Duke UNIVERSITY | PRATT SCHOOL of ENGINEERING

Automatic Speech Recognition Using the Kaldi Toolkit

Madeline F. Briere & Michael R. Gustafson, Ph.D.

Marvin AI Systems LLC

Abstract

This project explores the current technology available for *Automatic Speech Recognition* (ASR), the process of converting speech from a recorded audio signal to text.[1] Our goal is to identify a toolkit for use in the construction of a personal assistant system, similar to current on-the-market assistants, but with a smaller and more targeted lexicon to increase accuracy. In particular, we explore the Kaldi Speech Recognition Toolkit, written in C++ and licensed under the Apache License v2.0, developed for use by speech recognition researchers.[2] This toolkit was chosen on the grounds of extensibility, minimal restrictive licensing, thorough documentation (including example scripts), and complete speech recognition system recipes. We explore the ASR process used in Kaldi and assess three extensions of the Kaldi toolkit (*Digits, VoxForge*[3] and *CMU AN4*[4]). This project demonstrates that Kaldi can be extended in simple and complex situations and is flexible and easy to use in development. Kaldi is a viable choice for future extension.

Goals

Project Goals:
The ultimate end goal **for this project** is to assess the viability of the Kaldi ASR Toolkit.[2] Kaldi is a well-documented ASR toolkit that aims to provide complete speech recognition recipes to users. This system deals with the entire ASR process, from WAV file to text transcription. Kaldi seems the perfect solution to the homegrown vs. outsourced debate. Therefore, we explore extensions of Kaldi to demonstrate the viability of the toolkit for use in the team product.

Team Goals:
Our team is the Marvin AI Systems LLC engineering team. The ultimate end goal **for this team** is to develop a prototype system for ASR. This system is meant for use as a *personal assistant* in *automobile shops*.

This prototype must satisfy the following requirements:

- Lightweight and minimal
- Easy to develop and extend
- Maintains a balance between a *homegrown* (self-owned, but more prone to bugs and less extensive) and *outsourced* system (lack of ownership, but more extensive and well-tested)
- Accurate and fast (less than ten second wait time)

Kaldi Toolkit Layout

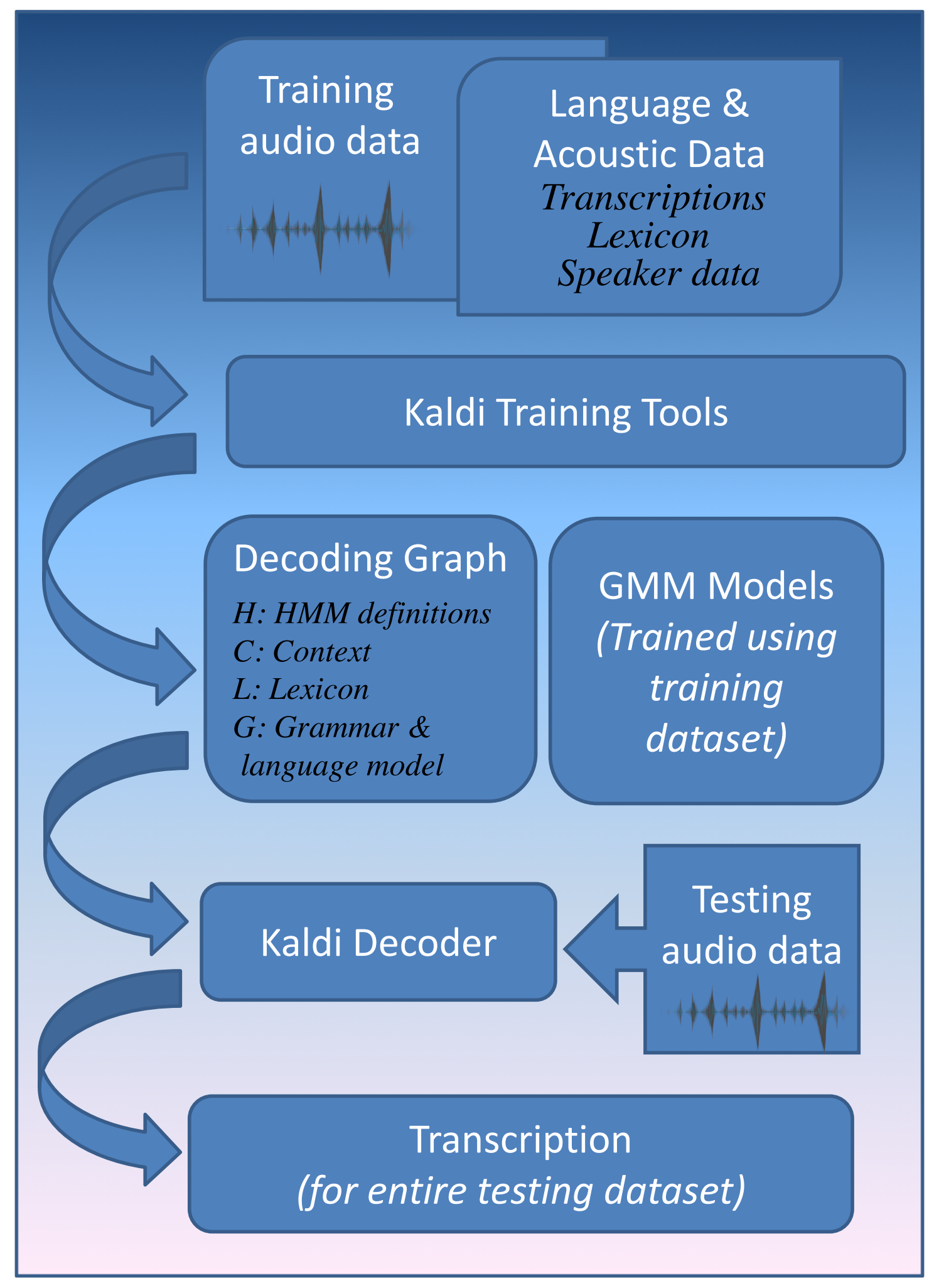


Figure 1: Layout of Kaldi Toolkit (based on NTNU diagram and Kaldi documentation).[2,5] Note that this diagram is hugely simplifying – optimizations and adjustments (e.g., using alignments) are not shown.

The general layout of the Kaldi Toolkit is displayed in *Figure 1*. It accepts a set of customizable audio data as input, along with accompanying language and acoustic data. This data is used to generate a decoding graph (of the *HCLG* format; see *Fig. 1*) and final GMM model. These pieces (*HCLG.fst* and *final.mdl*) can be fed into the decoder, along with testing features to produce transcriptions.[2] Of primary interest to us is the customizable input. In each extension, we have to define:

- Audio data(training and testing)
- Acoustic data
  - spk2gender: <speakerID><gender>
  - wav.scp: <utteranceID> <file_path>
  - text: <utteranceID> <transcription>
  - utt2spk: <utteranceID> <speakerID>
  - corpus.txt: <transcription>
- Language data
  - lexicon.txt: <word> <phone(s)>
  - nonsilence_phones.txt: <phone>
  - silence_phones.txt: <phone>
  - optional_silence.txt: <phone>
- (Optional) Configuration
- (Optional) Language model toolkit

Initial Assessment of Kaldi

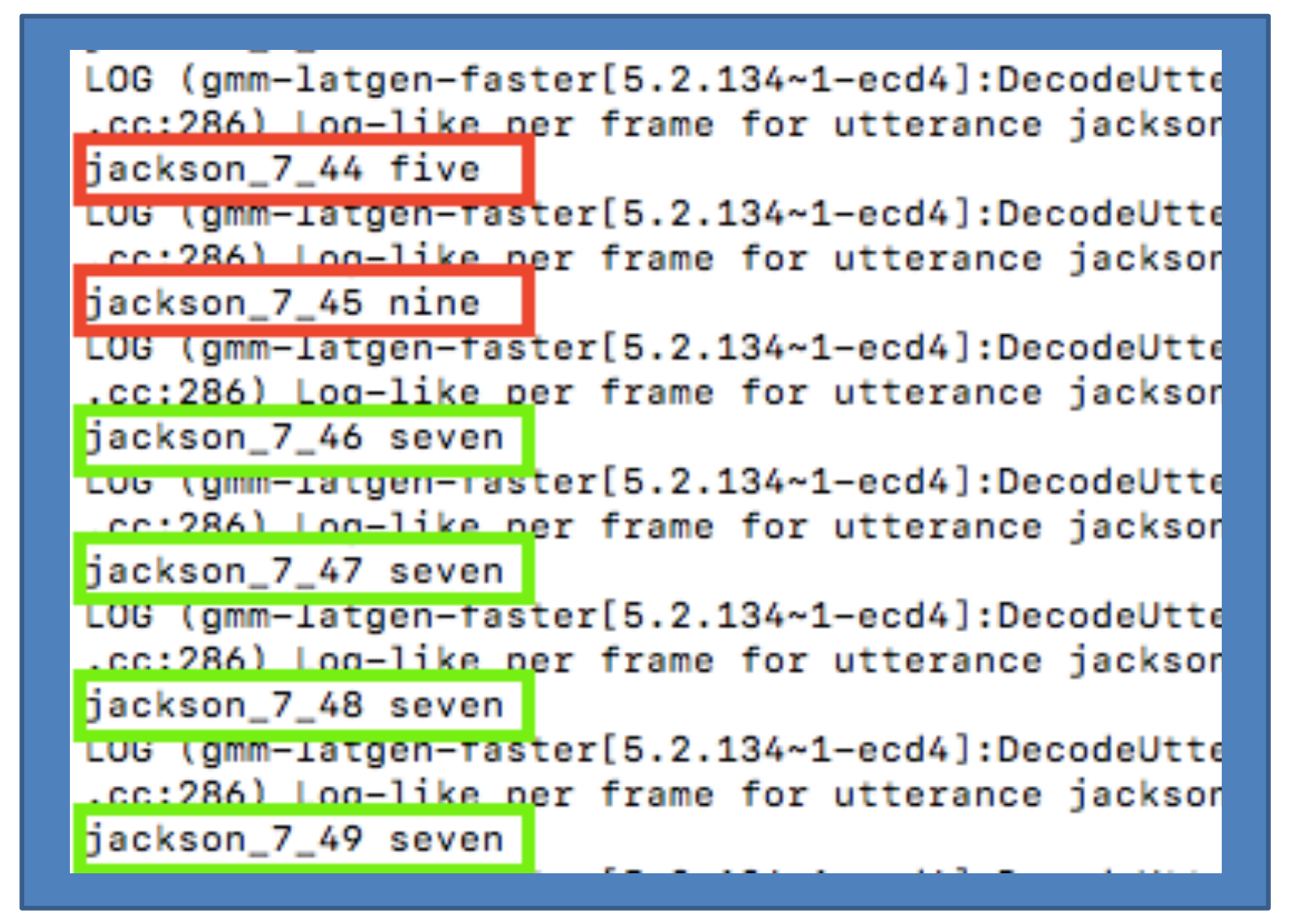
An initial assessment of Kaldi (see *Figure 2*) reveals it to be a viable system for the desired product. Kaldi includes a variety of utility scripts, including functionalities such as feature extraction, data preparation, transition modeling, construction of decoding graphs, and acoustic modelling. Extensions of Kaldi can incorporate custom training and testing data and use the corresponding lexicon. These extensions can still utilize the provided scripts, substituting in various decoding types, language models, etc.

Requirements	✓	✗
Easy to use	- Well-documented - Has extensive support system (Git, Kaldi homepage, help pages) - Many examples (including VoxForge, AMI, and Fisher)	- Requires knowledge of shell coding[5] - Not initially designed for "casual use" (meant to be used by full-time speech recognition researchers)[2]
Extensible	- Can reasonably build off of examples - Built specifically for extension with new datasets/models	- Complex extension requires intimate knowledge of Kaldi system - Commands change frequently[5]
Partly homegrown	- Extensions possible through customized scripting	- Customization leaves room for suboptimal configurations - Potentially buggy
Partly outsourced	- Extensive toolkit for feature extraction, decoding, etc. - Open license (limited legality concerns)[2]	- Less intimate knowledge of system

Figure 2: Assessing the viability of Kaldi (note that speed was not considered in this analysis)

Results

Digits Extension
The digits extension is a simple example, designed to translate audio clips of spoken digits 0 through 9. It is inspired by a "For Dummies" tutorial on the Kaldi site.[2] The audio data is from the Jakobovski Free Spoken Digit Dataset,[7] which provided a set of 1500 audio clips. 1000 clips (from speakers Nicolas and Theo) were used for training and 500 clips (from speaker Jackson) for testing. Each digit is spoken 150 times between the three speakers. This example required script-generated language and acoustic data in order to characterize the lexicon, phonetic mappings, etc. expected from the audio data. The Kaldi system uses this data as input (see *Fig. 1)*.
The decoding results were produced for two types of training, triphone and monophone (where the former includes contextual information, but the latter does not).[8] The decoding results for the training set were compared against the expected results (log examples shown in *Figure 3*) to produce a series of *Word Error Rates* for the ASR system. Because of the small lexicon and relatively large dataset, the word error rate for this example was low (optimal: 9.6%, triphone).



```
LOG (gmm-latgen-faster[5.2.134~1-ecd4]:DecodeUtte
.cc:286) Log-like per frame for utterance jackson
jackson_7_44 five
LOG (gmm-latgen-faster[5.2.134~1-ecd4]:DecodeUtte
.cc:286) Log-like per frame for utterance jackson
jackson_7_45 nine
LOG (gmm-latgen-faster[5.2.134~1-ecd4]:DecodeUtte
.cc:286) Log-like per frame for utterance jackson
jackson_7_46 seven
LOG (gmm-latgen-faster[5.2.134~1-ecd4]:DecodeUtte
.cc:286) Log-like per frame for utterance jackson
jackson_7_47 seven
LOG (gmm-latgen-faster[5.2.134~1-ecd4]:DecodeUtte
.cc:286) Log-like per frame for utterance jackson
jackson_7_48 seven
LOG (gmm-latgen-faster[5.2.134~1-ecd4]:DecodeUtte
.cc:286) Log-like per frame for utterance jackson
jackson_7_49 seven
```

Figure 3: A sample log from the digits example, showing transcriptions for several audio files of the number "7"

VoxForge Extension
VoxForge is an open source acoustic model (including a huge speech corpus), initially set up to collect transcribed speech for use with Free and Open Source Speech Recognition Engines.[3] This example was particularly interesting because it was much more complex than introductory examples: it had a lexicon of ~16,000 words and required the use of the Sun Grid Engine[4] for parallelization (requiring extensive configuration). The complexity of this system greatly increased its error rate (optimal: 22.53%, triphone last pass).

CMU AN4 Extension
The CMU AN4 (the Alphanumeric database)[5] is a series of census data recorded at CMU in 1991. This data was used to create a system capable of recognizing alphanumeric queries. This example provided insight as to how non-formatted audio and acoustic data can be funneled into Kaldi. A script was written to retrieve the dataset from the CMU site, renaming and sorting it into training and testing folders. This script also extracted the lexicon, phones, transcriptions, etc. from the /etc files. Because this system has such a small lexicon and a large training set, it yielded a low word error rate (optimal: 6.27%, triphone last pass).

In *Figure 4* below, we can see how the error rates from these Kaldi examples line up with current error rates for on-the-market systems. It should be noted that the comparison is not necessarily between completely similar systems – Google and Cortana are on much larger scales, designed to recognize all possible input (as opposed to our customized systems). In *Figure 5*, we can see how the training dataset size must increase with the lexicon size to maintain a reasonable word error rate for the system.

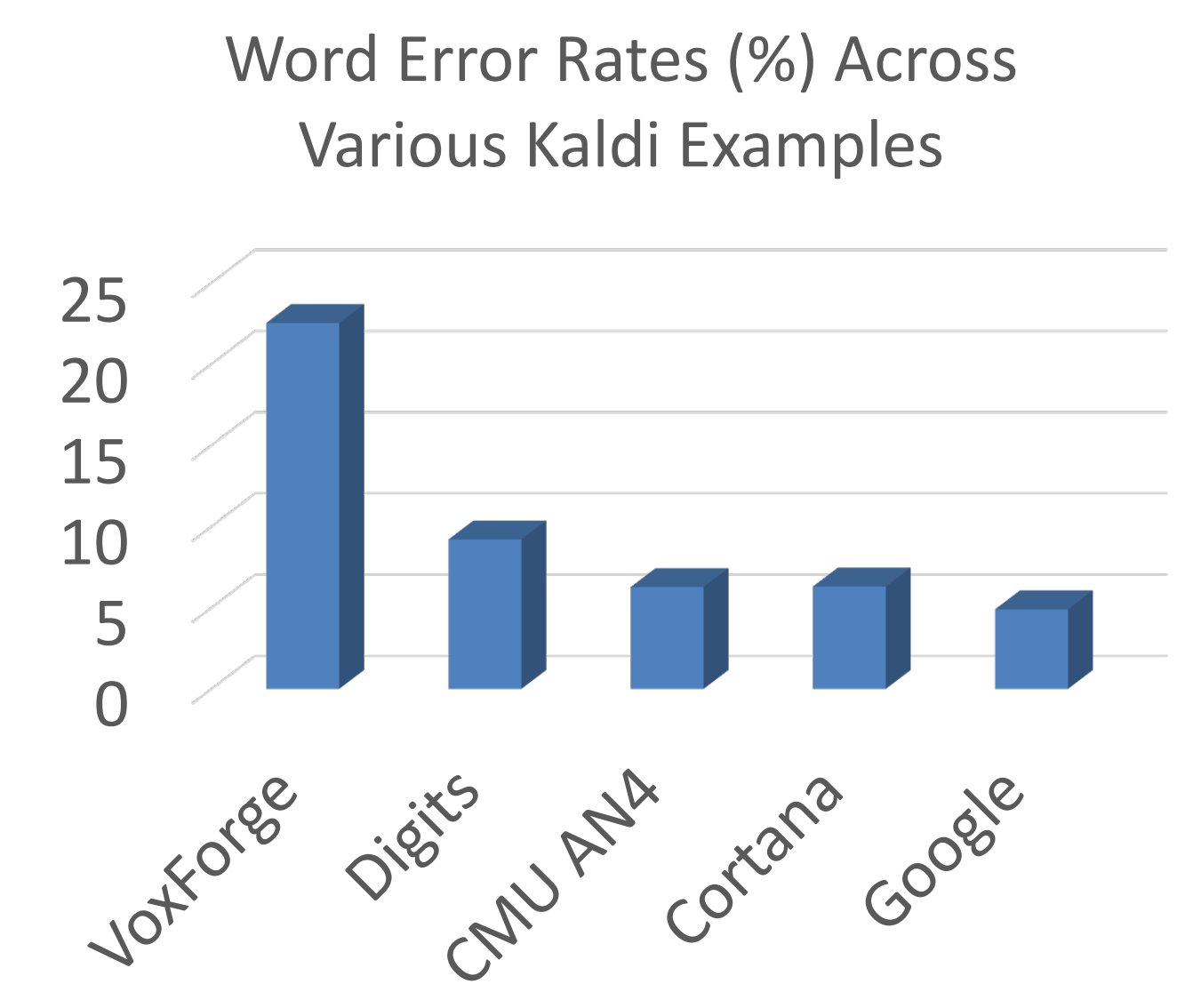


Figure 4: Word Error Rates in Kaldi examples compared to readily available systems[9,10]

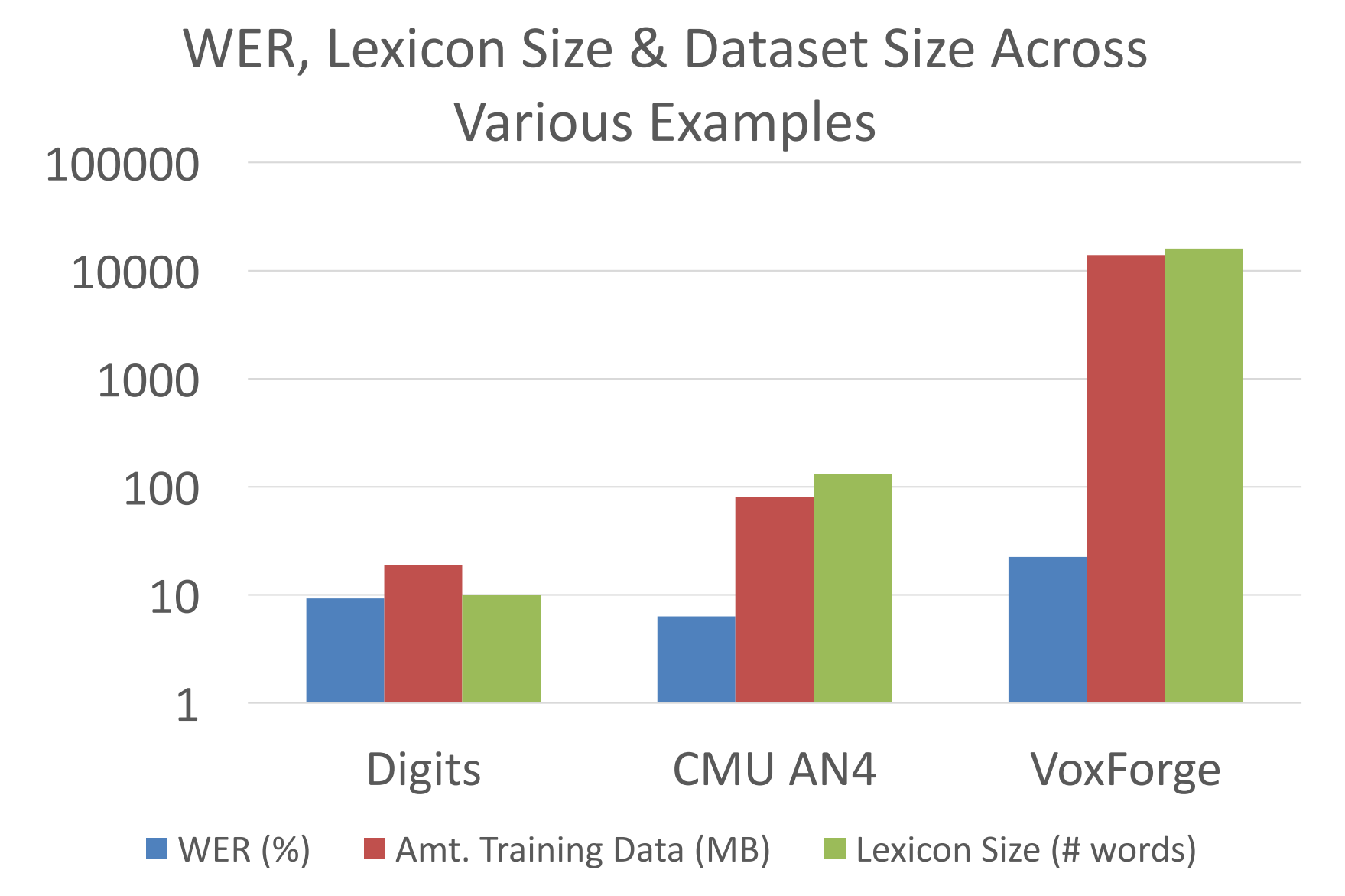


Figure 5: Word Error Rates, lexicon size and dataset size across Kaldi Examples (log scale)

Discussion

Given the flexibility demonstrated by the Kaldi toolkit, it is safe to say that further extensions and explorations will be possible. The ideal case will involve the incorporation of a large, custom training dataset, which we have shown to be possible. Another important extension will be real-time encoding – right now, this system is geared towards static, already-recorded datasets. Our product will require a dynamic system that can accommodate real-time decoding. Such examples are clearly possible, as indicated by the Kaldi Online Decoding Tutorial[2] and the Kaldi GStreamer (a real-time speech recognition server implemented using Kaldi and readily available on Github).[11] It should be noted that a variety of elements were *not* considered in this analysis, including speed. Future explorations must confirm that Kaldi real-time decoding is capable of supplying speech-to-text results in under ten seconds (given our custom dataset). The information we have seen so far, however, indicates that Kaldi is capable of providing accurate and flexible solutions to the problem of speech recognition.

Acknowledgements

Intellectual property of *Marvin AI Systems LLC*
Completed as Independent Study for *Duke University*

References

[1] Gruhn et. al. "Automatic Speech Recognition." *Statistical Pronunciation Modeling for Non-Native Speech Processing*, 2011.
[2] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. "Kaldi Speech Recognition Toolkit." *kaldi-asr.org*, 2011.
[3] "VoxForge: Open Source Acoustic Model and Audio Transcriptions." *VoxForge,* 2006-2017.
[4] "Sun Cluster Data Service for Sun Grid Engine Guide for Solaris OS." *Oracle,* 2010.
[5] "CMU Census Database." *Carnegie Mellon University Audio Database,* 1991.
[6] Chen, Berlin. "An Introduction to the Kaldi Speech Recognition Toolkit." *National Taiwan Normal University,* 2014.
[7] Jackson, Zohar. "Free Spoken Digit Dataset (FSDD)." 2017.
[8] Chodroff, Eleanor. "Corpus Phonetics Tutorial: Kaldi." *Northwestern University,* 2017.
[9] Sundar Pichai. Google's I/O Developer Conference, 2017.
[10] George Saon. "Recent Advances in Conversational Speech Recognition." *IBM,* 2016.
[11] Tanel Alumae. "Kaldi GStreamer server." 2017.