Automated Techniques for Creating Speech Corpora from Public Data Sources for ML Training

Lawrence Drabeck, Buvana Ramanan, Thomas Woo, and Troy Cauble

***Abstract*—For machine learning (ML) to work well, there is a need for large amounts of good quality training data. Obtaining such data is often the key bottleneck for the entire ML development process. Using humans to do explicit collection has been the main approach, but this tends to be expensive and time-consuming. Therefore, there is significant interest in creating alternative data collection techniques.**

We explore these alternative data collection techniques in the context of speech data in this paper. We were initially motivated by the problem of wake word engine training, where we need a large number of utterances for specific wake words. Given that there are already large public repositories of media data (e.g., YouTube, DailyMotion), we were curious as to how feasible it is to find the utterances that we need. Our results are encouraging as we found many different types of words can readily be found and downloaded in the quantity and quality needed to create training corpora for DL training. Usually > 30% of the found words are suitable for corpus creation. Greater than 80% of the top 10,000 ranks words and > 50% of the top 20,000 words we selected easily produced > 5000 found words, which is sufficient to train a high quality Wake Word Engine. Besides general words, we specifically looked for words used in wake word engine construction such as Name/Place/Product Name. Here, again, we find most common names/places/products return more than a sufficient number of words for corpus creation. Only uncommon names and places (like Atticus or Maximus) are difficult to find in sufficient quantities for corpus creation.

We demonstrate a wake word engine trained from words we found in YouTube has the equivalent performance to one trained with traditional human collected words. Even though we were focused on wake words, our approach is general. It can be applied to create speech corpus for various purposes.

***Index Terms*—Corpus, found data, training data, wake word engine, machine learning, deep learning.**

I. Introduction

Data is the key to modern machine learning, and in particular the rapidly emerging subfield of Deep Learning (DL). In DL, a key step is training where a large amount of training data is fed into a neural network for continuous updates to parameter weights. The final setting of these weights essentially determines the overall accuracy of the model. Thus having a large amount of high quality data is extremely critical to achieve good performance.

Creating a large high quality data set has traditi [redacted] a human task. For example, for image classificati will be asked to label the images; for speech recognition task, human will be asked to utter specific sentences. Systems have been created to make these easier. As an example, the Amazon Mechanical Turk (MTurk) system can be used to automatically dispatch tasks to interested individuals. Despite that, explicit human data collection is still expensive and time-consuming. To quote our actual experience, we used MTurk to collect utterance of specific wake words, we listed a price of $0.20 for 3 utterance of a wake word. Our whole campaign cost $140 in fees and took 3-4 weeks of time in total. As we need to evaluate multiple wake words, the cost and more importantly the time added up quickly and became non-trivial.

Outside of human collection, there are generally 2 ways to obtain data: (1) Using Found data - Trying to locate the needed data in pre-existing data; and (2) Synthesizing data - Trying to artificially "create" the needed data. In this paper, we explore the feasibility of using Found data for speech data collection.

Our work was initially motivated by our research on DL models for wake word engines (WWEs). A wake word is a word that is uttered to activate a voice assistant before a request is made. For example, the wake word "Alexa" is used for most Amazon voice assistants such as Echo. And the phrase "OK Google" is used for most Google voice assistant products. A WWE is a software component that listens continuously to incoming voice samples and determines if the specific wake word has been uttered. Modern DL-based solutions to WWE have outperformed traditional ones based on Hidden Markov Models. We started with MTurk initially, but found it too costly and time-consuming when significant iterations are needed.

Speech data for wake word training requires voice samples uttering the specific wake words. In most cases, the wake words tend to be actual words in the vocabulary, rather than "made up" words created specifically as wake words. For the latter case, even human collection would be difficult when people do not know how to pronounce the "made up" word. For the former, our work is to explore the use of public media data sources such as YouTube, and study how to effectively "locate" the needed utterances and how available the needed utterances are.

More specifically, our high level problem is: Given [redacted] ords,

elop techniques and tools to locate the utterances

ese words in public data sources such as YouTube

erstand the availability of the needed utterances

a stated purpose; and

ermine the quality of the available utterances for

stated purpose.

rk is mostly couched in terms of WWE training. But our tools, techniques, and general methodology can be

Manuscript received July 20, 2019; revised January 3, 2020 [redacted]

L. Drabeck and T. Cauble are with Nokia Bell Laboratories, 07733, USA (e-mail: lawrence.drabeck@nokia-b troy.cauble@nokia-bell-labs.com).

B. Ramanan and T. Woo are with Nokia Bell Laboratories, NJ 07974 (e-mail: buvana.ramanan@nokia-b thomas.woo@nokia-bell-labs.com).

easily applied to other purposes.

Indeed, there are other reasons for collecting specific utterances from public data. We mentioned two other unrelated applications here:

1. Language learning - To effectively learn the pronunciation of words, it is useful to hear it spoken in many real-life contexts by different people.
2. Editing - A media editing process typically require a large amount of raw clips, both video and audio. The ability to locate specific speech clips with specific utterances is critical.

The main contributions of this paper are as follows.

1. It studies the feasibility of locating specific utterances in public data sources, for the purpose of creating a speech corpus. Our study provides a picture of the overall feasibility, availability and quality.
2. We propose and study specific techniques and tools that allow us to effectively locate the needed utterances. The ability to locate the candidates is a key limiting factor for the overall data collection process.

The rest of the paper is organized as follows. Section II gives a brief overview of related work while Section III details our proposed methods, and their implementation considerations. Section IV presents the experiment results and analysis. Section V discusses other issues and future word and we conclude with Section VI.

II. Related Work

One key to Deep Learning (DL) success is to train using a carefully prepared dataset that (1) accurately represents the intended operating environment, and (2) is plentiful in capturing the diversity for generalization. For example, the Amazon Alexa team [1] utilizes 1 million positive examples and the XiaoMi team [2] utilizes 188k positive and 1 million negative examples, all specifically collected from humans. This data collection step is typically the Achilles' heel of DL as collecting and curating data at such massive scale involves time, money and other logistic challenges that is often a bottleneck. For a DL wake word application where there is a fixed set of pre-defined wake words, this is tedious but may still be workable. However, if a custom Wake Word Engine (WWE) is desired where the wake word can be chosen on demand by a user, such on demand targeted collection from human at a large scale is impractical.

A staggering 300 hours of video is uploaded to YouTube every minute [3], much of which has either uploaded or automatically generated subtitles. YouTube is used as a data source for many different applications. For video analytics, [4] used ~2000 videos obtained from YouTube of people doing "The Mannequin Challenge" [5], to train a model to predicting dense depth in scenarios where both a monocular camera and people in the scene are freely moving. Reference [6] uses video clips collected from YouTube depicting human actors performing various acrobatic stunts (e.g. flips and cartwheels) and locomotion skills (walking and running) to train DL models for character motion animation instead of using motion capture systems. Reference [7] uses the video comments as a basis to create a sentiment analysis corpus for autonomous vehicles, [8] creates a video/audio corpus for lip reading training and [9] creates a corpus of 1.8 million 10sec audio clips for 632 audio event categories.

For audio training, it has already been demonstrated that YouTube captions can be successfully used as a ground truth spoken transcription to train large scale ASR systems [10]. In Deep Learning speech and video applications, found-data from YouTube (YT) has successfully been applied for Automatic Speech Recognition (ASR), Speaker Recognition and Text-to-Speech (TTS) applications. Reference [11] created a YouTube dataset (AVSpeech), from roughly 290,000 YouTube videos of lectures (e.g. TED talks) and how-to videos to train neural nets to pick out a single speaker from a noisy environment (cocktail party effect). Reference [12] introduces a crawler for YouTube to curate training dataset for ASR and demonstrates a 40% improvement in Word Error Rate (WER) on the Wall Street Journal test dataset. In [13], the authors address the problem of operating ASRs in a wide range of developing languages, such as Swahili, by proposing to automatically scrape audio from YouTube and Voice of America and use ASR system confidence scores as the primary metric for the model components. The creators of the VoxCeleb1 and VoxCeleb2 datasets [14], [15], crawled YouTube to construct the datasets, which are now widely used in the field of Speaker Recognition [16]. Reference [17] discusses the effectiveness of YouTube based dataset for TTS, cloning former US president Barrack Obama's voice.

To the best of our knowledge, no literature exists that describes the feasibility of locating specific utterances in public data sources, for the purpose of creating a speech corpus. Our study provides a picture of the overall feasibility, availability and quality.

III. Data Curation Pipeline

We developed a data curation pipeline using YouTube videos with subtitles as the data source for creating different types of speech corpora. The first step is to obtain suitable video URLs, which we discuss in detail below. Then the audio and subtitles for these videos are downloaded. The subtitles are processed to remove non-ascii text, sound and speaker designations, and to convert numbers to words. Word timing alignment is performed on the cleaned subtitle text using a forced aligner [18] since subtitle timing is not accurate enough for our purposes. The word timing is used to slice individual words for Wake Word Engine training and used to slice utterances (typically 5-15 sec in length) for Automatic Speech Recognition (ASR) or Text-to-Speech (TTS) training. As a test of our data curation method, we recreated the TED-LIUM speech Corpus [19], [20] via downloading from YouTube. Using a small Spark computing cluster we were able to create this corpus in ~2 hours (326 hours of audio). We tested our curated speech corpus against the TED-LIUM corpus by training Baidu's Deep Speech ASR [21] and measuring WER on these trained Models. The model trained with our curated data had a WER within 1-2% the WER of the official TED-LIUM corpus trained model. As a further test, we trained an ASR on ~7000 hours of curated YouTube utterances and saw better WER performance than Google's default ASR model. We have found similar approaches in utilizing data from YouTube [12]. Our approach is more general in that it can search for and extract individual words for wake word corpus creation, in addition to gathering arbitrary voice/transcript pairs.

For training Wake Word Engines (WWE), individual words are needed. Navigating the billions of videos on YouTube to find the ones containing the wanted word is the challenge. Presently there is no method to search the subtitles of YouTube videos via their API, so we are limited to searching the titles and descriptions for the desired word or phrase. We have broken the search for the wanted word into 3 phases 1) search in already indexed YouTube subtitle material, 2) use YouTube search, which is limited to titles and descriptions, or other search engines to get relevant search terms, and 3) collected related videos from videos found in step2.

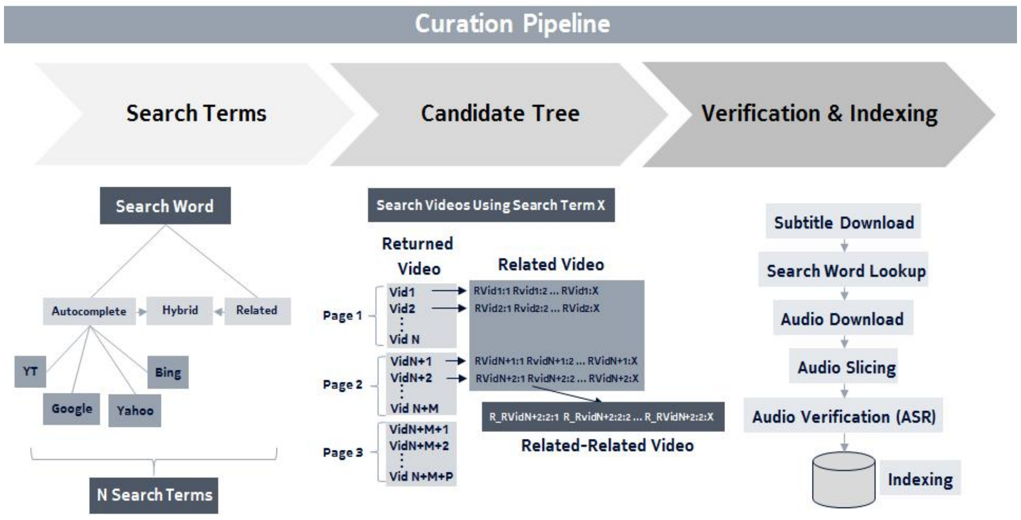


Fig. 1. Curation pipeline for words.

We first search for the wanted word in our existing indexed >10,000 hour corpus. To find additional words, we can use YouGlish [22], which is a web site to help users improve English pronunciation by hearing words in the context of English sentences. This site has indexed > 30M subtitled YouTube videos and is searchable for words and phrases. The returned YouTube URLs from this site are collected using their widget. Scraping URLs from the widget is slow and YouGlish is usually used as a backup method if the methods discussed below are not successful.

To find the wanted words in YouTube's vast library we need search terms. To start, one could use YT's API and search for the wanted word. YT's API returns a maximum of ~ 600 videos URLs for each search term. This is usually not a sufficient number of videos to collect the ~1600 quality wake words we would need to train a Wake Word Engine. One could manually expand the search terms, for example, looking for the word 'Toyota', one might try search terms like 'Toyota Prius' or 'Toyota Highlander' or 'Toyota vs Honda'. For a small number of additional search terms we can use YT's API. However, to expand our search terms and also find the most relevant ones, we utilize the autocomplete feature of different search engines instead of using manual searches. Also, to expand our search over a wider variety of search terms and avoid exhausting the limited number of searches per day allowed by YT's API, we use Selenium (a suite of tools to automate web browsers), which is slower, to scrape the video URLs for different YT searches.

Fig. 1 shows our overall methodology to generate the needed search terms, curated the video URLs and then download, clean and index our resulting data. In the first stage we need to generate search terms to lookup videos. We use either the autocomplete feature of different search engines or use a related term generator [23]-[25] to generate these search terms. The wanted word is usually the starting word for both methods.

In the autocomplete methodology, we search for 37 different autocompletes comprising the initial search string, which is usually the wanted word, followed by either a space, or by the letters from a to z or followed by the digits from 0 to 9. The reason for using a space after the search term is to avoid auto-completion of unwanted search terms. For example, YT autocompleting on 'nova' (without a trailing space) will return terms like Novak Djokovic or novacane, but what we really want is just "nova".

The four different searches engines we use are YouTube, Google, Bing and Yahoo. Other search autocomplete engines such as Wikipedia, Amazon, Ebay and many others could also be used. The YouTube and Google autocomplete engines return a maximum of twenty search terms so we can get a maximum 740 search terms for our 37 different searches. Bing returns a maximum of 25. YT and Google also return a ranking for the search terms so we can sort by ranking. For the Bing and Yahoo searches, no ranking is returned so in order to rank the returned search terms we could use an n-gram approach to look for the most relevant search terms. For many of the results below, we use the top 50 search terms from the YouTube autocomplete as they are usually the most relevant. We resort to the other search engines and related term generation when an insufficient numbers search terms or an insufficient number of wanted words are found in the videos.

In the second stage, we use the generated search terms to find videos with subtitles in YouTube. For a given search term, the most relevant ~20 videos are returned and displayed on the YT website. Scrolling down on the website will dynamically load more videos, depicted in Fig. 1 as different page brackets. Each subsequent scroll down loads ~ 20 more videos. After X scroll downs, you will eventually reach the limit of available videos for that search term. Each search term will have different number of maximum videos and thus number of scroll downs. We are using Selenium to scrape the URLs, so we need to scroll down to load more content. In the experimental section we explore the number of wanted words found versus the number of scroll downs.

If additional videos are required after exhausting the search terms and scroll downs then we can gather the related videos. Each video in YouTube has a list of multiple related videos suggested by YouTube (see Fig. 1Fig.). Each video found using the search terms is loaded and the related videos, suggested by YT, are scraped. Again, only the first ~20 related videos are initially loaded but subsequent scroll downs will load even more related videos. The related videos tend to be less relevant for the wanted wake word and we find the ratio of videos with wake words to total videos drop by a factor of 10x-20x. Even without using any scroll downs, scraping the related videos still increase the number of overall videos collected by 10x-20x, so we rarely use scroll downs in this stage.

It is also possible to find related videos for the related videos found above (Related-Related videos in Fig. 1). The relevance starts to really drop off for these cases and we rarely, if ever, go to this level.

The gathered video URLs from all the search terms and related scrapings are filtered for duplicates and checked for subtitles. The subtitles are downloaded and checked for the wanted word. This finally gives an indication of the number of wanted words found and indicates if further scrolling, searching or related videos need to be employed. The audio for any video with a wanted word in the subtitles is downloaded and the audio is aligned to the text using a forced aligner. For longer audio, the forced aligner becomes a bottle neck and can sometimes fail for long (>1.5 hour) audio. Therefore, for audio > X minutes in duration, we slice the audio into 2 minute clips around each wanted word using the course subtitle timing and align these 2 minute clips with the forced aligner.

Using the forced aligner timing, the wanted words are sliced out and padded (if necessary) to form 1 second audio files. We also slice out words other than the wanted word spoken by the same speaker for training data. By slicing these other non-wanted words from 15 seconds of audio on either side of a wanted word, there is a high probability of obtaining these non-wanted words spoken by the same speaker.

We verify the sliced wanted words by running them through several different Automatic Speech Recognition (ASR) systems. We use a DeepSpeech 2 model [21] we trained on ~ 7000 hrs of YouTube Audio (discussed above) as well as 3 commercial ASR systems; Google Standard, IBM and Houndify. Depending on the total number of sliced wanted words, we usually consider a good quality word to be recognized by at least 2 ASR engines. When too few sliced wanted words are found, we relax the 2 ASR matching to 1 ASR match. We have trained several wake word engines, that had a limited number of sliced wanted words, with only 1 ASR recognition and have obtained good results.

The final step is to clean up the background noise in the sliced wanted word audio. We have found that training a wake word engine with YouTube data showed reduced performance in a clean (non-noisy) test environments compared to the noisy test environment. This indicates a lack of non-noisy wake words. We use pyroomacoustics [26] python library to do the audio cleaning of the wake words to remove background noise. This leads to better overall WWE performance in both the clean and noisy environment. These steps lead to a well-constructed corpus for training DL models.

IV. Experimental Results

To study the effectiveness of our methodology, we look at a large group of general words, words that are Names, Places or Products, as well as diving deeper into just a few words. We also show that training a WWE with the collected words leads to a high quality WWE.

A. General Words

We randomly choose a set of 150 words, mostly nouns, from the Corpus of Contemporary American English (COCA) [27] which is a large, genre-balanced corpus of American English containing > 0.5 billion words equally divided among spoken, fiction, popular magazines, newspapers and academic texts. Fig. 2 shows the number of words found in the subtitles for each wanted word when searching the videos of YouTube. Here we used the top 50 YT autocomplete search terms and scrolling down on each YT page 11 times. As the word becomes less popular (the word ranking increases), the ability to find these words in YouTube decreases. There are a few exceptions that popup above this trend such as 'griddle' (rank 43398) with > 12k words found in the subtitles. This is not a term that is popular in everyday speech but there are many cooking/recipe videos on YouTube where a griddle is used to prepare some dishes. Another example is 'Pacific'(rank 29109) with > 11K words found in the subtitles. Although not high on the word ranking, there are many travel related videos in YouTube that may refer to the Pacific and hence the large number of found words.

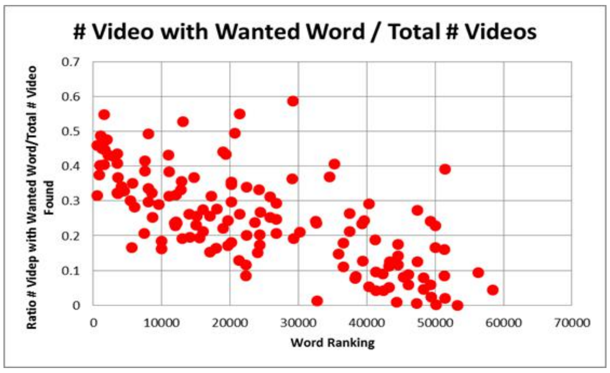


Fig. 3. Ratio of number of Videos with the wanted word to total Videos found for 150 word of different ranking. Top 50 search terms with 11 scroll downs.

Fig. 3 shows the ratio of the number of videos with the wanted word found vs total number of videos returned from the search terms. Again we see the same trend that less popular words are harder to find. For more popular words (rank<10k), we find that ~30%-50% of the videos found have at least one wanted word in them.

Even though we clean the subtitle text when downloading, when searching for words in the subtitles there are certain things to watch out for. A few issues we found were:

1. Compound words – not sure how these may be represented in the text so search not only the compound word, but a hyphenated version of the compound word and the compound word as two separate words (e.g. rockslide, rock-slide, rock slide) Table I gives a comparison of the number of wanted words found in the subtitles when searching only the compound word versus searching for the 3 version of the word. In some cases there is a large difference producing >2x as many words.
2. Words with apostrophes such as Kellogg's. In the subtitles this may be Kellogs or Kellogg's or Kelloggs'.
3. Numbers – we convert digits to numbers but numbers can be said in many different ways. For example, 100 can be pronounced one hundred or a hundred.
4. Money - \$1 can be pronounced one dollar, a dollar, one buck, or a buck
5. Year Dates like 1970 – This can be said Nineteen Seventy, Nineteen hundred and seventy, one thousand nine hundred and seventy or if referring to a flight for an airline could even be said as one nine seven zero.

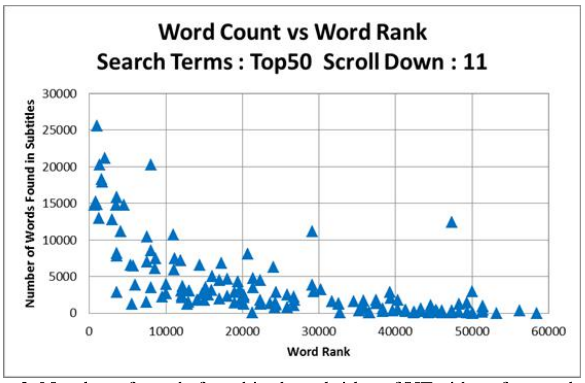


Fig. 2. Number of words found in the subtitles of YT videos for words of different ranking. We used the Top 50 search terms for each word and 11 scroll downs.

The numbers, dates and money issues are more related to creating corpora for ASR training where WER between spoken audio and the subtitles is an issue. For wake words, numbers, dates and money are not as big and issue as we check each word with an ASR so mislabeled words will be excluded. However, compound words and apostrophe words could be an issue if we don't identify them in the text.

Once we have identified the videos with the wanted words, the audio is downloaded and the subtitle text is aligned. The wanted words are sliced out of the audio and check by 3 different ASR engines. Usually we are able to download > 90% of the videos found. Those not downloaded are usually too long (>1.5 hours) or have other issues. Fig. 4 shows the percentage of words we are able to slice out of the downloaded videos and the percentage that match at least one ASR engine. The non-shaded section of the plot is for a previous study where the search terms were hand generated. For this study, 60-80% of the downloaded words were sliced from the audio and 40-60% were sliced and matched at least 1 ASR. There is quite a bit or variability here even for the same word. We did three different runs with different search terms for Toyota and the words with 1 ASR match vary from 30-50%

Other words show very poor ASR matching, most notably 'nova'. For 'nova', a large majority of the found words were words like Nova Scotia, Super Nova or Bossa Nova which all have very little gap in time between the two words. The forced aligner had difficulty separating the words without either chopping part of nova off or including a piece of the adjoining word. We had a similar problem when slicing out the word 'happy' since it is very common to find 'happy birthday'. Here usually the 'y' was chopped from the word and we ended up with 'happ'.

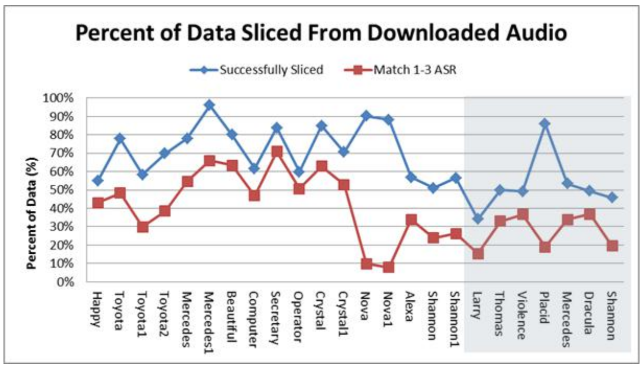


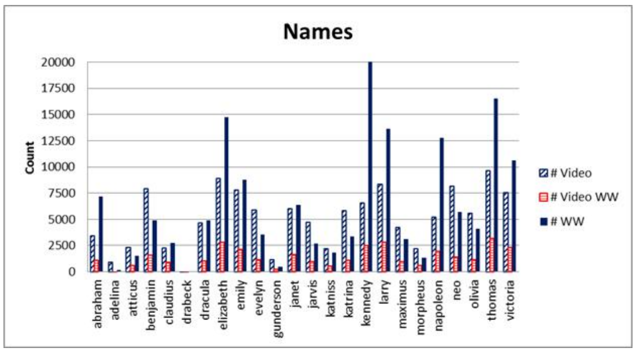
Fig. 4. Percentage of words that are sliced and also match at least one ASR from downloaded YT audio files.

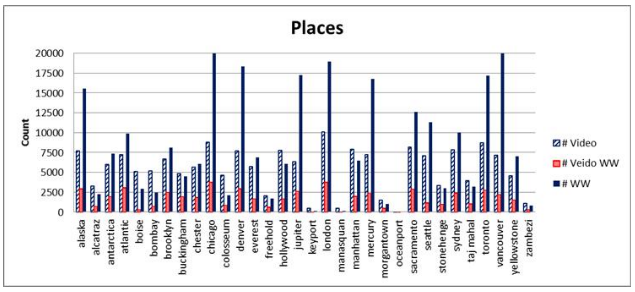
The shaded section of Fig. 4 is for search terms generated with the YT autocomplete method in this paper. Notice that the amount of words sliced and matching 1 ASR is lower, ~50% and ~30% respectively, compared to the non-shaded region. These auto-generated search terms returned many more videos with English subtitles but non-English audio and the forced aligner cannot align the audio. A few other reasons for non-alignment are excessive background noise, quickly spoken words and singing of words. Filtering the videos for spoken English should bring these results to similar levels as the non-shaded region.

B. Names/Places/Products

With an eye toward building a Wake Word Engine (WWE), we look at wanted words that are names, places, product names or corporations. From our previous work on wake word engines construction [26], we need ~1600 high quality words, 1200 for training and 400 for testing. From the shaded section of Fig. 4, we see we that ~ 30% of the words match at least on ASR so in order to get ~1600 good wake words we need to find ~5000 words in the subtitles.

Fig. 5 shows the number of wanted words (WW) found in the subtitles, the number of videos and the number of videos with wanted words for the three different categories. Here again, we use the autocompleted Top50 YouTube search results and 11 scroll downs. The common names easily have > 5k words found in the subtitles. We purposely chose many names on this list to be uncommon first names or last names to assess the difficulty in finding them. For these less common first names like Atticus, Morpheus and Katniss and last names like Drabeck and Gunderson, we don't quite find the needed 5k words and should try some of the other search techniques discussed above (i.e. >50 search terms, more scroll downs, different search engines or scrape related videos).





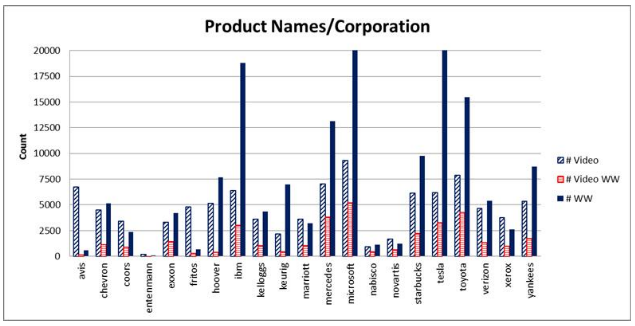


Fig. 5. Names, places, or products. The number of words, videos and videos with words in them for the top 50 YT search terms and 11 Scroll downs.

For the place words, the only ones that seem to come in under the 5K word limit are the exotic places, like Zambezi, or the local New Jersey towns of Freehold, Oceanport, Keyport, and Morganville. With lots of different travel blogs, places tend to be easy to find in YouTube videos.

For product names, the big tech companies and car companies are easy to find. It was surprising that certain food companies, like Entenmann's, Fritos, and Nabisco, were difficult to find. Also, 'Avis' seems difficult to find although many videos were found but few contained the word 'Avis' in the subtitles.

TABLE I: COMPOUND WORD SEARCHING

Compound Word	Num Found Search for Compound Word Only	Num Found Search for Compound, Hyphenated, 2 Words	% Increase in Found Words
rockslide	518	1047	102%
semifinal	757	2717	259%
coatrack	27	513	1800%
crawdad	68	425	525%
milkmaid	492	598	22%
breadbox	288	456	58%
blowpipe	128	139	9%
outrider	290	310	7%

C. Detailed Look at Several Words

We chose 'Shannon', 'Dracula', and 'Mercedes', two names and a brand name, to explore more deeply. We first explore the relationship of the number of scroll downs to the number of wanted words found in the subtitles and the number of videos with a least one wanted word in them. For each search term, there is a finite limit to how many videos are relevant to the search term. Each scroll down loads ~20 more videos onto the page. Fig. 6 shows, for 'Mercedes', 'Dracula' and 'Shannon', the number of words found in the subtitles and the number of videos containing these words for different levels of scrolling down. There is a large increase in the number of words found when going from 0 to 1 scroll downs followed by a linear rise until about 15-20 scroll downs. 'Dracula' seems to hit its asymptotic limit at 15, whereas for 'Mercedes', after 15 scroll downs, the number of words is still rising, albeit at a slower pace. 'Shannon' shows a linear rise until 20 scroll downs and then a decrease in the number of words found. We will discuss this decrease below.

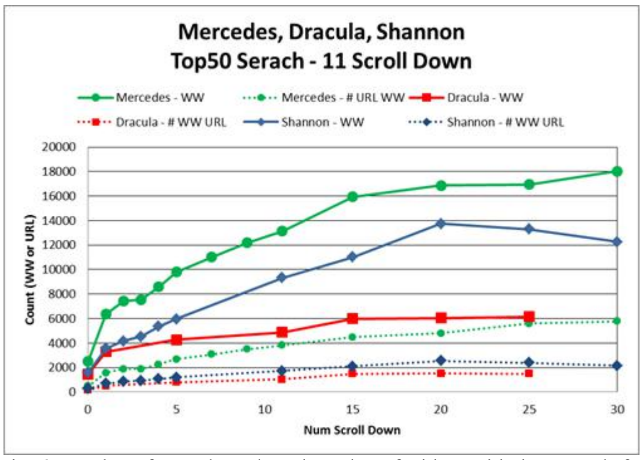


Fig. 6. Number of Found words and number of Videos with these words for different page scrolling for Mercedes, Dracula and Shannon.

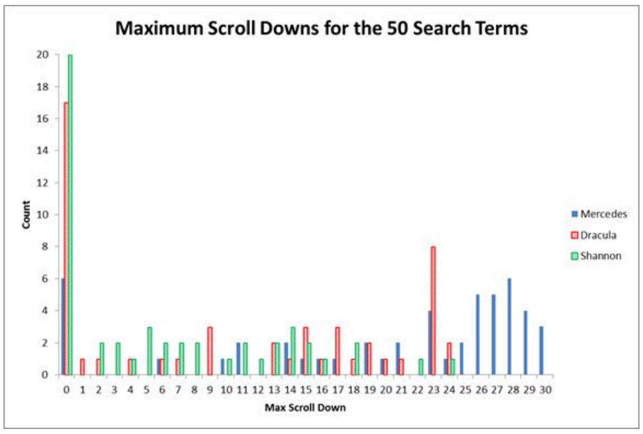


Fig. 7. Maximum Scroll Downs for the 50 Search terms for Dracula, Mercedes and Shannon.

By examining the maximum number of scroll downs per search term (see Fig. 7), we see that many search terms cannot reach the 30 scroll downs limit we set. Dracula and Shannon, for example, have >1/3 of the search terms returning less than 20 videos, thus having no scroll downs. One half of the videos have less than 10 scroll downs and none of the videos return enough videos to scroll down 25 times. Mercedes, which is more popular than Dracula or Shannon, has only 1/10 of the search terms having less than

20 videos and no scroll down and ~1/2 of the search terms returning enough videos for 25 or more scroll downs.

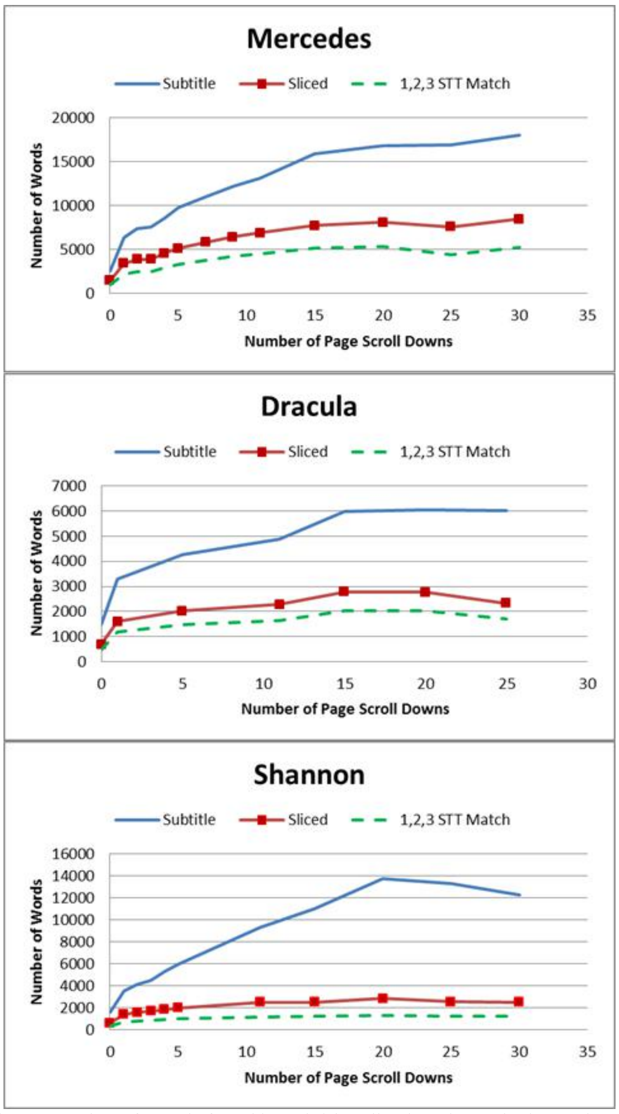


Fig. 8. Number of Words found in Subtitle, Sliced, and Matched to and ASR engine for Dracula, Mercedes and Shannon.

It should be noted, that when you search YouTube and use the autocomplete that the results are dynamic and change over time. Not only do the search term rankings change but also what videos are returned for a given search term change. When we were collecting data for Fig. 6 some of the 25 or 30 scroll downs were run a week after the other scroll downs. This sometimes lead to large differences in the scraped number of words, returning fewer words than the previously run 20 scroll downs. Also, many of the search terms return with the same ranking and you need to choose which will be in the top 50 (e.g. you have 30 terms with the same ranking and you only need 12 to fill in the top 50). We also tried to fix the search terms to the same terms from previous runs but this also returned different videos when run weeks apart. This variability lead to the decrease number of words found for Shannon beyond 20 scroll downs seen in Fig. 6 and Fig. 8.

We download the videos, slice out the words and check them against 3 different ASR engines. Fig. 8 shows the number of words identified in the subtitles, the number of words actually sliced out of the audio and the number of sliced words matching at least 1 of the ASR engines for each scroll down. Similar to the summary of this data shown in Fig. 4, Mercedes and Dracula show that ~50% of the words found in the subtitles are aligned by the forced aligner and sliced out. The other 50% that are not sliced are mostly due to the forced aligner not finding the word in the audio (discussed in section IV.A above). 33% of the words found in the subtitles are not only sliced but match at least 1 ASR engine.

For Shannon, a large number of words are found in the subtitles but only 20%-35% of the words are able to be sliced out. This lead to ~15% of the subtitle found words having at least one ASR match. The Shannon 15 to 30 scroll downs were run several weeks after the 0 to 5 scroll downs and we ended up with different search terms and not a lot of overlap in the found videos. Looking at all the videos for the different Shannon runs actually produces ~8500 sliced words with ~3800 passing the 1 ASR matching test. This is more than adequate for training a WWE. This shows that expanding the search terms beyond the top 50 for Shannon (this is what we effectively did by running the search term on different days) definitely helps find more words.

D. Alternate Search Methods

If an insufficient number of wanted words are found in the subtitles via the YouTube search autocomplete method, we try either expanding the number of search terms, using other search engines, using related search terms, or scrape the related videos found by the search terms. To explore these different methods, we chose 'dejection' (word rank 35255), a word from the 150 used in Section IV.A that had a low number of found words. These various methods are employed and the results are summarized in Table II. In all cases we use the maximum number of scrolls downs for each search term. For the top 50 YT search terms we only found 331 words in the subtitles. Expanding the number of YT search term to the maximum only produced 66 total search terms and 377 words found. Scraping the related videos from the YT max 66 search terms lead to 7x more videos but only increased the number of words found by 8. In this case scraping related videos was not very successful but we have found in other cases (such as for the word 'nova') more success with this method. Also looking up dejection in YouGlish only produced 17 instances of the word.

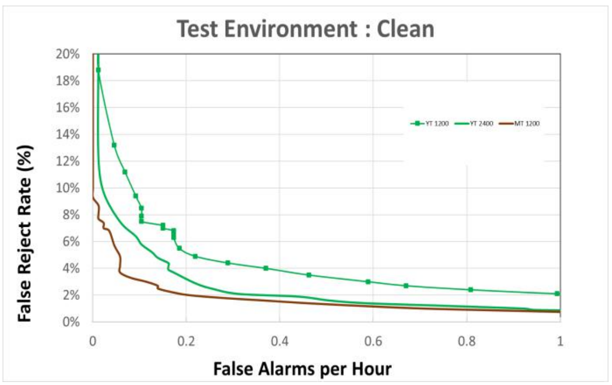


Fig. 9. ROC performance curve for a WWE with Shannon as the Wake Word.

Using the 3 different search engines (Google, Yahoo and Bing) has a large impact, almost doubling the amount of words found to >600. For Google we not only used the maximum number of search terms but also looked at the top 100 search terms. The top 100 Google terms were effective in finding many of the wanted words. Also, note that Yahoo only returned a maximum of 84 search terms, all containing dejection, whereas Bing and Google returned > 400. The search terms for Google and Bing returned non-related search terms for words like deception, ejection, deflection, and detection. These search terms were filtered out by removing any search terms that didn't contain at least one words containing the word fragment deject. The number of search terms in Table II reflects this filtering.

Combining the different search methods leads to 683 total wanted words found in the subtitles from 514 different videos. A final column in the table compares the overlap in the videos found between the different methods to the videos found in Google Max. Bing found 17 different videos from Google Max and Yahoo found 13 In this case the Bing or Google search methods we the most effective in finding words.

E. Corpus Testing - Wake Word Engine Training

To verify that our YouTube scraped words lead to good training corpora, we have used the 'Shannon' words found in the subtitles to train and test a Wake Word Engine. As a base line and for comparison we also train a WWE on 'Shannon' utterances gathered via Amazon Mechanical Turks. The corpora created for training are 1200 MTurk words, 1200 YouTube and 2400 YouTube scraped, tested and cleaned words. Note that the MTurk data collection took 3-4 weeks whereas the YouTube collection took several hours. For training, we utilized the DSCNN deep learning models in ARM's extended version [28] of TensorFlow keyword spotting tutorial [29]. For testing we used ~ 400 wake words collected from MTurk that are not used in the training corpus. We intersperse the wake works in non-wake words to create a testing set. For the non-wake words we use 500 episodes of Larry King (LK) downloaded from YouTube with the audio sliced into utterances ranging from 1 to 20 seconds in duration. We created long running test audios (89 hours) by mixing/overlaying the wake words with the non-wake words. Testing was done across 4 different test environments of clean, noisy, open air no reverberation and open air living room. For the complete details of this study see [30].

TABLE II: NUMBER OF WORDS FOUND FOR 'DEJECTION' USING VARIOUS SEARCH METHODS

Search Method	Num Search Terms	Num of Videos Found	Num Vid with Found Words	Num Words Found	Num Videos Not Found in Google Max
YouTube	50	521	246	331	9
YouTube - Max	66	848	284	377	3
Related Videos	66	6070	291	385	3
Google	100	2137	453	606	6
Yahoo - Max	84	1351	397	525	13
Bing – Max	168	3065	458	617	17
Google – Max	337	2539	483	636	0
Combine All Methods			514	683	

Fig. 9 shows the Receiver Operating Characteristics (ROC) curve for the resultant 3 WWE models in the clean test environment. The ROC curves display the False Reject Rate (i.e. percentage of time the WWE didn't respond to a wake word) versus the False Alarms per hours (i.e. how many time per hour the WWE responded to a non-wake word). The YT 2400 trained model performs equivalently to the MTurk model in this environment. Even though we needed twice as many training samples from YT to get the performance equivalence, the time to generate the test set went from weeks to hours. This shows that YouTube scraped data is a valid and viable alternative to corpus creation compared to the slow and costly MTurk data collection method.

V. DISCUSSION AND FUTURE WORK

For multi-word wake word (e.g. 'ok google'), finding enough of these exact phrases in YouTube may be problematic. We are also exploring creating these compound wake words by scraping each word individually from YouTube and then combining them together. The similarity of the speakers, background noise, gender, and other features need to be explored when combining the words in this manner to form a high-quality WWE.

One of the issues we discovered when training our wake word engine was that YouTube data has many cases where there is a substantial amount of background noise. Training a WWE and testing in a clean environment with clean test data didn't always perform as well as could be expected. We filtered the found words using an ASR check but never check for background noise level. We could further improve our corpus creation use SNR estimates like in [31] to separate noisy from clean data.

VI. CONCLUSION

We have explored alternative data collection techniques in the context of speech data in this paper. We show how to create data sets using found data and thus avoid the bottleneck in DL of data set creation. Our approach in this paper specifically looks at finding individual words for wake word engine creation, however, these techniques can be applied to create speech corpus for various purposes.

We show the feasibility of locating specific utterances in public data sources by searching for many common words, Name, Places and Product Names. These words can be found in sufficient quantities to construct high quality training sets for DL. We elaborate of different techniques to construct search terms and methods to efficiently locate additional utterances when faced with difficult words or phrases. We demonstrate that 40-50% of the words found in the subtitles can be extracted and sliced into individual words and > 30% of these subtitle found words can be sliced and recognized by at least 1 of the 3ASR engines we use for validation making it suitable for corpus creation. We use 'Shannon' as a test case to create a test corpus from found data and train a high quality Wake Word Engine. We show equivalent performance of this trained WWE to traditionally trained WWE.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Drabeck and Ramanan conducted the research and analyzed the data. Drabeck and Woo wrote the paper and all authors approved the final version.

References

[1] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Proc. IEEE Spoken Language Technology Workshop*, 2016, pp. 474–480.

[2] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," *INTERSPEECH*, 2018.

[3] S. Aslam, "YouTube by the Numbers," *OMNICORE*, 2018.

[4] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. Freeman, "Learning the depths of moving people by watching frozen people," arXiv:1904.11111v1, Apr. 2019.

[5] Wikipedia. Mannequin Challenge. [Online]. Available: https://en.wikipedia.org/wiki/Mannequin_Challenge

[6] X. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "SFV: Reinforcement learning of physical skills from videos," *ACM Trans. Graph*., vol. 37, no. 6, Article 178, November 2018.

[7] T. Li, L. Lin, M. Choi, K. Fu, S. Gong, and J. Wang, "YouTube AV 50K: An annotated corpus for comments in autonomous vehicles," in *Proc. 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, Pattaya, Thailand, 2018, pp. 1-5.

[8] B. Shillingford *et al.*, "Large-Scale Visual Speech Recognition," arXiv:1807.05162v3, 2018.

[9] J. Gemmeke *et al*., "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 776-780.

[10] H. Liao *et al*., "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription", in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 368-373.

[11] A. Ephrat, I. Mosseri, O. Lang *et al.*, "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation," *ACM Trans. Graph*., vol. 37, no. 4, August 2018.

[12] E. Lakomkin, S. Magg, C. Weber, and S. Wermter, "KT-Speech-Crawler: Automatic dataset construction for speech recognition for YouTube videos," in *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing*, Belgium, Oct. 2018, pp. 90-95.

[13] A. Ragni and M. Gales, "Automatic Speech recognition system development in the 'wild'," in *Proc. Interspeech*, 2018, pp. 2217-2221.

[14] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," in *Proc. Interspeech*, 2017, pp. 2616-2620.

[15] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," arXiv preprint arXiv:1806.05622.

[16] W. Xie, A. Nagrani, J.-S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2019

[17] J. Lorenzo-Trueba, F. Fang, X. Wang, I. Echizen, J. Yamagishi, and T. Kinnunen, "Can we steal your vocal identity from the Internet?: Initial investigation of cloning Obama's voice using GAN, WaveNet and low-quality found data," in *Proc. Odyssey 2018 the Speaker and Language Recognition Workshop*, 2018, pp. 240-247.

[18] R. Ochshorn and M. Hawkins. Gentle. [Online]. Available: https://lowerquality.com/gentle

[19] A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: an automatic speech recognition dedicated corpus," in *Proc. the Eighth International Conference on Language Resources and Evaluation (LREC)*, 2012, pp. 125–129.

[20] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *Proc. the Ninth International Conference on Language Resources and Evaluation (LREC),* 2014, pp. 3935–3939.

[21] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng, *Deep Speech: Scaling up End-to-End Speech Recognition*, Jun. 6, 2019.

[22] Youglish.com. Youglish. [Online]. Available: https://youglish.com/

[23] Related Words. [Online]. Available: https://relatedwords.org/

[24] Reverse Dictionary. [Online]. Available: https://reversedictionary.org/

[25] Describing Words. [Online]. Available: https://describingwords.io/

[26] PyRoomAcustics. [Online]. Available: https://readthedocs.org/projects/pyroomacoustics/

[27] Corpus of Contemporary English. [Online]. Available: https://www.english-corpora.org/coca/

[28] Keyword spotting for Microcontroller. [Online]. Available: https://github.com/ARM-software/ML-KWS-for-MCU

[29] Tensorflow Speech Commands Example. [Online]. Available:https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands

[30] B. Ramanan, L. Drabeck, T. Woo, T. Cauble, and A. Rana, "Eliminating Data Collection Bottleneck for Wake Word Engine Training Using Found and Synthetic Data," in *Proc. the 10th Conference on Speech Technology and Human-Computer Dialogue*.

[31] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. Freeman, and M. Rubinstein, "Looking to listen at the cocktail party: A Speaker-independent audio-visual model for speech separation," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 112:1-112:11, 2018.

Lawrence Drabeck got a B.S in 1986, M.S. in 1987 and Ph.D. in 1991 degrees in experimental condensed matter physics from the University of California at Los Angeles.

He is a member of the technical staff in the Applications Platforms and Software Systems of Nokia Bell Labs, Holmdel, NJ since 2014. Previously he was a member of the technical staff in the Wireless Research Department of Bell Labs since 1993. He was a postdoctoral fellow at MIT/Bell Labs working on high temperature superconducting filter technologies. In the past he has worked on wireless filters and front ends as well as wireless network optimization and big data network analysis. His present research interests are AI system development.

Buvana Ramanan holds a master of science degree from the Indian Institute of Science, Bangalore, India.

She is a research engineer at the Applications, Platforms, Software and Systems lab in Nokia Bell Labs, Murray Hill, NJ. Her research interests include deep learning, artificial intelligence, speech recognition and synthesis, data engineering for AI applications and big data systems. She has numerous referred publications in the fields of image processing, wireless network optimization and subscriber data analysis.

Thomas Woo holds a PhD in computer science from the University of Texas at Austin.

He is a group research leader in the Software and Systems Lab of Nokia Bell Labs. He has led research in areas such as AI systems, large scale distributed systems, networking and security. He has also done ventures and startups.

Troy Cauble got B.S. in 1985 and M.S. in 1986 degrees in electrical engineering from Purdue University, West Lafayette, IN, USA.

He is a member of the technical staff in the Applications, Platforms and Software Systems of Nokia Bell Labs, Holmdel, NJ since 2015. He has been a member of the technical staff in various organizations within Bell Labs for a very long time. His current research interests include distributed systems.