INTERSPEECH 2020: Show & Tell Contribution
October 25–29, 2020, Shanghai, China

Kaldi-web: An installation-free, on-device speech recognition system

Mathieu Hu[1], *Laurent Pierron*[1], *Emmanuel Vincent*[1], *Denis Jouvet*[1]

[1]Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
{mathieu.hu, laurent.pierron, emmanuel.vincent, denis.jouvet}@inria.fr

Abstract

Speech provides an intuitive interface to communicate with machines. Today, developers willing to implement such an interface must either rely on third-party proprietary software or become experts in speech recognition. Conversely, researchers in speech recognition wishing to demonstrate their results need to be familiar with technologies that are not relevant to their research (e.g., graphical user interface libraries). In this demo, we introduce Kaldi-web[1]: an open-source, cross-platform tool which bridges this gap by providing a user interface built around the online decoder of the Kaldi toolkit. Additionally, because we compile Kaldi to Web Assembly, speech recognition is performed directly in web browsers. This addresses privacy issues as no data is transmitted to the network for speech recognition.
Index Terms: speech recognition, human-computer interaction

1. Introduction

The past years have seen an increase in the number of applications relying on voice commands. This ranges from all-purpose personal assistants to more specialized applications. Today, the development of voice-enabled applications requires developers to rely on proprietary, cloud-based automatic speech recognition (ASR) software which sends user data across the network. This raises the issues of data protection and user privacy as well as the dependency on such proprietary software.

At the same time, researchers from the ASR community find new approaches to improve ASR systems. However, the relevance of their work to real-word scenarios may not be obvious to the developer and ASR enthusiast community as researchers would need to package their work in an intuitive interface to showcase it.

In this paper, we present an open-source, cross-platform package built around the popular Kaldi toolkit [1]. Our package aims to bridge the gap between these communities by providing

- an ASR engine running in web browsers that can load any small-sized Kaldi nnet3 model,
- a readily available Graphical User Interface (GUI) to control this ASR engine.

Our package is under a permissive licence: the source code we authored is under an Apache-2.0 license while the rest is either under an Apache-2.0 (Kaldi), a BSD-2.0 (libsamplerate[2]) or a CLAPACK license (CLAPACK, CBLAS, BLAS [2]).

The remainder of this paper describes our package in more detail. In Section 2, the general architecture of the package is shown. In Section 3, we detail how to showcase custom ASR models with our GUI as well as potential constraints that these models have to meet. Section 4 describes the resources used in the live demo of the package. This demo can be viewed at https://kaldi-web.loria.fr.

[1] https://gitlab.inria.fr/kaldi.web/kaldi-wasm

[2] http://www.mega-nerd.com/SRC/

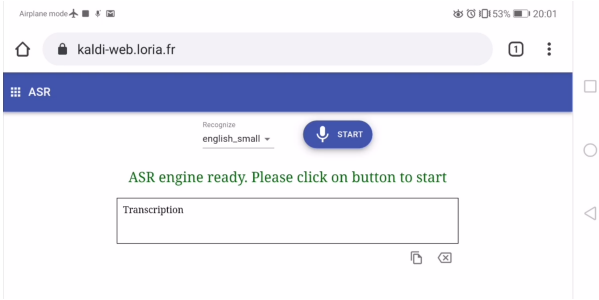

Figure 1: *Graphical user interface.*

2. Package architecture

2.1. Audio signal flow

The GUI is composed of a drop-down list of available ASR models, a button to start or stop the ASR system, and a text field displaying what has been recognized, as shown in Fig. 1.

To prevent heavy computations from freezing the GUI, and thus giving a poor user experience, the audio signal is processed in separate threads as shown in Fig. 2. These threads carry out the following tasks:

- audio formatting: the signal is resampled and encoded over 16-bit signed integers using libsamplerate;
- ASR: the formatted audio signal is converted to text using the `online2-tcp-nnet3-decode-faster` program of Kaldi that we adapted to our needs.

The GUI, displayed in the web page, runs in a thread that only processes user inputs, displays the recognized speech and passes data from a threads to another, i.e. sends the raw audio from the microphone to the formatting block before forwarding the formatted audio to the ASR block.

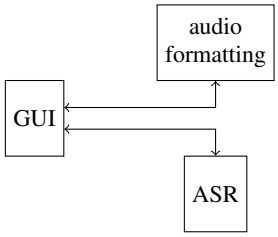

Figure 2: *Program architecture.*

2.2. Technology stack

To run libsamplerate and Kaldi in web browsers, the C/C++ programs are compiled to WebAssembly (WASM) [3] using the emscripten [4] toolchain. WASM is a new language for the web

that executes compiled programs at near-native speed and can even use hardware acceleration.

By working with WASM, the end application can run directly in web browsers. This means that the same source code can run on a wide range of devices. Additionally, in the case of ASR, it is privacy-preserving as the whole processing pipeline takes place on the device. Furthermore, since the application runs in the browser, no installation is required on the device.

In addition to Kaldi and libsamplerate, our package relies on the libraries CLAPACK, CBLAS and BLAS for matrix operations. Since emscripten only compiles from C and C++, we converted the FORTRAN source code of these libraries to C using the program `f2c`, then manually corrected inconsistencies and errors in the resulting code so that the corrected C source code passes all the tests in CBLAS as well as all the matrix operation tests in Kaldi.

The source code of these WASM and Kaldi compatible matrix libraries and the compilation scripts are also available in our repository.

2.3. Application architecture

The application as well as the ASR models are retrieved from the web. The web page is hosted on a server, which can be split in three main components as shown in Fig. 3:

- a web server, which sends the web page to the client;
- a model server, which sends the list of available ASR models to the client as well as each requested model. Note that, to avoid downloading an ASR model every time the page is loaded, the model is stored locally whenever possible;
- a frontend server, which hides the implementation details that the previous two servers represent to the client. The client only sends requests to the frontend server, which forwards the requests/response to/from the relevant server.

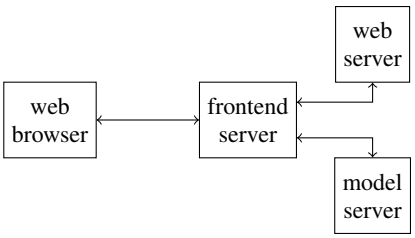

Figure 3: *Application architecture.*

3. Supported ASR models

3.1. Loading custom models in the interface

To use custom ASR models with our GUI, the source code of the package has first to be downloaded and installed. The installation steps are described in the wiki of the package and won't be detailed here.

Once installed, the frontend server, the web server and the model server described in Section 2 can be launched with the command line `npm start` entered at the root of the directory.

As the model server looks for ASR models bundled in a zip archive in the directory `dummy_serv/public`, custom models must be zipped and put in this directory too. These zip archives must have a `.zip` extension and follow a certain format, which is also detailed in the wiki.

3.2. Model specifications

The ASR program can run any Kaldi nnet3 model, whether it is chained or not and using i-vectors or not. The decoding graph must be built and the word-to-symbol map, which is usually named `words.txt`, must be available.

In addition to these requirements, it should be noted that the target device that will run the application will impose further restrictions. For example, in our live demo, the model `english` will run on most recent laptops and desktops but not on mobile phones as it may require more memory than available. In contrast, the model `english_small` can be used on most recent mobile phones. For reference, the model `english_small` contains a 6-layer time-delay neural network, a decoding graph of 105 MB, and close to 170,000 entries in `words.txt`.

4. Live demo

The live demo of the package, located at https://kaldi-web.loria.fr, distributes two models for the English language.

The larger model named `english` is the English ASR model of the Zamia speech project[3]. The smaller model named `english_small` relies on the same acoustic model as the larger one but the decoding graph has been downsized by reducing the size of the grammar and the lexicon.

5. Conclusion

In this demo, an open-source, cross-platform package for client-side ASR is presented. By compiling Kaldi to WASM, we show that it is possible to run an ASR system on the client directly without requiring the end user to compile or install any program. Moreover, web applications being cross-platform by nature, the application provided in our package runs on a wide range of devices. Our package can be used to demonstrate custom Kaldi nnet3 ASR models. The source code being available, it is of course also possible to modify the GUI, integrate it in other applications, or even compile other parts of Kaldi to WASM if needed.

6. Acknowledgements

The authors would like to thank Olivier Rochel for insightful discussions about C program compilation and linking and Philippe Schaeffer for advising them on licensing issues.

7. References

[1] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *ASRU*, 2011.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[3] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. F. Bastien, "Bringing the web up to speed with WebAssembly," in *PLDI*, 2017, pp. 185–200.

[4] A. Zakai, "Emscripten: An LLVM-to-JavaScript compiler," in *OOPSLA*, 2011, pp. 301–312.

[3] https://github.com/gooofy/zamia-speech