**Advantages / Disadvantages Of Speech Recognition**

**Advantages**

* Speech is a very natural way to interact, and it is not necessary to sit at a keyboard or work with a remote control.
* No training required for users!

**Disadvantages**

* Even the best speech recognition systems sometimes make errors. If there is noise or some other sound in the room (e.g. the television or a kettle boiling), the number of errors will increase.
* Speech Recognition works best if the microphone is close to the user (e.g. in a phone, or if the user is wearing a microphone). More distant microphones (e.g. on a table or wall) will tend to increase the number of errors.

**What is Speech Recognition?**

Automatic speech recognition is a process by which a computer takes a speech signal (recorded using a microphone) and converts it into words. Speech recognition is a hard problem for a number of reasons:

* Many different words can be spoken. The average person uses thousands or tens of thousands of words.
* In speech the boundaries between words are not obvious - one word runs on into another. So the problem is one of finding the words as well as identifying them. This usual case is called continuous speech recognition. Sometimes, to make the problem easier, systems demand that people leave pauses between words. This is an unnatural way of speaking. Recognizing such speech is referred to as isolated word recognition.
* When people speak casually or conversationally, it is a lot more difficult for the computer to recognize what they are saying compared to when they are dictating or reading from a script. We say that the acoustic variability is much greater.
* It is more straightforward for a system to be speaker-dependent - able to recognize the speech of a particular speaker. But it is much more useful if a system is speaker-independent - able to recognize anyone.
* Speech recognition works much better in a quiet room with a nearby microphone. If this is not the case, then other sounds may also be recorded and it is much harder to recognize such noisy speech.

Speech recognition systems work best in particular applications where a person is expected to be speaking about a particular subject (e.g. booking a doctor's appointment). In such cases, the speech recognition system can take advantage of the fact that there a person will most likely talk about only a very limited number of things.

The University of Edinburgh has a large group of researchers working on speech recognition. The work ranges from very basic research (building mathematical models of how speech works), through research into recognizing the speech of elderly users, to recognizing speech recorded using distant microphones (e.g. on a table top).

**Basic Principles of Speech Recognition**

To understand Speech Recognition software, we must understand the basic units of language and the methods to interpret it.

**Speech**

The smallest unit of spoken language is known as a **Phoneme**. The English language contains approximately 44 phonemes representing all the vowels and consonants that we use for speech. We can take the example of a typical word such as **moon** which can be broken down into three phonemes: **m, ue, n**.

**Interpreting Speech**

To interpret speech we must have a way of identifying the components of spoken words. Phonemes act as identifying markers that within speech since they remain at a constant value and can therefore be broken down further.

An algorithm has to be used to interpret the speech further. The University of Edinburgh is using[**mathematical models**](http://www.match-project.org.uk/resources/tutorial/Resources/Glossary/Glossary.html#MODEL) (Hidden Markov Models) to do this.

These models work on the basis of probability. To create a speech recognition engine, a large database of models is created to match each phoneme. When a comparison is performed, the most likely match is determined between the spoken phoneme and the stored one, and further computations are performed. This allows the system to break down the exact word that was uttered, to understand in what context it was used, and to understand the grammar if the word is part of a sentence.

**How Does Speech Recognition Work?**

A speech recognizer consists of a number of components. These are learned from data, using a **Speech Corpus** consisting of recordings of speech and their textual transcriptions. The Speech Recognizer learns to make correspondences between sounds and words.

**Signal Processing**

This processes the signals recorded by the microphone into **Feature Vectors** that provide a snapshot of what is going on in the speech signal, emphasising those features that are relevant to speech recognition. Typically, 100 feature vectors per second are produced.

**Acoustic Model**

This takes the stream of Feature Vectors and turns it into a stream of Phonemes (or Phoneme Hypotheses). A Phoneme is the unit that is used to construct words, and corresponds to a particular speech sound. An important aspect of the Acoustic Model is that it does not make definite decisions about what the stream of Phonemes is, but tells us how likely any particular Phoneme is at a point in the speech signal.

**Lexicon**

This tells us how words are constructed as a string of Phonemes. Alternative pronunciations are also possible.

**Language Model**

This states what sequences of words are likely and what are not. Just using a grammar is not possible, since people may say anything!

# Voice Recognition Tools Review. Alexa, PocketSphinx, Google API or Project Oxford?

Remember back when every sci-fi book, TV series and film had a computer that could be operated by voice? The hero simply speaks, and the computer answers (though, it [doesn't always give the answer the hero is looking for...](https://www.youtube.com/watch?v=ARJ8cAGm6JE)). Well, the future is here - and there’s much more to it than you probably thought when you were a kid.

# Why voice control matters

Well, if it were only a fancy feature, nobody would actually implement it - there are so many more useful things you can do with your developers’ time (and your app budget) than building features that don’t benefit the final user. However, voice control is really important for the following reasons:

* **Accessibility**. When building apps, you should always keep in mind that not all users will have perfect vision. Voice control makes operating the app easier for users with visual impairments.
* **Safety**. A hands-free interface is a huge convenience e.g. while driving a car or performing any other task that requires both hands and a high level of concentration. With the help of voice control, your app can be operated by more users at any time.

Sounds like something you wish your users had? There’s one more way to use voice control:

* **Interaction**. When building machines such as robots and moving vehicles, speech-recognition enables the user to communicate and control the devices easily. This is a very specific case of voice control implementation. To help you familiarise yourself with the topic, I’ll introduce you to the whole process.

### How to make a computer understand you

First of all, you need to understand the difference between **speech recognition** and **natural language processing**.

**Speech recognition** converts spoken word to written text. Using a speech-to-text (STT) engine, you can dictate messages or emails to your device and then send them. You can also use text-to-speech (TTS) techniques to imitate the voice. For example, with Google Translate’s TTS you can check how a word is supposed to sound.  
**Natural Language Processing** is a much more advanced field of computer science that is concerned with understanding the meaning of the user’s phrase. It uses artificial intelligence and machine learning to catch what you actually meant when you spoke to the device.

If you want your app to let the user order a pizza or book tickets on the next flight to Hawaii, it needs a natural language processing engine. The app can only help you when it understands the true meaning of your request. It should not only ***HEAR*** you, but also ***UNDERSTAND***you.

### Research requirements

Let’s assume that our goal is to create a personal assistant mobile application called Lucy (“à la Siri”). We’ve defined the requirements we need to achieve for the best user experience. The app needs to have the ability to:

* set a wake-up word (“Lucy”). The ideal solution should have a built-in feature for setting a custom keyword, which allows us to leave the push-to-talk button behind.
* answer open questions. This feature is connected to the application’s ability to recognize natural speech and respond to the request with an action.
* return a voice answer. The ideal solution should have a built-in “text-to-speech” engine. It should return an out-loud answer to questions and requests.
* return JSON code. This code could be used later to communicate with other applications.

We’ve compared the following technologies: PocketSphinx, Project Oxford (Microsoft), Alexa (Amazon), Google Voice Interaction, wit.ai, api.ai, IBM Watson, Nuance, and Sensory.

### PocketSphinx

The most difficult issue mentioned above is the first one: setting a custom wake-up word. We don’t want to use a cloud service for this feature because continuous connection with the Internet will negatively affect battery life. Instead of cloud services, we use [PocketSphinx](http://cmusphinx.sourceforge.net/" \t "_blank) - an open source solution for Android.

We are using PocketSphinx as an offline solution only for keyword recognition. Later, we’ll require some cloud service to handle the request.

Pros:

* With Sphinx it is possible to set a custom wake-up word.
* Sphinx works offline (lower battery consumption).

Cons:

* PocketSphinx is not accurate enough to get the effect we want to achieve. It reacts not only to the “Lucy” wake-up word.
* There is a pause after Sphinx recognizes a keyword and launches the cloud service.

### Project Oxford (Microsoft)

[Project Oxford](https://www.microsoft.com/cognitive-services/) is a speech-to-text engine from Microsoft. It works well enough, but we need to implement additional API’s to achieve the desired functionality.

Pros:

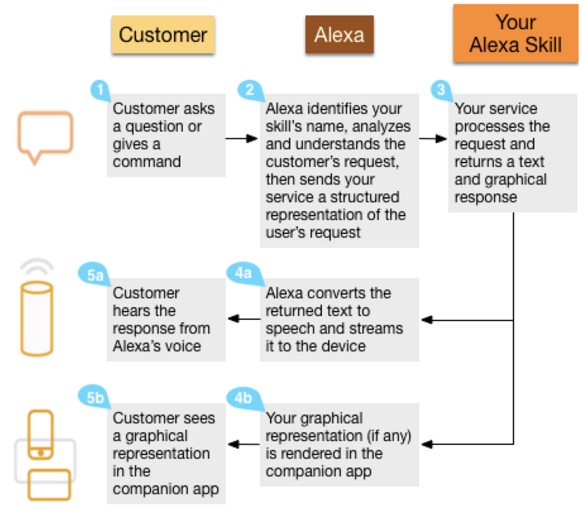
* Powerful in combination with [LUIS](https://www.projectoxford.ai/luis) (Language Understanding Intelligent Service) and [CRIS](https://www.projectoxford.ai/cris)(Custom Recognition Intelligent Service).

Cons:

* At the moment, we have a beta version of LUIS and CRIS, and we do not recommend using beta versions for commercial products due to stability concerns.
* We have to manually implement intents.

### Alexa

Alexa Voice Service (AVS) is a cloud speech-recognition service from Amazon. It is used in Amazon’s Echo. Here’s how it works:



“Alexa” is the wake-up word and starts the conversation. Our service gets called when customers use our invocation name, such as: “Alexa, ask Lucy to say hello world.” This example is a simple command-oriented one. ASK also supports more sophisticated multi-command dialogues and parameter passing. The above example would work like this:

* "Alexa" is the wake word that starts the conversation.
* "Ask...to" is one of the supported phrases for requesting a service.
* "Lucy" is the invocation name that identifies the skill we want (in our case it’s the name of our app).
* "Say hello world" is the specific request, question, or command.

Pros

* Alexa provides a set of built-in skills and capabilities available for use. Examples of built-in Alexa skills include the ability to answer general knowledge questions, get the current time, provide weather forecast information and query Wikipedia, among others.
* Returns an mp3 with an answer.

Cons

* To get custom intents within AVS it is necessary to create, register and test them with the Alexa Skill Kit.
* Complicated documentation.
* To capture the user's utterances, the device needs to have a button to activate the microphone (push-to-talk). We’ve contacted Amazon and got information that far-field voice recognition or using a spoken word to trigger activation of the Alexa Voice Service is currently unavailable.

### Google Voice Interaction API

This API works in a similar way to Alexa. [Google Voice Actions](https://developers.google.com/voice-actions/interaction/) recognizes many spoken and typed action requests and creates Android intents for them. Apps like Play Music and Keep can receive these intents and perform the requested action. Our app can support some of these actions too:

* Define an intent filter.
* Handle the intent in the app.
* Update the completion status of your app.

Unfortunately, with this solution we are not able to change the “OK, Google” wake-up word. The Google API is insufficient for us, as it only allows us to handle intents:

* User says ‘ok, google’
* Android will open its default search engine
* The user speaks a keyphrase that is matched to our app’s intents
* It will launch the application and handle the intent.

### Wit.ai

How [wit.ai](http://wit.ai/) works:

* Provide a sentence we want the app to understand. Then either select an existing intent from the Community or create our own.
* Send text or stream audio to the API. [Wit.ai](http://wit.ai/) gets structured information in return.
* [Wit.ai](http://wit.ai/) learns from usage and helps improve configurations.

Pros:

* Returns JSON.
* Already has a large number of built-in intents.
* Ability to learn from the user.

Cons:

* Not stable enough. During the test it was shut down after 30 sec.
* Not so comfortable to use.

### Api.ai

The [Api.ai](http://api.ai/) platform lets developers seamlessly integrate intelligent voice command systems into their products to create consumer-friendly voice-enabled user interfaces. We made a test application using [Api.ai](http://api.ai/) and it was closest in quality to Amazon Echo.

Pros:

* Easy to implement, clear documentation.
* Ability to learn and adapt (machine learning).
* Complete cloud solution (only without keyword activation): Voice recognition + natural language understanding + text-to-speech.
* Returns a voice answer.
* User friendly.

Cons:

* It has only push-to-talk input.

### IBM Watson

IBM Watson is a powerful tool for machine learning and analytics. Basically, it focuses on analyzing and structuring data and has [speech-to-text](http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html#how-it-is-used-block) and [text-to-speech](https://text-to-speech-demo.mybluemix.net/) solutions. It is good for big data analysis but it doesn’t fit the purpose of our application.

### Nuance

Nuance provides many voice recognition and natural language processing services. It has a ready solution for mobile speech-recognition: [VoCon Hybrid](http://www.nuance.com/for-business/speech-recognition-solutions/vocon-hybrid/index.htm), which could solve our most difficult issue - custom keyword recognition.

Pros:

* Always-listening mode with keyword activation removes the need for a push-to-talk button - a key advantage of this technology.
* All-inclusive main menu (a1M). Enables all commands to be spoken in a single utterance on the main menu. We want a similar interface.

Cons:

* Closed technology. It is not an open source API - if you want to use it in your project you need to contact Nuance and ask for samples to test it.
* Complicated documentation and set-up. We got the sample of this solution to test and we were unable to launch it and make it work. This solution requires additional time for testing.

### Sensory

Sensory is another expert in the speech-recognition field. TrulyHandsfree is one of the solutions they offer and it looks promising. It is a good alternative to PocketSphinx and we recommend it if you want a high quality application. Unfortunately, it’s not free.

Pros:

* Always-listening mode. With TrulyHandsfree you can set up a custom wake up word.
* High accuracy. The technology can respond to commands delivered from as far as 20 feet away or in high noise conditions.
* Ability to include pre-built commands. This is a great feature, you can set predefined commands and there’s no need for an Internet connection to handle them.

Cons:

* Closed solution. If you want to set a custom keyword, like “Lucy”, you need to contact Sensory.
* This solution solves only our first issue. It does not include a natural language processing engine and requires other services to handle complicated requests.

### Functionality comparison

### Conclusions

To sum up, we found the following:

* The Google Voice Interaction API and Alexa Voice Service provide similar functionality. It is possible to launch an application and make an intent through their systems but it’s impossible to customize it in the way we want. With these services, users will communicate mainly with Google and Alexa and not with our application. We believe that it is better not to use these solutions as they don’t allow us to build a strict connection with the application as an independent product.
* [Api.ai](http://api.ai/) and [wit.ai](http://wit.ai/) work in a similar way; they solve open questions and can return JSON code. We did tests and made a demo application and decided that [Api.ai](http://api.ai/) is better as it is more user friendly, works smoothly and provides us with ready solutions for voice recognition (text-to-speech), natural language processing and text-to-speech. We built a test application and obtained appropriate speech-recognition functionality. Unfortunately, [Api.ai](http://api.ai/) only uses a “push-to-talk” approach.
* Nuance provides a ready solution - VoCon Hybrid. The main advantage of this technology is that it has an always-listening mode and the ability to set up a custom wake-up word. It also has numerous advantages, such as:
  + An all-inclusive main menu (Enables all commands to be spoken in a single utterance on the main menu).
  + Multi-lingual and partial phonebook name selection (Recognizes partial contact names for multi-lingual phonebooks).
  + Natural language understanding (Recognizes natural speech, eliminating the restriction to predefined commands for all VoCon Hybrid languages).

However, Nuance technology is not available for free usage, and if we want to use it in the project it is obligatory to contact them.

* Sensory’s TrulyHandsfree is a great alternative to PocketSphinx, but it is also not free and fulfills only one item on the list of requirements.

### Our recommendations

* At this point, [api.ai](http://api.ai/) has the best result in tests and we would recommend its use as a cloud service for voice recognition and natural language processing. One of the biggest advantages of this service is that it is a ready to use product, unlike the other cloud services. [Api.ai](http://api.ai/) allows us to use an easily integrated solution.
* [Api.ai](http://api.ai/), like the rest of the cloud services, does not solve the wake-up word issue. At this point, we see the following options:
  + Use Sphinx as an offline solution, and make efforts to get it working as well as possible + [api.ai](http://api.ai/) for speech-recognition.
  + VoCon Hybrid from Nuance - as a ready solution for wake-up word + [api.ai](http://api.ai/) for natural language processing (answering open questions and returning voice answers)
  + Use Sensory’s TrulyHandsfree for wake-up word recognition + [api.ai](http://api.ai/)
  + Use a hardware solution for keyword detection. Implementation of a hardware solution could increase the quality of wake-up word detection, then we could use [api.ai](http://api.ai/).

# Google to Offer Speech-to-Text API

[News](https://www.programmableweb.com/category/all/news?articletypes=news), [Mobile](https://www.programmableweb.com/category/mobile), [Voice](https://www.programmableweb.com/category/voice)

Mar. 24 2016

By [Eric Zeman](https://www.programmableweb.com/profile/ericzeman), ProgrammableWeb Staff

Google's Cloud Speech API will allow developers to convert audio to text within their own apps. The offering from Google will bring its neural network smarts to apps large and small, and opens up a wide range of interesting new possibilities. It also brings the fight to Nuance Communications' front door.

Google is providing access to the limited preview of the [Cloud Speech API](https://cloud.google.com/speech/)through its developer website. Developers can take advantage of the API for free, for now, though presumably Google will start charging for access at some point. The API includes a number of key functions.

The automatic speech recognition is powered by learning, networks computers. Google claims it has unparalleled accuracy, and the learning computers become more accurate over time and more people use the API. At launch, the Speech API recognizes 80 languages with some regional variants. Google didn't say how big its vocabulary is other than to call it "extensive." [Nuance's mobile SDKs](https://developer.nuance.com/public/index.php?task=home), by way of comparison, only cover about 40 languages.

The API can capture audio from a microphone or in pre-recorded audio files, such as PCMU, FLAC, and AMR. Voice recordings are sent to Google's servers where they are transcribed into text, which is then streamed back to the app in real time. Google didn't say if or how the API handles voice recognition in an offline environment. The API can recognize spoken language even in noisy environments without hardware or software noise cancellation. Google says developers can set parameters to filter out inappropriate content if so desired. Developers can upload and store audio files. A future release of the API will allow developers to integrate those files with Google Cloud Storage.

The Cloud Speech API accesses the exact same toolset that Google uses for its own speech-recognition and voice-command tools in Google Search, Google Now, and the Google Keyboard. Anyone who's used Google voice search knows how quick and accurate its performance is. Developers can take advantage of this API to not only capture spoken words as text, but add support for voice-based commands.

Given how Google's description of the API's workflow says that the API can accept either real-time speech (to which it will respond with a text stream as the speech is recognized) or a complete audio file, it's possible that the API will be a streaming API instead of RESTful one. If that's the case, it's also possible that it will rely on [Google's Pub/Sub streaming API technology](https://cloud.google.com/pubsub/). In terms of abstracting the API, Google has yet to indicate if it will also provide SDKs for Android, iOS, or other mobile or server-side platforms.

Nuance has individual SDKs for Android, iOS, and Windows. (Google's API differs from the Alexa Voice Services API from Amazon. That API allows hardware makers to add the Alexa AI to their devices and nothing more.)

Access to the Google Cloud Speech API is limited, but Google didn't say how limited. Developers can [sign up to test the preview](https://services.google.com/fb/forms/speech-api-alpha/) at no cost.