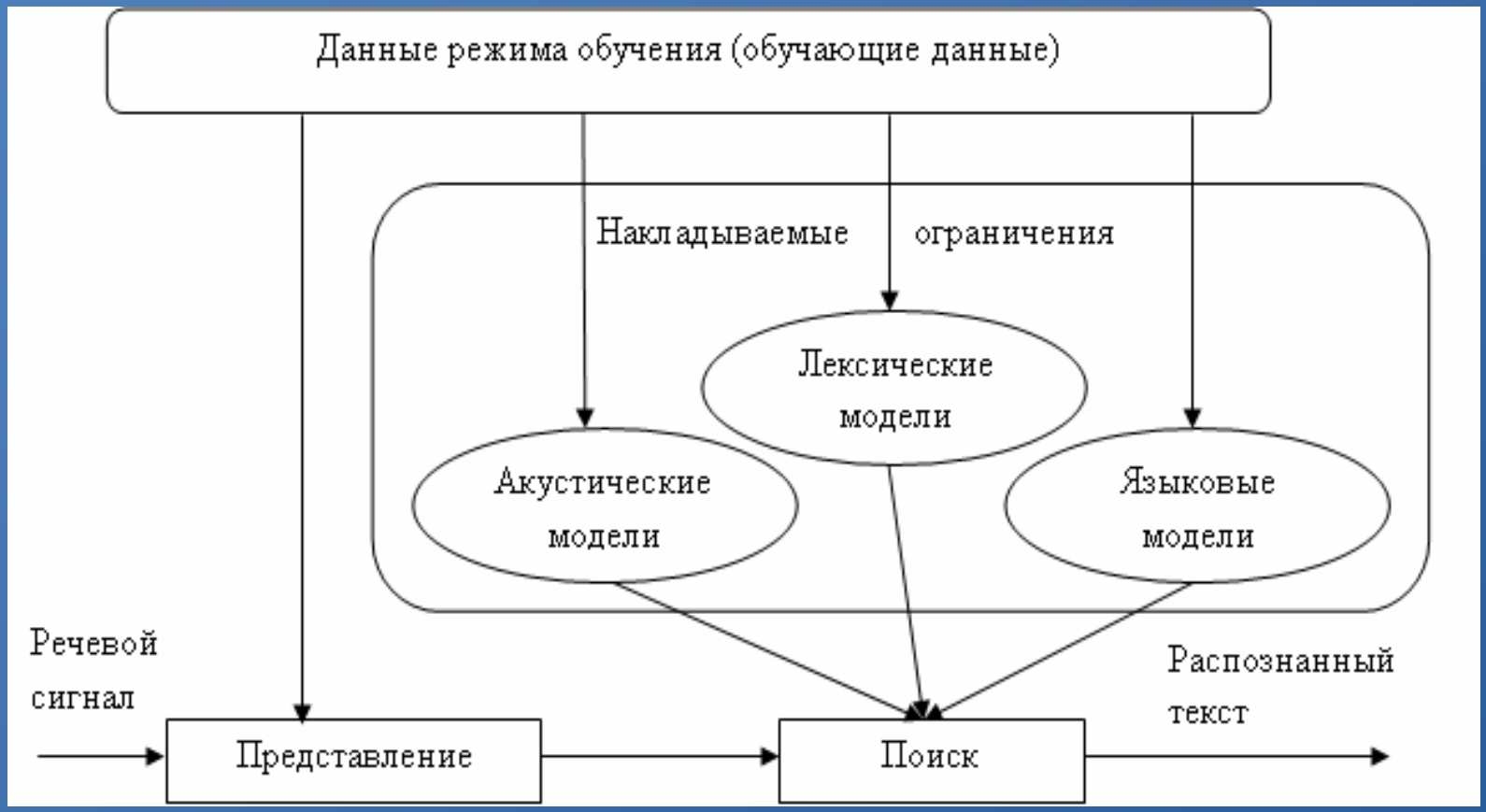
- перплексивность (степень неопределенности вероятностной модели):

популярная мера сложности задачи, объединяющая размер словаря и языковую модель;

- ряд внешних параметров, которые могут повлиять на производительность системы автоматического распознавания речи, включая характеристики шума окружающей среды, а также размещение и характеристики микрофона.



Основные компоненты системы автоматического распознавания речи



Процесс автоматического распознавания речи

Сначала оцифрованный речевой сигнал разбивается на множество фрагментов фиксированного размера, примерно по 10-20 мс каждый.

Для каждого фрагмента подбирается наиболее подходящее слово, в процессе подбора учитываются ограничения, накладываемые акустической, лексической и языковой моделями.

Для определения (уточнения) значений параметров моделей в процессе распознавания используются обучающие данные.

Модуль распознавания речи основан на Nuance* Dragon Assistant

- Nuance* Voice Command and Control

распознавание в пределах списка предопределенных команд;

- Nuance* Voice Dictation

распознавание коротких предложений (<30 секунд)

- Nuance* Voice Synthesis

озвучивание печатного текста (короткие предложения)

Программирование распознавания речи, класс UtilPipeline

Создание экземпляра модуля распознавания речи

Класс MyVoicePipeline наследник класса UtilPipeline

```
class MyVoicePipeline: public UtilPipeline {
   …
};
```

Программирование распознавания речи, класс UtilPipeline

Конструктор класса MyVoicePipeline

Функция EnableVoiceRecognition() – открывает приложению возможность распознавания речи

```
public:
MyVoicePipeline(PXCSession *session=0) :
    UtilPipeline(session)
{
  EnableVoiceRecognition();
}
```

Программирование распознавания речи, класс UtilPipeline

Класс MyVoicePipeline

Переопределение функции OnRecognized – получение и обработка событий распознавания

```
virtual void PXCAPI
OnRecognized(PXCVoiceRecognition::Recognition *data)
{
    wprintf_s(L"Recognized<%s>\n",data->dictation);
}
```

```
#include "stdafx.h"
#include "util_pipeline.h"
class MyVoicePipeline: public UtilPipeline {
public:
MyVoicePipeline(PXCSession *session=0):UtilPipeline(session) {
EnableVoiceRecognition();
}
virtual void PXCAPI
OnRecognized(PXCVoiceRecognition::Recognition *data) {
wprintf_s(L"Recognized<%s>\n",data->dictation);
}
};
int _tmain(int argc, _TCHAR* argv[]) {
MyVoicePipeline pp;
pp.LoopFrames();
return 0;
}
```

Процесс программирования распознавания речи. Шаг I

Создание сессии:

SDK сессия – ключевой объект любого приложения, использующего SDK, и должен создаваться в первую очередь. Для создания сессии необходимо вызвать функцию **PXCSession_Create**.

```
PXCSmartPtr<PXCSession> session;
PXCSession_Create(&session);
```

Процесс программирования распознавания речи. Шаг II

Задание реализации модуля:

для создания экземпляра модуля используется функция **PXCSession::CreateImpl**<Y>(&y) при этом создается экземпляр интерфейса Y и присваивается переменной y.

```
PXCVoiceRecognition *vrec=0;
session->CreateImpl<PXCVoiceRecognition>(&vrec);
```

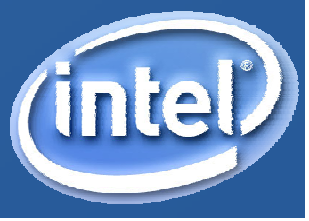
Процесс программирования распознавания речи. Шаг III

Инициализация модуля:

для инициализации модуля используются две функции

QueryProfile – возвращает доступные конфигурации и **SetProfile** – устанавливает текущую активную конфигурацию.

```
PXCVoiceRecognition::ProfileInfo pinfo;
vrec->QueryProfile(0,&pinfo);
UtilCapture capture(&session);
capture.LocateStreams(&pinfo.inputs);
vrec->SetProfile(&pinfo);
```



Процесс программирования распознавания речи. Шаг IV

Инициализация грамматики:

SetGrammar – активирует грамматический контекст;

CreateGrammar – создает пустую грамматику и возвращает идентификатор;

AddGrammar – добавляет слова в грамматику

Инициализация грамматики:

Режим диктанта:

```
pxcUID gid=0;
vrec->SetGrammar(gid);
```

Режим голосового управления:

```
pxcUID gid;
vrec->CreateGrammar(&gid);
vrec->AddGrammar(gid,1,L”One”);
vrec->AddGrammar(gid,2,L”Two”);
vrec->AddGrammar(gid,3,L”Three”);
vrec->SetGrammar(gid);
```

Процесс программирования распознавания речи. Шаг V

Обработка событий:

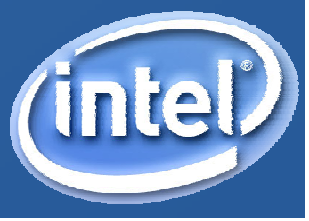
приложение реализует обработчик событий **Handler** и отслеживает появление события распознавания, используя функцию **SubcribeRecognition**

```
class MyHandler: public
 PXCVoiceRecognition::Recognition::Handler {
  public:
  virtual void PXCAPI
    OnRecognized(PXCVoiceRecognition::Recognition *data)
      {  // обработка распознавания данных}
};
…
vrec->SubscribeRecognition(0,new MyHandler);
```

Процесс программирования распознавания речи. Шаг VI

Цикл обработки данных:

```
PXCSmartPtr<PXCAudio> sample;
PXCSmartSPArray sps(2);
for (;;) {
  capture.ReadStreamAsync(sample.ReleaseRef(),
                sps.ReleaseRef(0));
  vrec->ProcessAudioAsync(sample,sps.ReleaseRef(1));
  sps.SynchronizeEx();
}
```



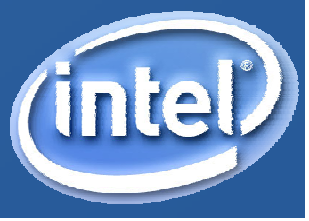
Процесс программирования синтеза речи

предполагает создание экземпляра модуля синтеза речи и передачи данных в него.

Шаг I

Создание сессии:

```
PXCSmartPtr<PXCSession> session;
PXCSession_Create(&session);
```



Процесс программирования синтеза речи. Шаг II

Определение реализации модуля:

в приложении используется функция **PXCSession::CreateImpl** для создания экземпляра интерфейса модуля распознавания речи **PXCVoiceSynthesis**:

```
PXCVoiceSynthesis *tts=0;
session->CreateImpl<PXCVoiceSynthesis>(&tts);
```

Процесс программирования синтеза речи. Шаг III

Инициализация модуля:

для инициализации модуля используются две функции **QueryProfile** и **SetProfile**. Первая функция возвращает доступные конфигурации. Вторая устанавливает текущую активную конфигурацию.

```
PXCVoiceSynthesis::ProfileInfo profile;
tts->QueryProfile(0, &profile);
tts->SetProfile(&profile);
```

Процесс программирования синтеза речи. Шаг IV

Обработка данных:

приложение использует функцию **QueueSentence** для создания очереди предложений для синтеза,

вызывает функцию **ProcessAudioAsync** для создания аудио буфера, который содержит синтезированную речь и передачи содержимого буфера на любое аудио устройство вывода

Цикл обработки данных:

```
pxcUID sid;
tts->QueueSentence(L”Speak this”,10,&sid);
PXCSmartPtr<PXCAudio> sample;
PXCSmartSP sp;
for (;;) {
pxcStatus sts=tts->    ProcessAudioAsync(sid,&sample,&sp) ;
if (sts<PXC_STATUS_NO_ERROR) break;
sp->Synchronize();
// передать участок синтезированной речи на устройство вывода
}
```

Контрольные вопросы:

1) Как выглядит схема речевого взаимодействия человека и компьютера?
2) На каком этапе может быть использован пакет Intel Perceptual Computing SDK?
3) Какие сложности возникают в процессе автоматического распознавания речи?
4) Каковы основные компоненты любой системы автоматического распознавания речи?
5) Каковы особенности и возможности модуля распознавания речи Intel Perceptual Computing SDK?
6) Что необходимо иметь в виду разработчику использующему модуль распознавания речи?