Fun with Speech

Hi everybody, thanks for coming.

First of all I want to warn you that this is not a serious talk. When you’re being serious, you think in a businesslike way, you go in the correct direction: identify a need, then find a solution. I’m going the other direction here: here’s some interesting technology, available for free, what can we make with it? Who cares if it’s practical, or even if it’s anything users want. Nobody wanted a spring that could walk downstairs, nobody was in need of a sticky goo that could “pick up” the Sunday comics.

Also, I’m not going to go deeply into code. I’m not even going to show you the inner workings of anything. What I am going to do is show some software building blocks and a couple of apps you can build with them.

The building blocks are these APIs. I have demos of each that are minimal, by which mean the code is just big enough to do something, and no bigger. Each of them shows how one API works on its own. The idea was to keep them clean and free of styling or inessential features, easy to pick up and drop into a larger project or chain together as I’ve done here. That’s my redeeming social value.

The toy apps are cobbled together from these APIs. The code to put together the APIs is not that complicated, and in fact I’m not going to try going through it now. I call them toy apps for two reasons: 1) because they really don’t do anything all that useful, and 2) they are like skill-based toys in that you have to learn an otherwise useless skill to enjoy them.

There are apps that change the world, and then there are other apps that have to wait for the world to change around them. These are the second kind. You might say they’re ahead of their time.

Last year about this time, I got excited about speech as a way to interact with a computer. There’s something magical about it. You don’t have to touch anything, move anything, do any work at all on objects external to you. Just saying something causes my computer to react. It’s literally action at a distance.

Speech has a lot of great qualities. It’s easy and high bandwidth. We would usually rather be talking than typing. If you talk for a living, you are lucky.

If you are interested in making interfaces that are easy to use, one aspect of ease-of-use is about ergonomics – reduce the number of clicks and keystrokes. Reduce the physical effort needed to get something done. One-click is a nice goal. Speech offers the possibility of zero clicks, which by traditional standards makes it a truly effortless interface. No clicking, no typing, just say the magic words.

When you start thinking about using voice input, the first thought is of voice commands. This is hard to do well. It can be incredibly frustrating for users. Because of the errors in speech recognition, we end up limiting our vocabulary, which means the user has to know or guess the available commands. If the computer gets a voice command wrong, it will do the wrong thing, or nothing at all. But the big problem is the same as for general transcription: how can you make it easy to recover from the errors in a) speech recognition, which will likely be unavoidable for some time into the future, and b) human speech production, which will never be avoidable, since we are human and we make mistakes.

Look at the difference in effort from the two interfaces, voice and text, when dealing with an error. With text, you hit up-arrow to bring back your failed command, go left a bit, and retype just the part that has the mistake, and resubmit it. With voice, there’s no go back, so you have to cancel your previous attempt and try again, and that can be super frustrating.

Here is Fred Armisen being a Glasshole on SNL, trying to get it to connect to the CBS wifi. The password is Peacock, peacock, peacock. Frustrating as hell. To see how silly this is, imagine a text input that required you to retype everything whenever you made a mistake. The keyboard really is a great interface for precise input. Voice input causes more trouble than it’s worth when you need precision.

The standard methods of using speech are for voice commands or transcription. But there is a third direction that might lead somewhere, some day: overheard conversation. That’s what we will use as input. With the new always-on voice recognition chips, your cellphone and computer will be able to pick up on any speech going on around them. There is this thing called Life Logging. And that brings us to my question:

Can a computer make itself useful by listening to your conversations with other humans?

Enough for the preamble. Let’s start the demos.

First, here is the easiest way to use speech in a browser.

“Town and Country”

It’s nice that it’s so simple, just this one addition to the input tag, but it’s hard to make use of it. It stops recording the moment you pause more than about half a second, and it gives up if you speak too long. It only accepts short phrases. There’s no feedback while you talk, so you don’t know if you are speaking clearly enough. Since it only works on short bursts of speech, you have to keep your hands on your mouse and keyboard, which means it can’t be used to make a hands-free interface.

Then, last year, Chrome came out with support for the Web Speech API, or at least, the speech recognition part of it, which is the most interesting. Let’s take a look.

“This time you are able to see what it thinks you are saying, while you talk. You get to see its various guesses about what you are saying. You can watch as the language model finds the most likely ngrams. And you can pause and it keeps on listening. So this really allows what you might call a passive voice interface.”

I want to show a little code. This is how the hypotheses are displayed. We get an onresult callback many times a second with results in two sections: one is the words speech recognition is sure about, called the final transcript. We get informed about new items that have made it to the final transcript only once, so we accumulate them in a buffer. The other is the system’s guess about what you just said. That gets refreshed on each onresult event, and displayed in the gray span. The result is that words bounce around in the gray area until things settle down and then they move to the black.

Okay, here is the central observation I’m making. If we can get conversational input from users, then we can apply standard information extraction tools to it, even though those tools were developed to handle written text. Here are a few. I’m going to use two of these: named entity recognition, and machine translation. I’ve also recently added sentence segmentation.

So let’s look at NER first. I’m using Alchemy. They make 1000 queries free per day, perfect for a toy like this.

“Barbara Mouchabouche moved with her husband Bob from Everett, Washington to Olympia”

You can see that it’s not just using a dictionary of names, it was trained to decide in this context that Mouchabouche is a name.

So now we have continuous speech recognition and named entity recognition. What can we do with them?

Well, clearly the intelligence community has a serious interest. With metadata about phone calls, they can build a network of the people you talk to. With NER on the content of phone calls, they can build a network of the people and things you talk about. There are research grants for that.

So it has an evil use. But can it also be used for good? What could your computer do for you if it could listen to what you say to other people? Hold that thought while I go off on something completely different.

If you have your own fake news show, then you have interns on your staff who manage this rectangle over your shoulder. People tell me it’s called the inset, a disappointing name.

Have you ever wished you had the same thing? Your friends don’t get your references? You end up explaining things over and over. Then you my friend feel the pain my app will relieve! It’s the first app in a new category, Computer Aided Conversation. The computer listens to your human conversation, when it hears a name, it searches for images of the thing you mentioned.

Yes it’s ridiculous, but for the sake of argument let’s say we’ve decided to show images based on names you mention in conversation. We have two of the three APIs we need. The missing one is search. How do we get relevant images? Microsoft has a free level for Bing that’s perfect for this. Take a look. And Flickr has a free API.

Now we’re ready for the TalkShow demo.

The setup is we’re trying to decide between two movies. First, there is the Joaquin Phoenix movie where he’s a computer nerd and Scarlett Johansson is the disembodied computer voice he falls in love with. By the guy who did Being John Malkovich. Then there’s the movie by Martin Scorsese, with Leonardo DiCaprio, Jonah Hill, Margot Robbie (who looks a lot like Jamie Pressly), and Matthew McConaughey. I heard someone say he’s this generation’s Robert Mitchum. Plus it has the French actor Jean Dujardin.

That could have been me talking to a friend. I wasn’t really using the computer. It was just there, listening, acting when it had something to offer to the conversation. Like an attentive servant.

Wittgenstein had something called the Picture Theory of Meaning, where a statement is meaningful, either true or false, if it pictures a state of affairs in the world. The verb “picture” is being used metaphorically here, but I’m going to go be unsophisticated and take it literally. Let’s say a statement is a kind of picture. If a statement pictures the world, what do the proper nouns in it do? How about picturing the things they refer to, using actual pictures? If that makes any sense to you, then you will believe me when I say this app is the first step in a grand project to turn speech into pictures on the fly.

In TalkShow, a name is meaningful if we can find a picture of that thing on the internet. Of course, with a little effort, we could add pictures of your family and friends, so they pop up when you talk about your vacation. Or a graph of last year’s sales figures for the office.

The theory that says this could be of value is that communication involves, among other things, calling up pictures in your mind of the things we’re talking about. Having a real picture on screen can avoid the miscommunication that happens when our two mental images are fundamentally different. We may have different objects in mind. That’s why, when Jon Stewart mentions Pat Buchanan, he shows a picture of him, to give you a clue as to whom he’s talking about.

In any case the benefit of seeing what you are talking about is small. So the effort has to be correspondingly small. At the moment, it means opening up your laptop and getting everyone to hover around it.

We have a general problem in our time with screens. We have many ways of handing them, none of them optimal. There are screens of every size.

Here we are now with our little screens in our little hands. When you find a video you want to share with everyone else, you end up passing your cellphone around and each person watches the video in turn. Hilarious. Someday this is going to change and it will look so 2014.

In the peace talks on Syria in Geneva, they scattered some screens around on the floor. A reasonable solution to the problem of sharing, when people are in a conversation configuration facing each other rather than a presentation one, where everyone is facing the same direction.

Some public spaces are being equipped with video walls. At some point, we will want to pair our cellphones with available screens.

Living rooms too.

One of the things I like to do is extrapolate much too far into the future from limited information in the present. I predict screens will get bigger and will take over more and more of our visual space during the day.

In 50 years, will screens cover every vertical space? All interiors will become holodecks? That’s when this app will find its niche. Ambient computing – computers take whatever input they can find, and try their best to give up something we’ll want.

Another warning to the reader: my predictions don’t always come true. When I saw that they started putting advertising on the floors of supermarkets, I thought it wouldn’t be long before they cover the floor wall-to-wall. Didn’t happen. The first time I got a robo-call, I thought what’s to prevent them from flooding the phone lines with these, until it becomes a general denial of service attack? We avoided that one too. So I may be wrong about screens covering every vertical surface.

Okay, so that’s something you can do with image search on names found by named entity recognition operating on the output of speech recognition.

The next app combines speech with video conferencing using machine translation.

For the communication platform, I picked Web Real Time Communication. It’s peer-to-peer voice, video, and data communications through a browser without plug-ins. You don’t have a phone number, you just navigate to the same url as someone else and you’re in contact.

The video isn’t really essential for my purposes, but this platform lets me control the web page displays for each user during a call. A purely audio call, with translation, is very hard to manage. Combining video conferencing and speech recognition is already interesting. Just getting an accurate transcript of the words spoken during a serious conference call would be great, but only if it is accurate. Add translation and you increase the power, but also increase the risk of error.

MT is particularly challenged by speech. Statistical Machine Translation models are trained on text that was translated by humans for their own human purposes. People working in MT were lucky to find all this existing data to leverage. But that meant they worked with text that was especially good. Written text that was worth translation by a human usually passes a bar of grammaticality. Sentences and paragraphs are well-formed. Speech on the other hand follows rules of grammar that are different from text. But it is rarely translated, except at the UN, which is a special kind of speech, so real-world training data is scarce. You might think of movie subtitles, but those are notoriously both compressed and noisy.

For translation, I’m using Bing. You have to register, but they give you 2 million characters a month for free. Bing works over JavaScript by issuing you a token that’s good for 10 minutes. There’s no cost for a token, except the back-and-forth to the token-issuing server, so you could get a token on each request. But you can remove that overhead by getting a 10-minute token. During those 10 minutes, you go directly to the translation server, using the token. So you can store your account id safely on your server, and use it to get the temporary token to pass down to the browser. The script requests a 10-minute token on load, and runs a timer that fires every 9 minutes to ask for another one. If I can’t get a new token in a minute then I just fail disgracefully.

So we add WebRTC and translation to our toolkit. We will also add text-to-speech, and those three will let us make a classic toy, the translating telephone.

For text-to-speech, it will soon be possible to use the Web Speech API. Until then, the tts in Google Translate is pretty nice. Several languages have real voice synthesis. You have to keep your buffers below 100 characters, and that produces some unnatural sounding transitions, but it’s okay for now. The quality of synthesized speech has improved dramatically recently, and it’s really the least of our worries.

Now we’re ready for the last demo. This one actually serves a real purpose. Unfortunately, it also places too heavy a demand on its underlying technologies. It requires a precision not yet achieved in speech recognition and machine translation. For one thing, translation means we have to make sense of everything being said, not just the names. More importantly, this interface runs two error-prone processes in series. Both ASR and MT have high error rates. Doing one after the other is just asking for trouble.

First I’m going to show you some things you can do with it, without having another person online. The main purpose of the app is to translate everyone’s speech during a conference call to your own language. In order to allow a single-user demo, I added this illogical feature to translate yourself to yourself. There’s no practical purpose to this feature, it’s just for fun.

If I click on my image, it maximizes, and the text shows up as subtitles on my video. This is “presentation mode”, appropriate when one person is presenting something without interruption. There’s also a multi-user “conversation mode”, where you see a transcript of everyone’s input, appropriate when several people are speaking and there is a lot of back and forth.

In either mode, speech is captured as source text in your native language and sent to all the other participants. Then it is translated on their browser into their language.

To end with, here’s how it works between and English speaker and a Korean speaker.

So that was 8 JavaScript APIs and 2 apps. I hope you get a chance to try them out and make your own useless inventions! Thanks.