# INTRODUCTION

## Welcome in Te Reo Māori

Tēnā koutou katoa.
Ko Barwon te awa. Ko Kathy Reid tōku ingoa.
Ngā mihi nui ki a koutou!
Kia kaha, te reo.

## Welcome in English

Greetings everyone. My name is Kathy Reid, I'm from Geelong, by the Barwon River.

Thank you so much for joining this talk today, when there are so many excellent talks on at the same time. Firstly I'd like you to join me in thanking ROOM MONITOR & AV CREW NAMES, for volunteering their time. We appreciate all of your help.

## Background

Up until recently, I led Developer Relations at Mycroft.AI, an open source voice assistant startup based in Kansas City, in Missouri. Mycroft are quite unique in that they provide an end to end open source voice assistant stack.

Many of the voice products that are available today make voice technology look really easy, and smooth, and simple, but I can assure you that it's not! Today we'll be diving into some of the key challenges in the end to end voice stack, how those challenges are *currently* being tackled, and what we're likely to see in the long term.

If the live demo gods are on my side, we'll also do a live demo of some of the Mycroft features.

## Questions

I'll be delighted to answer your questions, but please do save them up until the end of the talk.

# OVERVIEW

## Voice within the context of evolving user interfaces

Not so long ago - a couple of decades perhaps, speaking with a computer was relegated to the realms of science fiction. We saw cars that could talk;

IMAGE: KITT in Knight Rider

computers that could talk - "hello computer" *in Scottish accent*

IMAGE: Star Trek computer

and even holograms that could talk;

IMAGE: Time Trax

Just like many of the technologies showcased in these series - Knight Rider, Star Trek, Time Trax, voice user interfaces are quickly moving from science fiction to science fact. Advances in processor speed, compute power, machine learning and neural networks to act as a classifier are quickly making voice user interfaces a reality.

However, commercial, proprietary solutions have led the way here - Apple, with Siri, Amazon with Alexa, Microsoft - well until recently, with Cortana, Samsung with Bixby. There are many reasons for this - which I'll discuss throughout the presentation, but like many of you, I'd really like to see some open source solutions available which rival the maturity and feature completeness of the proprietary solutions, while **not** using the collection of private data for profiling and advertising purposes.

## Introduction to the voice stack

So,

*How many people in the audience are familiar with some type of speech recognition or text to speech service? count how many OK, that's great!*

OK, this will probably be a recap for many of you, but for the benefit of those who are new to voice, I'm going to provide an introduction to the voice stack.

Just like there are **stacks** of software for things like web servers or databases, or application servers, voice stacks need certain layers as well.

They're described in this diagram, by my Mycroft AI colleague, David Smehlik.

IMAGE: Anatomy of a voice interaction

In a voice interaction you begin with a **Wake Word** - also called a **Hot Word** - which is used to prepare the voice assistant to receive a command. Next, we use a **Speech to Text** engine to transcribe an **Utterance** from voice sounds into written language. Next, we use an **Intent parser** to determine what *type* of command the user wanted to execute. Then, we select a command to run and execute it. Next, we turn written language back into voice sounds again using a **Text to Speech** engine.

Does that make sense? OK, great.

Now, I'm going to dive a little deeper into each of the different layers of the voice stack, and we're going to explore some of the challenges at each layer of the voice stack, using open source software examples.

# WAKE WORD

SLIDE: Wake Word summary

## PocketSphinx

One of the earliest Wake Word engines used was PocketSphinx. PocketSphinx is part of the broader CMU Sphinx project from Carnegie Mellon University. PocketSphinx recognises Wake Words based on something called *phonemes*.

What's a *phoneme*? Good question!

## Phonemes

IMAGE: Phonemes

A phoneme is the smallest unit of sound that distinguishes one word from another in a particular language.

SLIDE: Phonemes in the English language

There are about 44 Phonemes in the English languages, and you can see them all here.

Different languages have different phonemes – for example, it's hard to approximate the Indonesian trill "r" in English (this phoneme is the sound you make when you "roll" your r sound). Using phonemes for Wake Word detection can also therefore be a challenge for people who are not native speakers of a language – as their pronunciation may differ from the "standard" pronunciation of a Wake Word.

The other challenge with using Phonemes in Wake Words is that certain Phonemes sound very similar to each other.

SLIDE: Similar sounding phonemes

This is one of the reasons why Wake Words for voice assistants are often very different-sounding to other words - that is, they're *differentiated*. Not much sounds like `Alexa` or `OK Google` or `Hey Mycroft`. This is a deliberate choice to make it easier to distinguish between phonemes. If you're thinking about training your own Wake Word, then this is a factor you need to consider as well.

But phonemes are not the only way to handle Wake Words.

## Snowboy

Snowboy is another Hot Word detection engine – available under both commercial and open source licenses. Snowboy differs from PocketSphinx in that it doesn't use phonemes for Wake Word detection; it instead uses a neural network that is trained on both false and true examples to differentiate between what is and isn't a Wake Word.

## Precise

WEBSITE: https://github.com/MycroftAI/mycroft-precise

Mycroft AI's Precise Wake Word engine works in a similar way – by training a recurrent neural network to differentiate between what is and isn't a Wake Word. This is then trained on samples that *are* and *are not* Wake Words to improve the detection accuracy.

## Challenges with Wake Words

SLIDE: Wake Word challenges

### Always listening

Even though all of these Wake Word detectors work `on device` - meaning that they don't need to send data to the cloud, they are `always listening`. That means that if the device on which the Wake Word listener is connected to the internet, it is *possible* that Wake Word recordings are sent over the cloud. So this is something that you need to be very aware of with a voice assistant - make yourself aware of how that information is being used. At Mycroft for example, our users must opt-in before recordings are transmitted back to our servers for training to improve accuracy. If the user hasn't opted in, then the recording is discarded.

You may not have thought much before about the sort of sounds that you make while your voice assistant is recording, but it can be both *enlightening* and *confronting*. While training Precise, our Wake Word listener, I've heard all sorts of quite personal sounds -

- people engaging in intimate acts
- people having quite heated arguments, although I've never heard domestic violence - that would be a really difficult position to be in ethically
- I've heard people losing their temper with disobedient children
- and I've heard myself swearing like a sailor when the device hasn't been working properly

Do you want to hear what Wake Word recordings sound like?
Should we have a listen to a few?

Now I have no idea what we'll hear, so I apologise in advance if we hear swearing, or, other things that might violate the Code of Conduct - but there is that risk - we don't have any minors in the audience?

DEMO of the Precise Wake Word tagger - https://home.mycroft.ai/#/precise

## Haber's classification of context

That makes us think carefully about the *context* in which voice assistants are used. One of the frameworks that provides some guidance here is from a researcher called JONATHAN HABER. He's put together a useful categorisation of space called the `Haber Classification of Contexts`

SLIDE: Haber's classification of Contexts

You can see immediately from this that where a voice assistant is located can dramatically change the *context* that it exists in. For example, a voice assistant in a lounge room is likely to be in *social* space, while one in a bathroom might be in *personal* space, and one in a bedroom, say on a nightstand, could very well be in *intimate* space, and collecting recordings of intimate activity.

We're actually seeing some interesting responses to this.

IMAGE: Project Alias

This is Project Alias, and this is like a "hat" for Alexa or Google Home, and what it does is run a small fan - it essentially "blocks" the device from hearing the Wake Word it's programmed with. Instead, you set a different Wake Word on the Project Alias device, and it acts essentially as a Wake Word proxy - and outputs the "real Wake Word" when you say the "proxy Wake Word" - it's really clever actually.

Or, I mean, you could just use a voice assistant that doesn't sell your privacy to the highest bidder I suppose ;-)

## Accuracy

IMAGE: Wake Word accuracy

I touched on accuracy before, but it's worth expanding on here, as it really is a challenge, not just for open source Wake Word engines, but for all Wake Word engines. It's a key challenge for a number of reasons.

SLIDE: False positive vs false negative

A Wake Word engine has to be trained for these four states *go through states*. The way we do this is to train it on both positive and negative samples - that is, recordings that are, and are not, the Wake Word. It sounds really simple, doesn't it? Yeah... nah!

What we've seen happen in reality at Mycroft AI - I can't speak for other voice platforms - is that there are lots of **biases** in the Wake Word samples;

- We find that there are significantly more samples of male-sounding voices than female - by a factor of at least ten to 1
- In our dataset, there are significantly more samples of American accents, as opposed to European, Asian or Latin American accents

Combined, what this means is that if you're a woman, who's not American, you're *much* less likely to have the Wake Word engine correctly identify you. Really, the only way to combat these sorts of biases is to have a greater range of samples to train on.

However, that also presents its own issues. At one point we considered filtering out a percentage of male Wake Word samples, to try and remove some of the bias in the data set - but this would mean tagging users and samples with a gender flag or some form of identifier - and as a privacy-focussed company, that wasn't something that we were comfortable doing - similar with accents which might indicate cultural or ethnic heritage.

## Longer term solutions

Longer term, where I think open source Wake Word software is going is that *individuals* will train their own Wake Word - for example, recording a few dozen examples of their Wake Word, which is then trained against a database of recordings which are known *not* to be that Wake Word.

IMAGE: Government could access your data

This too has privacy implications though; if a Wake Word is trained to a specific individual, then could a voice assistant be used to identify that individual? That is, to distinguish one user from another? What access would the government have to that data, particularly given the recent passage of the Access and Assistance Bill (#aabill), at least in Australia? This might sound alarmist at this point in history, but we've increasingly seen warrantless access to platform data by agencies which have peripheral or tangential claims to use that data.

You can see that even in the solutions to the challenges with open source voice, there are *moar* challenges - it's like an origami problem, every time we unfold one part, there's another corner that needs to be unfolded.

So that's some of the challenges we have with Wake Words.
Let's move on to Speech to Text.

## SPEECH TO TEXT

SLIDE: Speech to Text

Accurate Speech to Text conversion is one of the most challenging parts of the open source voice stack.

Kaldi is one of the most popular Speech to Text engines available, and it has several “models” to choose from. In the world of Speech to Text, a “model” is a neural network that has been trained on specific data sets, using a specific algorithm. Kaldi has models for English, Chinese and some other languages too.

One of Kaldi’s most attractive features is that it works “on-device” - that is, the Utterance that the user speaks doesn't need to go up to the cloud to be transcribed into text - whch has obvious privacy benefits. Another upside to having on-device STT is that it reduces the overall round trip time of a voice interaction. If you've ever used a slow internet connection with an AJAX type web application, the principle is very similar - **responsiveness** is a key feature of a voice user interface that's delightful to use.

However, Kaldi *does* require significant computing resources. This means that it's generally not suitable for running on a Raspberry Pi.

SLIDE: Common voice languages

This is Common Voice from Mozilla, which is one of several initiatives that they have in the open voice space. So, Common Voice is collecting lots of samples of different voices to aid in Speech Recognition in different languages, and Deep Speech is the Speech to Text engine they have that uses Common Voice data, as well as other data sets.

As at the time of writing, the compute requirements for DeepSpeech mean that it can only be used as a cloud implementation – it is too “heavy” to run ‘on device’, although a lot of work is going on to try and get it to work on ARM architecture and embedded devices.

## STT Challenges

SLIDE: STT Challenges

## Training a model

One of the biggest challenges with Speech to Text is accurately training the neural network that it uses to *accurately* recognise words. There are several challenges with this.

We covered earlier in Wake Words that several English phonemes sound quite similar - B and P sounds, K and G sounds, S and Z sounds and so on - and we see this issue again in speech to text as well.

### Accuracy of the model

One of the biggest challenges with Speech to Text is training the model. Mycroft AI have partnered with Mozilla, enabling our community to help train DeepSpeech. We then pass the trained data back to Mozilla to help improve the accuracy of their models. Even then, we're finding that the DeepSpeech model for English is not yet accurate enough for it to be our default Speech to Text engine.

SHOW SCREENSHOT: https://home.mycroft.ai/#/deepspeech

Inaccurate speech to text is really frustrating for an end user who's using a voice assistant - it results in really poor *voice user experience* - so it's something we need to be mindful of.

### Accents and slang variation

One of the toughest parts to deal with in Speech to Text is being able to recognise accents. Even though we might speak the same language, there can be a lot of regional difference, even within the same country - consider for example a midwestern American accent - y'all - and the nasal twang of Brooklyn *speaks in twang*. Or if you're Australian, there's a distinct difference between northern, broader accents and more urban accents in Melbourne or Sydney.

*maaaaaaaaaaate*

That can be really difficult for STT engines to recognise correctly.

IMAGE: Voice assistants can't understand accents

So, what we find is that we end up with STT engines for different regional pronunciations of a language - such as US English, Australian English or Great Britain English. But even that doesn't address *specific* accent groups.

## Slang

Another really challenging aspect in open source voice is slang.

*How many people in the room identify as Australian?*

OK, for those of you who are *not* Australian, tell me what this means?

OK, that's a pretty close guess.

SLIDE: Bingle in Broady

```
There's been a car accident in Broadmeadows
and the Western Freeway is congested
back to the service station
and as a result I will be late
to the social function at Mr Thompson's.
```

What sort of rules or deductions did you use to get to that conclusion?

- abbreviations - Broady Broadmeadows, Thompsom, Tommo, service station, servo
- slang - bevvies for beverages, yeah nah mate

So you can imagine the challenge that presents for a Speech to Text engine.

## Where are we going longer term?

I'm not quite sure where these challenges will head longer term - what I *suspect* we'll see is a similar approach to Wake Words - where people will end up training individual STT models - that are unique to a person and their particular accent and vocal patterns. I don't think we'll see this in the next year or two though - the amount of data that is required to create an STT model, and the amount of training it requires is huge - and it would probably take several months of recording utterances, and then training them, to build an individual STT model - and it's going to be a lot less accurate than other models which aggregate the data of many individuals.

## The role of open source voice technologies and endangered languages

SLIDE: languages

Mozilla's DeepSpeech implementation – along with the related Common Voice data acquisition project – aims to also support a wider range of minority languages. This in itself is a differentiator from proprietary and commercially-oriented STT engines, which have primarily targeted languages that have a significant number of speakers - or where there is a commercial imperative to supporting a language.

Open source technologies have a role to play in making speech recognition and voice assistant technology more accessible, more available, to smaller language communities. Particularly in Indigenous and remote communities, keeping a language alive becomes more difficult as older generations die out, and younger generations often adopt the majority language of the region as they integrate into schooling and employment functions. There's a whole system of classification around vulnerable and endangered languages, which I won't go into here, but I really think that's an emerging research topic - how can speech recognition and voice technologies help save endangered languages?

Wouldn't it be wonderful if there was a voice assistant available in Yolgnu Matha, or Warlpiri or Tok Pisin or Javanese or Pitjantjatjara?

Well, we're working on it ;-)

WEBSITE: Mycroft Translate

This is Mycroft Translate, a platform where we're crowdsourcing translations in about 40 different languages. Mycroft Translate is based on a platform called Pootle, which is used by the like of LibreOffice and KDE.

This is a really early implementation of a translation suite, and we've definitely encountered a few issues so far.

SLIDE: Issues with Translation

## Line by line translation

At the moment, the way we're handling translations is to import the vocab and dialog files, and parse them line by line, then we present them to translators line by line. This means that they don't provide a lot of **context** to the translator, and the translator isn't able to translate them as a whole. This means that the translation itself can be sub-optimal.

Going forward, we want to handle translations file by file - this should held address both the context issue, and help create a better translation.

After all, we don't want to fall into the trap that Coke fell into ;-)

SLIDE: Kia Ora, mate

By mixing Maori and Australian, they've ended up saying "Hello Death" in Maori. Or maybe it's just honest advertising, I'm not sure ;-)

## Gender and hierarchy

Another challenge we've found with translating dialog and vocab files is that gender and hierarchy are difficult to convey. In some languages, particularly Romance languages, objects have gender. For example, in French a bridge is feminine and in German a bridge is masculine.
Longer term, we will likely amend the Mycroft Translate platform to be able to hold different translations for different gender and hierarchy situations.

So, that's a little bit about Speech to text, and how we're making Mycroft a multilingual platform.

# INTENT PARSERS

SLIDE: Intent Parsers

A strong voice stack also needs to ensure that the **intent** of the user is accurately captured. There are several open source intent parsers available.

Rasa is open source and widely used in both voice assistants and chatbots.

Mycroft AI uses two intent parsers. The first, Adapt, uses a keyword matching approach to determine a confidence score, then passes control to the Skill, or command, with the highest confidence. Padatious takes a different approach, where examples of entities are provided, so that it can learn to recognise an entity within an utterance.

## Intent parser challenges

One of the challenges with Intent Parsers is that of *intent collisions* – imagine the utterance;

`“Play the news”`

Depending on what commands or Skills are available, there may be more than one that can handle the intent. How does the Intent Parser determine which one to pass to? At Mycroft AI we have recently implemented our `Common Play Framework`, which assigns different weights to different entities, leading to a more accurate overall intent confidence score.

SLIDE: Common Play Framework

So that's a little bit about intents and how we handle those.

# TEXT TO SPEECH

SLIDE: Text to Speech

At the other end of the voice interaction lifecycle is Text to Speech. Again, there are several opensource TTS options available.

In general, a TTS model is trained by gathering recordings of language speakers, using a structured corpus – or set of phrases. Machine learning techniques are then applied to synthesize the recordings into a general TTS model – usually for a specific language.

MaryTTS is one of the most popular, and supports several European languages.
Espeak has TTS models available for over 20 languages, although the quality of synthesis varies considerably between languages.

Mycroft AI’s Mimic TTS engine is based on CMU Flite and has two voices available for English. The newer Mimic 2 TTS engine is a Tacotron-based implementation, which is a less robotic, more natural sounding voice. Mimic runs on-device, while Mimic 2, due to compute requirements, currently runs in the cloud.

Additionally, Mycroft AI have recently released the Mimic Recording Studio, an open source, Docker-based application that allows people to take recordings which can then be trained using Mimic 2 into an individual voice.

DEMO: Mimic recording studio

## TTS Challenges

SLIDE: TTS Challenges

### Natural sounding voice

The Mimic Recording Studio is intended to help solve one of the many problems with TTS - having natural-sounding voices available in a range of genders and languages / dialects.

In our experience, we've found that the key to a natural sounding TTS voice is that the initial recordings themselves need to be really consistent - they need to be done at the same amount of gain, otherwise the resulting voice is distorted. Another factor that we've found contributes to a natural sounding voice is a consistent *pace* in recording.

### Poor pronunciation

Even controlling for volume and for pace, the trained voice will inevitably get word pronunciation wrong, and so we've implemented a platform to allow the community to correct pronunciation, which is then validated.

WEBSITE: https://mimic.mycroft.ai/pronounce

# CONCLUSION

As you can see, there is an emerging range of open source voice tools becoming available, each with their own benefits and drawbacks. One thing is for certain though – the impetus towards more mature open source voice solutions that protect privacy is here to stay!

As a parting note, I'm going to leave you with this quote from someone I admire greatly.

SLIDE: Malala Yousafzai quote

SLIDE: Thank you