INTERSPEECH 2019: Show & Tell Contribution
September 15–19, 2019, Graz, Austria

Elpis, an accessible speech-to-text tool

Ben Foley[1 2], *Alina Rakhi*[1 2], *Nicholas Lambourne*[1 2], *Nicholas Buckeridge*[1 2], *Janet Wiles*[1 2]

[1]The University of Queensland, Australia
[2]ARC Centre of Excellence for the Dynamics of Language (CoEDL), Australia

b.foley@uq.edu.au, a.ajayan@uq.net.au, n.lambourne@uq.edu.au,
nicholas.buckeridge@uqconnect.edu.au, j.wiles@uq.edu.au

Abstract

Elpis is a speech-to-text tool which has been designed to give language workers, including linguists and speech scientists, access to cutting-edge automatic speech recognition software, without the specialist training typically required to run these systems. Our presentation would demonstrate local and server-based versions of Elpis using sample data from the Abui (ISO 639: abz) language, about 17,000 speakers in Indonesia. Attendees would gain a sense of the benefits that a first-pass ASR transcription can bring to transcription workflows.

Index Terms: speech recognition, speech and language technology and systems, speech-to-text, co-design, accessibility

1. Introduction

The task of transcribing recorded audio is important in many language and speech science workflows, yet it can be very slow. A 2017 survey of 50 linguists found that it takes 40 hours to transcribe one hour of audio on average [1]. For some languages the ratio was reported up to 230:1. Using contemporary software techniques such as automatic speech recognition (ASR), we can improve the experience of the language worker, resulting in practical and psychological benefits for their work [2].

2. Elpis

2.1. Speech-to-text Technologies

ASR technologies such as Google Translate can transcribe 180 of the world's languages [3], but these tools aren't able to be used by language researchers or communities for languages not supported by the provider. Typically, third-party provided ASR services don't reach to any endangered languages, or for languages with small quantities of data for training ASR models. For languages not covered by third-party providers, speech recognition toolkits such as Kaldi [4] can be used given significant specialist training to install and configure the software. This requirement for highly specialist skills puts this approach out of reach of many language communities and researchers.

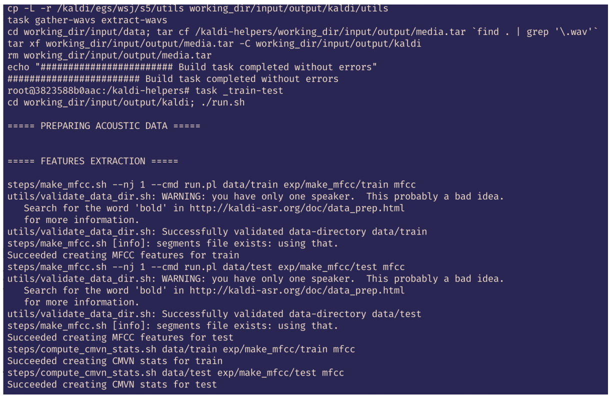
2.2. Elpis

To address this gap, the ARC Centre of Excellence for the Dynamics of Language has been developing Elpis [5], an ASR system which can be used to obtain a first-pass transcription on un-transcribed audio, without requiring a high degree of specialist training to implement. Elpis brings cutting-edge speech recognition technology within reach of low-resource language researchers who don't have backgrounds in speech engineering. With Elpis, we hope to enable community members to make a significant impact on transcribing their own languages from new recordings, or breathing life into archival, cultural heritage material.

Elpis has been developed to be user-friendly for novice and technical users. At the core of Elpis, Kaldi provides the technology required to train ASR models and obtain an inference on un-transcribed audio. Elpis provides a means to interact with Kaldi via either a graphical user interface (GUI) as a browser-based application, or via a Python application programming interface (API).

2.3. Command-line usage

Prior to the design of the GUI, Elpis consisted of a set of Python scripts which were used to process audio and annotation data into the formats and locations expected by Kaldi. Scripts were used to run training stages of a Kaldi 'recipe', and to infer a transcription for un-transcribed audio. Interaction with the Python scripts was by the command-line, requiring some degree of familiarity with command-line usage. For some users, this was a significant barrier to overcome, which led to the response of designing a graphical interface, more aligned with the types of interfaces with which many language workers are familiar.



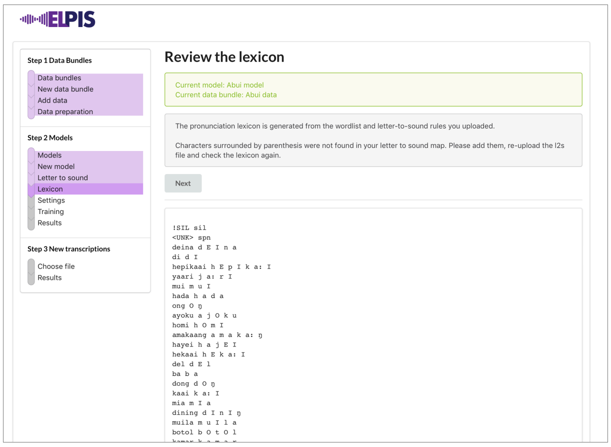
```
cp -L -r /kaldi/egs/wsj/s5/utils working_dir/input/output/kaldi/utils
task gather-wavs extract-wavs
cd working_dir/input/data; tar cf /kaldi-helpers/working_dir/input/output/media.tar `find . | grep '\.wav'`
tar xf working_dir/input/output/media.tar -C working_dir/input/output/kaldi
rm working_dir/input/output/media.tar
echo "######################## Build task completed without errors"
######################## Build task completed without errors
root@3823588b0aac:/kaldi-helpers# task _train-test
cd working_dir/input/output/kaldi; ./run.sh

===== PREPARING ACOUSTIC DATA =====


===== FEATURES EXTRACTION =====

steps/make_mfcc.sh --nj 1 --cmd run.pl data/train exp/make_mfcc/train mfcc
utils/validate_data_dir.sh: WARNING: you have only one speaker.  This probably a bad idea.
   Search for the word 'bold' in http://kaldi-asr.org/doc/data_prep.html
   for more information.
utils/validate_data_dir.sh: Successfully validated data-directory data/train
steps/make_mfcc.sh [info]: segments file exists: using that.
Succeeded creating MFCC features for train
steps/make_mfcc.sh --nj 1 --cmd run.pl data/test exp/make_mfcc/test mfcc
utils/validate_data_dir.sh: WARNING: you have only one speaker.  This probably a bad idea.
   Search for the word 'bold' in http://kaldi-asr.org/doc/data_prep.html
   for more information.
utils/validate_data_dir.sh: Successfully validated data-directory data/test
steps/make_mfcc.sh [info]: segments file exists: using that.
Succeeded creating MFCC features for test
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train mfcc
Succeeded creating CMVN stats for train
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test mfcc
Succeeded creating CMVN stats for test
```

Figure 1: *Elpis command-line output*

2.4. GUI

The Elpis GUI was designed in collaboration with linguists over the course of a Summer Research Program, following co-design processes to ensure the interface and intention of the tool were aligned with people's actual work needs. Co-design participants included language documentation linguists, phoneticians, archivists, early-career and highly experienced researchers. This coverage of experience and disciplines ensures that Elpis is adaptable to a wide range of situations.

Primary considerations in the design of the interface were to increase user confidence in the speech-to-text process, increase reproducibility of using the system, and give users opportunities to influence the training of models by adapting results of automated data preparation stages. The interface gives the same functionality as the prior set of scripts, processing and moving files into the formats and locations that Kaldi expects, with a significantly lower technical barrier to entry (and a lower fear factor).



Review the lexicon

Current model: Abui model
Current data bundle: Abui data

The pronunciation lexicon is generated from the wordlist and letter-to-sound rules you uploaded.

Characters surrounded by parenthesis were not found in your letter to sound map. Please add them, re-upload the l2s file and check the lexicon again.

```
!SIL sil
<UNK> spn
deina d E I n a
di d I
hepikaai h E p I k a: I
yaari j a: r I
mui m u I
hada h a d a
ong O ŋ
ayoku a j O k u
homi h O m I
amakaang a m a k a: ŋ
hayei h a j E I
hekaai h E k a: I
del d E l
ba b a
dong d O ŋ
kaai k a: I
mia m I a
dining d I n I ŋ
muila m u I l a
botol b O t O l
```

Figure 2: *The Elpis graphical user interface*

2.5. Python API

Maintaining programmatic access to the functionality provided by the Python scripts has been important in the development of Elpis. To satisfy this, the scripts previously used in the first command-line version have been re-designed to be used via an API, rather than being used directly. An application programming interface (or API) is a simple, standardised way of triggering more complex behaviour in an underlying software system. At a fundamental level, they allow software systems to "talk" to one another. APIs can be attached to nearly any software system, allowing people to programmatically perform actions like sending text messages, performing a bank transaction, or in our case, transcribing recorded audio.

2.6. Technologies

Elpis is composed of multiple technologies. At its core is the Kaldi ASR toolkit, itself a collection of Shell and C++ scripts. The API is written in Python, the GUI is a React app which interacts with a Python Flask server.

Elpis is deployed via a Docker [6] image, a virtualisation technology which enables one code base to be distributed to the multitude of computer operating systems that are evident in the field of language research.

Elpis can be installed locally on an individual computer or deployed onto a server for people to remotely access. For some people, local installation will enable them to use this technology for language research with restrictive ethics, that would otherwise preclude them uploading data to a third-party service's cloud storage and processing servers.

2.7. Usage

Using the system requires some amount of annotated data (currently in the form of Elan [7] transcriptions) to use as a training set to build an ASR model. The results from a first-pass inference can be exported in Elan format and edited, then the model can be re-trained. Future work includes an interface to edit the results in the Elpis interface and update the model based on the corrections.

3. Conclusions

Elpis has been developed to satisfy a need for accessible speech-to-text technologies, usable by people without specialist training. The design of Elpis provides a means for novice users to confidently use ASR tools, and supports power-user access to Kaldi ASR toolkit via an API.

4. References

[1] G. Durantin, 2017. Early results from survey exploring transcription processes. Available: http://www.dynamicsoflanguage.edu.au/news-and-media/latest-headlines/article/?id=early-results-from-survey-exploring-transcription-processes [Accessed 4.1.19]

[2] A. Michaud, O. Adams, T. Cohn, G. Neubig and S. Guillaume. 2018. Integrating Automatic Transcription into the Language Documentation Workflow: Experiments with Na Data and the Persephone Toolkit.

[3] Google. Google Cloud Speech-to-Text API Language Support [Online]. https://cloud.google.com/speech-to-text/docs/languages. [Accessed 4.1.2019].

[4] D. Povey, et al. The Kaldi speech recognition toolkit. No. CONF. IEEE Signal Processing Society, 2011.

[5] B. Foley, et al. "Building speech recognition systems for language documentation: The CoEDL Endangered Language Pipeline and Inference System (ELPIS)." 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages. 2018.

[6] Docker. Enterprise Container Platform for High Velocity Innovation [Online]. https://docker.com/. [Accessed 26.1.2019].

[7] ELAN (Version 5.2) [Computer software]. (2018, April 04). Nijmegen: Max Planck Institute for Psycholinguistics. https://tla.mpi.nl/tools/tla-tools/elan/. [Accessed 26.1.2019].