Natural Language Understanding/Translation

Gean Maidana Dollanarte

Abstract

Natural language understanding and translation is the process of computers understanding the syntax and semantics of a language. Natural language understanding and translation have helped create practical consumer products such as Google Translate, Apple's virtual assistant, Siri, and Amazon's virtual assistant, Alexa. A variety of factors have allowed natural language understanding and translation to flourish, including an increase in computing power, the availability of large amounts of linguistic data, and the development of highly successful machine learning methods. However, natural language understanding and translation are not perfect. It would be difficult for a bilingual person to not notice inaccurate translation when translating from one language to another, or to not notice that Apple, Amazon, or Google's virtual assistants do not pick up every word a person may say when speaking. Breakthroughs do happen though, such as Google Brain's machine translation system dubbed "GNMT," which significantly reduced the translation error rate to near-human quality when translating from one language to another.

This paper will primarily cover how machine learning is used to understand or translate languages. This paper will focus on the application of machine learning in consumer products that use natural language understanding or translation, such as Google Translate, Apple's Siri, and Amazon's Alexa.

Introduction

Because natural language understanding and translation systems are built on machine learning, it is important to understand what machine learning is and what neural networks are. IBM defines machine learning as, "A branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the ways that humans learn, gradually improving its accuracy" [2]. Machine learning models are built using neural networks. In his book, "Statistical Machine Translation," Philipp Koehn defines neural networks as, "A machine learning technique that takes a number of inputs and predicts outputs" [3]. Machine learning models are used to help computers understand language and machine learning models are also used to translate language. Therefore, machine learning and neural networks help build the foundation of natural language understanding and translation.

There are three well known consumer products that make use of natural language understanding or translation: Google Translate, Apple's Siri, and Amazon's Alexa. Google Translate makes use

primarily of natural language translation. Apple's Siri and Amazon's Alexa make use primarily of natural language understanding.

Google Translate in the past ran on a different translation system than it currently does. It was less accurate and as a result is no longer used. It was replaced by their current translation system called the Google Neural Machine Translation system. Google researchers have published papers and blog posts that describe, generally, the inner workings of how that system operates. Apple's Siri team has published a machine learning blog post that details, generally, the inner workings of how "Hey Siri," operates. Additionally, Amazon also has a blog post, similar to the Siri team, that details how Amazon handles speech recognition on their Alexa-enabled devices. Due to these consumer products being for-profit products, it is not in Google, Apple, or Amazon's interests to divulge the entirety of how these products operate, but they all provide enough useful information for us to understand how their natural language understanding and translation systems work on a surface level.

Google Translate

One of the most well-known consumer products that people have encountered is Google Translate. Almost everyone has encountered a sentence in a language different from their native language and has had to turn to Google Translate for help. While it is difficult to know precisely how Google Translate works in its entirety because it is copyrighted and proprietary technology, it is possible to have a general understanding of how it works. This is because researchers from Google have published papers discussing how Google Translate generally works using neural networks.

Google Translate has evolved a considerable amount since it launched in 2006. When Google launched Google Translate, it used Statistical Machine Translation as its translation system. Statistical Machine Translation had accuracy problems and a better solution was needed. In 2016, Google announced that it would be switching from Statistical Machine Translation to a Neural Machine Translation system called Google Neural Machine Translation, often abbreviated as GNMT for short. GNMT was an improvement over Statistical Machine Translation and to this day, Google Translate continues to operate using GNMT because the translation quality has been significantly improved [9]. The benefit that Neural Machine Translation provides compared to other techniques is that the system translates entire sentences at a time rather than translating a sentence piece-by-piece [7]. Figure 1 provides a glimpse into how Google Translate switching from a Statistical Machine Translation system to a Neural Machine Translation system increased the accuracy of translation.

Google Translate also offers additional features, all of which will also use the Google Neural Machine Translation system. In addition to the ability of typing text into an input box and receiving a translation, Google Translate also allows for the ability to speak and receive a translation. An additional feature is the use of photos which Google Translate will translate the text present in the image.

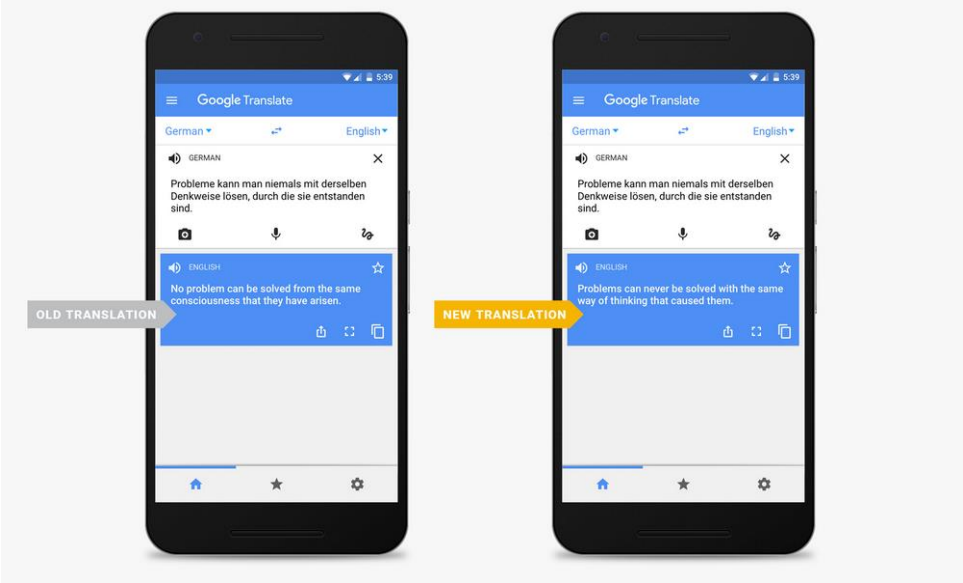

Figure 1: Google Translate output using the Statistical Machine Translation system (left) versus Google Neural Machine Translation system (right) [7].

An interesting aspect of Google Translate is the natural language understanding aspect. The team at Google asked the following question, “Can we translate between a language pair which the system has never seen before?” The researchers were curious about whether a sentence that the system has never seen before could be translated from Korean to Japanese. The answer to this question is yes, even though the system has never been taught this translation. This is referred to as “zero-shot translation”. Essentially, what has occurred is that the system is grouping points together. The meaning of a sentence translated from English to Korean, holds the same colored “points” as translating from Korean to Japanese, so the system could learn from that [4]. In Figure 2, the points are colored by the meaning. A sentence translated from English to Korean with the same meaning as a sentence translated from Japanese to English shares the same color.

Google Neural Machine Translation (GNMT)

Much of the knowledge about how GNMT works is in part due to Google researchers who have described the architecture of the GNMT system in a paper, “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation” [8]. The researchers noted the power of Neural Machine Translation (NMT) by stating, “The strength of NMT lies in its ability to learn directly, in an end-to-end fashion, the mapping from input text to associated output text” [8]. The researchers noted that, “Its architecture typically consists of two recurrent neural networks (RNN), one to consume the input text sequence and one to generate translated output text” [8]. A RNN is a neural network where the connection between the nodes forms a graph. NMT was not always the preferred choice and used to be worse in terms of accuracy for larger datasets for three reasons. According to the Google researchers, “Three inherent weaknesses of Neural Machine Translation are responsible for this gap: its slower training times, ineffectiveness in dealing with rare words, and sometimes failure to translate all words in the source sentence” [8]. Over time, however, NMT systems have gained wider

acceptance. One possible reason for this is because computational power has increased over the years, so it has become more feasible to train these types of systems. Additionally, the linguistic data that Google collected from their Statistical Machine Translation version of Google Translate undoubtedly helped in the development of GNMT [1].

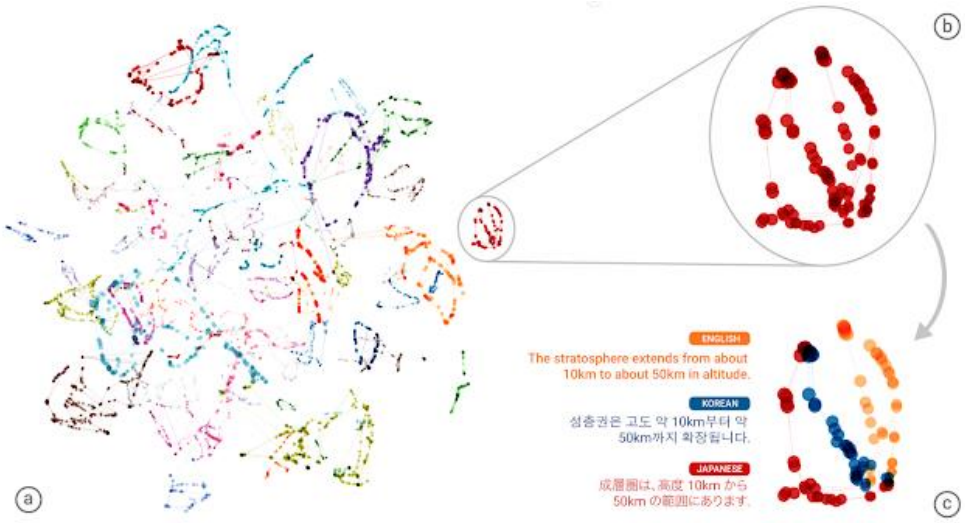

Figure 2: A visualization of sentence translations. A sentence translation from English to Korean with the same meaning as a sentence translation from Korean to Japanese share the same color, red [4].

As stated previously, Google Translate currently operates off of Google's Neural Machine Translation system. The researchers also revealed the components of the GNMT architecture. The researchers states that, "It has three components: an encoder network, a decoder network, and an attention network. The encoder transforms a source sentence into a list of vectors, one vector per input symbol. Given the list of these vectors, the decoder network produces one symbol at a time, until the special end-of-sentence symbol (EOS) is produced. The encoder and decoder are connected through an attention module which allows the decoder to focus on different regions of the source sentence during the course of decoding" [8]. Figure 3 shows the three components of the GNMT model architecture. The left dashed box is the encoder network. The right dashed box is the decoder network. They are connected by the blue box, which is the visualization of the attention network.

Another point noted by the researchers was that deeper neural networks often provide greater accuracy than shallow neural networks. This means that when there are more layers between the input and output (or encoder and decoder), it can usually give greater accuracy. However, the Google researchers noted that this is only true to an extent. If too many layers are added, the neural network becomes far too slow and difficult to train. For Long Short-Term Memory (LSTM) which is a type of RNN, the researchers noted that it works well up to 4 layers, barely with 6 layers, and very poorly beyond 8 layers.

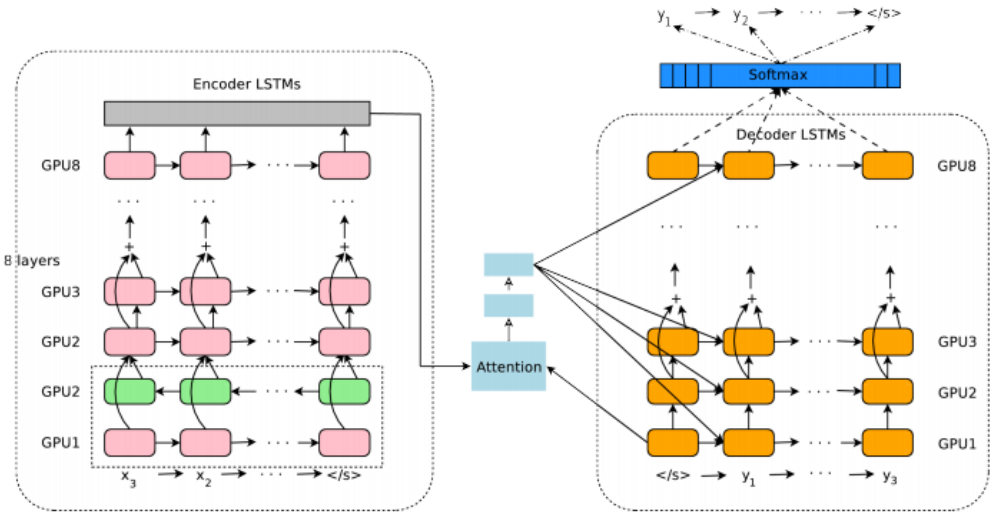

Figure 3: The encoder network (left dashed box), decoder network (right dashed box), and the attention network (middle blue box) [8].

However, GNMT uses 8 LSTM layers for the encoder and the decoder. A natural question that may arise is how they could achieve that if it is too difficult to train at 8 layers. This was possible because of the way the researchers approached how the LSTM was designed. A normal stacked LSTM looks like the figure on the left in Figure 4. The LSTM that the researchers implemented looks like the figure on the right in Figure 4. The difference is that the LSTM that the researchers implemented has a residual connection. With the residual connection, the input to a bottom LSTM layer is added to the output that goes to the top LSTM layer. The sum is then fed as input to the top LSTM layer. This reduces training times, and it makes it more feasible to add more layers as a result.

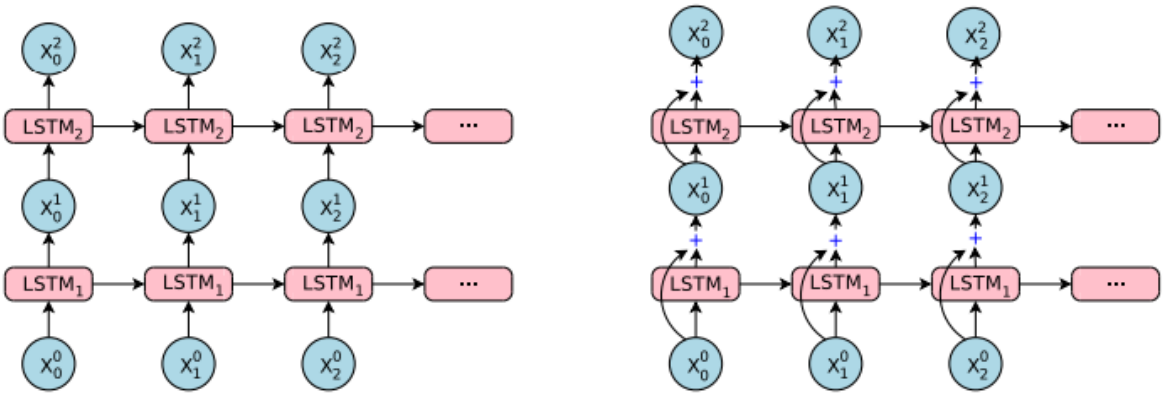

Figure 4: The left image represents a normal stacked LSTM, whereas the right image represents the LSTM that the Google research team developed with the residual connection [8].

Given the complexity of the GNMT model, the researchers make use of model parallelism and data parallelism in order to speed up training times. The researchers noted that encoder and decoder networks are partitioned on multiple graphics processing units (GPUs), effectively running each layer on a different GPU. The benefit of this is that it improves the training speed for the model.

An additional problem that needs to be taken into consideration when natural language translation occurs is rare words. Many times, rare words are just numbers or names. A simple solution to rare words is to use the copy approach to translation which means that the word is copied from source to target since copying the word is a most of the time a correct translation. For example, suppose a person is translating a sentence from English to Spanish, and that sentence contains the name of a person, Robert. The word "Robert" would be present in both translations because that person's name doesn't need to be translated, so it is copied in the translation.

Apple's Siri

Another well-known natural language understanding consumer product is Apple's Siri. Perhaps the most well-known feature of Siri is the feature of invoking it hands-free through the use of speaking aloud and saying, "Hey Siri". Apple's Siri team notes on the machine learning research part of their website that, "A very small speech recognizer runs all the time and listens for just those two words" [5]. When this recognizer acknowledges someone invoking it by saying, "Hey Siri," Siri will then execute what is said next as either a command or a query depending upon what the user said. The Siri team describes the inner workings of Siri by stating that, "The 'Hey Siri,' recognizer uses a deep neural network (DNN) to convert acoustic patterns of your voice at each instant into a probability distribution over speech sounds" [5]. The Siri team then goes on to describe how they determine whether the words you uttered were "Hey Siri," or not. The Siri team states that, "It then uses a temporal integration process to compute a confidence score that the phrase you uttered was 'Hey Siri'. If the score is high enough, Siri wakes up." [5]. The Siri team did not elaborate on what the threshold score is or how it is determined, all they revealed was that there is some threshold that they set for whether Siri is activated or not.

The Siri team explains the general infrastructure of Siri by saying, "Most of the implementation of Siri is 'in the cloud', including the main automatic speech recognition, the natural language interpretation, as well as the various information services" [5]. The Siri team explains that the if the iPhone detector is activated, then whatever was said by the user is sent, as an audio waveform, to a Siri server where a better speech recognizer will be activated to analyze what was said. The Siri team explains, "If the various stages on the iPhone pass it on, the waveform arrives at the Siri Server. If the main speech recognizer hears it as something other than 'Hey Siri' (for example 'Hey Seriously') then the server sends a cancellation signal to the phone to put it back to sleep" [5]. Figure 7, which shows the "Hey Siri" flow, shows how the server can send a signal back to the iPhone.

To understand how the detector works, it is first important to understand how Apple devices, including the microphone, work. The Siri team states that, "The microphone in an iPhone or

Apple Watch turns a person's voice into a stream of instantaneous waveform samples, at a rate of 16,000 per second. A spectrum analysis stage will then convert the waveform sample stream into a sequence of frames, each describing the sound spectrum of approximately 0.01 seconds. About twenty of these frames at a time (0.2 seconds) will be fed to the acoustic model" [5]. The Siri team states that the acoustic model is a DNN "which converts each of the acoustic patterns into a probability distribution over a set of speech sound classes" [5]. A simpler way to explain this is to imagine a soundwave is cut into many small pieces. A group of some of those small soundwave pieces are analyzed by the DNN and then the next batch of frames are analyzed and then the next group of frames are analyzed. This means that the frames are not all analyzed at once, but are rather analyzed in groups, one group after the other.

The Siri team states that the networks they use typically have 5 hidden layers. Hidden layers being all the layers in a neural network excluding the input and the output layers. The Siri team states that "we typically use five hidden layers, all the same size: 32, 128 or 192 units" [5]. Units are the individual pieces of each layer, often represented as a circle in pictures. The Siri team states that the unit sizes which will be used in a network depend on "memory and power constraints" [5]. There are two networks used by iPhones according to the Siri team, "One for initial detection and another as a secondary checker. The initial detector uses fewer units than the secondary checker" [5]. Regarding how the units for a layer are chosen, the Siri team states that "We choose the number of units in each hidden layer of the DNN to fit the computational resources available when the 'Hey Siri' detector runs" [5]. Figure 5 is a great visualization of the DNN that Apple provided that does the "Hey Siri" detection. The Siri team describes the figure by saying that "The hidden layers are fully connected, and the top layer performs temporal integration" [5]. The Siri team also stated when they provided Figure 5 that, "The actual DNN is indicated by the dashed box" [5].

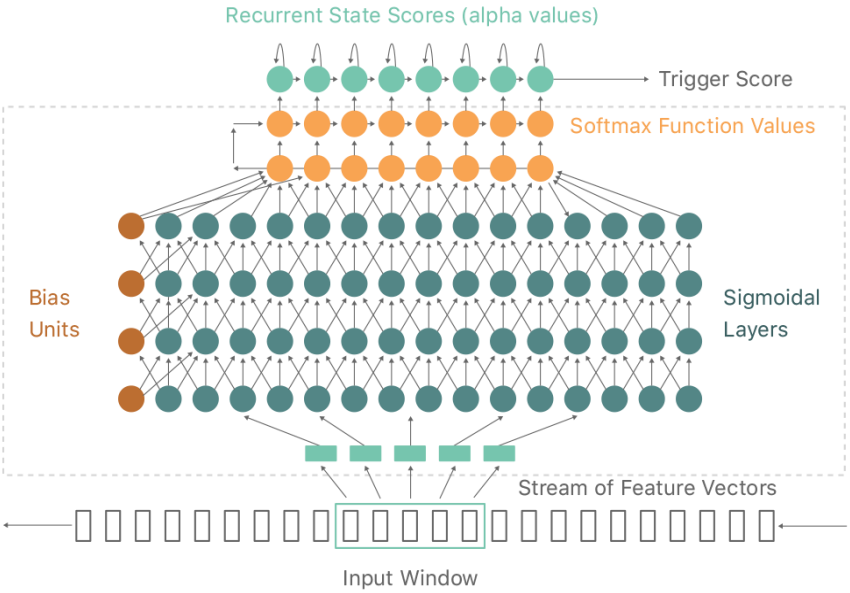

Figure 5: Deep Neural Network that the Siri team developed to be used during the detection of "Hey Siri" [5].

The Siri team developed an ingenious way to avoid false triggers, which are cases when someone else may trigger Siri by uttering, “Hey Siri”. To avoid this, an Apple device invites a user “to go through a short enrollment session. During enrollment, the user says five phrases that each begin with ‘Hey Siri,’” according to the Siri team and they also confirm that, “We save these examples on the device” [5]. The Siri team states that after this this enrollment session, “Any possible new ‘Hey Siri,’ utterances are compared with the stored examples” [5]. They state that, “The detector produces timing information that is used to convert the acoustic pattern into a fixed-length vector, by taking the average over the frames aligned to each state” [5].

The Siri team then describes how any new “Hey Siri,” utterances are compared to the stored audio that was saved after the enrollment session. The Siri team describes the process of how the utterance and the stored examples are compared by stating, “There is a separate, specially trained DNN that transforms this vector into a ‘speaker space,’ where by design, patterns by the same speaker tend to be close, whereas patterns by different speakers tend to be further apart” [5]. Siri then does a comparison between the distances. According to the Siri team, “We compare distances to the reference patterns created during the enrollment with another threshold to decide whether the sound that triggered the detector is likely to be ‘Hey Siri,’ spoken by the enrolled user” [5]. This procedure of having comparing an utterance with a stored example reduces the likelihood that the detector is triggered by someone else, but it also reduces the likelihood that “similar-sounding phrases trigger Siri” [5]. This intuitive idea of an enrollment session is vital to preventing users who do not own a specific Apple device from activating Siri by saying aloud, “Hey Siri”.

A problem with the “Hey Siri,” detector that the Siri team had to address is that not only must accuracy be prioritized, but it also, “needs to be fast and not have a significant effect on the battery life” [5]. In addition to preserving battery life, speed, and accuracy, the Siri team also prioritizes minimizing memory use and processor demand. The Siri team also describes how “Hey Siri,” works at any given time. The Siri team states, “To avoid running the main processor all day just to listen for the trigger phrase, the iPhone’s Always On Processor (AOP) (a small low-power auxiliary processor, that is, the embedded Motion Coprocessor) has access to the microphone signal (on 6S and later). We use a small proportion of the AOP’s limited power processing power to run a detector with a small version of the acoustic model (DNN). When the score exceeds a threshold the motion coprocessor wakes up the main processor, which analyzes the signal using a larger DNN” [5]. This approach to using a smaller, weaker processor prevents using a lot of battery life and processor power. When this smaller AOP detects, “Hey Siri,” a more accurate detector that resides in the main processor takes over.

Regarding the AOP, the Siri team details the specifics of it by stating, “In the first versions with AOP support, the first detector used a DNN with 5 layers and 32 hidden units, and the second detector had 5 layers with 192 units” [5]. The Siri team did not elaborate on details of the DNN used in the current AOP version. Figure 6 does a great job of visualizing how the AOP works, and after the confidence score passes the threshold set by the Siri team, the AOP passes the control over to the main processor. The left box is the low-compute DNN that the AOP runs, and if the confidence score meets the threshold, the AOP will call on the main processor which runs the more accurate DNN.

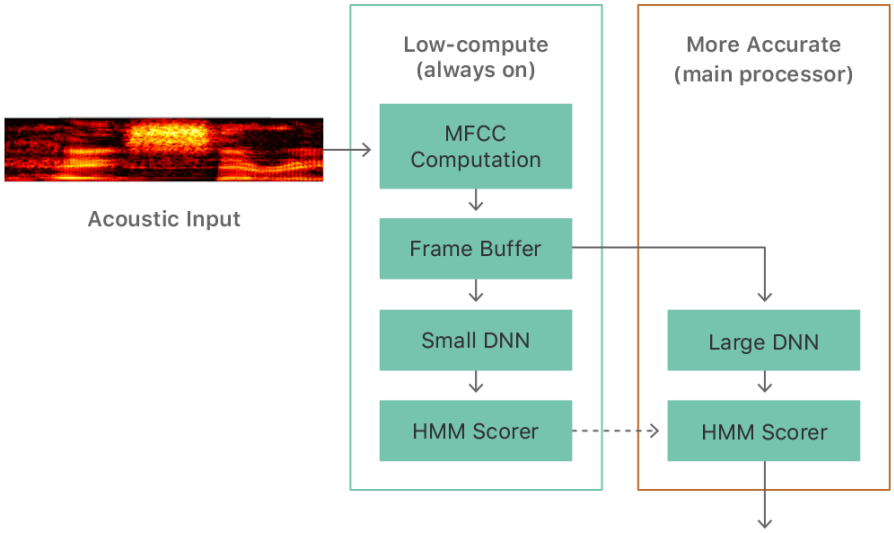

Figure 6: A visualization of how the low-compute DNN calls on the more accurate DNN in the main processor [5].

As stated previously, it is important to preserve battery life, accuracy, speed, and minimize both memory use and processor demand. That is already a difficult engineering task, but it is even more of a challenge for the Apple Watch because the battery is much smaller. The Siri team states that one way in which they handle this issue is that "The 'Hey Siri' detector runs only when the watch motion coprocessor detects a wrist raise gesture, which turns the screen on" [5]. After this wrist gesture, many things occur with WatchOS (Apple Watch's operating system), such as the powering up, preparing the screen and so on. Because of this, the system allots "Hey Siri," only a slight portion (~5%) of the rather limited compute budget. It is also a difficult task to start audio capture in time to catch the start of a trigger phrase, so the Siri team states that, "We make allowances for possible truncation in the way that we initialize the detector" [5].

Before the Siri team implemented the hands-free version "Hey Siri," a small percent of users with access to Siri would press the home button and say, "Hey Siri," before saying a request. Siri was activated by pressing the button, not by saying "Hey Siri,". Even though Siri was activated by a home button press, their speech was saved and used in datasets for developing the next iteration of Siri that did not require any button being pressed. These instances of "Hey Siri," were used in initial training sets for the DNN detector model meant for US English. This distinction of "for US English," needs to be made because not all languages can activate the detector by saying "Hey Siri," This means different models are needed for different languages since the language you use will affect what words are needed for the detector to recognize speech. For example, a Russian speaking user would need to say "Privet Siri," which means "Hey Siri," in Russian. The Siri team then used "automatic transcription on the training phrases" but had "some subset of the transcriptions checked for accuracy" by members of the Siri team [5].

Another important aspect of developing the detector is testing and tuning. The Siri team states that they measure the accuracy of the detector by two metrics, "firing at the wrong time and

failing to fire at the right time. The false-accept rate (FAR or false-alarm rate) is the number of false activations per hour and the false-reject rate (FRR) is the proportion of attempted activations that fail" [5]. The Siri team also explicitly points out that measurements for the false-accept rate and the false-reject rate are different, so it is not useful to compare these rates. Regarding determining accuracy, the Siri team states, "During development we try to estimate the accuracy of the system by using a large test set which is quite expensive to collect and prepare, but essential" [5]. There is positive data and negative data. Positive data contains the target phrase, such as "Hey Siri,". The difficulty with the positive data is that many times, the system does not capture attempts that failed to trigger, so to improve the system, the Siri team tries to collect as many failed attempts as possible.

Negative data is used to test false-activations, or Siri activations that occur when the user did not mean for them to happen. The data represents thousands of hours of recordings, from various sources, including podcasts and non-"Hey Siri" inputs to Siri in many languages. This is done to represent and imitate background sounds and speech and the kind of phrases that a user might say to another person. A lot of data was needed because the Siri team needs to try to estimate the false-alarm rate as low as one per week, meaning that the Siri team would like to have a near perfect system where false-activations do not occur.

Another difficulty in developing hands-free Siri was that the dataset that was created from their users initially required the press the home button, and then users needed to say, "Hey Siri". This is not ideal because it does not capture every possible condition that a hands-free Siri would have. For example, pressing a button and saying, "Hey Siri," will be captured entirely differently than being across the kitchen and saying, "Hey Siri" because of the distance that the individual would be from their phone in each instance. To combat this discrepancy, the Siri team made recordings in various conditions, close and far, in cars, bedrooms, restaurants, and many more places. All of these recordings were done in various languages by native speakers of each language.

After the models are developed, the next step is tuning the models. Tuning the models is a matter of what threshold the Siri team decides to use. Evaluating and improving "Hey Siri," and the model that powers it is a matter of training and testing using variations of approaches. The Siri team trains models in many different languages and tests under a wide range of conditions, although the Siri team did not specify what those wide range of conditions are. The Siri team also did not provide much detail on how they tune or evaluate and improve Siri.

As described, preparing the data for the models used in the detector is just as much, if not more, of a challenge than developing the deep neural networks used for natural language understanding. Additionally, a feature as simple as activating Siri hands-free requires an extraordinary amount of work and effort that the average consumer may take for granted. Figure 7 provides a visualization of the "Hey Siri" flow on the iPhone.

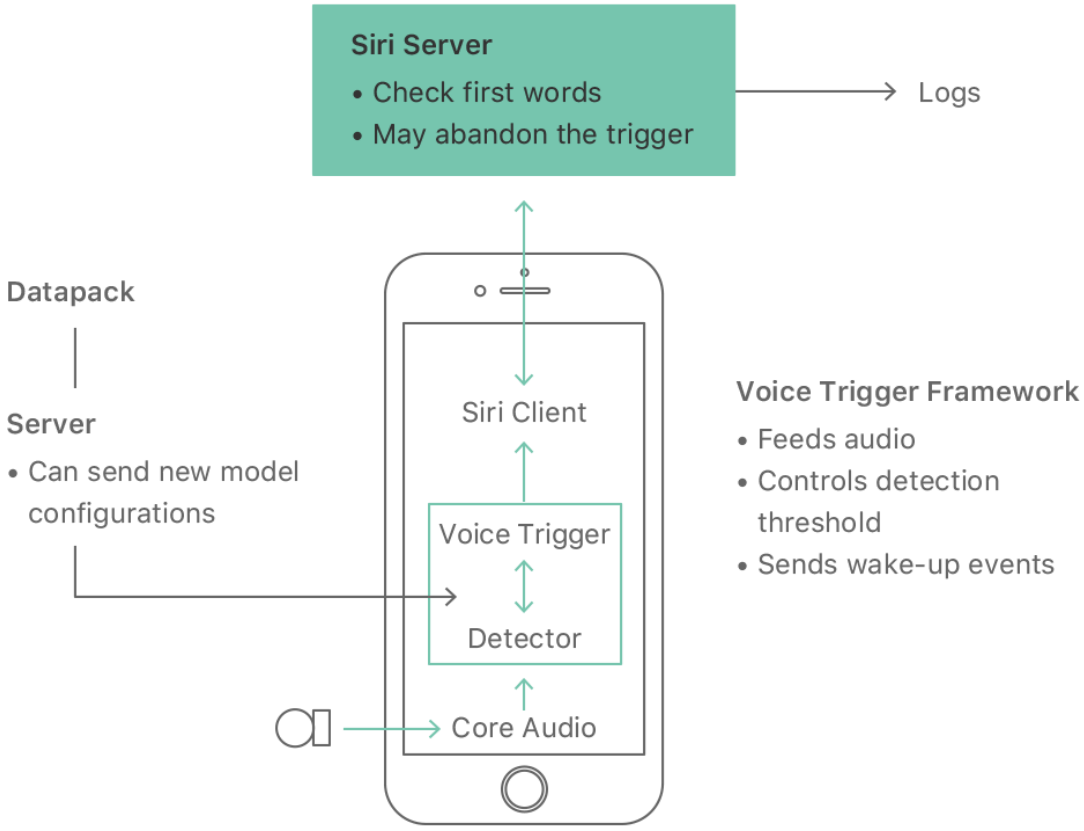

Figure 7: "Hey Siri" flow [5].

Amazon's Alexa

Amazon's Alexa is another well-known product. Alexa itself is a cloud-based voice service, however, Alexa is often associated with Amazon's Echo. It is important to remember the distinction between them. Alexa is the virtual assistant technology that Amazon developed. Echo is the physical product that Amazon sells Alexa with. Alexa works through the use of Alexa skills, which are essentially like apps for Alexa and provide a channel for content and services. Alexa is open for developers to integrate their products with. Amazon allowing developers access to work with Alexa is not unique, Apple allows developers access to Siri through SiriKit, however, a large difference is that SiriKit is confined to the Apple ecosystem, whereas Alexa allows developers to build products that interact with devices outside of just Apple products.

Amazon states that they have historically run Alexa's automatic-speech-recognition models, which are responsible for converting speech to text, in the cloud. However, in recent years, Amazon states that they have been "working to move more of Alexa's computational capacity to the edge of the network – to Alexa-enabled devices" [6]. The appeal of this is "faster response times since data does not have to travel to and from the cloud" [6]. Additionally, Amazon states that there is "lower consumption of internet bandwidth, which is important in some applications" [6]. Amazon explains the importance of why they care about lower consumption of internet

bandwidth by stating that, “This allows for availability on devices with inconsistent internet connections, such as Alexa-enabled in-car sound systems” [6].

Amazon describes the inner workings of their speech recognition model by stating that, “Neural automatic speech recognition (ASR) models are usually encoder-decoder models” [6]. This means that Amazon’s machine learning models for speech recognition are much like the Google Neural Machine Translation model because they both have an encoder and a decoder. Similar to how Apple’s “Hey Siri,” recognizer works, short speech snippets called frames are used as input to the encoder, which the encoder converts the frames into a representation that is useful for decoding. The decoder then translates that representation into text. Neural encoders can be massive, requiring millions of computations for each input, and humans do not speak continuously because we often pause between syllables. This means that not every frame is a useful input since some frames will contain those pauses. [6]

To get around this issue and prevent wasted computation, Amazon’s approach is the use of multiple encoders. The multiple encoder differ in complexity and they decide on the fly whether a frame of speech should be handled or not. That decision is made by a small neural network called an arbitrator, which must process every input frame before it can be processed and encoded. The arbitrator adds some computational overhead to the process, but it saves time by preventing unnecessary computation for frames that do not need to be encoded. The prevention of unnecessary computation makes the computation overhead of the arbitrator worthwhile.

The ASR models that power Alexa are constantly being updated by engineers who work on the Alexa team. Amazon provided an example of how often they are updated, such as the instance during the Olympics. Amazon anticipated a large spike in requests that used the words “Ledecky,” and “Kalisz,” both of which are the last names of two different Olympians. In response to this anticipation, Amazon updated their models accordingly.

Edge ASR models are models that reside on devices, such as phones, for example. Another way in which edge ASR models on devices differs from traditional cloud-based models is that in cloud-based models, when a model is updated, it is simply sent to a handful of data servers. However, with models that are on devices, Amazon may ultimately need to send updates to millions of devices simultaneously, so one of their research goals is to minimize bandwidth requirements for edge updates.

Conclusion

A commonality between all companies discussed in this paper is that they all use an encoder and decoder approach. Amazon elaborated more than Apple did on how they approach preventing the encoding of empty frames. Given the high quality of Siri, Siri likely also has ways to prevent empty frames (or pauses in speech) from being encoded, although they did not go into detail on how they do that. Google was fairly transparent in how they approach using a lot of layers in their translation system. They use residual connections to reduce the training time on their model, which was a clever approach to get around the complexity of training a computationally intensive machine learning model. Although it is impossible to know all the ins-and-outs of how

Google Translate, Siri and Alexa work because they are proprietary technology, the engineers who work on these products have been kind enough to describe how the systems for these products generally operate.

While Google, Apple and Amazon have different approaches to how they use machine learning models, they all share some commonalities. Google's machine learning model focuses more on translation, although of course Google still cares about speech because they do offer speech capabilities for Google Translate as well. Amazon and Apple both will have different ways in which they use machine learning models because their models rely more on speech.

References

[1] Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261-266.

[2] IBM. (n.d.). *What is machine learning?* IBM. https://www.ibm.com/cloud/learn/machine-learning

[3] Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

[4] Schuster (Google Brain Team), M., Johnson (Google Translate), M., & Thorat (Google Brain Team), N. (2016, November 22). *Zero-Shot Translation with Google's Multilingual Neural Machine Translation System*. Google AI Blog. https://ai.googleblog.com/2016/11/zero-shot-translation-with-googles.html

[5] Siri Team. (2017, October). *Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant*. Apple Machine Learning Research. https://machinelearning.apple.com/research/hey-siri

[6] Strimel, G. P. (2021, September 02). *How to make on-device speech recognition practical*. Amazon | Science. https://www.amazon.science/blog/how-to-make-on-device-speech-recognition-practical

[7] Turovsky, B. (2016). Found in translation: More accurate, fluent sentences in Google Translate. *Blog. Google. November*, *15*.

[8] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[9] Zong, Z., & Hong, C. (2018, September). On application of natural language processing in machine translation. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)* (pp. 506-510). IEEE.