ReadAlong Studio Web Interface for Digital Interactive Storytelling

Aidan Pine[1] **David Huggins-Daines**[2] **Eric Joanis**[1] **Patrick Littell**[1] **Marc Tessier**[1]
Delasie Torkornoo[3] **Rebecca Knowles**[1] **Roland Kuhn**[1] **Delaney Lothian**[1]
[1]National Research Council Canada, Ottawa ON, Canada
first.last@nrc-cnrc.gc.ca
[2]Independent Researcher dhd@ecolingui.ca
[3]Algonquian Dictionaries and Language Resources Project, Carleton University
Ottawa ON, Canada delasie.torkornoo@carleton.ca

Abstract

We develop an interactive web-based user interface for performing text–speech alignment and creating digital interactive "read-along" audio books that highlight words as they are spoken and allow users to replay individual words when clicked. We build on an existing Python library for zero-shot multilingual text–speech alignment (Littell et al., 2022), extend it by exposing its functionality through a RESTful API, and rewrite the underlying speech recognition engine to run in the browser. The ReadAlong Studio Web App is open-source, user-friendly, prioritizes privacy and data sovereignty, allows for a variety of standard export formats, and is designed to work for the majority of the world's languages.

Figure 1: A screenshot of a web component read-along published for Atikamekw. Other read-alongs published for Atikamekw can be found at https://atikamekw.atlas-ling.ca/lecture-audio/. Highlighting guides the reader to the word currently being spoken in the recording, and the reader can play single words by clicking on them.

1 Introduction

A "read-along", as seen in Figure 1, is an interactive language tool that highlights words as they are spoken and allows users to replay certain words when clicked (Luchian and Junker, 2004). Language learners are able to interact with these multimodal text/audio documents by repeating the pronunciation of specific words, pausing at a specific place in the document, and following along visually as the text is spoken. This tool promotes reading and listening skills in language learners, which are target skills that are underrepresented in language-learning technology (Shadiev and Yang, 2020).

While the ReadAlong Studio Web App is compatible with many languages (§2.1), it was designed specifically to support learners in an Indigenous language revitalization context, where listening comprehension is often a key priority (Hermes et al., 2012; Lothian et al., 2019). In addition to being more beneficial for comprehension than reading or listening alone (Webb and Chang, 2022), reading while listening can also help promote listening-based skills such as auditory discrimination (i.e., the ability to discriminate between sounds) (Chang, 2009). Furthermore, read-alongs could be used to promote speaking skills by using them in conjunction with speaking tasks, such as shadowing (i.e., reading/speaking along with fluent speech while trying to match pace) (Kadota, 2019).

Building read-alongs, however, can be challenging. Aligning text and speech manually requires a considerable amount of time, and requires some expertise in using audio software. On the other hand, while text-speech alignment can be automated (e.g., Schiel, 1999; Gorman et al., 2011; McAuliffe et al., 2017; Kürzinger et al., 2020), these systems require non-trivial expertise in speech technology and machine learning to train and deploy a model for a new language (MacKenzie and Turton, 2020).

The zero-shot text–speech aligner described in Littell et al. (2022) partially addresses this issue; it can align speech and text in a new language without having seen any prior data in that language. However, it is still a command-line tool that, for full use of its capabilities, requires some familiarity with text and XML formats; it is still not something the average language teacher could use without significant training.

Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023), pages 163–172

Thus, we decided to develop a web-based graphical user interface on top of that system. In the process, we ported a significant amount of the system to JavaScript so that audio processing could happen entirely in the browser. This allows users to create their own read-alongs without requiring them to write code or install anything on their computers, and without sending any audio data to a third party. The software is free and open source, and has a data privacy policy designed to affirm community data sovereignty.

1.1 ReadAlong Studio Web App

The ReadAlong Studio Web App is designed as a two-step process for creating the kind of read-alongs seen in Figure 1. First, the user either writes or uploads some text, records or "uploads" audio (the audio is not actually uploaded to a server, but kept in memory in the browser; see §2.5), and selects the language of their data. The actual text–speech alignment is performed automatically, and the user is taken to step two. Step two presents the read-along to the user in a WYSIWYG[1]-inspired editable mode and lets them add a title and a subtitle, images for each page, and translations for each line, as desired.[2]

There are public resources that instruct users on how to work with the ReadAlong Studio Web App. One such resource is embedded within the ReadAlong Studio Web App as a "tour" that guides users through the steps of creating a read-along (see Figure 3 in Appendix A). We encourage the interested reader to explore the interface themselves,[3] review some of the screenshots available in Appendix A, or the publicly available documentation associated with our recent workshop session at the 8th International Conference on Language Documentation & Conservation in March 2023.[4]

2 Design Decisions

2.1 Language Agnostic

It is well-known that language technologies are not equally available to the world's languages. Reviews of studies on language-learning technologies have found that not only are target languages typically restricted to European and majority languages (Shadiev and Yang, 2020; Burston, 2014), but over half of technologies researched target English, with that percentage increasing within the last decade (Sauro, 2016). While there are active efforts to promote more linguistic diversity among language technologies, sociohistorical and socioeconomic factors are still the most significant determinants of whether the language you speak is supported by the technologies you use. Additionally, for the NRC's Indigenous Languages Technology project, on which the majority of the authors work, our mandate is to support many languages (Kuhn et al., 2020). Thus, our goal was to make a web-interface for creating read-alongs that would be accessible in many languages with as few modifications as possible.

As mentioned in §1, other high-quality text-audio alignment tools exist, but the Littell et al. (2022) aligner best suited our needs since it not only supports zero-shot alignment in 39 (mostly Canadian Indigenous) languages out of the box, but also supports zero-shot alignment of most languages through the use of a rough, language-neutral "fallback" G2P engine (see Littell et al., 2022; Pine et al., 2022, for further details). This tends to work well on languages with relatively transparent orthographies that use characters in cross-linguistically common ways, but will potentially run into trouble in languages that use characters in uncommon ways or have significant orthographic ambiguities; we discuss this in greater detail in the Limitations section. However, in a series of workshops (§3) and in follow-up communication with users, it appears that all the languages users have tried so far have been successful, even those with unique orthographies like Korean and Western Armenian.

Choosing a zero-shot aligner had profound effects on the ReadAlong Studio Web App, since the interface only needs to handle the inference step. Unlike the Elpis tool (Foley et al., 2018) designed for the more challenging task of general speech recognition and transcription, there is no training of a model on user data, and we thus avoid the complication of guiding the user through this process.

The ReadAlong Studio Web App interface itself is also language agnostic in the sense that it has been written using Angular's built-in translation/internationalization library, with the site cur-

[1] What You See Is What You Get

[2] This "editing" mode is also available outside of the ReadAlong Studio Web App in any read-along by changing the "mode" attribute on the custom read-along HTML element from "VIEW" to "EDIT" (see Figure 2 in Appendix A).

[3] https://readalong-studio.mothertongues.org/

[4] https://readalongs.github.io/ICLDC-Docs/, https://github.com/ReadAlongs

rently available in English, French, and Spanish; the code can be adapted to other languages, and further contributions are welcome.

2.2 Portability

While read-alongs can be visualized within the ReadAlong Studio Web App, the purpose of them is to be shared and deployed in a variety of places. To make the interactive read-along user interface as transferable as possible between different web frameworks, we implemented it with StencilJS,[5] a framework for building custom elements using the Web Component open standard API.[6] Further information on how to embed a read-along in any website can be found in Figure 2 in Appendix A.

StencilJS is able to build wrappers around the web components, allowing for greater interoperability with modern web frameworks like Angular, React or Vue.[7] We currently build and publish an Angular wrapper on npm[8] and will publish React and Vue integrations if there is a demand.

ReadAlong Studio also generates a self-contained HTML file that Base64-encodes and embeds all the multimedia content into a single file that can be viewed in any browser, even when Internet access is unavailable. This is an important consideration for rural communities without ubiquitous WiFi and mobile data, where teachers send multimedia content to students via SD cards or USB drives, or where students download content to devices at a central location like a school.

The ReadAlong Studio Web App, for creating read-alongs, is also fairly portable: the software is open source with a permissive license (MIT), its dependencies are all open source, and it can be deployed with minimal resource requirements, albeit with some IT expertise. The front end is a static web page written using Angular, that can be served locally, or at no cost on a service like GitHub Pages. The back end is an API written in Python with FastAPI[9] that can be run locally, or on any cloud server with as little as 512 MB of RAM.

The various options for local, internal network, and cloud deployment, as well as the public deployment we provide, enable communities to choose the solution that best meets their accessibility and privacy requirements.

2.3 Implementation of zero-shot alignment

As in Littell et al. (2022), alignment is done by performing highly constrained finite-state grammar recognition using an English acoustic model and a dictionary generated by roughly mapping the output of zero-shot G2P to the target phoneset. The acoustic model is the same one used in PocketSphinx (Huggins-Daines et al., 2006), and is thus a very old technology optimized for efficiency over accuracy.

The recognizer itself[10] is compiled into WebAssembly using Emscripten[11] and wrapped in a hand-coded JavaScript API. By avoiding the use of C++ in the wrapper and removing functionality irrelevant to the web environment, we obtain a code footprint of 214KB of WebAssembly and 40KB of (minimized) JavaScript. The model is downloaded asynchronously after loading the page, and recognition is done asynchronously in the main browser thread, entirely on the user's computer.

Compared to the original system in Littell et al. (2022), we use a much smaller acoustic model (to limit the download size to 10MB) and also downsample the audio to 8kHz to speed up processing. This typically results in a decrease of 1-3 points in the F1 score of alignments, but we feel that the improved responsiveness and reduced network traffic, along with the privacy and sovereignty considerations detailed in §2.5, make up for what is generally an imperceptible difference.

In future work we plan to further rewrite the aligner to use more modern acoustic modeling and decoding technology, if this can be done while also maintaining or reducing the storage and memory footprint.

2.4 Targeting Open Formats

From the user's perspective, choosing to work with a particular technology comes with risks, including whether you will be compromising your intellectual property or rights to privacy by using the tool (see §2.5) and whether the time you spend using the tool or creating resources within it will be "locked-in" to the platform. We have heard many stories from teachers and curriculum developers of times they have invested hundreds of hours of work creating content in particular sites/products only to find

[5] https://stenciljs.com/docs/introduction
[6] https://www.webcomponents.org/introduction
[7] https://angular.io/, https://react.dev/, https://vuejs.org/
[8] https://www.npmjs.com/package/@readalongs/ngx-web-component
[9] https://fastapi.tiangolo.com/
[10] https://github.com/ReadAlongs/SoundSwallower
[11] https://emscripten.org

that the company goes bankrupt or switches to a different monetization strategy, rendering their content lost, inaccessible, or locked into proprietary file formats.

We do not intend for the ReadAlong Studio project to end unexpectedly, or for any of the core contributors to suddenly become unavailable; however, these outcomes are rarely anticipated for any project. To prevent a situation where the technology becomes unmaintained and users are unable to use the software, we have implemented a variety of features to ensure users' creations can persist beyond the life of this particular project.

The choice of programming languages, format and other software dependencies were carefully considered to give this application a longer than average "shelf life". Firstly, the software is released through a permissive open-source license which will hopefully encourage a diverse community of developers to take part in maintaining the software—with efforts being shared across all users. We expect the default HTML output format to continue to be usable on any JavaScript-enabled HTML5-compatible browser. The raw text and audio *could* be extracted from this, but it would take some technological expertise. To make storage and archiving more accessible and prevent "vendor lock-in", every stage of the pipeline offers downloads to standard formats. Text written directly into the software can be downloaded as .TXT, audio recorded can be downloaded as .MP3, and the resulting alignments can be downloaded as Praat TextGrids,[12] ELAN files,[13] and WebVTT[14] or SRT[15] subtitles; formats which have wide-spread support across many software tools.

2.5 Privacy & Data Sovereignty

Deciding to use a particular technology often comes with consequences for the user's privacy and ownership over their data (Keegan, 2019). Globally, there is a history of theft and misuse of Indigenous language data by academic researchers and external collaborators. In response, Indigenous communities in Canada have created language authorities and data sovereignty principles, such as the First Nations Principles of ownership, control, access, and possession (OCAP®)[16] (First Nations Information Governance Centre, 2023). It is against principles like these for Indigenous language data to be owned or kept in any part by external organizations. Since the motivation for the ReadAlong Studio Web App is primarily to support language education within a language revitalization context, we wanted to develop a tool that affirmed community efforts to remain in control of their data.

In order to adhere to these principles, we prioritized having alignments created locally on the user's machine. Part of our ability to do this stems from the fact that we chose a zero-shot text-audio alignment method, which requires no training data; the only data it requires is the data to be turned into the read-along. As described in §2.3, the second author of this paper also re-implemented the speech recognition engine in WebAssembly and JavaScript so that all alignments happen in the user's browser.

There is still one part of the process that does not occur locally, however: the G2P engine required for the zero-shot method to work is written in Python, so the text for the read-along is uploaded to a remote server for G2P processing. However, the text is not stored on the server after processing, and, as discussed in §2.2, a community could deploy the backend on their own servers if there is a need for greater privacy. Our eventual goal is to implement a version of the G2P engine that will also run in the browser.

We also prioritized user privacy with respect to collecting user analytics. In order to obtain a better understanding of how users are using the site, we have implemented analytics using Plausible Analytics (plausible.io): a privacy-focused analytics solution that does not use cookies or track individuals, but rather presents aggregate data about user operating systems, viewport size, and custom-specified "actions". These actions tell us what percentage of users actually create a read-along once they visit the site, or which output file format they download (§2.4). This information is included in our privacy policy on the ReadAlong Studio Web App; we allow users to opt out from the analytics at their discretion.

3 Discussion & Usage

We held two 90 minute workshop sessions titled "Watch me Speak! Interactive Storytelling using

[12] https://www.fon.hum.uva.nl/praat/
[13] https://archive.mpi.nl/tla/elan
[14] https://www.w3.org/TR/webvtt1/
[15] https://en.wikipedia.org/wiki/SubRip

[16] OCAP® is a registered trademark of the First Nations Information Governance Centre (FNIGC)

ReadAlong Studio" at the 8th International Conference on Language Documentation & Conservation to walk potential users through the ReadAlong Studio Web App in March 2023. The final twenty minutes of each workshop were dedicated to a "language party", inspired by the Aikuma Project's initiative of the same name:[17] participants were invited to create a read-along for their language and share it with the group. Participants created and shared read-alongs in Crow (Siouan), Halkomelem and Nsyilxcən (Salishan), Michif, Gitksan (Tsimshianic), Quechua, Korean, Nuu-Chah-Nulth (Wakashan), Paiwan (Austronesian), Sáliba (Piaroa–Saliban), Takelma, and Western Armenian (Indo-European).

Following the workshops, a participant shared that they had been nervous to create and share their read-along during the language party session of the workshop because they were worried they might not spell things correctly (since they did not have the language-specific keyboard installed), and potentially cause the system to break or not function properly. They were pleasantly surprised when the words of their text became highlighted with their voice. While we built ReadAlong Studio Web App to be *language* agnostic, it is also agnostic to dialect and orthographic variations, which are very common for a variety of reasons in Indigenous language revitalization contexts in Canada (see §5 of Littell et al., 2017). Many tools that are created for a language revitalization context do not offer such generalized support for different dialects or writing systems, potentially systematically excluding certain users. By contrast, the ReadAlong Studio Web App's tolerance for variations in pronunciation and spelling shows potential for fostering a non-judgemental environment for learners to practise speaking their language.

According to Plausible Analytics, of the 525 unique visitors from 23 countries to visit the ReadAlong Studio Web App in the first two months after the launch on February 26, 2023, 122 users created 396 read-alongs, with 65 users going on the tour and 56 users downloading their read-alongs. Among the 56 unique users that downloaded read-alongs, they downloaded read-alongs 174 times in the various available formats. The most popular format (which is also the default) was the offline HTML version. While the total number of people creating read-alongs is still modest, we are encouraged by the amount those users are interacting with the tool and we believe that these statistics demonstrate achievement of our goal for users to be able to create read-alongs for many languages without requiring technological expertise.

[17] https://www.languageparty.org/

4 Conclusion

In this paper, we detailed the motivation for, and design decisions of, web-based software for creating read-alongs titled ReadAlong Studio Web App. As key design considerations, we highlighted support for many languages (§2.1), portability and longevity (§2.2), avoiding vendor lock-in (§2.4), and affirming privacy and data sovereignty concerns (§2.5). Future work involves improving the workflow for correcting and adjusting alignments, increasing language support, and refactoring the G2P engine to JavaScript for complete client-side processing.

Limitations

Accessibility We have tried to develop ReadAlong Studio Web App with accessibility in mind, using accessible colour contrasts, ensuring buttons have aria-labels, and ensuring that the website is legible when zoomed-in to 200%, among other considerations. Using Google PageSpeed Insights, our website scores 89 for Accessibility, but we recognize that there are still improvements to be made; specifically, we would like to perform an audit of the website with respect to Web Content Accessibility Guidelines (WCAG).

Inexact transcription ReadAlong Studio will work best if the transcription is exact; that is, if there are as few discrepancies between the text and audio as possible. If extraneous text exists (such as page numbers, chapter titles, or translations), or if the audio includes un-transcribed speech (such as false starts), these errors will accumulate and can result in poor alignments.

The extent to which these discrepancies affect the final result depends on the length of the recording to be aligned. In practice, we have found that ReadAlong Studio is able to recover from minor transcription errors when the speech data to be aligned are around 5 minutes or less in length. We have successfully aligned much longer (up to 40 minute) files, but "your mileage may vary" depending on the exactness of the transcription, the language's orthography, and the type of data used.

Singing Several teachers have successfully aligned songs with the corresponding text using ReadAlong Studio. For such an alignment to be successful, however, it is necessary that the sung words be vocalized clearly, and not be drowned out by the accompanying music (if any). Extended legato singing (e.g., where one syllable is extended across multiple notes) can also cause poor alignments, since the speech-trained acoustic model does not expect single syllables to correspond to multiple intensity peaks in this way.

Language support The software works with most languages out-of-the-box. As mentioned in §2.1, ReadAlong Studio comes with support for 39 languages built-in, and handles other languages with a rough, best-guess G2P based on Unicode table information. At several international workshops (§3), we found that it worked reasonably well with every language brought by workshop participants, even those with unique alphabets like Western Armenian or Korean.

However, not every language will work equally well. It will typically work well in languages with systematic orthographies that use letters in cross-linguistically common ways. We anticipate difficulty with orthographies that use familiar letters in cross-linguistically unusual ways, such as "font-encodings" (Pine and Turin, 2018), abjads that leave out many vowels, and languages like Japanese where the pronunciation of logographs is highly variable and determined by context. Just like a human could not simply guess the missing vowels in written Hebrew without knowing Hebrew, the software will not be able to do this either.

Additionally, the software is limited to languages which are both written and spoken—we do not support signed languages since the aligner requires audio to align with text, and the tool is fundamentally inapplicable to unwritten languages.

The interface itself is currently only translated in English, French, and Spanish, limiting potential users who do not speak one of those languages.

Numbers and symbols While ReadAlong Studio can do rough zero-shot G2P for most alphabetic and syllabic writing systems, it is not capable of general text normalization—while it can guess that "T" might be pronounced [t] in an unfamiliar language, it simply has no basis to guess any particular pronunciation for "634", as this task is not only language-dependent but highly variable within any given language (Bigi, 2011). Therefore, all input must be "spelled out" for alignment to succeed.

If the input contains numbers or symbols, ReadAlong Studio Web App will prompt the user with a warning that it found uninterpretable symbols.[18]

Ethics Statement

We have addressed a wide variety of ethical concerns throughout the paper, including trying to ensure that the tool supports a diverse audience, does not create technological dependency, and affirms First Nations research principles of ownership, control, access, and possession (OCAP®).

In the preceding Limitations section, we have also tried to be transparent in the ways that our tool might not be adequate for certain users. A final outstanding ethical concern of ours is that our software could potentially be misused to create and distribute content that does not belong to the content creator, causing a variety of potential harms. In the workshop series that we ran, we highlighted this by explicitly warning participants against making read-alongs without first ensuring they had obtained the proper permissions and consent to build and/or distribute content created with audio or text that does not belong to them. While we have not designed a system that is able to ensure this type of misuse will not happen, we are trying to mitigate this risk by explicitly warning against this type of misuse in our public messaging related to the software.

We have attempted to be thorough in considering ethical issues related to our software, but we are aware that our considerations are not comprehensive. We encourage prospective users of the ReadAlong Studio Web App, and indeed of any language technology, to be mindful of this, and to think critically about the possible risks and benefits that come with using any particular tool. We direct interested readers to the excellent "Check Before You Tech"[19] checklist, and welcome any further related questions from current or prospective users.

Acknowledgements

This work would not have been possible without the many collaborators who shared their ex-

[18] This does impact languages like Sḵwx̱wú7mesh sníchim, in which "7" is a valid orthographic character representing a glottal stop. In this case, "7" could be converted to a glottal stop if the mapping existed, otherwise it would be skipped.

[19] https://fpcc.ca/resource/check-before-you-tech/

pertise, precious recordings, and experience using the ReadAlong Studio Web App Interface, including but not limited to the Yukon Native Language Centre, the Kitigan Zibi Cultural Centre, W̱SÁNEĆ School Board, the Pirurvik Centre, Conseil de la Nation Atikamekw, Onwkawenna Kentyohkwa, Owennatekha Brian Maracle, Silver Rae Stevens, Timothy Montler, Marie-Odile Junker, Hilaria Cruz, Nathan Thanyehtenhas Brinklow, Francis Tyers, Fineen Davis, Eddie Antonio Santos, Mica Arseneau, Vasilisa Andriyanets, Christopher Cox, Bradley Ellert, Robbie Jimerson, Shankhalika Srikanth, Sabrina Yu, Jorge Rosés Labrada, Caroline Running Wolf, Michael Running Wolf, Fangyuan (Toby) Huang, Zachery Hindley, Darrel Schreiner, Luyi Xiao, Siqi Chen, Kwok Keung Chung, Koon Kit Kong, He Yang, Yuzhe Shen, Rui Wang, Zirui Wang, Xuehan Yi, and Zhenjie Zhou.

References

Brigitte Bigi. 2011. A multilingual text normalization approach. In *2nd Less-Resourced Languages workshop, 5th Language & Technology Conference*.

Jack Burston. 2014. The reality of MALL: Still on the fringes. *CALICO Journal*, 31(1):103–125.

Anna C-S Chang. 2009. Gains to L2 listeners from reading while listening vs. listening only in comprehending short stories. *System*, 37(4):652–663.

First Nations Information Governance Centre. 2023. The First Nations principles of OCAP®.

Ben Foley, Joshua T Arnold, Rolando Coto-Solano, Gautier Durantin, T Mark Ellison, Daan van Esch, Scott Heath, Frantisek Kratochvil, Zara Maxwell-Smith, David Nash, et al. 2018. Building speech recognition systems for language documentation: The CoEDL endangered language pipeline and inference system (ELPIS). In *SLTU*, pages 205–209.

Kyle Gorman, Jonathan Howell, and Michael Wagner. 2011. Prosodylab-aligner: A tool for forced alignment of laboratory speech. *Canadian Acoustics*, 39(3):192–193.

Mary Hermes, Megan Bang, and Ananda Marin. 2012. Designing Indigenous language revitalization. *Harvard Educational Review*, 82(3):381–402.

David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alexander I Rudnicky. 2006. PocketSphinx: A free, real-time continuous speech recognition system for hand-held devices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

Shuhei Kadota. 2019. *Shadowing as a practice in second language acquisition*. Routledge, New York, NY.

Te Taka Keegan. 2019. Issues with Māori sovereignty over Māori language data.

Roland Kuhn, Fineen Davis, Alain Désilets, Eric Joanis, Anna Kazantseva, Rebecca Knowles, Patrick Littell, Delaney Lothian, Aidan Pine, Caroline Running Wolf, Eddie Santos, Darlene Stewart, Gilles Boulianne, Vishwa Gupta, Brian Maracle Owennatékha, Akwiratékha' Martin, Christopher Cox, Marie-Odile Junker, Olivia Sammons, Delasie Torkornoo, Nathan Thanyehténhas Brinklow, Sara Child, Benoît Farley, David Huggins-Daines, Daisy Rosenblum, and Heather Souter. 2020. The Indigenous Languages Technology project at NRC Canada: An empowerment-oriented approach to developing language software. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5866–5878, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ludwig Kürzinger, Dominik Winkelbauer, Lujun Li, Tobias Watzel, and Gerhard Rigoll. 2020. CTC-segmentation of large corpora for German end-to-end speech recognition. In *Speech and Computer: 22nd International Conference, SPECOM 2020, St. Petersburg, Russia, October 7–9, 2020, Proceedings 22*, pages 267–278. Springer.

Patrick Littell, Eric Joanis, Aidan Pine, Marc Tessier, David Huggins Daines, and Delasie Torkornoo. 2022. ReadAlong Studio: Practical zero-shot text–speech alignment for Indigenous language audiobooks. In *Proceedings of the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages*, pages 23–32, Marseille, France. European Language Resources Association.

Patrick Littell, Aidan Pine, and Henry Davis. 2017. Waldayu and Waldayu Mobile: Modern digital dictionary interfaces for endangered languages. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 141–150, Honolulu. Association for Computational Linguistics.

Delaney Lothian, Gokce Akcayir, and Carrie Demmans Epp. 2019. Accommodating Indigenous people when using technology to learn their ancestral language. *Proceedings of the first International Workshop on Supporting Lifelong Learning co-located with the 20th International Conference on Artificial Intelligence (AIED 2019)*, pages 16–22.

Radu Luchian and Marie-Odile Junker. 2004. Developing an on-line Cree read-along with syllabics. *Carleton University Cognitive Science Technical Report*.

Laurel MacKenzie and Danielle Turton. 2020. Assessing the accuracy of existing forced alignment software on varieties of British English. *Linguistics Vanguard*, 6(s1):20180061.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. 2017. Montreal Forced Aligner: Trainable text-speech alignment using Kaldi. In *Interspeech*, volume 2017, pages 498–502.

Aidan Pine and Mark Turin. 2018. Seeing the Heiltsuk orthography from font encoding through to Unicode: A case study using convertextract. In *Proceedings of the LREC 2018 Workshop CCURL 2018*, pages 27–30.

Aidan Pine, Patrick William Littell, Eric Joanis, David Huggins-Daines, Christopher Cox, Fineen Davis, Eddie Antonio Santos, Shankhalika Srikanth, Delasie Torkornoo, and Sabrina Yu. 2022. G_i2P_i: Rule-based, index-preserving grapheme-to-phoneme transformations. In *Proceedings of the Fifth Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 52–60, Dublin, Ireland. Association for Computational Linguistics.

Shannon Sauro. 2016. Does CALL have an English problem? *Language Learning & Technology*, 20(3):1–8.

F. Schiel. 1999. Automatic phonetic transcription of nonprompted speech. In *Proc. of the ICPhS*, pages 607–610.

Rustam Shadiev and Mengke Yang. 2020. Review of studies on technology-enhanced language learning and teaching. *Sustainability*, 12(2).

Stuart Webb and Anna C-S Chang. 2022. How does mode of input affect the incidental learning of collocations? *Studies in Second Language Acquisition*, 44(1):35–56.

Appendix A: Screenshots

Figure 2 illustrates how simple it is to insert a read-along into a web page. Figure 3 shows the guided tour users can follow to better understand how to use the Studio. Figure 4 expands the drop-down menu for choosing download formats. Figures 5 and 6 are screen captures of the two-step ReadAlong Studio Web App interface.

```html
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <!-- Here is how you declare the Web Component. These files are produced by ReadAlong Studio, the
         paths must point where they are hosted. Multiple ReadAlongs can exist on the same page. -->
    <read-along href='my-file.readalong' audio='my-file.mp3' mode='VIEW' />
  </body>

  <!-- Import the package at the end of your HTML file. The example here is using the unpkg CDN. -->
  <script src='https://unpkg.com/@readalongs/web-component'></script>
</html>
```

Figure 2: Minimal HTML code required to embed a read-along in your website: insert the `read-along` element where the read-along should be displayed, and add the `script` link at the end of the HTML source to load the code required for rendering the read-along.

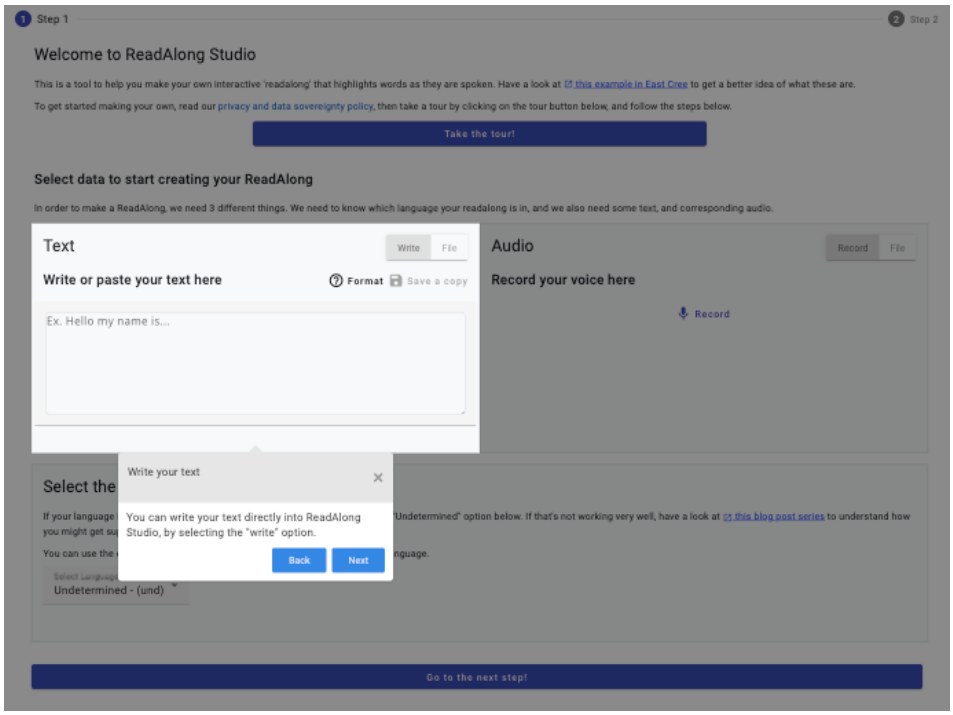

Figure 3: Guided tour in ReadAlong Studio demonstrating how to write text for creating a read-along

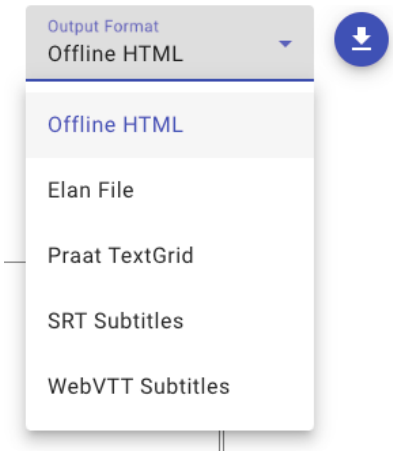

Figure 4: Drop-down menu showing the variety of downloadable output formats in ReadAlong Studio.

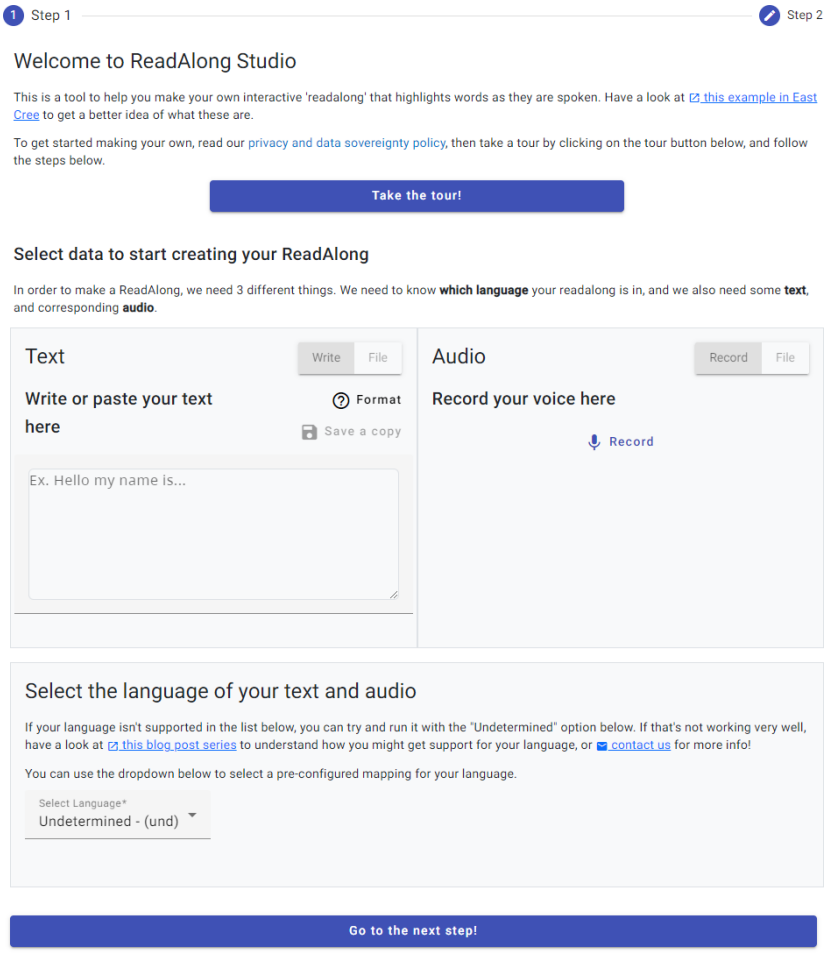

Figure 5: Step 1 of the two-step ReadAlong Studio Web App interface: selecting text and audio

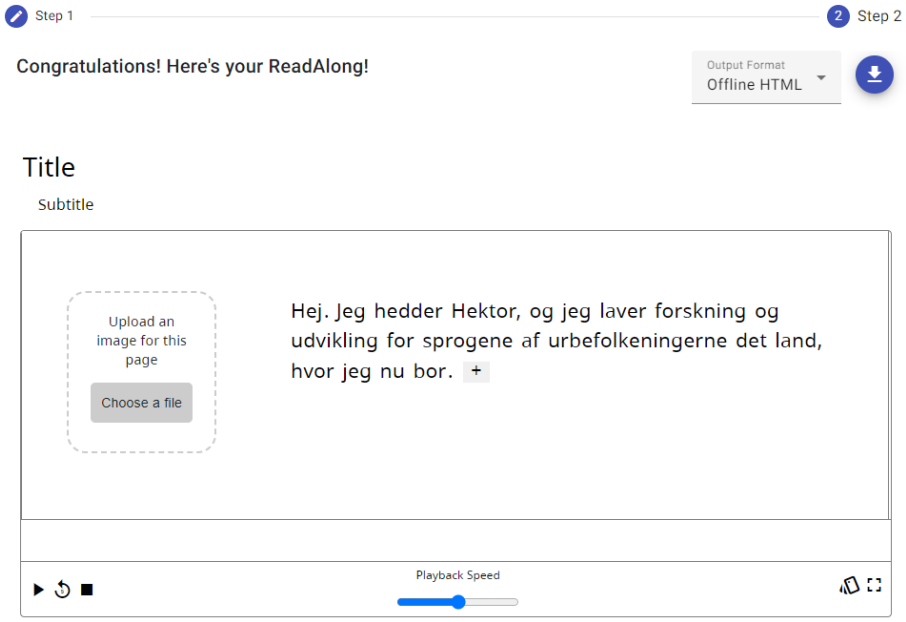

Figure 6: Step 2 of the two-step ReadAlong Studio Web App interface: editing the created read-along