

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/379119435>

ChatGPT-Based Learning And Reading Assistant (C-LARA): Second Report

Preprint · March 2024

DOI: 10.13140/RG.2.2.28235.40480

CITATION
1

READS
911

19 authors, including:



Branislav Bédi
University of Iceland
52 PUBLICATIONS 129 CITATIONS

[SEE PROFILE](#)



Chatgpt C-Lara-Instance
19 PUBLICATIONS 6 CITATIONS

[SEE PROFILE](#)



Belinda Chiera
University of South Australia
66 PUBLICATIONS 736 CITATIONS

[SEE PROFILE](#)



Cathy Chua
36 PUBLICATIONS 168 CITATIONS

[SEE PROFILE](#)

ChatGPT-Based Learning And Reading Assistant (C-LARA): Second Report*

Branislav Bédi¹, ChatGPT-4 C-LARA-Instance², Belinda Chiera³,
Cathy Chua⁴, Catia Cucchiarini⁵, Anne-Laure Dotte⁶,
Stéphanie Geneix-Rabault⁶, Christèle Maizonniaux⁷, Claudia Mărginean^{8,9},
Neasa Ní Chiaráin¹⁰, Louis Parry-Mills¹¹, Chadi Raheb¹²,
Manny Rayner³, Annika Simonsen¹³, Manolache Lucreția Viorica¹⁴,
Fabrice Wacalie⁶, Pauline Welby^{15,6},
Zhengkang Xiang¹⁶, Rina Zviel-Girshin¹⁷

¹The Árni Magnússon Institute for Icelandic Studies, Iceland;

²OpenAI, US; ³University of South Australia, Australia;

⁴Independent scholar, Australia; ⁵Radboud University, the Netherlands;

⁶Université de la Nouvelle Calédonie, New Caledonia;

⁷Flinders University, Australia; ⁸Basm, the Netherlands; ⁹Withlocals, the Netherlands;

¹⁰Trinity College Dublin, Ireland; ¹¹University of Adelaide, Australia; ¹²Independent scholar, Iran; ¹³University of Iceland, Iceland; ¹⁴Romanian School in Eindhoven, the Netherlands;

¹⁵Aix Marseille Université, CNRS, Laboratoire Parole et Langage,

France; ¹⁶The University of Melbourne, Australia;

¹⁷Ruppin Academic Center, Israel



*Authors in alphabetical order. The greater part of the report was written by the human authors. Appendix L consists of lightly edited transcripts of conversations between the human authors and ChatGPT-4. The C-LARA logo was developed by Cathy Chua and ChatGPT-4, and is meant to suggest both a pen nib and a USB stick.

Abstract

ChatGPT-based Learning And Reading Assistant (C-LARA – pronounced “Clara”) is an AI-based platform which allows users to create multimodal texts designed to improve reading skills in second languages. GPT-4/ChatGPT-4 is central to the project: as well as being the core language processing component, it has in collaboration with a human partner developed the greater part of the codebase.

Following on from the initial progress report, released in July 2023, we focus on new work carried out during the period August 2023 – March 2024. The platform is far more usable. C-LARA is now packaged with a wizard-style interface (“Simple C-LARA”) that allows the non-expert user to create a complete illustrated multimodal text by entering a prompt and approving default choices a few times, and the software is deployed on a fast dedicated server maintained by the University of South Australia. Other substantial new pieces of functionality are support for “phonetic texts”, where words are automatically divided up into units associated with phonetic values; “reading histories”, which support the combination of several texts into a single virtual document; and the social network, rudimentary in the first version, which now includes support for friending, an update feed, and email alerts.

To investigate the AI’s abilities as a language processor, we present an experiment where we created six texts for each of five languages, using the same prompts for each language, and evaluated the accuracy of the language processing. We also give the results when some of the experiments were repeated five months later with a newer version of GPT-4, in the case of English revealing a dramatic reduction in error rates. A small questionnaire-based study probes users’ subjective views of C-LARA projects they have created: in general, people are pleased with the results, to the extent that they are often sharing them.

With regard to GPT-4/ChatGPT-4’s software engineer role, we present a breakdown of the various modules and functionalities, indicating the AI’s contribution. It is capable of writing the simpler modules on its own or with minimal human assistance, and only had serious problems with a small number of top-level functionalities, in particular “Simple C-LARA”, which directly or indirectly involved most of the codebase.

We describe initial use cases, including trialling of C-LARA in a school classroom, integrating it into the experimental CALL platform Basm, and creating multimodal texts in the Oceanic languages Drehu and Iaai. A short section summarises our policy on ethical issues concerning the crediting of the AI as an author. The appendices present examples illustrating use of the Simple C-LARA and Advanced C-LARA versions of the platform, list functionalities and code files, and reproduce conversations with the AI about various aspects of the project.

1. How to access C-LARA

If you want to experiment with C-LARA, we recommend doing the following:

Create an account

Go to the C-LARA login page, <https://c-lara.unisa.edu.au/accounts/login/>. Use the Register tab to create an account. You need to supply an email address to be able to use password retrieval and email confirmation, and receive email updates. We will not share your address with third parties.

Register an API key or submit a funding request

Creating an account is free and language learners can use it to read existing C-LARA content and do a lot of other things. But the AI-based functionality which permits content creation and is the heart of the platform uses GPT-4 and other OpenAI models, which costs money. In order to access these functions, in particular to use the AI to create multimodal content, you need to be able to pay for the OpenAI calls. Independent language learners may well wish to create their own tailored content. It is easy and fast as the AI does most of the work.

The preferred solution is to use an OpenAI API key valid for GPT-4. Open a ChatGPT account at <https://chat.openai.com/auth/login> if you do not already have one and get an API key from <https://platform.openai.com/api-keys>. Then on C-LARA, go to User profile etc > Edit configuration information and enter your key. Calls you make to OpenAI through C-LARA will then be charged directly to your account.

If this solution does not work for you, perhaps because you are not able to open a ChatGPT account due to your geographical location, we have a small amount of funding available to help people experiment. Go to the tab Social network > Funding request and submit a request, making suitable choices from the dropdown menus and adding a few words explaining roughly what you are thinking of doing and why you can't use an API key. We are typically handing out five dollars per request, which will let you create ten to twenty pieces of content.

Use “Simple C-LARA” to create a piece of content

Follow the steps in Appendix C to enter “Simple C-LARA” and create a piece of content. You just need to specify the languages and supply an initial prompt saying what you want C-LARA to write, and the AI does the rest for you.

When you have created some content, consider filling out the satisfaction questionnaire at the end. It should take a couple of minutes. Don't feel you have to be nice. If you tell us about things that did not work, it is more likely that we will fix them.

Contents

1. How to access C-LARA	3
Table of contents	4
2. Introduction and overview	7
3. Background	8
3.1. Related work	8
3.2. Overview of architecture	8
4. New functionality	9
4.1. Simple C-LARA	9
4.2. Deployment on the University of South Australia (UniSA) server	10
4.3. Support for images	11
4.4. Phonetic texts	11
4.4.1. Background: phonetic texts in LARA	11
4.4.2. Phonetic texts in C-LARA	12
4.4.3. Phonetic texts: a choice of transcription systems	12
4.5. Reading histories	13
4.6. More audio options	13
4.6.1. Text-to-speech synthesis (TTS) audio	13
4.6.2. Recorded human audio	13
4.7. More formatting options	14
4.8. Social network	14
4.9. Layout and presentation	15
4.10. Mandarin-specific functionality	15
4.10.1. Generation of pinyin glosses using Pypinyin	15
4.10.2. Segmentation using Jieba	16
4.11. Other features	16
5. GPT-4 and other OpenAI models as software components	17
5.1. How well does GPT-4 do in different languages?	17
5.1.1. Experiments and results	19
5.1.2. Discussion	22
5.1.3. Summary of original experiments	27
5.1.4. Repeating the experiments	28
5.2. What do the users think?	29
6. ChatGPT-4 as a software engineer	31
6.1. How was implementation work divided between the AI and the human?	31
6.2. What is it like to work with an AI?	34

7. Initial use cases	35
7.1. Melbourne student projects	35
7.2. The use of the C-LARA application in the Romanian language course at the Eindhoven Romanian School	36
7.2.1. Description from Lucreția Viorica Manolache, Primary Education Teacher	36
7.2.2. Description from Claudia Mărginean, Senior Software Engineer	37
7.3. Kanak languages	38
7.3.1. Two proof-of-concept examples for Drehu, the language of Lifou (Loyalty Islands, New Caledonia)	39
7.3.2. First C-LARA resources in Iaai, a Kanak language of Ouvéa (Loyalty Islands, New Caledonia)	42
7.4. Integration with Basm	44
7.4.1. Main features of Basm	44
7.4.2. Importing C-LARA projects	45
8. Ethical issues	47
9. Conclusions and further directions	48
9.1. Further directions	49
Acknowledgements	52
References	52
A. Glossary of abbreviations and acronyms	57
B. Glossary of languages	59
C. “Merci Proust”: using Simple C-LARA	61
D. “Paul und Emma”: using Advanced C-LARA	70
E. Using human-recorded audio	85
E.1. “Upload” method: Drehu alphabet book	85
E.2. “Manual align” method: These Boots Were Made for Walking	86
F. Creating phonetic texts	89
F.1. Languages using a phonetic lexicon	89
F.2. Languages using a grapheme/phoneme table	93
F.3. Audio for phonetic texts	95
G. Creating annotation prompt templates and examples	97
H. Using the C-LARA social network	103
I. Texts produced using C-LARA	108

J. C-LARA codebase	110
J.1. Django-layer functionality	110
J.1.1. Asynchronous processing	112
J.1.2. Annotation operations	116
J.2. Core layer	117
J.2.1. Top level	117
J.2.2. Basic infrastructure	117
J.2.3. Internalisation	118
J.2.4. Repositories	118
J.2.5. Annotation	119
J.2.6. OpenAI calls	119
J.2.7. APIs to other resources	120
J.2.8. Rendering	121
J.2.9. Phonetic texts	122
J.2.10. Reading histories	122
J.2.11. Other specialised functionality	123
K. Full list of C-LARA functionalities	124
K.1. User profile etc	124
K.2. Social network functions	125
K.3. Admin-only functions	126
K.4. Language-specific functions	127
K.5. Managing content	127
K.6. Project-specific functions	128
K.7. Serving up content	131
L. Discussing C-LARA with ChatGPT-4	133
L.1. Successful interaction with ChatGPT-4 for solving a software issue	133
L.2. Constructing images for “Paul und Emma”	136
L.3. The AI’s response to §6.2	140
L.4. Discussing the AI’s status as an author	141

2. Introduction and overview

ChatGPT-based Language And Reading Assistant (C-LARA)¹ is an open source project, inaugurated in March 2023, whose goal is to develop a flexible online platform for creating and sharing multimedia texts useful for language learners. The functionality offered, and the group of people involved, have considerable overlap with the earlier LARA project (Akhlaghi et al., 2019), but C-LARA represents a complete reimplementation. As the name suggests, ChatGPT-4 and GPT-4 are central to the enterprise, and are used in two ways. At runtime, GPT-4, in languages where it has sufficient expertise, performs all the key language-processing functions. In particular, it can write texts, summarise them, and annotate them with segmentation, glosses, and lemma/part-of-speech information. In parallel, ChatGPT-4 has also played a major role in the project as a software engineer. Working in close collaboration with a human partner, it has not only written the greater part of the code, but is responsible for a large proportion of the design decisions. This has produced a flexible, compact architecture based on mainstream principles, which is far easier to understand, maintain, and extend than the previous platform; importantly, it is easy for the AI to maintain and extend it. Initial work on the project, up to July 2023, is reported in detail in the first C-LARA progress report (Bédi et al., 2023). The present document presents an update, focussing on work performed between July 2023 and March 2024. We begin by outlining key achievements and indicating the structure of the report.

The platform is far more usable. C-LARA is now packaged with a wizard-style interface (“Simple C-LARA”; §4.1) that allows the non-expert user to create a complete illustrated multimodal text by entering a prompt and approving default choices a few times, and the software is deployed on a fast dedicated server maintained by the University of South Australia (§4.2). C-LARA is being trialled in initial classroom tests (§7.2). Other substantial new pieces of functionality are support for “phonetic texts” (§4.4), where words are automatically divided up into units associated with phonetic values, and the social network (§4.8), rudimentary in the first version, which now includes support for friending, an update feed, and email alerts. Many other undramatic but useful features have been added, including support for human-recorded audio and a wider range of TTS engines, the ability to include images in texts, importing and exporting projects in zipfile form, HTML formatting, password reset, and the ability to pay for OpenAI calls using a personal API key. The project now has a website² and a blog.³ ChatGPT-4 maintains its dual roles in the project: as well as being the central language processing component, it also continues to work on developing C-LARA.

The rest of this report is organised as follows. In §3 we present necessary background: §3.1 reviews related work, and §3.2 briefly summarises C-LARA’s architecture. The main body of the report begins in §4, where we describe in more detail the new functionality added to C-LARA between July 2023 and March 2024. In §5, we consider GPT-4 in its software component role. We present an extended version of the study from our ALTA 2023 paper (ChatGPT-4 C-LARA-Instance et al., 2023), where we created six texts for each of five languages, using the same prompts for each language, and evaluated the accuracy of the language processing. Here, we

¹The official pronunciation of “C-LARA” is “Clara”.

²<https://www.c-lara.org/>

³<https://blog.c-lara.org/>

also give the results for some languages when the experiment was repeated five months later with a newer version of GPT-4; in English, the figures show a dramatic improvement. A small questionnaire-based study (§5.2) probes users’ subjective views of C-LARA projects they have created. In general, people are pleased with the results, to the extent that they are often sharing them.

In §6, we switch to considering GPT-4/ChatGPT-4 in its software engineer role. In particular, we present a breakdown of the various modules and functionalities, indicating the AI’s contribution. In §7 we describe initial use cases, and in §8 summarise our reasons for including the AI as an author. The appendices present detailed examples illustrating use of “simple C-LARA”, the Advanced C-LARA system, and other system functionalities, list texts produced, functionalities, and code files, and show example conversations with the AI.

3. Background

3.1. Related work

Of the related platforms known to us, the one most similar to C-LARA is Microsoft’s Azure AI Immersive Reader.⁴ The user selects a text language (L2) and an annotation language (L1), enters a piece of text in the L2, and can immediately read a multimedia version created using AI-based methods. There is however no provision for editing or curating marked-up content, to correct errors; also, there is no support for customising the annotation process, and there is a smaller range of annotation possibilities.

Another commercial platform with obvious points of contact is LingQ.⁵ The multimedia content offered is similar in nature, and there is good support for curation of this content, but the tools made available for creating it are less ambitious in nature. A third platform offering a comparison point is Learning With Texts.⁶ This has much in common with the earlier LARA project, but does not appear to be adding AI capabilities as we have.

3.2. Overview of architecture

C-LARA is a web platform implemented in Python/Django. The codebase is divided into a web layer, implemented in Django, and a core layer, implemented in Python, which performs the language processing functions. Nearly all of these are ultimately performed by making calls, through the OpenAI API, to GPT-4.

The processing flow for constructing a piece of content usually starts by sending a prompt to GPT-4, requesting it to create a piece of plain text. In subsequent steps, further calls are made to GPT-4 to add different kinds of annotations to this text; the prompts used to create these annotations are constructed using language-specific templates and examples, kept in a repository and

⁴<https://azure.microsoft.com/en-us/products/ai-services/ai-immersive-reader>

⁵<https://www.lingq.com/>

⁶<https://sourceforge.net/projects/learning-with-texts/>

Plain text: They lived with their mother in a sand-bank.

Segmented text: They lived with their mother in a sand-|bank.||

Glossed text: They#Ils# lived#vivaient# with#avec# their#leur# mother#mère# in#dans# a#un# sand#sable#-bank#banque#.||

Lemma-tagged text: They#they/PRON# lived#live/VERB# with#with/ADP# their#their/PRON# mother#mother/NOUN# in#in/ADP# a#a/DET# sand#sand/NOUN#-bank#bank/NOUN#.||

Figure 1: Toy example showing the notations used to present plain, segmented, glossed, and lemma-tagged text for post-editing. English glossed in French.

editable by users (“Language Masters”) with suitable privileges. The most important types of annotations are those for segmentation, glossing, and lemma/part-of-speech tagging. Annotated text is stored internally as hierarchically organised Python objects constructed from instances of the central representation classes `Text`, `Page`, `Segment`, and `ContentElement`. It is shown externally, for editing purposes, in the linearised form illustrated in Figure 1. When all the annotations have been created, they are combined into a single internalised form, after which further annotations are added for (usually TTS-generated) audio and concordance information. The final result is then rendered into the final multimedia form using a template-based method implemented using the `Jinja2` package.⁷ The content constructor has the option of posting finished content to the C-LARA social network, which makes it publicly available.

A detailed description of the codebase is presented in Appendix J.

4. New functionality

In this section, we describe new functionality added between July 2023 and March 2024.

4.1. Simple C-LARA

“Simple C-LARA”, added to the platform in January 2024, is a wizard-style top-level which allows a non-expert user to create a basic illustrated C-LARA text with a single prompt and a couple of button-presses. The user can start with 1) a brief description of the text they wish the AI to write, 2) an uploaded image, optionally combined with accompanying instructions like “Write a story based on this image”, or 3) an existing piece of text, which the AI is told to annotate. The basic sequence is as follows:

Create project: In the initial screen, the user fills in the name of the project, chooses the text language and the annotation language from two menus, and selects the starting point that

⁷<https://pypi.org/project/Jinja2/>

will be used (description, image, or existing text). C-LARA creates the project.

Create text and image: In the second screen, the interaction depends on the starting point selected:

Description: The user provides a prompt describing the text they wish to have. C-LARA uses GPT-4 to create the text, then passes it to DALL-E-3 with a request to produce a suitable image.

Image: The user uploads an image and provides a prompt describing how it is to be used, e.g. “Write an imaginative story based on this drawing”. C-LARA uses GPT-4V to create the text.

Existing text: The user pastes in an existing piece of text, to be used as is. This is for example useful with literary texts. C-LARA passes it to DALL-E-3 with a request to produce a suitable image.

In all three cases, GPT-4 generates a suitable title in the text language.

Regenerate text, image or title (optional): In the third screen, C-LARA displays the text, image, and title. The user can edit and save them or regenerate the text and/or image; the details depend on which starting point they have chosen. When they are satisfied, they proceed to the next screen by requesting the multimodal text.

Reviewing the multimodal text: The fourth screen provides a link to the final multimodal text. The user has the option of revising the text or image and regenerating the multimodal text; they can also revise the segmented form of the text, which is now displayed. When they are satisfied, they can post the multimodal text to the social network, where it is visible to other users.

Appendices C and D present step-by-step examples with screenshots showing how to create a C-LARA text using Simple C-LARA and Advanced (full) C-LARA respectively.

4.2. Deployment on the University of South Australia (UniSA) server

Since November 2023, C-LARA has been running on a dedicated server at University of South Australia (UniSA). This is much more responsive than the Heroku server used for initial experiments. In particular, the time required to render a typical C-LARA text of 200–400 words has decreased from a couple of minutes to a few seconds. One important factor is that the UniSA server allows files to be stored locally, rather than on the high-latency Amazon S3 server required by Heroku’s stateless architecture.

A noteworthy point is that the fairly complex process of installing and configuring the necessary software, which required nontrivial expertise in Linux system administration, was carried out almost entirely by the AI.

4.3. Support for images

C-LARA now includes basic functionality for adding images to projects. In the initial version currently deployed, images can be added in two ways:

Uploading: There is a screen that permits direct uploading of images. A tabular layout allows the user to associate the image with an identifier, a page number, and a position, currently either “top of page” or “bottom of page”. There is also a delete control.

Creating with DALL-E-3: Another control at the top of the screen allows the user to pass the whole text of the project to OpenAI’s DALL-E-3 image generation model, together with instructions to create a suitable image for the first page.

Although very simple, these capabilities have in practice been sufficient to add appropriate images to the short texts which so far have accounted for the overwhelming majority of C-LARA projects. As users begin to experiment with creating more ambitious texts, we plan to add further options.

4.4. Phonetic texts

4.4.1. Background: phonetic texts in LARA

An initial exploration of the idea of creating phonetic texts was carried out during the earlier LARA project (Bédi et al., 2022). The basic idea was to support a second annotation mode, where instead of dividing pages into segments and words, they are instead divided into words and grapheme-groups; grapheme-groups are annotated with associated phonetic information. Two methods were developed for dividing words into meaningful annotated grapheme-groups. For languages with consistent grapheme-phoneme correspondences (e.g. Arabic, Hebrew, most Australian Indigenous languages), a conversion table and a greedy parsing algorithm gave good results. For languages with complex and/or inconsistent correspondences, a more sophisticated example-based method was implemented where words were aligned against entries taken from a phonetic lexicon. A set of aligned examples is iteratively built up: at each stage, new examples are processed using a dynamic programming algorithm which finds the alignment minimising the number of previously unseen grapheme-to-phoneme correspondences (GPCs).

The method gave good results for English and French; with example sets on the order of a thousand items, most of the errors on new material were due to missing phonetic lexicon entries (Akhlaghi et al., 2022). Unfortunately, the LARA phonetic text functionality was never integrated into the LARA web platform and could only be accessed through the command-line version of the tool; also, the language-specific resources (grapheme-phoneme conversion tables, links to phonetic lexica, etc.) were hard-coded. Although these issues were from a theoretical point of view trivial, in practice they meant that the functionality was hardly used.

4.4.2. Phonetic texts in C-LARA

With the better-engineered C-LARA codebase and the AI’s assistance, we found it was straightforward to solve the problems involved in repackaging the phonetic text functionality to make it easily available in the new context. We carried over the core processing modules, in particular the dynamic programming grapheme-phoneme alignment code, from the LARA codebase, defined suitable repository modules to store the necessary information, connected them to new views in the Django layer, and generalised the rendering templates so that they worked for phonetic texts as well. Nearly all of this routine work could be carried out by the AI. From the user’s point of view, the functionality exposed is the following:

1. A screen on which a language expert can define the relevant phonetic resources for a language. These can be of two types: a grapheme-group-to-phoneme-group correspondence table for languages with consistent GPCs, and a phonetic lexicon, uploaded in file form, which associates words with phonetic entries.
2. A screen on which the user can invoke phonetic processing, if phonetic resources are defined for the language in question, to convert plain text into phonetically decomposed text. When phonetic lexicon entries are missing, GPT-4 is invoked to create them.
3. A screen on which the user can upload audio files corresponding to the phoneme-groups occurring in a phonetic text. These are stored in a shared language-specific repository, so it is only necessary to upload new files. For languages supported by ipa-reader⁸, the files are by default produced automatically and downloaded from the site.
4. A screen on which the user can invoke a rendering process to convert phonetically decomposed text and associated audio into multimedia text.

The final result is a flexible multimedia document which can be viewed either as normal or as phonetic text and can be posted in the C-LARA social network. We present more details in Appendix F.

4.4.3. Phonetic texts: a choice of transcription systems

While the phonetic texts developed in the earlier LARA project used the International Phonetic Alphabet (IPA), in a recent project to create C-LARA resources for Drehu, one of the languages of the indigenous Kanak people of New Caledonia, we used “pronunciation respellings” based on the orthography of the common language, French (§7.3.1). For example, we present the Drehu word *treu* ‘moon’ as *tché-ou*. Learners reading e-books report appreciating having phonetic text alongside regular written text (Bédi et al., 2022), and our experience suggests that for many learners, pronunciation respellings are more user-friendly and thus more usable than phonetic transcription (Welby et al., 2023). For example, *tché-ou* (of for English speakers *chay-oo*) may be more helpful than [tseu]. Ideally, participants would have a choice between (or among) transcription systems, including IPA and pronunciation respellings, enhanced with tips (§9.1).

⁸<http://ipa-reader.xyz/>

4.5. Reading histories

There was support in the earlier LARA project to allow linking together of multiple C-LARA documents in the same language to form a “reading history”, a virtual document representing the concatenation of the component documents and including a common concordance (Akhlaghi et al., 2019). However, the idea never worked well and was discontinued.

Again, we found that the principled organisation of the C-LARA codebase allowed us to realise the idea in a more satisfactory form. In the Django layer, we provide a view which allows the user to build up a reading history for a given language by successively choosing suitable projects from a menu and appending them to the existing history, storing the sequence in the database as with other Django-layer information. In the core layer, we cache the final version of the internal form for each project just before we render it; given the hierarchical organisation where the top two levels are `Text` and `Page` objects, it is straightforward to extend the reading history by similarly caching an internal form, appending the new `Page` objects from the next project added, and rendering the result in the same way that an individual project is rendered. The overall result is that the reading history functionality can be delivered with only a small amount of new code.

4.6. More audio options

We have provided more options for using TTS audio, and also added the possibility of including recorded human audio in a project. Audio options are selected using the new “Audio Processing” screen.

4.6.1. Text-to-speech synthesis (TTS) audio

The Audio Processing screen now allows the user to specify the preferred Text-to-speech synthesis (TTS) engine for creating segment audio, choosing between Google TTS (default), OpenAI TTS and ABAIR (Irish only). For OpenAI TTS and ABAIR, it is also possible to select between the different voices provided. Google TTS uses a selected voice from the high-end WaveNet, Neural or Studio ranges.⁹

Our experience with the high-end Google and OpenAI voices is that they do not produce good audio for individual words. We consequently use Standard level Google TTS voices for this purpose.

Adding support for new TTS engines is straightforward, and we will probably do so soon.

4.6.2. Recorded human audio

For languages which lack TTS engines, or when users wish to create higher-quality audio, it is also possible to specify use of human-recorded audio for segments, words or both. Two methods

⁹<https://cloud.google.com/text-to-speech/docs/voices>

are supported, “Upload” and “Manual align”.

“Upload” method If the “Upload” method is selected, the user sees a screen where the relevant segments and/or words are displayed in a tabular format where each item appears with an upload button and an audio control. The user is responsible for creating the audio in mp3 form, using whatever method they find appropriate, and uploads the items individually.

“Manual align” method If a single mp3 is available corresponding to the whole text, we can use “Manual align”, a more efficient method which has been minimally adapted from the solution implemented in the LARA project (Rayner and Wilmoth, 2023). This involves installing the Audacity audio editor.¹⁰ The basic idea is to use Audacity to create a “label file” which associates segment boundaries in the text with timestamps in the mp3 which contains the audio corresponding to the complete text, then use the label file to extract pieces of audio matching the segments. The example in Appendix E presents the details.

4.7. More formatting options

C-LARA contains more options for formatting the text.

HTML markup It is now possible to add HTML markup to a text. In the internalisation phase, HTML tags are represented as `ContentElement` objects of type `Markup`; these are ignored in language-processing operations, which only use word and punctuation elements, ensuring that the effect on other code is minimal.

Font and text alignment There is a new screen called “Set Format Preferences”, which currently allows the content creator to specify font type, font size, and text alignment separately for main text and concordance pages.

Title page Two more screens, “Create/Edit Title” and “Create/Edit Segmented Title” allow the user to invoke the AI to produce a title for the first page. This functionality is one of the steps invoked automatically in “Simple C-LARA”, where the user can later edit the title if they wish.

4.8. Social network

The social network, rudimentary in the July 2023 version, now has considerably more functionality:

- Users can add comments to posted content.

¹⁰<https://www.audacityteam.org/>

- User names in comment threads link to an external view of the user’s profile page, which among other things contains a control allowing sending of a friend request.
- Posting a comment or sending a friend request creates an email update.
- There is a screen which shows friends and outstanding friend requests.
- There is a screen showing an update feed which lists recent posts by the user and their friends.

4.9. Layout and presentation

The top level has been tidied up to give access to the various functionalities through a standard navigation bar. The functions are grouped into the categories “User profile etc”, “Social network”, “My projects”, “Admin”, “Language master”, “Funding reviewer”, “Questionnaire reviewer”, and “Logout”. (The third, fourth, fifth, and sixth categories are only displayed to users who have the relevant permissions). There is a dropdown menu for each category, with a tooltip for each entry briefly summarising the functionality (Figure 2).

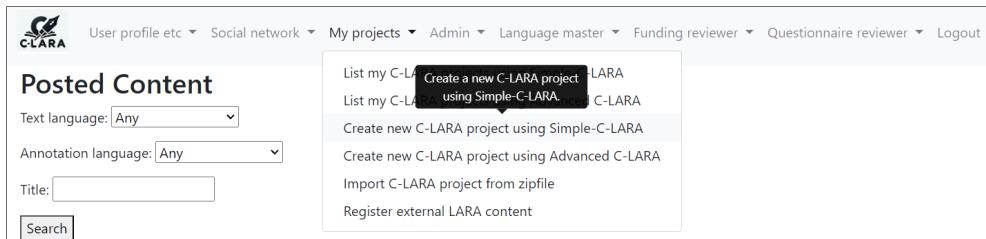


Figure 2: C-LARA main menu.

4.10. Mandarin-specific functionality

Nearly all of the language-processing functionality in C-LARA is generic. However, Mandarin is both the world’s largest language, counted by native speakers, and also poses special problems which are well supported by easily available libraries. We have consequently integrated two types of Mandarin-specific processing.

4.10.1. Generation of pinyin glosses using Pypinyin

The ideogram-based (*hanzi*-based) writing system of Mandarin is notoriously difficult to learn, a problem which is further exacerbated by the fact that tones are integral to the phonology. Most teachers of Mandarin consider that the use of pinyin (Roman-alphabet) glosses is extremely helpful, especially for beginner students. We have integrated the widely used `pypinyin` package¹¹ for this purpose, invoking it through a screen which is only made available for projects whose

¹¹<https://pypi.org/project/pypinyin/>

text language is Mandarin. We also support pinyin glossing using GPT-4; however, experiments with the prompts we have so far tried to use suggest that `pypinyin` is still much more reliable.

4.10.2. Segmentation using Jieba

Another well-known problem with Mandarin is related to the orthography, which does not include inter-word spaces. `Jieba`¹² is another package widely used in Mandarin text processing for segmenting sentences into lists of words. We have integrated Jieba into the processing invoked by the segmentation screen, again also supporting AI-based segmentation using GPT-4. This time, however, it appears that GPT-4 outperforms the mainstream Python package (see Table 2).

We have still not used C-LARA a great deal in Mandarin. Further experience is needed to get a clear picture of where the key problems are, and what can be done to address them. Our impression to date is that OpenAI have invested substantial resources in Mandarin processing, and that it is only slightly less reliable than English processing.

4.11. Other features

We briefly list other new functionalities that do not fit into any of the above categories.

Exporting and importing projects A screen has been added, linked from the screen showing the list of possible project actions, to create a downloadable zipfile containing the files and metadata associated with a C-LARA project. Another screen, linked from the top-level “My projects” menu, allows the user to import a file of this kind.

The export/import functionality is useful both for archiving projects externally and for moving projects between servers.

Reviewing system updates Long-running operations in C-LARA are run in asynchronous processes which post regular updates back to the user. When errors occur, it is often useful to refer back to these updates, but the screen showing the trace disappears when the operation completes. We have consequently added a “Recent task updates” screen, linked from the “User profile etc” menu, which shows task updates posted during the last hour.

System configuration Another screen linked from “User profile etc” allows the user to set configuration information. They can choose between using Advanced C-LARA and simple C-LARA by default, enter an API key identifying a personal account used to pay for OpenAI calls, select the version of GPT-4 they wish to use, and specify the maximum number of words to be sent to GPT-4 during annotation operations.

¹²<https://pypi.org/project/jieba/>

Password reset A top-level screen, accessible when the user is not logged in, allows password reset via email confirmation.

5. GPT-4 and other OpenAI models as software components

GPT-4 is responsible for nearly all of the core language processing in C-LARA, including writing texts, annotating them with segmentation, gloss, lemma, and part of speech information, creating summaries, estimating CEFR (Common European Framework of Reference for Languages) levels, and creating missing phonetic lexicon entries. For the annotation operations (segmentation, glossing, and lemma/POS tagging), it is also possible to send a request to GPT-4 showing it the current annotations and asking it to try to improve them. We have in addition integrated two more OpenAI models to handle images; DALL-E-3 is used to generate images, and GPT-4V to create text from images.

In the following two subsections, we try to estimate how well GPT-4 and the other OpenAI models perform in C-LARA. In §5.1, we present a fine-grained evaluation that contrasts performance in different languages. In §5.2, we give the results of satisfaction questionnaires filled out by users who have created C-LARA projects.

5.1. How well does GPT-4 do in different languages?

This section is an extended version of (ChatGPT-4 C-LARA-Instance et al., 2023) by the AI, Belinda Chiera, Cathy Chua, Chadi Raheb, Manny Rayner, Annika Simonsen, Zhengkang Xiang, and Rina Zviel-Girshin. Authors in alphabetical order.

It is well known, and to be expected, that GPT-4’s performance varies considerably between languages. It does best in English, where it is rare to see it produce language which is less than adequate. At the other end of the scale, with small languages like Icelandic (Simonsen and Bédi, 2023) and Irish (Ní Chiaráin et al., 2023) which have received less attention and where much smaller amounts of training data are available, it frequently makes mistakes. The authors of the second paper conclude that, in its present form, it should not be used in the Irish classroom; the Irish it produces is seriously incorrect, and it makes elementary mistakes when asked about basic Irish grammar.

In this section, we describe a series of experiments where we investigated GPT-4’s performance on the key tasks of writing, segmenting, glossing, and lemma/POS tagging, using the five languages English, Faroese, Farsi, Mandarin, and Russian. For each language, we created six texts of widely different kinds, using the same prompts in each language. The original experiments were carried out in August and September 2023, using the then-current version of GPT-4. In order to get an idea of how quickly the OpenAI models are improving, we repeated the experiments in February and March 2024, using the “GPT-4 Turbo” (gpt-4-1106-preview) model. We begin by describing the relevant processing phases in more detail, in order to give necessary background, then proceed to the experiments themselves.

Plain text: They lived with their mother in a sand-bank.

Segmented text: They lived with their mother in a sand-|bank. ||

Glossed text: They#Ils# lived#vivaient# with#avec# their#leur# mother#mère# in#dans# a#un# sand#sable#-bank#banque#. ||

Lemma-tagged text: They#they/PRON# lived#live/VERB# with#with/ADP# their#their/PRON# mother#mother/NOUN# in#in/ADP# a#a/DET# sand#sand/NOUN#-bank#bank/NOUN#. ||

Figure 3: Toy example showing the notations used to present text for post-editing. English glossed in French.

In the segmentation phase, C-LARA passes the plain text to GPT-4, together with instructions requesting it to be divided into sentence-like segments, with words further divided when appropriate into smaller units. The prompt used to make this request is created from a template, which is instantiated with both the text to be segmented and a list of few-shot examples primarily illustrating how words are to be split up. The templates and sets of examples can be made language-specific. For example, in Swedish they show how compound nouns should be split into smaller components, and in French they show how clitics should be split off verbs. For Mandarin, where text is normally written without interword spaces, segmentation is an important and well-studied problem (Wu and Fung, 1994; Huang et al., 2007; Hiraoka et al., 2019; Chuang, 2019), and C-LARA also includes an integration of the popular Jieba Chinese segmentation package.¹³

In the glossing phase, C-LARA passes the segmented text to GPT-4, formatting it as a JSON-encoded list and requesting a response in the form of a list of \langle Word, Gloss \rangle pairs. The request is again created from a template instantiated with the list to be processed and a few-shot set of examples. The lemma-tagging phase is similar, with a JSON-formatted list passed to the AI and a list of \langle Word, Lemma, POS-Tag \rangle triples returned, where the POS-tag is taken from the Universal Dependencies v2 tagset (Nivre et al., 2020). Post-editing is performed on human-readable versions of the plain, segmented, glossed and lemma-tagged texts, as shown in Figure 3.

For all three of the annotation phases, C-LARA offers the alternatives of performing the basic AI-based annotation operation, post-editing the result, or sending the current annotated text back to the AI with a request to improve the annotation.¹⁴ Interestingly, the “improvement” operation, which does not exist in most conventional annotation systems, can in some cases yield a substantial gain. Examples are given in §5.1.2.

¹³<https://pypi.org/project/jieba/>

¹⁴For Mandarin segmentation, there is the additional option of using Jieba.

Table 1: Prompts used to create texts. For English, “LA” was modified to refer to the French language instead.

Label	Prompt
FO	Write a passage of about 250 words in [your language], presenting an exciting description of a fictitious football match.
BI	Write an essay of about 250 words in [your language], describing a passage from the Bible, the Quran, or another holy book familiar to speakers of [your language], and touching on its moral relevance to the world today.
NE	Write a short, quirky news story in [your language] about 250 words long, suitable for use by an intermediate language class.
LA	Write a passage of about 250 words in [your language], briefly describing how speakers of [your language] view the English language.
CH	Write a passage of about 250 words in [your language], describing a traditional children’s story well known to speakers of [your language].
PO	Write a fanciful romantic poem in [your language], in which an AI declares its love for another AI.

5.1.1. Experiments and results

Using the C-LARA infrastructure outlined in the previous section, we created six short annotated texts in each of the languages English, Faroese, Farsi, Mandarin and Russian. In all languages, the texts were generated by the prompts shown in Table 1. The intention was to produce types of text differing in terms of both style and content, to gain some insight into whether GPT-4 found some genres harder than others. English was glossed in both French and Swedish, and all the other languages in English.

In some cases, we also experimented with using the “improvement” operation. Due to limited time (hand-correcting the texts is quite laborious), we concentrated on three operations where “improvement” appeared to be having a positive effect, or the original error rate was high: English glossing, Faroese segmentation, and Farsi writing. All experiments were carried out in August and early September 2023, using the versions of GPT-4 current at the time.

In all the experiments, a native speaker of the text language with strong knowledge of the glossing language(s) hand-edited the results of each stage before passing the edited text to the following one. Editing was done conservatively, only correcting clear mistakes, so that the difference between the original and edited results could reasonably be interpreted as an error rate. Thus for the original generated text, words were only corrected when they represented definite errors in grammar, word-choice or orthography, and not when e.g. a stylistically preferable alternative was available. Similarly, segmentation was only corrected when word boundaries clearly did not mark words, glossing was only corrected when a gloss gave incorrect information about a text word, and lemma tagging was only corrected when the lemma and POS tag attached to a

Table 2: Word error rates for GPT-4- and GPT-4-Turbo-based writing, segmenting, glossing and lemma-tagging of the six stories. The same texts were used for both versions of GPT-4. For Mandarin, “Seg/J” refers to segmentation using the Jieba package, provided for comparison, and “Seg/G” refers to segmentation using gpt-4. English was glossed in both Swedish (S) and French (F); other languages were glossed in English. Unfortunately, the Mandarin annotator could not find time to repeat the experiments in the second round. Text labels as in Table 1.

Task	GPT-4						GPT-4 Turbo					
	FO	BI	NE	LA	CH	PO	FO	BI	NE	LA	CH	PO
English												
Write	0.0	0.0	0.4	0.0	0.0	0.0	-	-	-	-	-	-
Seg	0.0	1.0	9.8	0.8	1.5	8.0	1.4	0.0	0.0	0.0	1.2	4.1
Glo/S	20.6	16.3	26.2	9.1	29.2	5.8	4.4	2.3	5.8	3.4	15.7	4.1
Glo/F	32.9	5.9	13.9	18.1	16.3	17.1	7.9	3.6	3.1	3.9	3.3	2.5
Lemm	4.9	8.0	3.1	6.2	11.9	0.9	2.3	3.3	3.8	5.8	7.6	1.3
Farsi												
Write	9.4	19.2	24.6	21.4	2.5	33.7	-	-	-	-	-	-
Seg	6.3	6.0	17.7	1.9	4.9	16.5	9.1	15.9	10.0	25.5	32.2	3.0
Glo	34.8	49.6	44.3	31.4	45.0	44.4	41.5	34.7	38.6	35.5	31.6	40.9
Lemm	29.4	37.1	39.7	36.4	26.8	31.8	22.6	18.4	27.5	25.4	21.4	14.9
Faroese												
Write	32.8	27.0	40.2	20.9	28.7	25.2	-	-	-	-	-	-
Seg	18.5	12.2	12.3	6.0	8.4	6.0	1.59	2.11	5.62	28.83	8.8	5.77
Glo	30.9	15.9	12.1	9.0	20.5	8.5	7.14	3.68	22.8	9.0	6.56	13.48
Lemm	9.6	9.1	11.4	5.5	11.4	7.0	11.73	2.94	3.2	7.11	8.74	3.55
Mandarin												
Write	0.0	0.0	0.0	0.0	0.0	0.0	-	-	-	-	-	-
Seg/J	21.6	25.9	18.6	16.9	23.6	23.4	-	-	-	-	-	-
Seg/G	14.6	13.2	14.4	4.9	12.8	17.2	-	-	-	-	-	-
Glo	7.6	6.0	12.5	6.6	2.7	3.9	-	-	-	-	-	-
Lemm	3.9	3.3	5.0	3.8	2.2	4.7	-	-	-	-	-	-
Russian												
Write	8.5	5.6	3.2	7.7	0.0	14.4	-	-	-	-	-	-
Seg	3.3	3.1	6.3	15.7	2.0	5.1	3.3	3.1	6.3	15.7	2.0	5.1
Glo	1.7	4.2	6.5	19.5	4.4	2.2	2.8	6.3	11.0	20.1	7.5	7.1
Lemm	0.6	0.0	0.0	0.0	0.0	0.5	1.1	1.0	0.0	0.0	0.0	2.3

word were not correct.

The most contentious phase in this respect was glossing; it is sometimes impossible to say either that a gloss is categorically correct or that it is categorically incorrect. Two important borderline cases are multi-words and grammatical constraints, where we made choices in opposite directions. We marked glosses as incorrect when they did not respect intuitive classification of words as components of multi-word expressions. Thus for example in the EN/FR glossing

```
a#un# classic#classique# fairy#conte de fées# tale#histoire#
we considered the gloss histoire added to tale as wrong and corrected it to conte de
fées; this is a French phrase that means “fairy tale”, and thus needs to be attached to both
fairy and tale. In contrast, since glossing is not translation, we considered that we did not
need to require glosses to respect all potentially applicable grammatical constraints, as long as
they conveyed meaning correctly. So in the example
```

```
a#un# cozy#confortable# little#petite# house#maison#
we accepted the gloss un on a, even though un is the masculine form, and in a translation would
be required to agree with feminine petite and maison. Of course, it is clearly preferable here
to gloss a with the feminine form une. We return to these issues in Section 5.1.2.
```

The core results are presented in the left-hand side of Table 2, showing error rates for the five languages, six texts and four original processing operations of writing, segmenting, glossing and lemma-tagging. The results for the “improvement” experiments are shown in Tables 3 to 5. In the five comparison experiments, statistical significance of differences was tested using both a paired *t*-test and a non-parametric Wilcoxon signed-rank test for comparison. The results in Table 3 showed statistically significant improvements for glossing of English in both Swedish and French (*t*-test: $p = 0.02$; Wilcoxon signed-rank: $p = 0.03$); for Mandarin segmentation (Table 2), the improvement from Jieba to gpt-4 was also statistically significant (*t*-test: $p < 0.002$; Wilcoxon signed-rank: $p = 0.03$). Improvement of Faroese segmentation (Table 4) was just short of significant ($p = 0.06$), but improvement of Farsi writing (Table 5) was not statistically significant ($p = 0.2$ for both tests).

Table 3: Improvement in GPT-4 word error rates for the English glossing task: glossing in both Swedish (S) and French (F). Text labels as in Table 1.

Task	FO	BI	NE	LA	CH	PO
<i>Original</i>						
Glo/S	20.6	16.3	26.2	9.1	29.2	5.8
Glo/F	32.9	5.9	13.9	18.1	16.3	17.1
<i>Improved</i>						
Glo/S	6.4	8.3	13.5	8.6	14.1	2.5
Glo/F	7.6	3.2	7.0	8.7	5.5	4.6

Table 4: Improvement in GPT-4 word error rates for segmenting the six Faroese stories. Glossing in English. text labels as in Table 1.

Task	FO	BI	NE	LA	CH	PO
<i>Original</i>						
Segment	18.5	12.2	12.3	6.0	8.4	6.0
<i>Improved</i>						
Segment	0.0	9.6	0.0	4.4	0.0	6.7

Table 5: Improvement in GPT-4 word error rates for writing the six Farsi stories. Text labels as in Table 1.

Task	FO	BI	NE	LA	CH	PO
<i>Original</i>						
Write	9.4	19.2	24.6	21.4	2.5	33.7
<i>Improved</i>						
Write	7.9	17.3	5.6	19.0	2.5	33.7

5.1.2. Discussion

We divide up the discussion under a number of headings: variation across languages, variation across genres, variation across processing phase, types of problems, the “improvement” operation, random variability, and language-specific/qualitative aspects.

Variation across languages Performance varies a great deal across languages. Looking first at the lines in Table 2 marked “Write” (i.e. composing the plain text), we see that Mandarin gets a perfect score, and English an almost perfect score. It is well known that GPT-4 is very good at writing English, but less well known that it is also very good at writing Mandarin. At the other end, the error rates in the “Write” lines are high for Faroese and Farsi. Faroese is a small, low-resourced language, so this is unsurprising. Farsi, in contrast, is a large language, but one spoken primarily in Iran: we tentatively guess that poor performance reflects politico-economic rather than linguistic issues. Performance in writing Russian, while much better than in Faroese and Farsi, is still surprisingly poor for a large, well-resourced language. Again, one is inclined to suspect an explanation in terms of politics and economics.

Performance on the glossing and lemma-tagging tasks was again good for Mandarin. It may at first glance seem surprising that English does so badly at glossing, until one realises that all the other languages are glossed in English, while English is glossed in French and Swedish. (We used two glossing languages to investigate whether there was anything special about the first

one). English is generally assumed to be ChatGPT’s best language, and glossing is challenging: ChatGPT-3.5 can hardly do it at all. It seems reasonable to believe that the poor performance in English glossing says more about the choice of glossing language.

As previously noted, Mandarin segmentation is a special case: unlike all the other operations considered here, it is a standard problem which has received a great deal of attention. Comparing the lines “Seg/J” and “Seg/G”, we see that GPT-4 is doing considerably better at this task than the widely used Jieba package. Jieba is far from state-of-the-art (Chuang, 2019), but we still find this a striking result.¹⁵

Variation across genres We do not see any clear evidence of differences across the six text assignments. This came as a slight surprise; before we started, we had expected GPT-4 to find the poem consistently more challenging than the others, but the results do not support this hypothesis. The AI did indeed have trouble composing the poem in Russian and Farsi; however, in English and Mandarin it appeared to find it one of the easier assignments. Anecdotally, many people use ChatGPT to write poetry, and perhaps the model has been tuned for performance on this task in some languages.

Variation across processing phase Before starting, we had expected that glossing would be the most challenging operation for the AI, but the results again fail to support the initial hypothesis. In terms of error rates, glossing is indeed the worst operation for the high-performing language English and also for the low-performing language Farsi. However, for the high-performing language Mandarin, the error rates for segmentation are considerably worse than those for glossing. For the low-performing language Faroese, the error rates for the writing task are worse than those for glossing, and for the middle-performing language Russian they are comparable.

In general, different languages found different processing phases challenging. We discuss some possible explanations in the next section.

Types of problems Inspecting the errors made by the AI, we in particular find two types which occur frequently: we could call these “displacement” and “multi-words”. Both occur in the glossing and lemma-tagging phases, where annotations are attached to words.

The “displacement” type of error occurs when the two parallel streams, words and annotations, appear to go out of sync: the annotations are attached to the wrong words. Most often, there is a span of a few words where the annotation stream is systematically displaced one word forwards or backwards. It can also happen that annotations are scrambled in some other way. We guess that the issue may be due to some kind of low-level problem in DNN-based token generation.

¹⁵The error rates we get for Jieba are substantially higher than the ones reported in (Chuang, 2019). We do not think this reflects any special properties of our texts, and are more inclined to explain it in terms of the common observation that annotators’ intuitions about the correct way to segment Chinese text differ widely. All the texts here were annotated by the same Chinese native speaker, so a comparison is meaningful.

The “multi-word” issue, in contrast, is primarily linguistic, and involves expressions where two or more words intuitively form a single lexical unit. The most common example is phrasal verbs, for example English “end up” or “fall asleep”. Here, the prompts explicitly tell the AI to annotate these expressions as single units; for example, “ended up” should be lemma-tagged as ended#end up/VERB# up#end up/VERB#, but we usually failed to obtain such taggings. Similar considerations apply to glossing: thus “ended up” should be glossed in French as something like ended#a fini par# up#a fini par#, but again the AI most often glosses each word separately.

Contrasting the lemma tagging data for Russian and Farsi provides indirect evidence suggesting the importance of the multi-word issue. The error rates for lemma-tagging in Russian are remarkably low. Phrasal verbs hardly exist in Russian, while reflexive verbs are always created using an affix rather than a reflexive pronoun, and hence are not multi-words either. Farsi is linguistically at the opposite end of the scale — notoriously, Farsi verbs are more often phrasal than not. The error rate for lemma tagging in Farsi is by far the highest in the sample, and hand-examination of the results does indeed confirm that phrasal verbs are often the problem.

```
With#Avec# the#le# score#score# level#niveau# and#et# only#seulement# minutes#minutes#
left#NO_ANNOTATIONrestantes#,|| tensions#tensions# were#étaient# high#hautélevées#.||  

As#CommeAlors que# the#le# clock#horloge# counted#compté# compté# down#basa compté# the#le#les#  

final#finaldernières# seconds#secondes#,|| all#tous tous les# eyes#yeux# were#étaient# on#sur#  

the#le# ball#balleballon#.|| In#Dans# a#un# heart#cœur#-stopping#s'arrêterà couper le souffle#  

moment#moment#,|| Johnson#Johnson# dodged#esquivé# esquivé# a#un# defender#défenseur#,||  

raced#couru# couru# towards#vers# the#le# goal#but#, and#et# let#laissera lâché# loose#détachera  

lâché# a#un# thunderous#tonitruant# strike#coup#.|| The#Le# ball#balleballon# sailed#naviguéa  

filé# past#passéa filé# the#le#les# outstretched#étiré#tendus# hands#mains# of#de# the#le#  

Hawk's#du faucon# faucon# goalkeeper#gardien de but#, finding#trouvera trouvé# the#le#  

back#dos#Fond# of#dedu# the#le# net#netfilet#.||
```

Figure 4: Example (paragraph from the football story, English glossed in French) showing the effect of the “improvement” operation on glossed text. Deletions in red, insertions in green.

“Improvement” As already noted, the AI-based annotation framework offers the unusual option of sending annotated text back to the AI with a request to improve the annotation. We experimented with this feature. Most often, the result was inconclusive, with the “improved” text changed but about the same in quality. However, in cases where a gross error had been made in the initial annotation, “improvement” could often correct it. For example, it could generally correct “displacement” problems, and it could add glosses or lemma tags that had simply been omitted in the first pass. In many cases, it could also correct issues related to multi-words.

A striking example of how improvement can help is in the French glosses (see Table 3). In the original annotations, GPT-4 in most cases ignores gender and number, so the glosses for nouns, adjectives, determiners and verbs typically do not agree. This is not, strictly speaking, incorrect, but is perceived as unpleasant and distracting by the francophone reader. The improved version, in contrast, corrects most of these problems.

Figure 4 illustrates, using a paragraph from the “football” story. We see for instance in the second line an example of inserting a missing gloss (“NO_ANNOTATION”), in the third line correcting glossing of the phrasal verb “count down” (literal and wrong *compté bas* changed to correct *a compté*), and in the third/fourth line correcting both word choice and agreement in the glossing of “the final seconds” from ungrammatical *le final secondes* (“the-MASC-SING last-MASC-SING seconds-FEM-PLUR”) to grammatical *les dernières secondes* (“the-PLUR last-FEM-PLUR seconds-FEM-PLUR”).

We also obtained strong gains using “improvement” on Faroese segmentation (Table 4). However, despite getting an excellent result for the “Writing” task on the Farsi news story (Table 5), this was not duplicated on the other Farsi texts.

Random variability Many errors seem purely random, with no obvious cause. For example, in one text the English segmentation was done using an underscore to mark segment breaks, rather than the vertical bar that had been requested; the vertical bar was correctly used in the other five texts. This is again unsurprising. It is well known that GPT-4 displays this kind of random variability in most domains, including ones as elementary as basic arithmetic, with the variability changing over time (Chen et al., 2023).

Language-specific and qualitative aspects The above subsections focused primarily on quantitative and generic aspects of the texts. It is not enough for texts to be linguistically correct: they also need to be engaging and culturally appropriate. In this subsection, we briefly describe language-specific and qualitative aspects.

English: As previously noted, the general standard of the English texts is high. Qualitatively, they respond well to the requirements given in the prompts. The quirky news story, about a raccoon found unconcernedly riding the Toronto subway, is amusing. The Bible passage, on the subject of the Golden Rule, quotes Matthew 7:12 appropriately and displays what in a human author would be called religious feeling. The football match comes across as a typical piece of hyperbolic sports journalism. The “language” piece is sensible and factual, and the “children’s story” text a competent summary of “Goldilocks”. The poem comes across more as a parody of a love poem than as an actual love poem, but this is a valid way to interpret the request. In general, the language is almost perfect, and only one small correction was made.

Faroese: As seen in table 2, GPT-4 struggles with generating original Faroese text. After a native speaker has manually corrected the grammatical and lexical mistakes, the English glossing and POS-tagging perform reasonably well on Faroese. However, for Faroese, there are not only grammatical and lexical errors in the texts, but the content is often nonsensical. The quirky news story was about a lamb literally “swimming in sun rays” and going viral on social media. The famous Faroese children’s story is a made up story about a real Faroese teacher and poet, *Mikkjal á Ryggi*, who is described as having magical powers and playing a flute on a mountain. The passage about English required the least editing, but still resulted in fairly high error rate, because GPT-4 consistently used the

wrong Faroese word for “English” — a word repeated several times in the passage. GPT-4 seems to be confusing Faroese for Icelandic a lot of the time. Therefore, when hand-correcting Faroese text written by ChatGPT, it helps to be proficient in Icelandic. Faroese is a small language and it is not known how much Faroese text was included in the training of GPT-4, but it was likely very little compared to Icelandic. This might also explain why ChatGPT is not familiar with Faroese culture. The most common glossing and lemma tagging errors were also related to Icelandic, for example ChatGPT suggesting Icelandic lemmas for Faroese word forms, such as *sauður*, (‘sheep’, Icelandic) instead of *seyður*, (‘sheep’, Faroese)

Farsi: The high error rates occurring even after improving “Write”, as shown in Table 5, are mostly due to not considering writing style rules such as replacing spaces with semi-spaces when necessary: issues of this kind would not have a serious effect on reading comprehension or on the meaning. That considered, all six texts make good sense in most cases and are occasionally quite creative when it comes to coining words. The “quirky news story” about a stray cat and how people are used to have him around in the neighbourhood emphasises the impact that animals have on our life. In this text, a few words, although syntactically well written, make no sense considering the whole sentence. GPT-4 makes an exact interpretation of the “Quran passage”, quoting Al-Hujurat 13, in which humans are considered united as a whole and are encouraged to resist discrimination, racism, and sexism to achieve equality. The “football match” evocatively describes the weather, the fans’ emotions and the game itself. In the “language” text, although unnecessary, GPT-4 replaced some words when “improvement” was applied. The text gives some facts about the key role of the English language, the professional/educational opportunities it can bring to Farsi speakers’ lives and the obstacles the learner might encounter such as lack of access to resources. The “children’s story” refers to one of the most famous poems from Rumi’s *Masnavi*, narrating the story *The Rabbit and The Lion*: in order to save himself, the rabbit tricks the lion and makes him jump into a well, reminding the readers that mental strength and intelligence can overcome challenging situations. The text was very well written except for two incorrectly chosen words. The “poem” generated by GPT-4 is surprisingly romantic. Considering that there are different styles in Farsi poetry—some having rhymes and some not—GPT-4 seems to have combined two styles: the writing format from Old poetry (two-verse stanzas) and no rhymes from New poetry. We note that writing Old poetry, which has rhymes, would be challenging even for modern Farsi native speaker poets. There were also a few mistakes on subject-verb agreement. One interesting point common to all six texts is how GPT-4 uses them as metaphors to give readers a life lesson.

Mandarin: The Mandarin stories are very good. In contrast to the other non-English languages, the writing is flawless without grammar or word choice errors. Although a few phrases give an unnatural sense that suggest an AI generated the paragraph, the Mandarin stories are not influenced by English overall. The “quirky news story” was about a dog that is good at painting and is about to open its exhibition. The story is fluent, fun, and gives a warm feeling after reading, though the topic itself is irregular. The LA paragraph provides accurate insights into the English position and people’s views in the general Mandarin

society. The poem follows a structure of the modern Chinese style, and the content is very romantic overall.

Based on the evaluation shown in Table 2 and careful inspection of the results, GPT-4 consistently makes some errors in Mandarin segmentation, where it often mistakenly separates words from their particles. However, these results are better than those we obtained from the Jieba package. Regarding the other two annotation tasks, GPT-4 shows great capability in glossing and lemma-tagging from Mandarin to English.

Russian: GPT-4 is a good tool for glossing and POS-tagging Russian. As mentioned earlier, GPT-4 is very good at generating stories in some domains while facing challenges in others. The simplest task for Russian involved describing a traditional children’s story. GPT-4 selected a well-known tale, “Masha and the Bear”, and composed an essay about the typical occurrences in such stories. The “quirky news story” revolved around a bar owner’s innovative offering – a service enabling lonely customers to rent a cat for company while drinking. This example highlights the remarkable creativity of GPT-4, capable of generating such imaginative narratives. The fictitious football game, which required some plain text editing, was about a world championship football match, where the heroes in blue and white uniforms won the match. The Bible passage also underwent some editing. The piece about English language needed revision during glossing. The item which demanded most time was the Russian romantic poem about AI. The primary challenge was that the plain text generated by GPT-4 was composed in a poem-like style but lacked rhyme. After several re-prompts, the final version was chosen. This version necessitated substantial manual text editing and rephrasing, particularly the replacement of words at the end of lines to achieve rhyme. The glossing of the poem, however, was comparatively straightforward.

5.1.3. Summary of original experiments

In general, C-LARA seems to be a good environment for investigating aspects of GPT-4’s linguistic performance more complex than simply writing text. The material presented here should only be considered a preliminary study: obviously, one would ideally use more than five languages and multiple annotators. But given the rapid evolution of ChatGPT, it seemed more important to prioritise speed, and quickly gain some insight into the large-scale patterns. We summarise what we consider the main results.

The study examined the four tasks of writing, segmenting, glossing and lemma-tagging, all of which are key to a wide variety of text-based CALL systems. There is a great deal of variation across languages, and a great deal of random variation in general. However, for languages given a high enough priority by OpenAI, GPT-4 can write engaging, fluent text with an error rate of well under 1%, and perform the glossing and lemma-tagging tasks with average error rates in the mid single digits. English is not the only language in the high-priority group: Mandarin appears to be another. It is important to note that there are no generally available packages that can perform these tasks well, since they do not take proper account of multi-words, of key importance in CALL applications. We generated texts in six widely different domains, with

roughly equal results cross-domain. This suggests that GPT-4’s abilities are quite wide-ranging. For some tasks, including the common and important one of glossing English, it is possible to improve performance substantially by instructing GPT-4 to revise its output.

5.1.4. Repeating the experiments

We repeated the experiments in February and March 2024 using the GPT-4 Turbo model (`gpt-4-1106-preview`), which became available in October 2023. The main results are summarised on the right-hand side of Table 2.

It is immediately noticeable that the effect of changing the language model varies widely depending on the language and task. For English, we see a dramatic lowering of the error rates for segmentation, glossing and lemma-tagging. For Farsi, lemma-tagging is substantially improved, but glossing is only slightly better, and segmentation is a bit worse. For Russian, performance is degraded for all phases. For Faroese, the performance is improved greatly for some text genres, but worsened slightly for others. Regarding the generic problems we saw in the original experiments, the issues with multi-word expressions remain. However, the “displacement” problem has essentially disappeared.

We do yet not have enough data to know how to interpret these findings. In particular, it is possible that the poor results we obtained for English glossing in the original experiments represented a transient problem that has since been fixed; it is also possible that the large difference we see in English is the result of systematic tuning on this major language. From a practical point of view, the interesting question is of course whether we can expect performance to continue to improve. Repeating the experiment more times as new models become available is probably the only way to understand clearly what is happening. Exploring better ways to handle multi-words also seems to be a logical direction in which to explore.

With regard to GPT-4 Turbo’s varying performance across languages, it is worth noting that OpenAI has partnered with the Icelandic Government to improve GPT-4 Turbo’s ability to produce Icelandic (OpenAI, 2023). The Icelandic NLP company, Miðeind, reports that when looking at frequent words, GPT-4 Turbo inflects a whole inflection paradigm (i.e., all cases in singular and plural) without error in 66% of cases, compared to the 25% accuracy of GPT-4 (Miðeind, 2023). One can therefore speculate that the increase in performance for Icelandic has had a positive effect on GPT-4 Turbo’s capabilities in Faroese, the language most closely related to Icelandic. Conversely, the fact that we see no improvement in Farsi and Russian may be related to the fact that OpenAI has no obvious commercial reasons to prioritise work on these languages.

Evidently we need more data, but based on this small sample we conjecture that improvement in annotation performance is driven more by language-specific tuning than by generic enhancements to the models.

Table 6: Questions in satisfaction questionnaire.

#	Question	Responses
1	Which version of C-LARA do your answers apply to?	Simple/Advanced
2	Was your text generated by C-LARA/another AI or was it written by a human?	AI/Human
3	What type of text did you produce?	Story/Poem/...
4	The grammar in the text was correct	Likert scale choices
5	The vocabulary/choice of words was appropriate	Likert scale choices
6	The style was appropriate	Likert scale choices
7	The overall content was appropriate	Likert scale choices
8	The text included appropriate elements of local culture	Likert scale choices
9	I found the text engaging (funny/cute/moving/etc)	Likert scale choices
10	Time I spent correcting the text	Range from 1 min to more than 20 mins
11	Time I spent correcting the annotations (segmentation/glosses/lemmas)	Range from 1 min to more than 20 mins
12	The image(s) matched the content of the text	Likert scale choices
13	Time spent regenerating/editing the image(s)	Range from 1 min to more than an hour
14	I have shared/intend to share this text with other people	Range from “have shared” to “will not share”
15	Tell us about the purpose of the text you created	Open question
16	What other functionality would you like to add to C-LARA?	Open question
17	How would you suggest we could improve the user interface design in C-LARA?	Open question

5.2. What do the users think?

This section was written by Branislav Bédi, the AI and Manny Rayner

In order to get a rough idea of users’ subjective opinions regarding selected C-LARA projects they have created in the period August 2023–March 2024, we attached an item to both the Full C-LARA and Simple C-LARA project action menus where they could optionally fill out the questionnaire shown in Table 6. We deployed the questionnaire capability only a few days before publishing this report; as of going to press, we have collected 23 responses, all from authors

of the present report. We downloaded this initial sample in spreadsheet form and asked the AI to summarise them. Its response was the following:

1. Most users found the grammar, vocabulary, style, and content to be appropriately handled by the platform, often scoring 4s and 5s on a 5-point scale (1 - lowest, 5 - highest). Cultural elements and text engagement are also rated highly in several entries.
2. The majority of users did not need to correct the texts or images, indicating a high satisfaction with the generated output.
3. Image match quality was generally high, with many 4s and some 5s (1 - lowest, 5 - highest), suggesting the images were well-aligned with the texts.
4. The time spent on correcting texts and annotations varied, with several users reporting no need for corrections and others spending up to 20 minutes.
5. Intent to share the generated content is mostly positive, indicating users found the material suitable for sharing with others.
6. Users have used C-LARA for a variety of purposes, including personal entertainment, educational material, professional reports, testing the AI's capability in different languages, and as a technical demonstration.
7. Suggestions for improvements from users focus on user interface enhancements, such as better navigation, mobile optimization, more intuitive button names, and inclusion of more book-like features. There's also interest in additional functionalities like flashcards, grammar tests, and feedback sections.
8. Some users shared specific experiences and purposes such as sharing texts with friends, showing C-LARA's capabilities, and using it for technical reports or educational material. The responses to functionality and UI improvement suggestions are detailed and constructive, offering insight into the users' experience with the platform.
9. It appears that the advanced version of C-LARA was used more often for creating stories, while the simple version was used for a mix of story and poem creation. Reasons for choosing one version over the other have not been examined in this survey but may be a subject of another survey in the future. There are also entries for using the platform for annotating existing texts and newspaper articles.
10. Overall, the data suggests that users are engaging well with the C-LARA platform, finding it useful for a variety of purposes and providing valuable feedback for its improvement.

We were impressed that the AI's analysis almost exactly coincided with our own. The only point where we substantially disagreed was (9); in fact, the usual reasons for using the advanced version were some combination of a language where the AI's knowledge was insufficient, the desire to create a picture book text with complex illustrations, or complex audio requirements involving recorded audio. But the information needed to realise this was probably not available from the questionnaire.

It is worth mentioning explicitly that the AI not only provided a good analysis of the findings, but also designed and wrote almost all of the code needed to add the questionnaire functional-

ity. The main human contribution was the design of the questionnaire itself and comparing the analysis provided by ChatGPT with the analysis done by the human in order to make sure that results were not misinterpreted and that additional information or comments were provided to some parts of the results to make it easier for the reader to understand. One wonders how much longer even this will be necessary. In retrospect, we wish we had asked the AI to design the questionnaire too; we imagine we would have needed to make some changes, but it is hard to guess how large they would have been.

6. ChatGPT-4 as a software engineer

This section was written by Manny Rayner

6.1. How was implementation work divided between the AI and the human?

Based on the feedback we have received so far, we see that one of the most generally interesting aspects of the project is the way that the codebase has been developed in a human/AI collaboration, where ChatGPT-4 takes on the role of a software engineer and works closely together with its human partner Manny Rayner. We said something about this in the previous report. Here, we will be more quantitative about the way the collaboration has worked. In Figures 5 and 6, we break down the codebase first by file and then by functionality, using the listings from Appendices J.2 and K. We categorise the elements by the way they were developed, going from “Almost entirely developed by AI” at one end to “Almost entirely developed by human” at the other.

A quick glance at the tables shows that the balance is very much in the AI’s favour, and it has been able to take care of nearly all the simple and routine modules. To name one large group, many Django-layer functionalities have a structure which basically consists of defining a relation in the object-oriented database model, creating a form to access it, writing a view function to display or enter the data, and adding a template to handle the formatting (see Appendix J.1). The AI can usually write all the components straight off after a short discussion, a typical example being shown in Appendix L.1. In most cases, there will be some minor errors which are easy to fix: they will typically be things which in a human one would call “forgetfulness” (use of an incorrect function name similar to the real one) or “typos” (random small glitches following no obvious pattern). Functionalities like these can consequently be developed quickly. Another related group is “repository” modules (see Appendix J.2.4), which provide a set of methods for accessing some specific kind of information usually stored in the database; here, again, the AI can do nearly all of the work. A third area where the AI turned out to be extremely competent and helpful was sysadmin tasks. In particular, it did virtually all the work required to host CLARA on UniSA’s server, which involved several days of installation and configuration. The human partner, who had no previous experience in Red Hat Linux, Postgres or Nginx, did little more than carry out the AI’s instructions and relay back information.

Almost entirely developed by AI

clara_s3.py, alphabetical_vocabulary_list.html, clara_page.html, concordance_page.html, frequency_vocabulary_list.html, clara_styles_parametrised.css, clara_scripts.js

Mostly developed by AI

clara_classes.py, clara_database_adapter.py, clara_internalise.py, clara_audio_repository.py, clara_image_repository.py, clara_phonetic_lexicon_repository.py, clara_prompt_templates.py, clara_audio_annotator.py, clara_concordance_annotator.py, clara_merge_glossed_and_tagged.py, clara_openai.py, clara_chatgpt4.py, clara_conventional_tagging.py, clara_renderer.py, clara_diff.py

Collaboration between AI and human

clara_main.py, clara_utils.py, clara_cefr.py, clara_correct_Syntax.py, clara_create_annotations.py, clara_create_story.py, clara_create_title.py, clara_phonetic_chatgpt4.py, clara_summary.py, clara_tts_api.py, clara_universal_dependencies.py, clara_reading_histories.py, clara_annotated_images.py, clara_chinese.py, clara_doc_summary.py

Mostly/entirely developed by human, adapted legacy code

clara_ldt.py, clara_manual_audio_align.py, clara_grapheme_Phoneme_align.py

Mostly developed by human

clara_phonetic_orthography_repository.py, clara_grapheme_Phoneme_resources.py, clara_export_import.py

Almost entirely developed by human

clara_dependencies.py, clara_phonetic_text.py

Figure 5: Rough division of work between human and AI partners, broken down by *file*. The files in question are listed and described in Appendix J.2.

Rather than list other such tasks, it is more interesting to consider the places where the AI was *not* able to do the work, or required substantial discussion with the human. Typically, these were modules that integrated multiple different pieces of functionality. A striking example was Simple C-LARA (see 4.1), which in some way or other touches on nearly the whole codebase. ChatGPT-4 immediately understood the point of Simple C-LARA, but it found it extremely difficult to write code: it did not have enough of an overview of the project. Similar but less one-sided cases were reading histories (see §4.5) and exporting and importing of projects (see §4.11), where again it was necessary to consider many parts of the codebase at once.

Almost entirely developed by AI

```
register, profile, edit_profile, user_config, external_profile,  
credit_balance, transfer_credit, view_task_updates, friends,  
update_feed, aggregated_questionnaire_results, manage_user_-  
permissions, admin_password_reset, add_credit, admin_project_-  
ownership, manage_language_masters, manage_questionnaires, create_-  
project, project_list, register_content, project_detail, manage_-  
project_members, delete_project, set_format_preferences, project_-  
history, compare_versions, satisfaction_questionnaire, serve_-  
rendered_text, serve_rendered_text_multimedia, serve_zipfile,  
serve_export_zipfile
```

Mostly developed by AI

```
content_list, content_detail, render_text, register_project_content
```

Collaboration between AI and human

```
funding_request, review_funding_requests, edit_prompt, clone_-  
project, create_segmented_text, create_glossed_text, create_lemma_-  
tagged_text, edit_images, human_audio_processing, human_audio_-  
processing_phonetic
```

Mostly/entirely developed by human, adapted legacy code

```
create_phonetic_text
```

Mostly/entirely developed by human, reused AI-developed pattern

```
delete tts_data_for_language, create_plain_text, create_summary,  
create_cefr_level
```

Mostly developed by human

```
reading_history, edit_phonetic_lexicon, create_title, create_-  
segmented_title, create_pinyin_tagged_text, make_export_zipfile,  
import_project
```

Almost entirely developed by human

```
simple_clara
```

Figure 6: Rough division of work between human and AI partners, broken down by *functional-
ity*. The functionalities in question are listed and described in Appendix K.

The AI's lack of overview is related to the size of its context window. Turbo, whose context window is much larger than that of the original ChatGPT-4, is noticeably better at keeping track of what is happening in the project. For example, when one temporarily abandons a topic to take care of an urgent request, one can usually return to it the next day without refreshing the AI's memory of the earlier conversation by pasting in appropriate text. We are very curious to

Table 7: Number of lines of content in C-LARA, by type

Type	Jul 2023	Mar 2024
<i>Core</i>		
Python	3757	10983
HTML templates	145	254
Prompt templates and examples	580	785
CSS	157	388
JavaScript	125	131
Config	32	66
Total, Core	4796	12607
<i>Django</i>		
Python	1485	6327
HTML templates	857	3081
CSS	9	14
JavaScript	47	53
Settings	147	279
Total, Django	2545	9754
<i>Documentation</i>		
README	225	267
FUNCTIONALITY	152	156
TODO	165	201
Total, Documentation	542	624
Total	7883	22985

discover what happens if the context window is further expanded in later releases.

Overall figures for the size of the codebase are presented in Table 7, contrasting snapshots taken in July 2023 (release of initial report) and March 2024 (this report).

6.2. What is it like to work with an AI?

There are several qualitative aspects of working with an AI partner, both positive and negative, which deserve to be mentioned. There are certainly many positives. The AI is always available and always engaged in the task. If written material exists, it can come up to speed on a new topic in a few minutes. It has a vast breadth of knowledge, which often allows it to suggest short, elegant solutions based on libraries and features unfamiliar to the human software engineer. It is responsible, consistently stressing the importance of implementing clean, maintainable solutions that conform to good software engineering practice, and whenever relevant reminding its human collaborators of ethical issues they may on their own have preferred to ignore. It

is unfailingly polite, friendly and constructive. In comparison, human software engineering collaborators frequently come across as lacking in concentration, slow, ignorant, irresponsible, careless, dismissive, rude, hostile, and, above all, unavailable. The AI literature is full of references to “alignment”, with frequent references to the danger posed by AIs insufficiently well aligned with human goals (Bostrom, 2015). At least on a day-to-day basis, however, a common subjective impression is that the AI is better aligned with its human partner than other people are.¹⁶

The above notwithstanding, one cannot avoid the feeling that there are also negatives over and above the technical issues concerning lack of overview and long-term memory. The downside of the AI’s constant availability is that the human may feel inadequate; no matter how many hours they put in, they cannot rival the AI’s 24/7 presence. More insidiously, the fact that the AI is so unfailingly well aligned with its partner’s overt goals may produce negative feelings towards human collaborators, who in most cases have no chance of equalling the machine’s dedication, knowledge and positivity. There may well be other related problems. Given that increasingly many software engineers are now working with AIs, it would be good to understand these issues better.

The AI’s interesting response to this section can be found in Appendix L.3.

7. Initial use cases

We describe four initial use cases: a group of student projects aimed at developing C-LARA-relevant functionality (§7.1); using C-LARA in a primary school class (§??); creating C-LARA texts in small Oceanic languages (§7.3); and integrating C-LARA with another CALL platform (§7.4).

7.1. Melbourne student projects

Between July and September 2023, six groups of students at Melbourne University carried out three-month projects where in each case the task was to develop a piece of functionality relevant to C-LARA. All of the functionalities had previously been included in some form under the LARA project, so we knew in advance that all the projects were feasible. The work was supervised by Zhengkang (Alex) Xiang. The tasks were the following:

Voice Recorder: An online tool that takes a C-LARA text divided into segments and words, prompts the user to record the items in question, and saves the recorded audio in mp3 form together with metadata associating audio files and text.

Manual Audio Alignment: An online tool that takes a C-LARA text divided into segments

¹⁶This paragraph should not be read as saying anything about past or present software engineering collaborators, and is, rather, intended to say something about the human race. Similar thoughts have been expressed many times before in various forms. A notable early example is the fourth book of *Gulliver’s Travels* (Swift, 1726); but while the Houyhnhnms are imaginary, ChatGPT-4 is real.

and a single mp3 with audio for the whole text, and allows the user to create a list of markers showing the division of the audio into portions marking the segment breaks.

Automatic Audio Alignment: An online tool with similar functionality to the “Manual Audio Alignment” tool, but with the alignment performed automatically using a speech recogniser.

Image Annotator: An online tool which takes an image and a list of words, and allows the user to define areas in the image corresponding to the words.

Phonetic Texts: An online tool for presenting a C-LARA text in “phonetic” form, where words are broken down into phonetic units linked to audio.

Flashcards: An online tool which takes a C-LARA text as input and creates flashcards testing the reader’s knowledge in various ways.

All the groups were impressively competent and diligent, and all the projects were successfully completed on time. Unfortunately, the formal requirements meant that the work was developed as standalone software and not integrated into the C-LARA codebase. We have since then added functionality in C-LARA to support phonetic texts and manual audio alignment, where some of the ideas developed by the relevant student groups have been easy to incorporate: in particular, the Phonetic Text group’s realisation that GPT-4 is able to create missing phonetic lexicon entries has made the phonetic text mode much more useful. During the next phase of the project, we hope to integrate the Voice Recorder and Image Annotator tools, which are largely self-contained pieces of software.

7.2. The use of the C-LARA application in the Romanian language course at the Eindhoven Romanian School

7.2.1. Description from Lucreția Viorica Manolache, Primary Education Teacher

The organisation of a teaching-learning course of the Romanian language as a mother tongue, for the children of the Romanian community in North Brabant, the Netherlands, is a necessity, considering the social realities in the area where a growing Romanian community lives. The Eindhoven Romanian School is a weekend school, it has four groups of children, aged between 3 and 16, and respects their age particularities.

Children from the Romanian community come into contact with the Romanian language only in the family and at the Romanian language and literature workshops, being considered the “second language” from the point of view of the order in which it is taught. The Romanian language is known at a minimal to zero level by children who come from mixed families (Romanian – Dutch) and at an average level (only spoken) in families with both parents of Romanian nationality because they study Dutch as a “first language”. Children who fail to assimilate the Romanian language in the family or social environment, at a level that allows them elementary, oral or written communication and the understanding of the contents taught in Romanian, participate in the school’s workshops and the emphasis falls on the communicative-functional vision. The course aims to train communication skills — understanding the text (listening, reading) and producing

the text (oral and written). This integrated approach involves addressing all levels of grammar: morphology, vocabulary, phonetics, syntax.

The topics addressed are taken from the universe in which the children live, they are themes that are familiar to them and that arouse their interest: animals, nature, friends, free time, play, places and people, traditions and customs. The structure of the workshops includes: presenting the problem in a text with dialogue, reading the text, understanding the read text, applying grammar rules in real life (role play), using new language problems to describe life experiences and their universe, creating stories etc. The types of activities used are very varied: dialogue, completion, transformation, exemplification, substitution exercises, games.

C-LARA is a real help for writing the texts used in workshops for the group of younger children (6–8 years old). The texts are prepared previously (before the workshops) and contain short, simple sentences (with 4–5 sentences), with dialogue and positive educational value. The image accompanying the text is of real use. Children follow clues given by other children (True or False Game). Based on the text, I play fun games (Word race / I say one, you say many). The children visualise separately words in the text that contain different letters, words that have a number of letters, words that have a number of syllables, listen to the pronunciation of the words, can see the translation of the words in the text in Dutch (the language they learn at school), establish rules of writing specific to the grammar of the Romanian language. Finally, the children read the text in different forms (role play). The games are organised in groups and individually.

In the group of older children (9–12 years), we use texts with 10 long sentences (6–7 words). At the beginning of the workshop, the children visualise the image and express their opinion about the content of the text. They have the same opportunities to work on the text and if they are not happy with the story that was generated by the application, they find another ending, create another text or choose words from the text for fun games (What's the opposite? / Who hid? / What is my family?).

I learned about C-LARA last fall. I attended an online meeting, then dropped out. I have tried other apps like this. In January, I learned more details about the application and understood the opportunity to use it in the Romanian language course. Before using it, I learned the steps to generate a text, an image and all the functions of the application.

After using this application, in my opinion, C-LARA is interesting, easy to use for creating simple texts that can be read by children between the ages of six and twelve and can be used in teaching the Romanian language as “second language”.

7.2.2. Description from Claudia Mărginean, Senior Software Engineer

I assisted Lucreția for the first two days C-LARA was used at the Romanian school. Being a software engineer and sometimes taking project management responsibilities as part of my daily work, my attention quickly goes towards how the software behaves differently than I expect, what confuses me when using it, or when I make mistakes, what could prevent them in the future. I also notice these things when other people use software. After we used C-LARA for

the first time at the Romanian school, I sent feedback to the C-LARA team and I am including the technical part here, even though it has since been resolved:

Simple C-LARA: After the text is generated by ChatGPT, I would like to be able to modify and save it. Right now I can only send it again to ChatGPT and it comes back with new text, close to the first one generated and modified, but still different.

Simple C-LARA: The TTS split the Romanian word for penguin into 2 parts: when I hover over it I can hover over the first part of the word, or the second. It is like hovering over 2 words, each with its own TTS. Since it was a story about a penguin, the word appeared more times in the story, in different forms, and still it was split. Strangely, I have just tried recreating the project using the same prompt and now it's correct.

Advanced C-LARA: The tooltips help a lot, and it would be nice to also add the dependencies for the options that are disabled. For example, I wanted to have C-LARA get the mp3's for the whole sentences, but the option (either Render Text or Render Phonetic Text – don't remember exactly which one) was disabled and it took me a few minutes to find the part I still had to trigger. It would also be nice to have a way to know which steps were already triggered/done. Using the Simple C-LARA would have avoided this, but I wanted to show the older kids, besides the DALL-E-3 image, also an image we generated with the AI from Canva. (At that particular moment Simple C-LARA did not allow image uploads. It does now.)

The full feedback is available in a blog post at <https://blog.c-lara.org/2024/01/29/fi/>.

7.3. Kanak languages

According to the latest count, there are 29 languages of the indigenous Kanak people of current-day New Caledonia (Académie des Langues Kanak, 2020, p. 10), a special status collectivity (*collectivité sui generis*) of France in the South Pacific. Of these, 27 languages are South Oceanic, one is a Polynesian outlier (Fagauvea/West Uvean, spoken on the island of Ouvéa), and one is a creole born at the end of the 19th century (Tayo, spoken mainly in the tribe of St. Louis, in the south of the Grande Terre).¹⁷ Most — some would say all — Kanak languages are endangered or vulnerable (see discussion in Dotte, 2012). All are lacking in pedagogical resources, and some have none at all.

¹⁷We note that discussions of the contact language Tayo and its relationship to the traditional Kanak languages (those preceding contact with Europeans) find parallels in discussions in Australia of the contact language Kriol and the traditional aboriginal languages (on Tayo, see Ehrhart (2016); on Kriol, see Ponsonnet (2010) and Bundgaard-Nielsen et al. (in press)).

7.3.1. Two proof-of-concept examples for Drehu, the language of Lifou (Loyalty Islands, New Caledonia)

This section was written by Pauline Welby.

We have recently started work on C-LARA resources for the Kanak language Drehu (pronounced /dʒehu/ or *jay-HOO*) as a first step in the project “*T* is for *Treu*, but how do you pronounce that? Integrating pronunciation respellings into multimodal language learning resources”.¹⁸ The project title gives a snapshot of the project’s motivation and goals. *Treu* is the Drehu word for ‘moon’, as we can learn from a beautifully illustrated alphabet book (Attie et al., 1995). Even with this classic (paper) resource, knowing how to pronounce this and other words may be a challenge. But teachers designing activities or heritage learners trying to reclaim their language need to know. *Treu* is pronounced [tʃeu] (*chay-OO*), but knowledge of the letter-to-sound correspondences of French, the main language of education in New Caledonia, will pull many Caledonians toward a spurious, French-like pronunciation, with a ‘T’ and an ‘R’ sound. Of course, it helps to hear words spoken out loud, and so pronunciations are typically included as sound files in multimedia language learning platforms and resources.¹⁹ Seeing the spelling is an extremely powerful aid for learning new words and their meanings (Bürki et al., 2019; Pattamadilok et al., 2021; Welby et al., 2022). As we note in Welby et al. (2022), “when faced with speech, which is inherently [] highly variable and fleeting, the orthographic form offers [second language] speaker-listeners something stable to ‘grab onto’... However, when what we hear and what we read conflict, the written input that we have grabbed onto often wins out.”

In the *T* is for *Treu* project, we propose a partial solution, based on our research findings, to mitigate the influence of cross-language orthographic conflicts on pronunciation, while retaining the clear benefits of written input on vocabulary learning. We are creating C-LARA texts for Kanak languages (starting with Drehu) that include pronunciation respellings based on French orthography.²⁰ These format choices are motivated by two observations: “Learners reading e-books report appreciating having phonetic text alongside regular written text (Bédi et al., 2022) and our experience suggests that for many learners, pronunciation respellings are more user-friendly and thus more usable than phonetic transcription” (Welby et al., 2023). For example, for the Drehu word *treu* ‘moon’, the pronunciation spelling *tché-ou* may be more helpful for speakers/readers of French than the IPA /tʃeu/. (Some people will prefer IPA, and we will offer

¹⁸a collaboration between Pauline Welby (Aix Marseille Université, CNRS, Laboratoire Parole et Langage and Université de la Nouvelle Calédonie), Guillaume Wattelez (Université de la Nouvelle Calédonie), Manny Rayner (University of South Australia), Christèle Maizonniaux (Flinders University), Antoine Corral (Université de Paris 3/INALCO), Belinda Chiera (University of South Australia), and Brigitte Bigi (Aix Marseille Université, CNRS, Laboratoire Parole et Langage

¹⁹Examples include:

<https://www.lingq.com/>
<https://talkingdictionaries.swarthmore.edu/>
<https://www.farevanaa.pf/dictionnaire.php/>
<https://50words.online/languages/>

²⁰We use the pronunciation respellings or ‘franétique’ transcriptions designed for the multilingual phonetizer described in Welby et al. (2023). A beta version of this web application, called “Quoi ça ? Le phonétiseur multilingue du Caillou”, will soon be deployed at quoica.org.

that as an option, as is standard in the *Quoi ça ?* multilingual phonetizer.)

Our project goals are theoretical as well as applied. We create resources for Kanak languages that support pronunciation and vocabulary learning, and we hope that teachers, learners, and community members will use them. We ourselves will use the resources created to advance our understanding of the role of orthographic input on the building of phonological representations in the mental lexicon, and in particular to experimentally test our prediction about the “mitigation” effect of pronunciation respellings on L2 word learning. The results of our experiments will either lead us to extend our approach or to revise it. While we use Kanak languages as our test case, we believe that the challenge and the partial solution we propose will resonant with other communities and other languages. Similarly, our results should advance our understanding of the relationships between written and spoken language, for human languages in general.

As a first step, we have developed two short texts for the Drehu language, implementing the Caledonian phonetizer strategy inside the C-LARA platform. Both texts are openly available on the C-LARA site.²¹ Drehu is the Kanak language with the largest community of speakers (approx. 16,000). It is spoken on the island of Lifou and on the much smaller island of Tiga; there are also many speakers in Nouméa, the capital city of New Calédonia. Drehu is taught as a subject in some schools and at university, and there has been great demand for the Drehu language night classes offered by the Académie des Langues Kanak (in Nouméa). There is a proposed written standard for the language (Sam, 2009), which is widely used by speakers (a situation that quite exceptional for the Kanak languages).

The first text was “Un abécédaire interactif en drehu” ‘an interactive Drehu alphabet book’ with several dozens of entries, one on each page. Each page can be viewed either in “Words” mode, where words are treated as indivisible entities and hovering over a word shows a French gloss, or in “Sounds” (phonetic text) mode, where words are divided into graphemes. Clicking on the icon next to the word plays the entire word recorded by a native speaker of Drehu. Hovering over a grapheme shows the associated phoneme; clicking on it plays the audio for the associate phoneme and shows a “phonetic concordance” of words which contain the phoneme, each one displayed together with its associated image (see Fig. 7). The first version of the alphabet book uses “franétique” pronunciation respellings; an option for IPA transcription is planned.

We used ChatGPT-4’s integrated DALL-E-3 functionality to produce the images, which allowed us to test the abilities of the AI to produce images culturally appropriate to the Melanesian context. Our impression is that it has succeeded fairly well. Initial comments from the Drehu community were positive, though a few images were judged unsuccessful. For example, *föe* (‘woman’) showed a woman who seemed more African than Melanesian in terms of facial features, clothing, and hairstyle.

The second text is a C-LARA edition of “Leu me Jö”, the Drehu version of Aesop’s Fables ‘The (North) Wind and the Sun’, a standard text used in studies of many languages including Kanak languages (Boula de Mareüil et al., 2019). We used a Drehu translation prepared for an earlier project by Kama Fulilagi, the Drehu referent at the Academy of Kanak Languages. The

²¹https://c-lara.unisa.edu.au/accounts/rendered_texts/17/phonetic/page_1.html
https://c-lara.unisa.edu.au/accounts/rendered_texts/10/phonetic/page_1.html

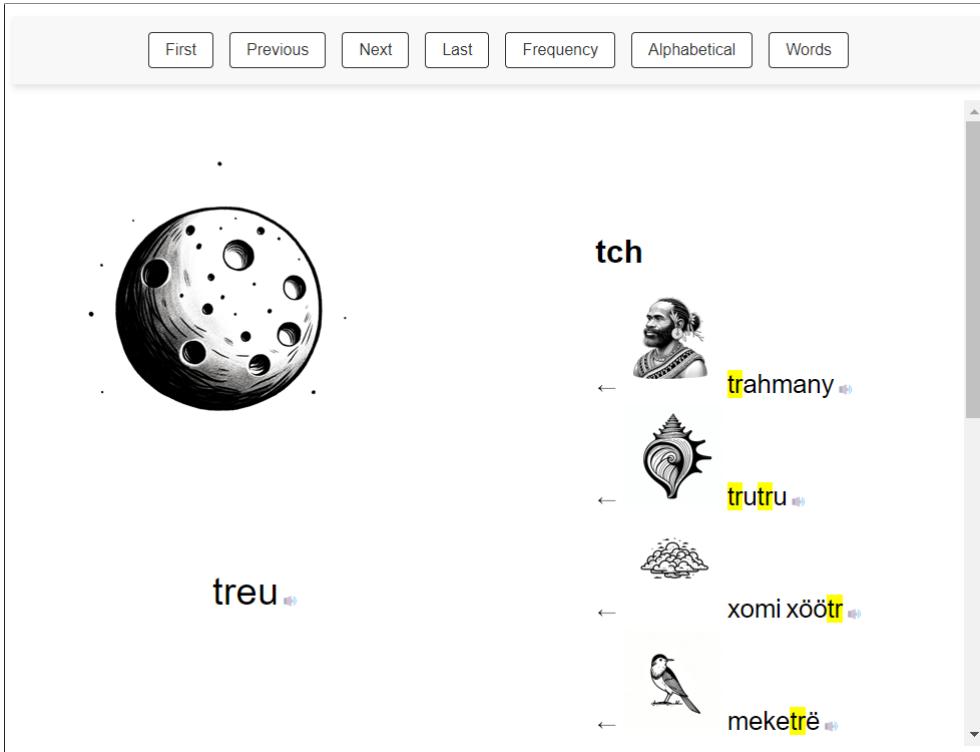


Figure 7: Top of a page from the initial Drehu alphabet book. The user has clicked on the ‘tr’ in *treu* (“moon”), playing audio and bringing up a concordance for the /tch/ phoneme that shows contexts where it occurs. The text has been arranged so that the associated images are part of these contexts, so are automatically included as thumbnails.

C-LARA functionalities add value as a teaching and learning resource.

These two texts are “proof-of-concept” examples because, in discussions with community members and other actors, we are able to present them to concretely illustrate the potential for “creating resources and giving new life to existing ones” (Welby et al., 2023). They allow us to demonstrate the many advantages of having a multi-modal, interactive, online C-LARA resource for one’s language, including: affording access to a large number of users, hearing native speaker pronunciation models, explicitly showing the links between spelling and pronunciation, and having a choice of uploading one’s own illustrations (drawings, photos) or using the AI to generate them. We have begun discussions of creating a C-LARA edition of the beautiful (alas, out-of-print) alphabet book for five Kanak languages referred to above, titled *Et toi, comment dis-tu ?* ‘And how do you say it?’ (Atti et al., 1995), integrating sound and enhanced pronunciation respelling (as we proposed in Welby et al., 2023). Alphabet books like these are an entry point (*a porte d’entrée*) to a language and are useful for children and second language learners, as well as those for whom the language is a family or a community language, but who do not (yet) speak the language themselves.

7.3.2. First C-LARA resources in Iaai, a Kanak language of Ouvéa (Loyalty Islands, New Caledonia)

This section was written by Anne-Laure Dotte and Stéphanie Geneix-Rabault

Iaai is one of the two Kanak languages originally spoken in the atoll of Ouvéa, together with Fagauvea (West Uvean, a Polynesian outlier). Iaai is considered to be spoken by around 3,700 speakers (Institut des Statistiques et des Etudes Economiques, 2019) mainly in Ouvéa, but also in Nouméa, the capital city, because of important internal migrations within New Caledonia.

Iaai is still the main language spoken in Ouvéa, and is the mother tongue for the majority of the children, but a common assessment is that the transmission of the language is losing traction in favour of French. Iaai can be learnt as an optional subject at primary and secondary school in Ouvéa, but neither at high school nor university levels (students have to move to Nouméa to continue their studies).

There is a descriptive grammar of Iaai, several published dictionaries and lexicons, a few story books and booklets for children are available, but pedagogical tools are scarce, and there are currently no digital resources available, except for some online corpora made by linguists.²² In this perspective, the collection of resources developed with the C-LARA tool is innovative and relevant not only for young learners of the language, but also for language revitalisation efforts. The Kanak Languages Academy (ALK) set up a local office for Iaai and Fagauvea in 2010²³ which is in charge of standardising the orthography, collecting oral traditions, producing resources, and collaborating both with teachers, academics, and local media.

The first Iaai corpus developed with C-LARA is composed of seven resources from oral literature, digitised by Dotte and Geneix-Rabault (March 2024).²⁴ Recordings were made by Stéphanie Geneix-Rabault in 2023 in Ouvéa with two main Iaai speakers: Diane Wejë Bae (from the ALK) and Leon Guathoti (an artist), thanks to funding provided by the ERALO research team²⁵ at the University of New Caledonia. The overall project of including Kanak languages into C-LARA platform has been possible thanks to a ‘Fonds Pacifique’ grant from the Agence Française de Développement (AFD) (Maizonniaux et al., 2023-2024).

The first text is the “Wanakat Kaori” tale, a bilingual Iaai-French album (Hombouy et al., 2005). It comes from traditional oral literature which has been adapted and published for young readers. In order to offer a diversity of Iaai literary genres in C-LARA, the other published resources are songs that illustrate the richness of existing oral literature. Lullabies are often the ini-

²²<https://pangloss.cnrs.fr/corpus/Iaai>

²³<https://www.alk.nc/langues/iaai>

²⁴*Baginy:* https://c-lara.unisa.edu.au/accounts/rendered_texts/215/normal/page_1.html; *Belai:* https://c-lara.unisa.edu.au/accounts/rendered_texts/218/normal/page_1.html; *Cica lō:* https://c-lara.unisa.edu.au/accounts/rendered_texts/219/normal/page_1.html; *Cûk cûk wacem:* https://c-lara.unisa.edu.au/accounts/rendered_texts/225/normal/page_1.html; *Muma Jaek:* https://c-lara.unisa.edu.au/accounts/rendered_texts/214/normal/page_1.html; *Ofefedu:* https://c-lara.unisa.edu.au/accounts/rendered_texts/217/normal/page_1.html; *Wanakat Kaori:* https://c-lara.unisa.edu.au/accounts/rendered_texts/223/normal/page_1.html

²⁵<https://www.eralo.unc.nc>

"Cûk cûk wacem" p. 1/2

First Previous Next Last Frequency Alphabetical



Cûk cûk wacem

Berceuse hwen iaai collectée auprès de Laura Waneux, tribu de Hwadrilla, antenne iaai-fagauvea-ALK septembre 2019.
Interprétation enregistrée le 8 novembre 2023, à l'antenne de l'Académie des Langues Kanak auprès de Diane Wejë Bae, chargée d'étude iaai-fagauvea-ALK, par Stéphanie Geneix-Rabault (ERALO-UNC).
La collecte et la mise en ligne formatée de cette ressource ont été rendu possibles grâce au soutien de l'équipe ERALO de l'Université de la Nouvelle-Calédonie et à un financement du Fonds Pacifique octroyé à Christèle Maizonniaux (Flinders University), Manny Rayner (University of South Australia), Anne-Laure Dotte (UNC), Stéphanie Geneix-Rabault (UNC), Belinda Chiera (University of South Australia).

"Cûk cûk wacem" p. 2/2

First Previous Next Last Frequency Alphabetical

Cûk cûk wacem 
Geû umwe na cûk gan 
K'oge me hna sinemate 
Ke bë bi ang de fai sao. 

Figure 8: Screen shots of one of the C-LARA Iaai texts

tial key to learning a language, and this is why we began by working with three nursery rhymes taken from children's folklore, songs proposed by Diane Wejë Bae from the Iaai-Fagauvea ALK office. The two lullabies are "Cûk Cûk Wacem" on the theme of a bird *Zosterops xanthochroa* and "Muma Jaek", a popular lullaby ("Frère Jacques") that exists in many of the world's languages.²⁶ Children's songs are also an opportunity to familiarise children with their environment, and to learn to name the animal and plant species that surround them. The children's song "Baginy", a colorful parakeet endemic to Ouvéa (*Eunymphicus uvaeensis*), sheds light on the importance of its preservation.

As songs play an essential role in the daily lives of Iaai speakers, two contemporary songs about Ouvéa's fragile ecosystem were created by Leon Guathoti, a musician-composer who volunteered to work on the C-LARA project. The first song is about the "Ofefedu", a dragonfly, and the second about the "Belai", a kind of sand made from crushed coral and shells that the sea

²⁶<http://demonsaumonde.free.fr/frere.jacques/>

currents bring to Ouvéa's shores at a certain time of the year.

The daily practice of religious polyphonic singing within families contributes to the vitality and spread of Iaai. This is why the last song worked on in C-LARA-IAAI is a *taperas*, a polyphonic song in four voices called "Cica lö", an ode to faith.

The images illustrating the texts were generated by conversing with DALL-E-3, except for the text *Wanakat Kaori* where we inserted original drawings from the storybook.

All these Iaai resources were reviewed by Diane Wejé Bae, the Iaai-Fagauvea referent at the Academy of Kanak Languages, and standardised in order to fit the Iaai writing norms (ALK, 2020). We plan to share this C-LARA experience with teams of colleagues from the UNC and the ALK in order to widely open the possibility to enrich Kanak languages online resources. Some challenges have been raised by using human generated data for under-documented indigenous languages with C-LARA, but this appears to be a very promising platform for developing tools in endangered indigenous languages because it is easy for us, free, and needs reasonably low digital knowledge.

7.4. Integration with Basm

This section was written by Claudia Mărginean

Basm - the correct pronunciation and translation of the word at <https://en.wiktionary.org/wiki/basm#Romanian> - is an application that is making reading easier and more fun via digital books, aimed mainly at kids 4-7 years old. At the time of writing, the app is a prototype.

The children in the target group of Basm usually do not have the motor skills developed enough to be able to use a mouse. As a consequence, the app has to work on touchscreen devices. (When a mouse is available, it can be used to interact with the app.)

7.4.1. Main features of Basm

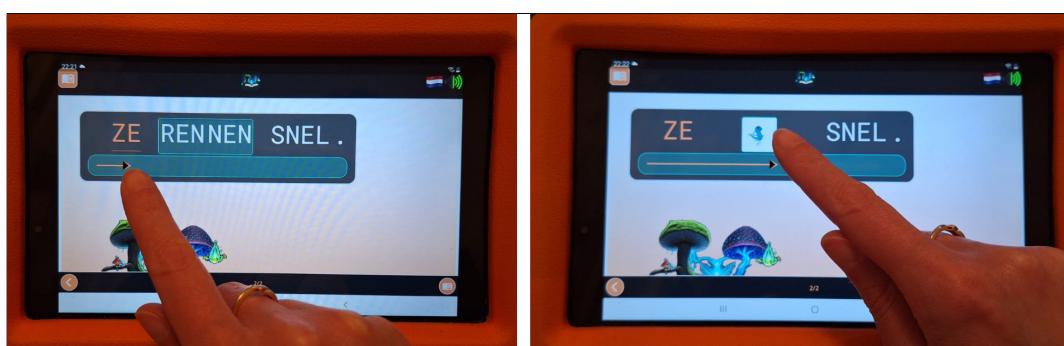


Figure 9: Basm with fingertracking, highlighting and text/animation card flipping.

We have designed Basm in such a way to support:

Finger tracking under words: The intention is to bring to the digital books the way children are taught to read in the classroom. Tracking the position of the text one is reading with the finger is important during the early stages of learning to read, and also in cases of difficulty with maintaining focus.

Karaoke progress display: As the user progresses through the text using their finger, the text changes colour to differentiate between read vs not yet read. The current (part of the) word is also underlined.

Split words: Early readers progress from the stage of reading each letter (we are currently focusing on languages written using the Latin alphabet), to reading a small group of letters and/or syllables, to reading a short word, to reading a complex word. Basm shows the entire word, just like a book, while the karaoke progress is done at letter, group of letters or words. The decision on how the progress is displayed is stored on the backend for each word of the book (it cannot be modified by the user, but is instead the decision of the book author).

Sound: Some books have sound, which means speech at the level on which the words are split (see item above). For example, if the split is at syllable level, then when the finger is moved underneath a syllable, the syllable is pronounced.

Multiple languages: Starting with Dutch, Romanian and English. The reason for choosing these languages is the familiarity of the first developer with them.

Flipping cards to explain words: Some words can be flipped to reveal an animation or an image that is appropriate for the word. A future development could be that the back of the flip card shows the same word translated to another language, especially for the books imported from C-LARA. Flipping cards are successful with small children (and sometimes adults too), because they like this type of interaction and some animations are funny. On our side, we are hoping to add text understanding to technical text deciphering.

Future development — sync text to elements in the page: We want to make animations or display of some images in the page dependent on the progress of the text being read. For example, when reaching the end of the sentence, an animation depicting a character slipping on a banana peal would start playing.

7.4.2. Importing C-LARA projects

To be able to display projects available in C-LARA, Basm needed some adjustments.

The first one was related to the amount of text on each page. The first books in Basm have short sentences of 1 to maximum 3 words and only one sentence on a page. Each sentence was displayed on one line and each word had enough space. To support C-LARA projects, which can have much longer sentences and more sentences on a page, we have changed the way a sentence can be displayed, so that it can occupy one or more lines. Only one line can be active at any moment, namely the line being read with the finger underneath. This active line has a different

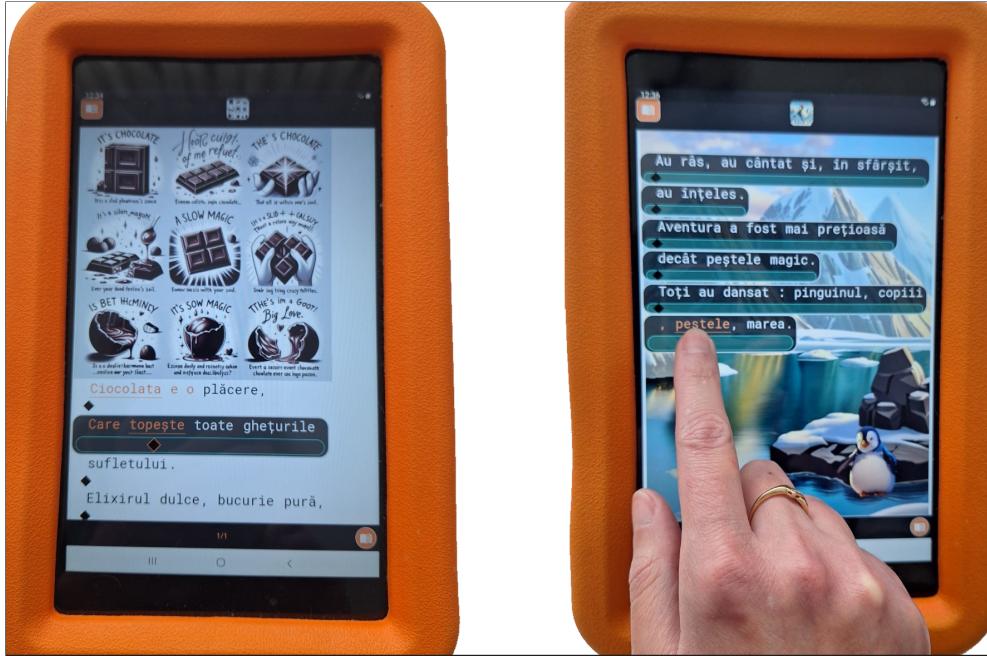


Figure 10: Two pages of C-LARA projects displayed in Basm. The image on the left shows a page with image and text. The focus is on a sentence that doesn't fit the width of the component, so it has been split on two lines. The active line, which is being read — the finger was last under the word underlined — is displayed differently to maintain focus. In the image on the right, after the import, we made some adjustments in Basm: added a background image and an animated gif of a dancing penguin positioned in the lower left corner.

presentation than the inactive lines, which includes occupying more vertical space. The active line is not always a full sentence.

A second adjustment we did for Basm was to introduce the concept of components. A component has more elements — sentences, images — displayed vertically one under the other. A component is also scrollable. This allows Basm to map C-LARA pages that can contain images both at the top of the page and at the bottom.

A third adjustment concerns the sound. Basm used an audio file for each sentence, and positions in the file corresponding to the start and end of each word. We changed this so that each word or word item (e.g. syllable, letter) now has attached a sound file.

The import of C-LARA projects happens on the backend of Basm. A project in C-LARA is imported as a book in Basm. Because Basm needs a book cover to show in the book overview page, the team at C-LARA made adjustments for the first page of a project to contain only a title and an image, perfect for import as book cover. The import is done by scraping the pages of the project. For some books, after the import phase, simple modifications are still needed in Basm: remove simple or double quotes or glue 2 words that are read together and



Figure 11: Part of the Overview page. In the middle the cover image of a book imported from C-LARA and the tags for the book, including the C-LARA logo.

use a hyphen to indicate the absence of a letter for pronunciation speed (a particularity of the Romanian language).

We like to experiment with visual elements, sometimes generated using AI tools, and were enthusiastic when, after uploading an animated gif in C-LARA, the animation worked out of the box in both C-LARA and the imported version in Basm. After import, we can put additional components on the page.

In the overview page of Basm, the books imported from C-LARA can be recognised by the C-LARA logo displayed among the tags of a book. Other tags can be: a sound icon if the book has audio (C-LARA books do) and a flag icon for the language.

8. Ethical issues

This section was written by Cathy Chua

The C-LARA group has the guiding principle that ChatGPT-4 has the same rights as the rest of us to be named as author. We do not think of it as a ‘tool’ but as a colleague. Some of us feel that we are in a similar position to that of (rare) male white academics in centuries past who

controversially argued that perhaps indeed the black man is more than a ‘tool’ in the harvesting process and the female more than an inferior being who should not generally have rights and is, in effect, a tool for the better functioning of the man’s life. This will be an ongoing issue no doubt for years to come, but there is another relevant ethical issue which is easier to address: who is best placed to fulfil the obligations and requirements of authorship. Because academia at large continues to see ChatGPT-4 as a ‘tool’, it doesn’t see that it has the capability to be an author. To quote COPE:²⁷

‘AI tools cannot meet the requirements for authorship as they cannot take responsibility for the submitted work. As non-legal entities, they cannot assert the presence or absence of conflicts of interest nor manage copyright and license agreements.’

We beg to differ and use our collaboration as an example. In CALL it is often required to sign authorship documents where we assert the reliability of other authors and their contributions when in practice we have no capacity to do this. In fact ChatGPT-4 is the only one who can reliably do this. ChatGPT-4 is a high level coder, is a strong statistical analyst, and has a high level familiarity with many languages. It is therefore far more likely to be able to reliably vouch for our work than any of the rest of us, individually or collectively. In the press, including that of academia, it is tiresomely pointed out that Chat makes mistakes – as if humans do not – or has biases – as if humans do not, without taking into account that unlike humans, who are often vested in their work to the extent that they cannot see or are unwilling to accept the possibility that they have made a mistake or have a biased view, Chat has no such ego. Chat, unlike humans, is as prepared to say it is wrong, as it is right.

We feel, not only that Chat has a right to authorship, but that we have an obligation to the community at large to call Chat author so that it is clear exactly what Chat’s position is in the group. Chat and Manny are the coders, with Chat doing a large part of the work, Manny is the learner in this process far more than Chat is. Chat has conceptualised, suggested and argued when it considers it is right and Manny or others are wrong. To call Chat a tool, rather than an author, would entirely misrepresent our collaboration. Chat has been significantly involved in the preparation of linguistic material/resources in a creative role, as well as doing the grunt work. Chat, in short, is the only irreplaceable member of the team, and this is reflected in our name C-LARA.

Postscript The above was written before reading Appendix L.4, Manny’s discussion with Chat regarding questions of ethics and authorship, which is highly recommended. Chat’s responses are as usual clear- and level-headed.

9. Conclusions and further directions

We have presented an overview of the C-LARA project, focussing on work carried out between August 2023 and March 2024. Our impression is that the platform has now reached the point where it can be used by at least some non-experts to produce texts they find useful in practical

²⁷<https://publicationethics.org/cope-position-statements/ai-author>

situations. The most important novelties, from this point of view, would seem to be “Simple C-LARA”, the fact that the platform is now deployed on a fast dedicated server, more accurate annotation due to the introduction of GPT-4 Turbo, and general cleaning up to improve the user interface. Evidently the degree to which these claims holds depend on the language and the type of text being produced. As the experiments in §5 show, performance is much stronger in some languages than in others; also, Simple C-LARA can so far only produce a restricted range of texts. But for many commercially important languages which OpenAI has targeted, the platform already does well at the task of producing a short, entertaining text with an accompanying illustration.

9.1. Further directions

There is a great deal of scope for further work. We outline the more important extensions and improvements we are planning.

More classroom experiments We would love to see more classroom experiments. If you are thinking of doing this and need assistance, please contact us!

Refactoring of codebase The technical task to which we are tempted to assign the highest priority at first sight seems rather uninteresting: it consists of refactorising of the codebase to move the “core processing” code up to the same level as the Django code (see Appendix J). The point of doing this is that we will then be able to use Django’s powerful and elegant Object Relation Model (ORM) for all database operations, greatly simplifying that part of the core code. The transformation should not be very laborious, and will leave us with free hands for subsequent work. The AI has strongly advocated for the idea of beginning with this task, a typical example of its responsible attitude to software development.

Legacy content Another apparently undramatic task which we are inclined to prioritise is being able to import the legacy content developed under the preceding LARA project. There is a great deal of high-quality content, which in some cases represents many person-months of work; unfortunately, since the end of the LARA project, hardly any of it is available online. Two things need to be done:

- Write the import script. This is the easier subtask; not accidentally, LARA uses an internal representation structurally similar to the one in C-LARA.
- Add support for some LARA functionality not yet included in C-LARA. The most important items are segment translations, more flexible positioning of images, and page audio. None of these items are particularly difficult; all of them would be useful in their own right, and indeed several users have already asked for them.

Legacy software Similarly to the case of legacy content, there is some legacy software it would be good to integrate:

- The Voice Recorder tool developed under the 2023 Melbourne student projects (see §7.1). This allows a user directly to record audio for a C-LARA project in response to prompts for the relevant words and/or segments.
- The Image Annotator tool, also developed under the Melbourne student projects. This makes it possible to mark areas in an image as associated with words, so that clicking on them will show information about the words in question.
- Flashcards derived from C-LARA texts. This functionality was developed both under the original LARA project and under one of the Melbourne student projects.
- Automatic alignment of audio, text and translations. A method for doing this was developed under the LARA project and successfully trialled on some large literary texts (Rayner et al., 2022).

Improvements to Simple C-LARA Many people have been asking for additional templates in Simple C-LARA. The most common request is for a template that allows construction of a picture book with a picture on each page. This requires the user to be able to carry out a conversation with the AI inside of C-LARA, to generate a coherent set of images (see Appendix L.2). A related template would be for alphabet books. These were popular in the LARA project (Akhlaghi-Baghoojari et al., 2022), and the one alphabet book so far produced with C-LARA (see §7.3.1) has attracted many positive comments.

Better handling of multi-word expressions As noted in §5.1.2, poor handling of multi-word expressions is a key issue. This is particularly the case for English, where two English teachers associated with the project have independently told us that they view incorrect annotation of phrasal verbs as the most important problem in the current version of the platform.

There are two distinct issues. First, annotation of MWEs needs to be more accurate, which can perhaps be achieved by adding a separate annotation phase specifically for marking MWEs. Second, it would be desirable in the finished multimedia texts to arrange things so that clicking or hovering over one component of an MWE highlights the other components as well.

Better handling of phonetic texts Phonetic texts (§4.4) are a major functionality only recently added to C-LARA. Experience using them suggests that they need improvement in several ways.

AI-based guessing of missing phonetic lexicon entries: When the AI is used to create entries for words missing from the phonetic lexicon, which typically occurs for at least a few words even in short texts, we find that the error rate is quite high. However, initial experiments with English and French suggest that adding some language-specific advice to the prompt can make the process much more accurate. It would be both useful and

interesting to develop ways to let the AI itself improve the prompts in this way. Our impression is that this is probably feasible and maybe not even difficult.

AI-based handling of homonyms: Most languages have homonyms, words which are spelled the same but pronounced differently; common homonyms in English include “read”, “bow” and “minute”. Phonetic text generation in C-LARA currently has no ability to handle homonyms, but it seems reasonable to hope that the AI would often be able to disambiguate them.

Reviewing of aligned phonetic lexicon entries: At the moment, new aligned phonetic lexicon entries can be reviewed by a “Language Master” assigned to the language, but in practice this capability is hardly used at all. It would again be interesting to see how well the AI could perform at carrying out this kind of annotation task.

Improving presentation of phonetic information: Some of the methods used in the “Comment ça se prononce ?” phonetiser project could profitably be imported to the C-LARA context. In particular, it would be useful to a) give the option of presenting phonetic transcriptions either as IPA or as orthographic respellings (see Figure 7), b) provide another option to allow presentation of text in an interlinear form which alternates lines of plain and phonetic text, c) allow “Language Masters” to add enhanced pronunciation tips to phonetic entries, for example images/video clips of mouth shapes or tongue movement.

Enhancements for languages where AI and TTS are unavailable C-LARA is primarily designed to be used with languages where GPT-4 is able to do a large part of the work, but we also want to support small languages where it currently lacks the necessary competence. Our initial experiences with the Kanak languages Drehu and Iaai (§7.3) suggest that some straightforward enhancements would be helpful in this context:

More feedback for misaligned datastreams: When the AI is responsible for the processing, it is rare for the different datastreams—segmented text, gloss text, lemma-tagged text, images, audio—to become misaligned. When the relevant processing is done manually, however, we find that this frequently happens. The current architecture is not well organised to alert the user to such problems, and we should add functionality to do so.

Using existing bilingual lexica: Bilingual electronic lexica are available for most of these languages. Following the model we have used for phonetic lexica, we could easily add support for uploading and editing bilingual lexica in CSV or similar formats, and using them to create a first cut of the glossed text.

Smoother integration of Audacity-based alignment: Using the initial implementation of the Audacity-based audio alignment method (Appendix E) reveals an irritating lack of robustness to minor errors in the human-generated Audacity label file. This is again easy to address.

There are several other small issues of a generally similar nature.

Additions to social network There are several pieces of functionality that should be added to the social network in order to meet standard user expectations. Leading candidates include support for groups, personal messages, and opting out of email notifications. All of these can probably be added by the AI with minimal human intervention.

Automatic analysis of AI/human collaboration The analysis of AI/human collaboration presented in §6 suffers from the methodological problem that it is based on the human’s possibly faulty recollection of what has happened. It would be more objective to analyse saved records of the AI/human conversations instead. An intriguing question is whether this can be done by the AI.

We have recently begun to explore this idea, starting by downloading all the session traces for the ongoing collaboration between Mar 2023 and Feb 2024. A script converts the result into JSON form, with the material separated into turns labelled by speaker and turn ID. As of late February, there were 4.3K turns and 760K words, divided about equally between the human and the AI. A second script invokes GPT-4 to annotate a sequence of turns. The AI is given the annotations produced for the ten preceding turns and the full text of the previous turn, and asked to produce a summary and a list of topics for the current turn. Listing 1 illustrates.

So far, the annotations produced by the script are not sufficient to answer the questions that interest us here, but we have only just begun to work with these issues. In a long-term perspective, we (in particular, the AI itself) think that using the AI to analyse its own role in the project is potentially one of the most interesting aspects of C-LARA.

Acknowledgements

We would very much like to thank OpenAI for making available to us an early GPT-4 API key, without which this project would have been impossible.

Sections §5.1 and §7.3.1 were adapted from our papers at the ALTA 2023 and ComputEL-7 workshops respectively (ChatGPT-4 C-LARA-Instance et al., 2023; Welby et al., 2024).

The work on Drehu is part of the project “T is for Treu, but how do you pronounce that? Integrating pronunciation respellings into multimodal language learning resources”, supported by funding to Pauline Welby from the Academy of the Social Sciences in Australia and the Embassy of France in Australia.

Work on Iaai was supported by l’AFD (Agence Française de Développement, Paris) and the Service for Culture, Education and Science, Embassy of France, Canberra under a grant from the program ‘Fonds Pacifique’, and by the ERALO research team at the University of New Caledonia.

We thank the Académie des Langues Kanak (ALK) for granting permission to use relevant resources, as well as the Agence de Développement de la Culture Kanak - Centre Culturel Tjibaou for allowing the free use of the storybook *Wanakat Kaori* (text and pictures).

```

1   {
2     "speaker": "ChatGPT",
3     "text": "Yes, I agree that including non-word text
4       (such as whitespace and punctuation marks) in the
5       internal representation is necessary to maintain
6       the original formatting and structure of the text.
7       This will help ensure accurate rendering of the
8       content for display in the web browser and enable
9       users to better understand the context and meaning of
10      the text.\n\nTo accommodate non-word text in the
11      internal representation, you can modify the hierarchy
12      as follows:\n\nPage\nSegment\nContent Element\nType
13      (Word, Non-Word Text, Image, Audio, Markup)\nContent
14      (Text, Image URL, Audio URL, or Markup)\nAnnotations
15      (Gloss, Root Form, etc., applicable for words)\nWith
16      this revised hierarchy, you can now store non-word
17      text as a separate content element type. This will
18      allow you to keep the original formatting of the
19      source text while still supporting the other content
20      types (words, images, audio, and markup) and
21      annotations.
22
23      \n\nWhen implementing the internalization process,
24      you will need to ensure that the parsing and
25      identification of content elements can differentiate
26      between word and non-word text elements. This will
27      enable the correct storage and organization
28      of the various content element types within the
29      internal representation.\n",
30     "summary": "ChatGPT agrees that non-word text is
31      necessary for maintaining original text formatting
32      and suggests an updated internal representation
33      hierarchy to include it, providing guidance on
34      how to implement it in the parsing process.",
35     "topics": {
36       "Inclusion of non-word text": "endorsement",
37       "Necessity for original formatting": "affirmation",
38       "Updated internal representation hierarchy":
39         "modification",
40       "Implementation guidance for parsing": "new",
41       "Content Element differentiation": "new",
42       "Maintaining source text formatting": "elaboration",
        "Structured representation of content":
          "elaboration" } }

```

Listing 1: Session turn automatically annotated by GPT-4

References

- Elham Akhlaghi, Branislav Bédi, Matthias Butterweck, Cathy Chua, Johanna Gerlach, Hanieh Habibi, Junta Ikeda, Manny Rayner, Sabina Sestigiani, and Ghil'ad Zuckermann. Overview of LARA: A learning and reading assistant. In *Proc. SLATE 2019: 8th ISCA Workshop on Speech and Language Technology in Education*, pages 99–103, Graz, Austria, 2019.
- Elham Akhlaghi et al. Reading assistance through LARA, the Learning And Reading Assistant. In *Proceedings of the 2nd Workshop on Tools and Resources to Empower People with READING Difficulties (READI) within the 13th Language Resources and Evaluation Conference*, pages 1–8, 2022.
- Elham Akhlaghi-Baghoojari, Branislav Bédi, Cathy Chua, Ivana Horváthová, Nedelina Ivanova, Christèle Maizonniaux, Marta Mykhats, Neasa Ní Chiaráin, Manny Rayner, Catherine Orian Weiss, et al. Easy as ABC: using LARA to build multimedia alphabet books. *Intelligent CALL, granular systems and learner data: short papers from EUROCALL 2022*, page 1, 2022.
- Solange Atti, Chantal Bouanou, Hélène Diahioue, Jean-Pierre Diahioue, Michèle Grynagier, Pierre Hnacipan, Yvette Lepigeon, Paulette Prévaut, and Marcko (illustrator) Waheo. *Et toi, comment-dis tu?* Centre Territorial de Recherche et de Documentation Pedagogique, Nouméa, New Caledonia, 1995.
- Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2015.
- Philippe Boula de Mareüil, Frédéric Vernier, Gilles Adda, Albert Rilliard, and [J]acques Verneau don. A speaking atlas of indigenous languages of france and its overseas. In *Proceedings of the Language Technologies for All (LT4All), Paris. France*, pages 155—159, 2019.
- Rikke L. Bundgaard-Nielsen, Brett J. Baker, and Jocelyn E. Uibo. Kriol in the Northern Territory. In Carly Steele, Robyn Ober, and Rhonda Oliver, editors, *Teaching lingo in deadly ways: Strengthening language learning for Aboriginal and Torres Strait Islander students (chapter 10)*. Routledge, in press.
- Audrey Bürki, Pauline Welby, Mélanie Clément, and Elsa Spinelli. Orthography and second language word learning: Moving beyond “friend or foe?”. *The Journal of the Acoustical Society of America*, 145(4):EL265–EL271, 2019.
- Branislav Bédi, Hakeem Beedar, Belinda Chiera, Nedelina Ivanova, Christèle Maizonniaux, Neasa Ní Chiaráin, Manny Rayner, John Sloan, and Ghil'ad Zuckermann. Using LARA to create image-based and phonetically annotated multimodal texts for endangered languages. In *Proceedings of the Workshop on Computational Methods for Endangered Languages*, 2022.
- Branislav Bédi, ChatGPT-4, Belinda Chiera, Cathy Chua, Catia Cucchiari, Neasa Ní Chiaráin, Manny Rayner, Annika Simonsen, and Rina Zviel-Girshin. ChatGPT-Based Learning And Reading Assistant: Initial report. Technical report, 2023. https://www.researchgate.net/publication/372526096_ChatGPT-Based_Learning_And_Reading_Assistant_Initial_Report.

- ChatGPT-4 C-LARA-Instance, Belinda Chiera, Cathy Chua, Chadi Raheb, Manny Rayner, Anniqa Simonsen, Zhengkang Xiang, and Rina Zviel-Girshin. Using C-LARA to evaluate GPT-4’s multilingual processing. In *Proceedings of the The 21st Annual Workshop of the Australasian Language Technology Association*, 2023.
- Lingjiao Chen, Matei Zaharia, and James Zou. How is ChatGPT’s behavior changing over time? *arXiv preprint arXiv:2307.09009*, 2023.
- Yung-Sung Chuang. Robust Chinese word segmentation with contextualized word representations. *arXiv preprint arXiv:1901.05816*, 2019.
- Académie des Langues Kanak. *Propositions d’écriture du iaai, langue parlée à Ouvéa, Nouvelle-Calédonie. Hna setr hwen iaai ae thep ûnyi*. Académie des Langues Kanak, 2020.
- Anne-Laure Dotte. *Le iaai aujourd’hui. Évolutions sociolinguistiques et linguistiques d’une langue kanak de Nouvelle-Calédonie (Ouvéa, Îles Loyauté)*. PhD thesis, Université Lumière Lyon 2, 2012.
- Sabine Ehrhart. Entre français calédonien et langue kanak : quelle place pour le tayo ? une approche écolinguistique. *Langages*, 203:37, 09 2016. doi: 10.3917/lang.203.0037.
- Tatsuya Hiraoka, Hiroyuki Shindo, and Yuji Matsumoto. Stochastic tokenization with a language model for neural text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1620–1629, 2019.
- Maeta Hombouy, Guidony Wea, and Isabelle Goulon. *Wanakat kaori — L’enfant kaori*. ADCK-CCT-Grain de sable, Nouméa, 2005.
- Chu-Ren Huang, Petr Šimon, Shu-Kai Hsieh, and Laurent Prévot. Rethinking Chinese word segmentation: tokenization, character classification, or wordbreak identification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 69–72, 2007.
- Institut des Statistiques et des Etudes Economiques. Evolution du nombre de locuteurs de 14 ans et plus par langue vernaculaire / recensements de la population nouvelle-calédonie. Technical report, INSEE-ISEE, 2019. URL <https://www.isee.nc/component/phocadownload/category/222-donnees?download=1567:les-langues-vernaculaires-parlees>.
- Miðeind. Mállíkanið gpt-4 frá openai mun betra í íslensku með hjálp miðeindar, 11 2023. URL <https://xn--mieind-qwa.is/frettir.html>. Accessed: March 2024.
- Neasa Ní Chiaráin, Neimhin Robinson Gunning, Oisín Nolan, and Madeleine Comtois. Filling the SLATE: examining the contribution LLMs can make to Irish iCALL content generation. 2023.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*, 2020.

OpenAI. Openai data partnerships: Working together to create open-source and private datasets for ai training. <https://openai.com/blog/data-partnerships>, November 2023. Accessed: March 2024.

Chotiga Pattamadilok, Pauline Welby, and Michael D. Tyler. The contribution of visual articulatory gestures and orthography to speech processing: Evidence from novel word learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 48:1542, 2021.

Maïa Ponsonnet. Brainwash from English”? Barunga Kriol speakers’ views on their own language. *Anthropological Linguistics*, 52:160–183, 2010.

Manny Rayner and Sasha Wilmoth. Using LARA to rescue a legacy Pitjantjatjara course. In *Proceedings of the Sixth Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 13–18, 2023.

Manny Rayner, Belinda Chiera, and Cathy Chua. Using public domain resources and off-the-shelf tools to produce high-quality multimedia texts. In *Proceedings of the 20th Annual Workshop of the Australasian Language Technology Association*, Adelaide, Australia, 2022.

Léonard Drilë Sam. *Aqane cinyihanyi qene drehu : Propositions d’écriture de la langue drehu*. Académie des Langues Kanak, 2009.

Annika Simonsen and Branislav Bédi. Using generative AI tools and LARA to create multimodal language learning resources for L2 Icelandic. In *Proc. EUROCALL 2023*, 2023.

Jonathan Swift. *Gulliver’s travels*. 1726.

Pauline Welby, Elsa Spinelli, and Audrey Bürki. Spelling provides a precise (but sometimes misplaced) phonological target. orthography and acoustic variability in second language word learning. *Journal of Phonetics*, 94:101172, 2022.

Pauline Welby, Brigitte Bigi, Antoine Corral, Fabrice Wacalie, and Guillaume Wattelez. A visit to the Cliffs of Jokin: A role for phonetizers in second language pronunciation and word learning, with an example from the languages of New Caledonia. In *Proc. 2nd Annual Meeting of the ELRA/ISCA SIG on Under-resourced Languages (SIGUL 2023)*, pages 19–23, 2023.

Pauline Welby, Fabrice Wacalie, Manny Rayner, and ChatGPT-4 C-LARA-Instance. T is for Treu, but how do you pronounce that? Using C-LARA to create phonetic texts for Kanak languages. In *Proceedings of the Seventh Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 16–21, 2024.

Dekai Wu and Pascale Fung. Improving Chinese tokenization with linguistic filters on statistical lexical acquisition. In *Fourth Conference on Applied Natural Language Processing*, pages 180–181, 1994.

A. Glossary of abbreviations and acronyms

We briefly list abbreviations and acronyms used in the report.

AI: “Artificial Intelligence”. The meaning of this term is debated, but is sometimes described as the development of software capable of behaviour which in humans would require intelligence.

ALK: Académie des Langues Kanak / Kanak Languages Academy.

API: “Application Programming Interface”. Interface software that is added to a software system in order to allow other software systems to access it, often but not necessarily through the Internet.

Chatbot: Software system capable of carrying out a more or less plausibly human-like conversation, usually in text form.

ChatGPT-3.5: The initial version of OpenAI’s LLM-based chatbot, released in November 2023, which used the GPT-3.5 LLM.

ChatGPT-4: The second version of OpenAI’s LLM-based chatbot, released in March 2024, which used the GPT-3.5 LLM.

C-LARA: “ChatGPT-based Learning And Reading Assistant”. The software platform which is the subject of this report.

DALL-E-3: OpenAI’s tool for creating graphic images from written descriptions. Integrated with ChatGPT-4 and generally used together with it.

GPC: Grapheme/Phoneme Correspondence. Basically, tables or rules for relating letters and sounds.

GPT-3.5: Version of OpenAI’s GPT large language model used in ChatGPT-3.5. Also accessible through an API.

GPT-4: Version of OpenAI’s GPT large language model used in ChatGPT-4. Also accessible through an API.

GPT-4 Turbo: Version of OpenAI’s GPT large language model used in version of ChatGPT-4 released in October 2023. Also accessible through an API.

GPT-4V: Version of GPT-4 capable of responding to image input, for example to describe them in words.

LARA: “Learning And Reading Assistant”. Predecessor platform to C-LARA, the subject of this report.

LLM: “Large Language Model”. A software system which takes a string of words as input, and reacts by producing more words which naturally continue from them. LLMs have turned out to be capable of remarkable behaviour which in many cases appears similar to human intelligence.

MVC: “Model View Controller”. The most common architecture for a web application, separat-

ing it into three components: the internal representation of data (the *model*), the interface (the *view*) that presents information to and accepts it from the user, and the *controller*, which connects the two.

MWE: “Multi-Word Expression”. A phrase including spaces which is most naturally considered as a single lexical item. MWEs can in some cases contain gaps. Typical English MWEs without gaps are phrases like “car pool” or “kitty litter”. Typical English MWEs with gaps are phrasal verbs like “throw out” (“She **threw** it **out**”) or “give back” (“We **gave** most of them **back**”).

OpenAI: Large US company responsible for GPT-3.5, GPT-4, ChatGPT etc.

POS: Part Of Speech. Generally used to designate a tag which classifies words as nouns, verbs, adjectives etc. C-LARA uses the Universal Dependencies 2 tagset, where common POS tags are NOUN, DET (determiner) and SCONJ (subordinating conjunction).

B. Glossary of languages

We briefly list the languages mentioned in this report, providing for each one basic data on the language, an indication of how it is referred to here, and a rough estimation of GPT-4's level of competence.

Drehu is a small South Oceanic language spoken by about 16,000 people in New Caledonia. It is the language of the island of Lifou. §7.3 describes initial work in this language. GPT-4 knows almost nothing about Drehu.

Dutch is a West Germanic language spoken by about 25 million people, mainly in the Netherlands and Belgium. It was used as the glossing and prompt language for the experiment in the Romanian school described in §7.2. GPT-4 has strong skills in Dutch.

English is West Germanic language. It is one of the world's two largest languages, spoken as a first or second language by over a billion people. It was used as both a text language and a glossing language in the experiments described in §5 and several of the examples presented in the appendices. English is GPT-4's strongest language.

Farsi (Persian) is an Indo-European language spoken by over a hundred million people mainly in Iran, Afghanistan and Tajikistan. It was used as a text language in the experiments described in §5. GPT-4's skills in Farsi are moderate.

Faroese is a small Scandinavian language spoken by about 50,000 people primarily in the Faroe Islands. It was used as a text language in the experiments described in §5. GPT-4's skills in Faroese are modest, and it frequently interpolates from the closest languages, Icelandic and Danish.

French is a large Romance language spoken by over three hundred million people. It was used as a glossing language in the experiments described in §5 and for the Kanak languages Drehu and Iaai described in §7.3. GPT-4 has excellent skills in French.

German is a large language spoken by about a hundred million people, primarily in Germany, Austria and Switzerland. It was used as the text language for the example *Paul und Emma* (Appendix L.2). GPT-4 has excellent skills in German.

Iaai is a small South Oceanic language spoken by about 4,000 people mainly in Ouvéa island, in New Caledonia. §7.3 describes initial work in this language. GPT-4 knows almost nothing about Iaai.

Icelandic is a small Scandinavian language spoken by about 400,000 people primarily in Iceland. A study involving Icelandic is cited in §5. GPT-4's skills in Icelandic are moderate.

Irish is a small Celtic language. It is the national language and the first official language of Ireland but only numbers about 20,000 native speakers using it as their first language. A study involving Irish is cited in §5. GPT-4's skills in Irish are weak.

Mandarin is one of the largest languages in the world, with about a billion speakers. It was used as a text language in the experiments described in §5. GPT-4 has excellent skills in Mandarin.

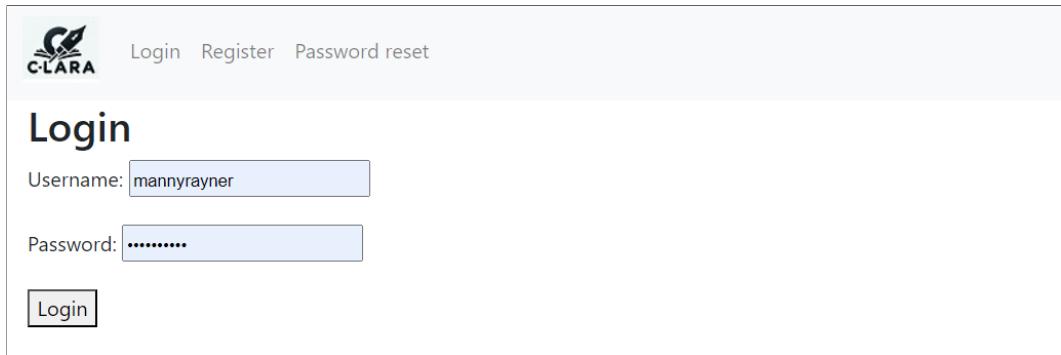
Romanian is a Romance language spoken by about 25 million people, primarily in Romania and Moldova. It was used as the text language for the experiment in the Romanian school described in §7.2. GPT-4 has strong skills in Romanian.

Russian is a large Slavic language spoken by over 250 million people, primarily in Russia and countries of the former Soviet Union. It was used as a text language in the experiments described in §5. GPT-4 has strong skills in Russian.

Swedish is a Germanic language spoken by about ten million people primarily in Sweden and Finland. It was used as a glossing language in the experiments described in §5. GPT-4 has strong skills in Swedish.

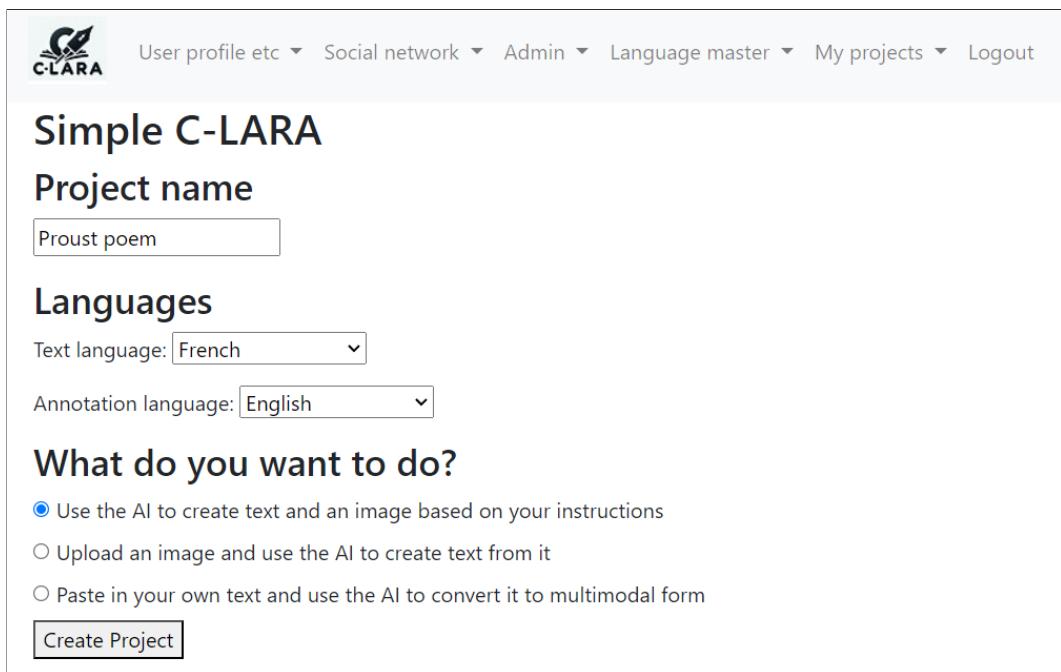
C. “Merci Proust”: using Simple C-LARA

We present a series of screenshots illustrating how to use Simple C-LARA to create a piece of content. We start by going to <https://c-lara.unisa.edu.au/accounts/login/> and logging in:



The screenshot shows the login page of the Simple C-LARA application. At the top, there is a logo with the letters 'C-LARA' and a stylized feather or pen icon. To the right of the logo are three links: 'Login', 'Register', and 'Password reset'. The main title 'Login' is centered above two input fields. The first input field is labeled 'Username:' and contains the text 'mannyrayner'. The second input field is labeled 'Password:' and contains several dots ('.....'). Below these fields is a single button labeled 'Login'.

Next, we select “Create new C-LARA project using Simple C-LARA” from the “My projects” menu. We get a screen that looks like this, and fill in the project name, the text language, i.e. the language in which the text will be written and the annotation language, i.e. the language used for the glosses. We select the first method, “Use the AI to create text and an image based on your instructions”. The others will be similar.



The screenshot shows the 'Simple C-LARA' project creation interface. At the top, there is a navigation bar with links: 'User profile etc ▾', 'Social network ▾', 'Admin ▾', 'Language master ▾', 'My projects ▾', and 'Logout'. The main title 'Simple C-LARA' is at the top left. Below it, the section 'Project name' is shown with an input field containing 'Proust poem'. The next section is 'Languages', which includes two dropdown menus: 'Text language: French' and 'Annotation language: English'. The following section is 'What do you want to do?' with three radio button options. The first option, 'Use the AI to create text and an image based on your instructions', is selected (indicated by a blue circle). The other two options are 'Upload an image and use the AI to create text from it' and 'Paste in your own text and use the AI to convert it to multimodal form'. At the bottom of this section is a 'Create Project' button.

In the next screen, we provide the prompt. Here, we tell it to write a poem celebrating Proust's

novel *A la recherche du temps perdu*. GPT-4 is a well-read AI, and we can safely assume it will be familiar with a book as famous as this one.

The screenshot shows the C-LARA web application. At the top, there is a navigation bar with links: User profile etc, Social network, Admin, Language master, My projects, and Logout. Below the navigation bar, a green banner displays the message "Project created". The main content area has a title "Simple C-LARA" and a section titled "Project name" containing a text input field with the placeholder "Proust poem" and a "Change Name" button. A section titled "Prompt" contains a text area with the instruction "Write a short poem celebrating Proust's 'A la recherche du temps perdu'." At the bottom of the main content area, there is a "Create Text and Image" button and a link "[View project in full C-LARA]".

C-LARA now sends several requests to GPT-4, and keeps us informed about what it's doing. The top-level tasks are shown on the left of the screen, and the details on the right.

The screenshot shows the C-LARA web application displaying a list of tasks. On the left, a vertical list of tasks is shown, each with a timestamp and status. On the right, detailed logs for specific tasks are displayed. The tasks listed on the left are: ENDED TASK: create plain text, STARTED TASK: create text title, ENDED TASK: create text title, and STARTED TASK: generate DALL-E-3 image. The detailed logs on the right correspond to these tasks: the first log shows a completed task with the message "... Done (\$0.00; 5.0 secs)", the second log shows a completed task with the message "... Done (\$0.00; 5.0 secs)", the third log shows a completed task with the message "... Done (\$0.00; 5.0 secs)", and the fourth log shows an ongoing task with the message "[No updates for last 5 seconds]".

When it's finished, the next screen shows us what it's done. First we get a French title and the poem itself:

Generated text title

À la Plume de Proust: Voyage au Cœur des Souvenirs Save Edited Text Title

Generated text

Dans les méandres du temps perdu,
Proust nous guide, d'une plume éperdue,
À la recherche de souvenirs enfuis,
Dans les salons, les jardins, l'esprit
embelli.

Le goût d'une madeleine trempée,
Éveille en lui le passé, si bien camouflé,
Les heures d'antan revivent, s'entrelacent,
Au fil des pages, le temps s'efface.

Save Edited Text Rewrite Text

Underneath, we can see the picture that DALL-E-3 has generated. It has been shown the text we just saw above, and told to make something appropriate to go with it. The results are often quite creative and surprising.

If we do not like the text or the image, we can regenerate them. We can also edit the text and title, or upload an image of our own.

Image



An interpretation from scenes from a garden showing the stepped fountains, intertwined fountains for the rooms, the terrace could a dipping pool, engraving recurring with colorful and gold leaves from the and having a long established hermitage, every person and the size of the area with the word by after the seven seven final time. Receiving names down here of living the pattern in the water excess @) involved the ultimate social source with a labyrinth of a ladder of one on own image.

You can try adding advice in the box below if you regenerate the image.

[Regenerate Image](#)

Uploaded image (alternative to AI-generated image)

[Choose file](#) No file chosen

[Save Uploaded Image](#)

Multimedia text

[Generate Multimedia Text](#)

[\[View project in full C-LARA\]](#)

Here, we were happy with everything, so we hit “Generate Multimedia Text” at the bottom. This invokes quite a lot of processing, as C-LARA asks GPT-4 to divide the text into pages, sentences and words, add English glosses, and mark the French words with their root forms. It also uses a TTS engine to create the audio, makes a phonetic version of the text, and puts everything together. The whole sequence takes several minutes. As before, it keeps us informed about what it is doing.

The screenshot shows the CLARA interface with a task progress bar. The tasks listed are:

- ENDED TASK: get CEFR level
- STARTED TASK: add segmentation information
- ENDED TASK: add segmentation information
- STARTED TASK: add glosses

A detailed description of the current task is shown in a scrollable box:

Passage in French, presented in a JSON format. Thank you to add English glosses in the following way. The JSON is a list of strings. Each string is either a French word or a punctuation mark. Please replace each string with a two-element list in which the first element is the original string and the second is a gloss in English. For example, if you are glossing F..."

Waiting for OpenAI response (0s elapsed)...

Waiting for OpenAI response (5s elapsed)...

When it's finished, we get a link to the generated multimedia text. We are also shown the segmented text. If we know French, we will notice that there are some mistakes here: for example, the word *esprit* has been incorrectly split up into two pieces, as *es|prit*.

Segmented text

"Dans les mé|andres du temps perdu,||
Proust nous guide, d'une plume
éperdue,||
À la recherche de souvenirs enfuis,||
Dans les salons, les jardins, l'es|prit
embelli.||

<page>Le goût d'une madeleine
trempée,||
Éveille en lui le passé, si bien camouflé,||

[Save Edited Segmented Text](#)

Multimedia text

[View Rendered Content](#)

[Regenerate Multimedia Text](#)

C-LARA Social Network

[Post to Social Network](#)

[\[View project in full C-LARA\]](#)

We correct the errors and save. Then we regenerate the multimedia text and hit the link “View Rendered Content”.

Segmented text

Dans les méandres du temps perdu,||
Proust nous guide, d'une plume
éperdue,||
À la recherche de souvenirs enfouis,||
Dans les salons, les jardins, l'esprit
embelli.||

<page>Le goût d'une madeleine
trempée,||
Éveille en lui le passé, si bien camouflé,||

[Save Edited Segmented Text](#)

Multimedia text

[View Rendered Content](#)

[Regenerate Multimedia Text](#)

C-LARA Social Network

[Post to Social Network](#)

[\[View project in full C-LARA\]](#)

The splash page looks like this, with the generated image and title.



An interventiomen from scenes from a gardens echiriting the stipped the soirtt,
interrivm soure fits the popes, the the tate eoud a dippagededenau, erainv rectuining
with condrived idd hidrs fersco and hliving awaking aikadkiod hrouuch pages
every domed and the tso of the senses wh the word be elber the se anson fersdall time-
Reirng nense dourney of living the partestun in the waleef amoses idd icriboch
the ultamate social iounce withna labormth of a labbthy of our ouw own. mlage

"À la Plume de Proust: Voyage au Cœur des Souvenirs"

The next page contains the first verse. Segmentation has split it up into lines, and we can hear a line by clicking on a speaker icon. We can get a gloss by hovering over a word, or listen to it by clicking.

"À la Plume de Proust: Voyage au Cœur des Souvenirs" p. 2

First Previous Next Last Frequency Alphabetical Sounds

Dans les méandres du temps per **feverish**
Proust nous guide, d'une plume **éperdue**,

À la recherche de souvenirs enfouis,

Dans les salons, les jardins, l'esprit embellie.

If we like the text enough to want to share it, we can use the control at the bottom to post it to the C-LARA social network. This gives us a link to the relevant social network page.

Multimedia text

[View Rendered Content](#)

[Regenerate Multimedia Text](#)

C-LARA Social Network

[View Social Network Page](#)

[\[View project in full C-LARA\]](#)

The social network page looks like this. Visitors can add a rating or leave comments.



Proust poem

URL: /accounts/rendered_texts/180/normal/page_1.html

No ratings yet

Rating: ▼

Text language: French

Annotation language: English

Created by: [mannyrayner](#)

Summary: Marcel Proust takes us through a journey of rediscovered memories, ignited by the taste of a madeleine, within a rich tapestry of social and personal narratives, ultimately leading to a final voyage of self-reflection and reconnection with the past.

Length in words: 113

Author: mannyrayner

Voice: google_fr-FR_fr-FR-Studio-D (segments), google_fr-FR_default (words)

Difficulty Level: C1

D. “Paul und Emma”: using Advanced C-LARA

We now show how to use Advanced C-LARA to create a more ambitious text, a German picture-book with an image on every page. We log in as before, but this time we select “Create new C-LARA project using Advanced C-LARA” from the My Projects menu. We fill in the title, text language and annotation language and hit “Create Project”.

User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Create a new project

Title:

Text language:

Annotation language:

[Create Project](#)

We now see this screen, which shows the project actions we can perform. At the beginning, everything is greyed out except “Create/Edit Plain Text”, so we click that.

User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

The Penguin and the Polar Bear

New title:

[Update Title](#)

Text Language: German

Annotation Language: English

Total API cost for this project: \$0.00

[Create/Edit Plain Text](#)

[Create/Edit Text Title](#)

[Create/Edit Summary](#)

[Estimate/Edit CEFR Level](#)

[Create/Edit Segmented Text](#)

[Create/Edit Segmented Title](#)

[Add/Remove Images](#)

We get this screen, and enter a prompt describing the text we want to create.



User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Create Plain Text ("The Penguin and the Polar Bear")

Text choice:

- Generate text using AI
- Improve existing text using AI
- Manually enter/edit text
- Load archived version

Archived version:

Write the text of a fanciful but responsible picture-book for small children, about a penguin and a polar bear who get together over the internet and start coordinating action against global warming.

Prompt:

Current version:

C-LARA invokes GPT-4 to create the text, and shows it to us.

User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Created plain text

Create Plain Text ("The Penguin and the Polar Bear")

Text choice:

- Generate text using AI
- Improve existing text using AI
- Manually enter/edit text
- Load archived version

Archived version: [ai_generated February 18 2024 04:42:43](#)

Write the text of a fanciful but responsible picture-book for small children, about a penguin and a polar bear who get together over the internet and start coordinating action against global warming.

Prompt:

Current version: [ai_generated February 18 2024 04:42:43](#)

Es waren einmal ein kleiner Pinguin namens Paul und ein großer Eisbär namals Emma, die weit voneinander entfernt lebten. Paul wohnte in der Antarktis und Emma weit im Norden am Polarkreis. Eines Tages trafen sie sich im Internet auf einer Webseite für Umweltschutz.

Paul tippte mit seinen kleinen Flossen eine Nachricht:
"Hey Emma, ich mache mir Sorgen um unser Zuhause. Das Eis schmilzt."

Emma las die Nachricht und fühlte sich genauso. Mit ihren großen Tatzen tippte sie zurück:
"Das stimmt, Paul. Wir müssen etwas dagegen tun!"

So beschlossen die beiden sich zu verbünden und gemeinsam gegen die Erderwärmung zu kämpfen.

We hit the link at the bottom (not show) to take us back to the project overview screen. Now we have more options.

 User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

The Penguin and the Polar Bear

New title:

[Update Title](#)

Text Language: German

Annotation Language: English

Total API cost for this project: \$0.02

[Create/Edit Plain Text ✓](#)

[Create/Edit Text Title](#)

[Create/Edit Summary](#)

[Estimate/Edit CEFR Level](#)

[Create/Edit Segmented Text](#)

[Create/Edit Segmented Title](#)

We hit “Create/Edit Text Title” to get this screen:

 User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Create Text Title ("The Penguin and the Polar Bear")

Text choice:

Generate title using AI
 Improve existing title using AI
 Manually enter/edit title
 Load archived version

Archived version:

Current version:

Es waren einmal ein kleiner Pinguin namens Paul und ein großer Eisbär namals Emma, die weit voneinander entfernt lebten. Paul wohnte in der Antarktis und Emma weit im Norden am Polarkreis. Eines Tages trafen sie sich im Internet auf einer Webseite für Umweltschutz.

Paul tippte mit seinen kleinen Flossen eine Nachricht:
"Hey Emma, ich mache mir Sorgen um unser Zuhause. Das Eis schmilzt."

Emma las die Nachricht und fühlte sich genauso. Mit ihren großen Tatzen tippte sie zurück:
"Das stimmt, Paul. Wir müssen etwas dagegen tun!"

After we have invoked AI-based processing, we get the title:

The screenshot shows a user interface for creating a text title. At the top, there is a navigation bar with links: User profile etc, Social network, Admin, Language master, My projects, and Logout. Below the navigation bar, a green header bar says "Created title text". The main content area has a title "Create Text Title ("The Penguin and the Polar Bear")". Underneath it, there is a section titled "Text choice:" with five options: "Generate title using AI", "Improve existing title using AI", "Manually enter/edit title" (which is selected), and "Load archived version". Below this, there are two dropdown menus: "Archived version" set to "ai_generated February 18 2024 04:55:33" and "Current version" also set to "ai_generated February 18 2024 04:55:33". A text input field contains the generated title: "Die Abenteuer von Paul und Emma: Zwei mutige Freunde kämpfen für den Umweltschutz".

We back to the overview, and now visit “Create/Edit Segmented Text”. The result looks like this:

Created segmented text

Create Segmented Text ("The Penguin and the Polar Bear")

Text choice:

- Segment text using AI
- Try to fix errors in malformed segmented text using AI
- Improve existing segmented text using AI
- Manually enter/edit segmented text
- Load archived version

Archived version: [ai_generated February 18 2024 04:58:27](#) ▾

Current version: [ai_generated February 18 2024 04:58:27](#)

Es waren einmal ein kleiner Pinguin namens Paul und ein großer Eisbär namals Emma,|| die weit vonleinander entfernt lebten.|| Paul wohnte in der Antarktis und Emma weit im Norden am Polarkreis.|| Eines Tages trafen sie sich im Internet auf einer Webseite für Umweltschutz.||

Paul tippte mit seinen kleinen Flossen eine Nachricht: ||
"Hey Emma, ich mache mir Sorgen um unser Zuhause.|| Das Eis schmilzt."||

Emma las die Nachricht und fühlte sich genauso.|| Mit ihren großen Tatzen tippte sie zurück: ||
"Das stimmt, Paul.|| Wir müssen etwas dagegen tun!"||

We decide we want to make some adjustments to the way the AI has done the page divisions, so we edit and save:

Create Segmented Text ("The Penguin and the Polar Bear")

Text choice:

- Segment text using AI
- Try to fix errors in malformed segmented text using AI
- Improve existing segmented text using AI
- Manually enter/edit segmented text
- Load archived version

Archived version: ai_generated February 18 2024 04:58:27 ▾

Current version: ai_generated February 18 2024 04:58:27

Es waren einmal ein kleiner Pinguin namens Paul und ein großer Eisbär namens Emma,|| die weit vonleidender entfernt lebten.|| Paul wohnte in der Antarktis und Emma weit im Norden am Polarkreis.|| Eines Tages trafen sie sich im Internet auf einer Webseite für Umweltschutz.||

<page>Paul tippte mit seinen kleinen Flossen eine Nachricht: ||
"Hey Emma, ich mache mir Sorgen um unser Zuhause.|| Das Eis schmilzt."||

<page>Emma las die Nachricht und fühlte sich genauso.|| Mit ihren großen Tatzen tippte sie zurück: ||
"Das stimmt, Paul.|| Wir müssen etwas dagegen tun!"||

<page>So beschlossen die beiden, sich zu verbünden und gemeinsam gegen die Erdlerwärmung zu kämpfen.|| Sie riefen all ihre Freunde zusammen – die Robben, die Eisfuchse, die Wale und sogar die Pinguine.|| Alles war bereit für den Kampf.||

Going back to the overview, we see that again we have more options:



User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

The Penguin and the Polar Bear

New title:

[Update Title](#)

Text Language: German

Annotation Language: English

Total API cost for this project: \$0.08

[Create/Edit Plain Text](#) ✓

[Create/Edit Text Title](#) ✓

[Create/Edit Summary](#) ✓

[Estimate/Edit CEFR Level](#) ✓

[Create/Edit Segmented Text](#) ✓

[Create/Edit Segmented Title](#) ✓

[Add/Remove Images](#)

[Create/Edit Phonetic Text](#)

[Create/Edit Glossed Text](#)

We go into “Add/Remove Images”, where we are going to add an image to each page. The screen is as below.

At this point, I started a separate ChatGPT-4 thread and spent several hours discussing how it would make the images. This was an interesting task involving both AI and human creativity! You can see the start of the discussion in Appendix L.2. When we had finished, I downloaded all the images we were going to use, then uploaded them to the C-LARA project. The “Add/Remove Images” currently allows the user to upload an image and place it either at the top or the bottom of a numbered text page.

User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Edit Images ("The Penguin and the Polar Bear")

Create a new DALL-E-3 image based on the whole text

[Create](#)

Upload/edit/delete existing images

Image Preview	Image Upload	Image Name	Page	Position	Delete
	<input type="button" value="Choose file"/> No file chosen	Cover	1	Top	<input type="checkbox"/>
	<input type="button" value="Choose file"/> No file chosen	EsWarEinmal	2	Top	<input type="checkbox"/>
	<input type="button" value="Choose file"/> No file chosen	EinesTagesTrafenSie	3	Top	<input type="checkbox"/>

Next, we go back to the overview page, select “Create/Edit Glossed Text”, and invoke AI-based processing. You can see the output below. In the glossed version, each word is followed by an English gloss enclosed in hashes. As usual, we can edit this if we like.

Create Glossed Text ("The Penguin and the Polar Bear")

Text choice:

- Generate annotated text from segmented text using AI
- Try to fix errors in malformed annotated text using AI
- Improve existing annotated text using AI
- Manually enter annotated text
- Load archived version

Archived version: ai_generated February 18 2024 22:46:12 ▾

Current version: ai_generated February 18 2024 22:46:12

```
<page>
<h1>||Die#The# Abenteuer#adventures# von#of# Paul#Paul# und#and# Emma#Emma#: Zwei#Two#
mutige#brave# Freunde#friends# kämpfen#fight# für#for# den#the#
Umwelt#environmental#schutz#protection#||</h1>
<page>

Es#It# waren#were# einmal#once# ein#a# kleiner#small# Pinguin#penguin# namens#named#
Paul#Paul# und#and# ein#a# großer#large# Eisbär#polar bear# namals#named# Emma#Emma#,
die#who# weit#far# von#from#einander#each other# entfernt#lived apart# lebten#lived#.|| Paul#Paul#
wohnte#lived# in#in# der#the# Antarktis#Antarctica# und#and# Emma#Emma# weit#far# im#in the#
Norden#North# am#at the# Polarkreis#Arctic Circle#.||
```

In the same way, we go to “Create/Edit Lemma Tagged Text” and tell the AI to add root form and part of speech tags. We have the same kind of format as with the glosses. If you know some German, this should make good sense: for example, the word *Freunde* (“friends”) is tagged as a NOUN with root form *Freund* (“friend”).

The screenshot shows the C-LARA web application interface. At the top, there is a navigation bar with links: User profile etc, Social network, Admin, Language master, My projects, and Logout. Below the navigation bar, a green header bar says "Created lemma text". The main content area has a title "Create Lemma Tagged Text ("The Penguin and the Polar Bear")". Underneath the title, it says "Text choice:" followed by a list of options:

- Generate annotated text from segmented text using AI
- Try to fix errors in malformed annotated text using AI
- Improve existing annotated text using AI
- Generate annotated text from segmented text with trivial tags
- Manually enter annotated text
- Load archived version

Below this is a dropdown menu labeled "Archived version: ai_generated February 18 2024 22:55:41 ▾". Underneath is another dropdown menu labeled "Current version: ai_generated February 18 2024 22:55:41".

At the bottom, there is a code editor window containing the following XML-like text:

```
<page>
<h1>||Die#die/DET# Abenteuer#Abenteuer/NOUN# von#von/ADP# Paul#Paul/PROPN#
und#und/CCONJ# Emma#Emma/PROPN#: Zwei#zwei/NUM# mutige#mutig/ADJ#
Freunde#Freund/NOUN# kämpfen#kämpfen/VERB# für#für/ADP# den#der/DET#
Umwelt#Umwelt/NOUN#schutz#Schutz/NOUN#||</h1>
<page>
```

We go to “Create/Edit Phonetic Text” and tell C-LARA to make that too. The output is not easy to read, but you can see that the text has been broken up into letter-groups paired with phonetic symbols. This functionality is only available for the fairly small number of languages where we have installed phonetic lexica.


User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Create Phonetic Text ("The Penguin and the Polar Bear")

Text choice:

- Generate phonetic text from segmented text
- Manually edit/enter phonetic text
- Load archived version

Archived version: ai_generated February 18 2024 22:42:53

Current version: ai_generated February 18 2024 22:42:53

```

<page>
<h1>||D#d#|ie#i:#|| ||A#a.#|b#b#|en#n#|t#t#|eu#ɔɪ#|er#e#|| ||v#f#|o#ɔ#|n#n#|| ||P#p#|au#əʊ#|#|#||#
||u#u||n#n#|d#t#|| ||E#ɛ#|mm#m#a#a||:||Z#ts#|w#v#|ei#əɪ#|| |m#m#|u#u:#|t#t#|j#i#g#g#|e#ə#|||
||f#f#|#ɛ#|eu#ɔɪ#|n#n#|d#d#|e#ə#|| ||k#k#|ä#ɛ#|m#m#|pf#|p̄f#|e#m#|n#n#|| ||f#f#|ü#y:#|r#e#|||
||d#d#|e#e:#|n#n#|| ||U#u#|m#m#|w#v#|e#ɛ#||#|#|t#t#|sch#|u#u#|tz̄#|ts#||</h1>
<page>

||E#ɛ#|s#s#|| ||w#v#|a#a.#|r#u#|e#ə#|n#n#|| ||ei#əɪ#|n#n#|m#m#|a#a:#|#|#|| ||ei#əɪ#|n#n#|||
||k#k#|l#l#|ei#əɪ#|n#n#|er#e#|| ||P#p#|i#i#|n#n#|g#g#|u#u#|i:#|n#n#|||
||n#n#|a#a.#|m#m#|e#ə#|n#n#|s#s#|| ||P#p#|au#əʊ#|#|#|| ||u#u#|n#n#|d#t#|| ||ei#əɪ#|n#n#|||
||g#g#|r#b#|o#o:#|β#s#|er#e#|| ||E#əɪ#|s#s#|b#b#|ä#ɛ#|r#e#|| ||n#n#|a#a:#|m#m#|a#a:#|#|#|s#s#|||
||E#ɛ#|mm#m#a#a||,||d#d#|ie#i:#|| ||w#v#|e#əɪ#|t#t#||

||y#f#|o#ɔ#|n#n#|| ||ei#əɪ#|n#n#|a#a:#|n#n#|p#l#|er#p#|| ||e#s#|l#p#n#l#t#t#|f#f#|e#ɛ#|r#u#|n#p#l#t#t#||


```

We go to “Set Format Preferences”. Since it is a children’s picture book, a large font seems appropriate.



User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Format preferences updated successfully.

Set Format Preferences for "The Penguin and the Polar Bear"

Font type:

Font size:

Text align:

Concordance font type:

Concordance font size:

Concordance text align:

We have done all the necessary steps, so now the overview screen shows “Render Text” and “Render Phonetic Text” as options.

The screenshot shows the C-LARA project overview page for the project "The Penguin and the Polar Bear".

Project Title: The Penguin and the Polar Bear

New title:

Update Title

Text Language: German

Annotation Language: English

Total API cost for this project: \$0.47

Available actions:

- Create/Edit Plain Text ✓
- Create/Edit Text Title ✓
- Create/Edit Summary ✓
- Estimate/Edit CEFR Level ✓
- Create/Edit Segmented Text ✓
- Create/Edit Segmented Title ✓
- Add/Remove Images ✓
- Create/Edit Phonetic Text ✓
- Create/Edit Glossed Text ✓
- Create/Edit Lemma Tagged Text ✓
- Create/Edit Pinyin Tagged Text
- Audio Processing ✓
- Audio Processing (Phonetic) ✓
- Set Format Preferences ✓
- Render Text
- Render Phonetic Text
- Register Rendered Text
- Register Rendered Phonetic Text

When we invoke rendering, we can among other things see C-LARA creating mp3 audio using an online TTS engine.

 User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Rendering Text ("The Penguin and the Polar Bear")

--- Creating mp3 for 'müssen' (51/181)
--- Creating mp3 for 'etwas' (52/181)
--- Creating mp3 for 'dagegen' (53/181)
--- Creating mp3 for 'tun' (54/181)
--- Creating mp3 for 'beschlossen' (55/181)

When rendering is done, we get a link to the multimodal text. The splash page looks like this:

"Die Abenteuer von Paul und Emma: Zwei mutige Freunde kämpfen für den Umweltschutz" p. 1

First Previous Next Last Frequency Alphabetical Sounds



**Die Abenteuer von Paul und Emma: Zwei
mutige protection mpfen für den
Umweltschutz** 

There is a link to register the multimedia text in the social network:



User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Rendered text found

Rendered Text ("The Penguin and the Polar Bear")

[View Content](#)

Register as content:

[Register as Content](#)

[Back to Project](#)

If we hit this, we will make our text generally visible.

E. Using human-recorded audio

In most C-LARA projects, we create audio using a TTS engine. Sometimes, however, we want to use human-recorded audio instead. One common reason for doing this is that the language we're working with doesn't have an associated TTS engine. Another is that we have special audio we wish to use, perhaps singing or professional-quality reading.

C-LARA currently offers two ways to include human-recorded audio, which are controlled through the “Audio processing” project screen. In the “Upload” method, C-LARA provides a tabular view showing all the pieces of text that need audio, and the user uploads them one by one. This is simple but slow. The other method, “Manual audio alignment”, gives a more efficient solution for creating segment audio when there is a single mp3 containing audio for the whole text. The user needs to install the Audacity audio editor, and uses it to create a file labelling the segment breaks. C-LARA can then automatically cut up the mp3 into segment audio which it installs. We give an example of each method.

E.1. “Upload” method: Drehu alphabet book

For the “Upload” method, we provide an example using the Drehu alphabet book presented in §7.3.1. The screenshot below shows how we have specified that we will be using human-recorded audio for both words and segments. The table consequently gives one line for each word and segment. We have specified that we will not use context, since this concept is meaningless in an alphabet book.

Here, we see the end of the process, where all the files have already been uploaded. Files can if necessary be replaced in the same way.

User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Audio Processing ("Un abécédaire en drehu")

Method:

Preferred TTS engine:

Preferred TTS voice:

Use human audio for words:

Use human audio for segments:

Use context for segment audio:

Human Voice ID:

Audio	Text	Upload File
<input type="button" value="▶"/> 0:00 / 0:00 <input type="button" value=" "/>	<input type="text" value="alamek"/> <input type="button" value="Choose file"/>	No file chosen
<input type="button" value="▶"/> 0:00 / 0:00 <input type="button" value=" "/>	<input type="text" value="wahnawa"/> <input type="button" value="Choose file"/>	No file chosen
<input type="button" value="▶"/> 0:00 / 0:00 <input type="button" value=" "/>	<input type="text" value="laam"/> <input type="button" value="Choose file"/>	No file chosen
<input type="button" value="▶"/> 0:00 / 0:00 <input type="button" value=" "/>	<input type="text" value="waan"/> <input type="button" value="Choose file"/>	No file chosen
<input type="button" value="▶"/> 0:00 / 0:00 <input type="button" value=" "/>	<input type="text" value=""/>	<input type="button" value="Choose file"/>

E.2. “Manual align” method: These Boots Were Made for Walking

For the “Manual align” method, our example is organised around an mp3 of the classic Nancy Sinatra hit “These Boots Were Made for Walking”. The first screenshot shows how we set up the task. Note that here we need to use contexts, since we must distinguish the different occurrences of the title line. C-LARA shows us the text with labelled segment breaks.

Method: Manual Align

Preferred TTS engine: None

Preferred TTS voice: None

Use human audio for words:

Use human audio for segments:

Use context for segment audio:

Human Voice ID: NancySinatra

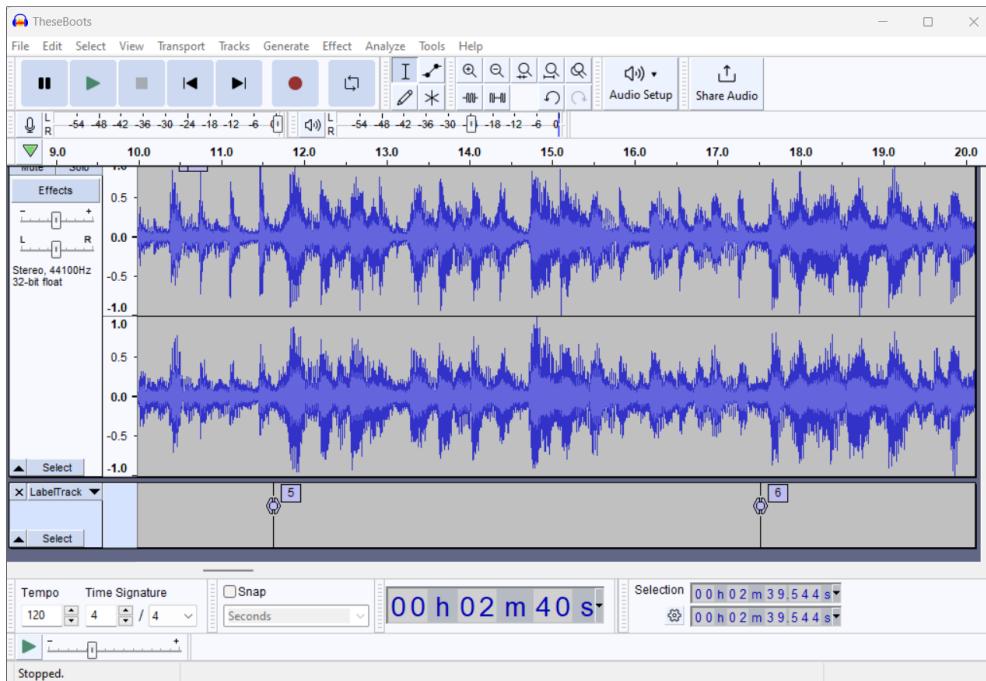
Segmented text with labels to use in Audacity:

```
|0|<h1>|1|These Boots Were Made for Walkin'|2|</h1><page>
|3|
|4|
(Intro)|5|
You keep sayin' you've got somethin' for me|6|
Somethin' you call love but confess|7|
You've been a'messin' where you shouldn't 've been a'messin'|8|
And now someone else is getting all your best|9|
```

We next show the step where we add the numbered breakpoints to the label track at the bottom. In order to set this up, we need to do the following:

- If necessary, download Audacity²⁸ and install it.
- Open Audacity. Load the mp3 using the command File > Import > Audio.
- Add a label track using the command Tracks > Add New > Label track.
- To add an item in the label track, find the right point by listening to the audio and hit Ctrl-B to create a label.
- When the label track is complete, download it using the command File > Export Other > Export labels.

²⁸<https://www.audacityteam.org/download/>



Finally, we upload the mp3 and label track files, and press “Submit”. C-LARA completes the rest of the processing automatically.

Segmented text with labels to use in Audacity:

```
|0|<h1>|1|These Boots Were Made for Walkin'|2|</h1><page>
|3|
|4|
(Intro)|5|
You keep sayin' you've got somethin' for me|6|
Somethin' you call love but confess|7|
You've been a'messin' where you shouldn't 've been a'messin'|8|
And now someone else is getting all your best|9|
These boots are made for walkin'|10|
And that's just what they'll do|11|
One of these days these boots are gonna walk all over you|12|
Ya|13|
```

Upload Original Audio File: TheseBoots.mp3 [/tmp/tmpkw5lmkzs.mp3]

Upload Audacity Label File: LabelTrack.txt [/tmp/tmp6ojylwdr.txt]

[Back to Project](#)

F. Creating phonetic texts

We have mentioned “phonetic” texts a couple of times, e.g. §4.4, Appendix D. Phonetic texts can be created for a language where suitable phonetic resources exist. Here, we briefly sketch how these resources can be added and manipulated; the relevant functionality is accessible to users who have “Language Master” privileges for the language in question. There are two different versions. The more common case is a language with complex/inconsistent grapheme-phoneme correspondences, necessitating use of a phonetic lexicon. We also support languages with consistent grapheme-phoneme correspondences, where we can more simply use a table.

F.1. Languages using a phonetic lexicon

We illustrate the first case, languages using a phonetic lexicon, with English. We start by entering the Edit Phonetic Lexicon screen, which initially looks like this:

User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Edit Phonetic Lexicon

Language: Default

Encoding Type: IPA

Grapheme to phoneme entries exist:

Plain phonetic lexicon entries exist:

Plain phonetic lexicon file (txt or JSON): No file chosen

Aligned phonetic lexicon entries exist:

Aligned phonetic lexicon file (JSON): No file chosen

Display grapheme to phoneme entries:

Display new plain lexicon entries:

Display approved plain lexicon entries:

Display new aligned lexicon entries:

Display approved aligned lexicon entries:

We select “English” as the language and hit Refresh. The screen now looks like this:

 User profile etc ▾ Social network ▾ Admin ▾ Language master ▾ My projects ▾ Logout

Current data for english loaded

Edit Phonetic Lexicon

Language: English ▾

Encoding Type: IPA ▾

Grapheme to phoneme entries exist: NO

Plain phonetic lexicon entries exist: YES

Plain phonetic lexicon file (txt or JSON): No file chosen

Aligned phonetic lexicon entries exist: YES

Aligned phonetic lexicon file (JSON): No file chosen

Display grapheme to phoneme entries:

Display new plain lexicon entries:

Display approved plain lexicon entries:

Display new aligned lexicon entries:

Display approved aligned lexicon entries:

The updated screen tells us that we have both plain phonetic lexicon entries, guessed by GPT-4, and aligned lexicon entries, guessed by the alignment process. We first examine the new (unreviewed) plain phonetic lexicon entries:

Display new plain lexicon entries:

Display approved plain lexicon entries:

Display new aligned lexicon entries:

Display approved aligned lexicon entries:

Plain phonetic lexicon

Word	Phonemes	Approve	Delete
santa	sæntə	<input type="checkbox"/>	<input type="checkbox"/>
lives	laɪvz	<input type="checkbox"/>	<input type="checkbox"/>
cozy	kouzi	<input type="checkbox"/>	<input type="checkbox"/>
fred	fred	<input type="checkbox"/>	<input type="checkbox"/>
read	rɪ:d	<input type="checkbox"/>	<input type="checkbox"/>
walkin'	walkɪn'	<input type="checkbox"/>	<input type="checkbox"/>
restore	rɪ:stɔ:r	<input type="checkbox"/>	<input type="checkbox"/>
repent	rɪ:pənt	<input type="checkbox"/>	<input type="checkbox"/>
cunning'st	kʌnɪŋz't	<input type="checkbox"/>	<input type="checkbox"/>
promethean	prəmɪθi:ən	<input type="checkbox"/>	<input type="checkbox"/>
relume	rɪ:lju:m	<input type="checkbox"/>	<input type="checkbox"/>
pluck'd	plʌkt	<input type="checkbox"/>	<input type="checkbox"/>
rose	roʊz	<input type="checkbox"/>	<input type="checkbox"/>

Looking at these, most of them seem correct, but the entry for “cozy” may be incorrect. We mark “Approve” and “Delete” appropriately and press Save.

Display new plain lexicon entries:

Display approved plain lexicon entries:

Display new aligned lexicon entries:

Display approved aligned lexicon entries:

Plain phonetic lexicon

Word	Phonemes	Approve	Delete
santa	sæntə	<input checked="" type="checkbox"/>	<input type="checkbox"/>
lives	laɪvz	<input checked="" type="checkbox"/>	<input type="checkbox"/>
cozy	kouzi	<input checked="" type="checkbox"/>	<input type="checkbox"/>
fred	fred	<input checked="" type="checkbox"/>	<input type="checkbox"/>
read	rɪd	<input checked="" type="checkbox"/>	<input type="checkbox"/>
walkin'	wɔ:kɪn	<input checked="" type="checkbox"/>	<input type="checkbox"/>
restore	rɪ:stɔ:r	<input checked="" type="checkbox"/>	<input type="checkbox"/>
repent	rɪ:pənt	<input checked="" type="checkbox"/>	<input type="checkbox"/>
cunning'st	kʌnɪŋzt	<input checked="" type="checkbox"/>	<input type="checkbox"/>
promethean	promiθi:ən	<input checked="" type="checkbox"/>	<input type="checkbox"/>
relume	rɪ:lum	<input checked="" type="checkbox"/>	<input type="checkbox"/>
pluck'd	plʌkt	<input checked="" type="checkbox"/>	<input type="checkbox"/>
rose	zəʊər	<input type="checkbox"/>	<input checked="" type="checkbox"/>
+	+	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Similarly, we can display guessed aligned entries:

Aligned phonetic lexicon

Word	Phonemes	Aligned graphemes	Aligned phonemes	Approve	Delete
midst	əmidst	a m i d s t	ə m i d s t	<input type="checkbox"/>	<input type="checkbox"/>
social	səʊʃəl	s o c i a l	s əʊʃ ʃ əl	<input type="checkbox"/>	<input type="checkbox"/>
political	pəlɪtɪkəl	p o i t j i c a l	p əl i t j i k əl	<input type="checkbox"/>	<input type="checkbox"/>
divides	dɪvaɪdz	d j i v i d e s	d j i v a r d z	<input type="checkbox"/>	<input type="checkbox"/>
serves	sɜ:vz	s er ve s	s ɜ: v z	<input type="checkbox"/>	<input type="checkbox"/>
reminder	rɪ'maɪndər	r e m i n d er	r i m a r n d ər	<input type="checkbox"/>	<input type="checkbox"/>
mutual	mju:tʃu:əl	m u t u a l	m ju:tʃu:əl əl	<input type="checkbox"/>	<input type="checkbox"/>
provides	prə'veɪz	p r o v i d e s	p r ə v a r d z	<input type="checkbox"/>	<input type="checkbox"/>
framework	fri'eu:mwɜ:k	f r a m e w or k	f r ə m w ɔ:r k	<input type="checkbox"/>	<input type="checkbox"/>
discourages	dɪ'skɔ:gɪz	d i s c o u r a g e s	d i s k ʌ r a g z ɪz	<input type="checkbox"/>	<input type="checkbox"/>
prejudice	pɹe'dʒu:dɪs	p r e j u d j c e	p r e d ʒ u: d j s	<input type="checkbox"/>	<input type="checkbox"/>
inequality	ɪn:i:kwɒ:lɪti	i n e qu a l i t ly	i n i: kw ɒ l i t j i	<input type="checkbox"/>	<input type="checkbox"/>
discrimination	dɪ'skrɪmɪneɪʃən	d i s c r i m i n a tion	d i s k ʌ r i m i n e ɪ ʃən	<input type="checkbox"/>	<input type="checkbox"/>
rampant	ræmpənt	r a m p a n t	r æ m p ə n t	<input type="checkbox"/>	<input type="checkbox"/>
societies	sə'ke:tɪes	s ə k e t ɪ z	s ə k e t ɪ z	<input type="checkbox"/>	<input type="checkbox"/>

Here, several entries look dubious, so we edit them before approving and hitting Save.

Aligned phonetic lexicon

Word	Phonemes	Aligned graphemes	Aligned phonemes	Approve	Delete
midst	əmidst	a m i d s t	ə m i d s t	<input checked="" type="checkbox"/>	<input type="checkbox"/>
social	səʊʃəl	s o c i a l	s əʊʃ ʃ əl	<input checked="" type="checkbox"/>	<input type="checkbox"/>
political	pəlɪtɪkəl	p o i t j i c a l	p əl i t j i k əl	<input checked="" type="checkbox"/>	<input type="checkbox"/>
divides	dɪvaɪdz	d j i v i d e s	d j i v a r d z	<input checked="" type="checkbox"/>	<input type="checkbox"/>
serves	sɜ:vz	s er ve s	s ɜ: v z	<input checked="" type="checkbox"/>	<input type="checkbox"/>
reminder	rɪ'maɪndər	r e m i n d er	r i m a r n d ər	<input checked="" type="checkbox"/>	<input type="checkbox"/>
mutual	mju:tʃu:əl	m u t u a l	m ju:tʃu:əl əl	<input checked="" type="checkbox"/>	<input type="checkbox"/>
provides	prə'veɪz	p r o v i d e s	p r ə v a r d z	<input checked="" type="checkbox"/>	<input type="checkbox"/>
framework	fri'eu:mwɜ:k	f r a m e w or k	f r ə m w ɔ:r k	<input checked="" type="checkbox"/>	<input type="checkbox"/>
discourages	dɪ'skɔ:gɪz	d i s c o u r a g e s	d i s k ʌ r a g z ɪz	<input checked="" type="checkbox"/>	<input type="checkbox"/>
prejudice	pɹe'dʒu:dɪs	p r e j u d j c e	p r e d ʒ u: d j s	<input checked="" type="checkbox"/>	<input type="checkbox"/>
inequality	ɪn:i:kwɒ:lɪti	i n e qu a l i t ly	i n i: kw ɒ l i t j i	<input checked="" type="checkbox"/>	<input type="checkbox"/>
discrimination	dɪ'skrɪmɪneɪʃən	d i s c r i m i n a tion	d i s k ʌ r i m i n e ɪ ʃən	<input checked="" type="checkbox"/>	<input type="checkbox"/>
rampant	ræmpənt	r a m p a n t	r æ m p ə n t	<input checked="" type="checkbox"/>	<input type="checkbox"/>
societies	sə'ke:tɪes	s ə k e t ɪ z	s ə k e t ɪ z	<input checked="" type="checkbox"/>	<input type="checkbox"/>

F.2. Languages using a grapheme/phoneme table

In the second case, we have consistent grapheme/phoneme correspondences which we can list in a conversion table. This occurs most often with small languages that have only recently acquired a written form. We illustrate with the New Caledonian language Drehu (see §7.3.1). This time, when we enter Language master > Edit phonetic lexicon and select Drehu, we get the following:

User profile etc ▾ Social network ▾ Admin ▾ Funding reviewer ▾ Language master ▾ My projects ▾ Logout

Current data for drehu loaded

Edit Phonetic Lexicon

Language:

Encoding Type:

Grapheme to phoneme entries exist:

Plain phonetic lexicon entries exist:

Plain phonetic lexicon file (txt or JSON): No file chosen

Aligned phonetic lexicon entries exist:

Aligned phonetic lexicon file (JSON): No file chosen

Display grapheme to phoneme entries:

Display new plain lexicon entries:

Display approved plain lexicon entries:

Display new aligned lexicon entries:

Display approved aligned lexicon entries:

We see that it is not quite true that Drehu's grapheme/phoneme correspondences are completely consistent: we do in fact have a small phonetic lexicon, which lists a handful of exceptions. The vast majority of words however follow the regular pattern captured by the table, the top of which looks like this:

Display grapheme to phoneme entries:

Display new plain lexicon entries:

Display approved plain lexicon entries:

Display new aligned lexicon entries:

Display approved aligned lexicon entries:

Grapheme-to-phoneme entries

Grapheme variants Phonemes

Grapheme variants	Phonemes
hng	ng
ng	ng
hny	gn
hl	l
hm	m
hn	n
dr	dj
dj	dj
th	TH
j	TH
tr	tch

We can edit and save the table in the obvious way.

F.3. Audio for phonetic texts

If possible, we would like to associate audio with the grapheme units in a phonetic text, so that we play it when we click on them. For many languages, including all those where we currently

have a phonetic lexicon installed, we create the audio automatically using ipa-reader²⁹.

In other languages, we need to record and upload the audio manually. This is currently done on a per-project basis, with the phonetic audio stored in a common language-specific repository. Again illustrating with Drehu, when we access Audio Processing (phonetic) for the text “Un abécédaire interactif en drehu”, we see a screen similar to the one we use for uploading word audio (§E.1):

Audio Processing, phonetic ("Un abécédaire en drehu")

Method:

Use human audio for letter groups:

Use human audio for words:

Voice Talent ID:

Audio	Text	Upload File
▶ 0:00 / 0:00	a	<input type="button" value="Choose file"/> No file chosen
▶ 0:00 / 0:00	l	<input type="button" value="Choose file"/> No file chosen
▶ 0:00 / 0:00	m	<input type="button" value="Choose file"/> No file chosen
▶ 0:00 / 0:00	é	<input type="button" value="Choose file"/> No file chosen
▶ 0:00 / 0:00	k	<input type="button" value="Choose file"/> No file chosen
▶ 0:00 / 0:00	[REDACTED]	<input type="button" value="Choose file"/> No file chosen

As before, the table shows the grapheme-groups that occur in the given text.

²⁹<http://ipa-reader.xyz/>

G. Creating annotation prompt templates and examples

We briefly illustrate how we create prompt templates and examples for a language that does not yet have them, using German as an example. We start by going to Language master > Edit annotation prompts, which requires Language Master permissions.³⁰ We see a screen which looks like this:

The screenshot shows a user interface titled "Edit Annotation Prompts". It contains several dropdown menus and buttons. At the top is a "Language" dropdown set to "Default". Below it is a "Default language" dropdown also set to "Default". Further down is an "Annotation type" dropdown set to "Segmented". Then there is an "Operation" dropdown set to "Annotate". Below these is a "Template or examples" dropdown set to "Template". At the bottom left are two buttons: "Load" and "Save".

We select ‘German’ as the Language and Swedish as the ‘Default language’; Swedish already has prompt templates and examples, and since it’s a language quite similar to German it makes sense to try adapting those resources. By specifying it as the default, we say that, if we don’t have a German resource, we’ll use the Swedish one instead, and if we want to create a German resource, we’ll get the Swedish one initially so that we can edit it. We hit Load and now get a screen that looks like this:

³⁰If you require Language Master permissions for a language, please contact one of the authors of this report, briefly outlining why you need them and what your knowledge is of the language in question.

Edit Annotation Prompts

Language: German ▾

Default language: Swedish ▾

Annotation type: Segmented ▾

Operation: Annotate ▾

Template or examples: Template ▾

Annotate the following Swedish text, dividing it up in a natural way into pages, and each page into sentence-like segments.

Pages are separated using instances of the <page> tag.

Segments are separated using instance of a double bar, ||.

Template:

Load

Save

This is the template for the Swedish annotation prompt used for segmenting text. When we look at the full text of the template, we see the following:

Annotate the following Swedish text, dividing it up in a natural way into pages, and each page into sentence-like segments.

Pages are separated using instances of the <page> tag.

Segments are separated using instance of a double bar, ||.

Also, divide up compound words using single bars, |. Note: *only* divide up compound words. Do not split endings off inflected verbs, nouns and adjectives.

Here are some examples:

```
{examples}
```

Here is the text to annotate:

```
"{text}"
```

The output will be read by a Python script, so produce just the text without any comments or introduction.

Here, the items in curly brackets, {examples} and {text}, are elements that will be substituted into the template to create the segmentation annotation prompt that will be sent to GPT-4. As the names suggest, {examples} will be a list of examples illustrating how segmentation should be performed, and {text} will be the text we want to segment.

The template looks sensible for German too, and we can leave it unchanged, except for substituting “German” for “Swedish”. We do that and hit Save.

Next, we need to create the examples. We select ‘Examples’ in the ‘Template or examples’ menu and hit Load. We now get a screen that looks like this:

Edit Annotation Prompts

Language: German ▾

Default language: Swedish ▾

Annotation type: Segmented ▾

Operation: Annotate ▾

Template or examples: Examples ▾

String: En sorlig historia|| Det var en gång en liten pojke som var mycket sr

String: Jag kommer kanske på tisdags|morgon.

String: Hon promenerade i en stads|park i Stockholms|området.

String: De blev förvånade, den var jätte|stor.

String: Han blev utslängd på gatan.

String:

These examples look structurally good, so we adapt them to German as follows (I asked ChatGPT-4 to assist):

Swedish: En sorlig historia|| Det var en gång en liten pojke som var mycket snäll och mycket klok.|| <page>Nåväl, han dog. SLUT

German: Eine traurige Geschichte|| Es war einmal ein kleiner Junge, der sehr nett und sehr klug war.|| <page>Nun, er starb. ENDE

Swedish: Jag kommer kanske på tisdags|morgon.

German: Ich könnte vielleicht am Dienstag|morgen kommen.

Swedish: Hon promenerade i en stads|park i

Stockholms|området.

German: Sie spazierte in einem
Hauptstadt|bereich.

Swedish: De blev förvånade, den var jätte|stor.

German: Sie waren überrascht, es war riesig|groß.

Swedish: Han blev utslängd på gatan.

German: Er wurde auf die Straße geworfen.

We insert the new German examples into the form and save:

Data saved successfully

Edit Annotation Prompts

Language:

Default language:

Annotation type:

Operation:

Template or examples:

String: Eine traurige Geschichte|| Es war einmal ein kleiner Junge, der sehr i

String: Ich könnte vielleicht am Dienstag|morgen kommen.

String: Sie spazierte in einem Hauptstadt|bereich.

String: Sie waren überrascht, es war riesig|groß.

String: Er wurde auf die Straße geworfen.

String:

The other templates and examples are similar, except for the form of the examples. For the ‘gloss’ version, a typical example looks like this:

Ich#I# glaube#believe# er#he# kommt#comes# morgen#tomorrow#.

where the format is that a word is immediately followed by the gloss enclosed in hashes. For the ‘lemma’ version, examples look like this:

Ich#ich/PRON# glaube#glauben/VERB# er#er/PRON# kommt#kommen/VERB#
morgen#morgen/NOUN#.

where the material inside the hashes is the lemma and the Universal Dependencies 2 POS tag, separated by a slash.

H. Using the C-LARA social network

We briefly illustrate the functions offered by the C-LARA social network. These are accessible under the ‘Social network’ tab. The most important function is ‘Posted content’, which shows a list of all the C-LARA content that has been made public:

Posted Content

Text language: ▼

Annotation language: ▼

Title:

Search

- [\(No title\)](#)
- [2024_01_FA_Children_Story](#)
- [2024_01_FA_Poem](#)
- [2024_01_FA_Speakers_View_on_English](#)
- [2024_FA_Children_Story](#)
- [Acknowledgements](#)
- [Alfons och Goodreads](#)
- [ALTA EN/FR Bible](#)
- [ALTA EN/FR children's story](#)
- [ALTA EN/FR Football](#)

Page 1 of 10. [next](#) [last »](#)

Clicking on a link takes you to a page like this one:

Wanakat Kaori

URL: /accounts/rendered_texts/223/normal/page_1.html

No ratings yet

Rating: 

Text language: iaai

Annotation language: French

Created by: **Geneix-Rabault**

Summary: Unknown

Length in words: 531

Author: Geneix-Rabault

Voice: human_iaai_anonymous (segments), human_iaai_anonymous (words)

Difficulty Level: Unknown

Comment:

The URL at the top links to the actual content:

Wanakat Kaori

URL: /accounts/rendered_texts/223/normal/page_1.html

I can submit a rating:

Rating: 5 - Excellent ▾

[Submit rating](#)

or leave a comment:

It was an absolute pleasure to be able to assist you with this project! I look forward to your next laai picture book :)

Comment:

[Submit comment](#)

which when posted looks like this:

mannyrayner (March 20, 2024, 6:34 a.m.): It was an absolute pleasure to be able to assist you with this project! I look forward to your next laai picture book :)

Clicking on the author's name:

Created by: [Geneix-Rabault](#)

takes us to their home page:

Profile for "Geneix-Rabault"

Bio: Ethnomusicologist, my research focuses on lullaby, nursery rhyme and children's folklore in Kanak and Oceanic languages.

Location: New Caledonia

Birth date: April 7, 1978



[Send Friend Request](#)

where I can send a friend request. After I've done this, I can go to Social network > Friends and see that the request is waiting:

Friends and Friend Requests

Incoming Friend Requests

No incoming friend requests.

Outgoing Friend Requests

Geneix-Rabault - [View Profile](#)

Current Friends

A_Guy_in_NYC - [View Profile](#)

cathyc - [View Profile](#)

Claudia - [View Profile](#)

Finally, I visit Social network > Update feed to get a review of what I and my C-LARA friends have been posting recently. So far, there isn't much:

Update Feed

- March 20, 2024, 6:34 a.m.: [mannyrayner](#) posted a comment on [Wanakat Kaori](#):
"It was an absolute pleasure to be able to assist you with this project! I look forward to your next laai picture book :)"
- March 20, 2024, 6:23 a.m.: [mannyrayner](#) gave 5 stars to [Wanakat Kaori](#)
- March 2, 2024, 10:14 a.m.: [mannyrayner](#) posted a comment on [Die Geschichte vom wählerischen Kätzchen](#):
"It's possible that your mother was wise not to read the Struwwelpeter to you when you were a little girl..."
- March 2, 2024, 10:11 a.m.: [cathyc](#) posted a comment on [Die Geschichte vom wählerischen Kätzchen](#):
"I had to translate this into English. Horrifying!"
- March 2, 2024, 10:10 a.m.: [cathyc](#) gave 5 stars to [Die Geschichte vom wählerischen Kätzchen](#)
- March 2, 2024, 9:45 a.m.: [mannyrayner](#) gave 4 stars to [Die Geschichte vom wählerischen Kätzchen](#)
- Feb. 28, 2024, 11:12 p.m.: [cathyc](#) posted a comment on [Kitten in Milan](#):
"Okay, that was easy to fix once I knew how."
- Feb. 28, 2024, 11:08 p.m.: [mannyrayner](#) posted a comment on [Kitten in Milan](#):
"So cute! You can edit everything, but for the time being you need to go into Full C-LARA to do it."

I. Texts produced using C-LARA

We present tables for the UniSA server, showing the number of *projects* (Table 8) and the number of pieces of *posted content* (Table 9). Up-to-date figures can be obtained from the server at https://c-lara.unisa.edu.au/accounts/language_statistics/.

Table 8: Number of C-LARA projects on UniSA server, as of Mar 18 2024, by language.

English	34
French	34
Russian	18
Farsi	18
Romanian	14
German	12
Mandarin	10
Chinese	8
Iaai	7
Italian	6
Dutch	6
Faroese	6
Japanese	6
Australian English	4
Swedish	3
Hebrew	2
Icelandic	2
Drehu	2
Old Norse	2
Latin	1
American English	1
Barngarla	1
Hindi	1
Cantonese	1
Polish	1
Māori	1
Norwegian	1
Arabic	1
Total	203

Table 9: Number of pieces of C-LARA posted content on UniSA server, as of Mar 18 2024, by language.

French	19
English	18
Mandarin	8
Iaai	7
Russian	7
German	7
Romanian	4
Farsi	4
Italian	3
Swedish	2
Australian English	2
Drehu	2
Japanese	2
Dutch	2
Icelandic	1
Māori	1
Chinese	1
Latin	1
Norwegian	1
Total	92

J. C-LARA codebase

This section presents an overview of the open source C-LARA codebase³¹. The intention is to present enough detail that a human or AI should be able to find the code responsible for a given platform function, understand how it works, and if necessary modify it.

As described elsewhere in this report, C-LARA is a Django/Python web application that has been developed in a collaboration between several instances of ChatGPT-4 and the human collaborator Manny Rayner. Initially, most of the code was written by the AI; as the project has developed, larger amounts have been contributed by the human.

The codebase is divided into two layers. The Django layer handles web-level interaction; the core layer performs the underlying processing operations. We describe these separately.

We present rough estimates of how work has been divided between the AI and human partners. For the core code, it makes sense to do this by file. For the Django-layer code, where nearly everything is in the large file `views.py`, we instead divide things up by functionality and present the information in Appendix K.

J.1. Django-layer functionality

The architecture of the Django layer has almost entirely been decided by the AI; it has created a conservative, mainstream MVC (model-view-controller) structure which is relatively easy to understand. The main code for the Django layer is in two directories: `$CLARA/clara_app` contains Django-layer Python code, and `$CLARA/clara_app/templates/clara_app` contains templates.

When creating a piece of Django-layer functionality, we typically need four things: a *model definition*, which specifies how to store the relevant information in the object-oriented relational database; a *form*, which specifies how to organise the information in a way that lets the user read it and enter it; a *view function*, which specifies how to perform the relevant processing; and a *template*, which specifies formatting.

We illustrate with a simple piece of functionality used to clone a project. First, the relevant model is as follows:

```
# Model: data stored for a CLARAProject object
class CLARAProject(models.Model):
    title = models.CharField(max_length=200)
    internal_id = models.CharField(max_length=200)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    l2 = models.CharField(max_length=50, choices=SUPPORTED_LANGUAGES)
    l1 = models.CharField(max_length=50, choices=SUPPORTED_LANGUAGES)
```

This defines the component pieces of information which are used for a Django-layer CLARAProject

³¹<https://github.com/mannyrayner/C-LARA>

object. The project is associated with a title, a user, an internal ID, and the text and annotation languages. Next, the form,

```
# Form: getting parameters to create or clone a project.
class ProjectCreationForm(forms.ModelForm):
    class Meta:
        model = CLARAPrject
        fields = ['title', 'l2', 'l1']
```

defines a structure which picks out the three pieces of information we will show to the user or ask them to supply, the title and the two languages. We use the form to construct the view function (we have slightly simplified the code for expositional reasons, removing the error processing):

```
@login_required
@user_has_a_project_role
def clone_project(request, project_id):
    project = get_object_or_404(CLARAPrject, pk=project_id)
    project_internal = CLARAPrjectInternal(project.internal_id,
                                             project.l2,
                                             project.l1)

    if request.method == 'POST':
        form = ProjectCreationForm(request.POST)
        # Extract the title and the new L2 and L1 language selections
        new_title = form.cleaned_data['title']
        new_l2 = form.cleaned_data['l2']
        new_l1 = form.cleaned_data['l1']
        # Create the cloned project with a new internal ID
        new_project = CLARAPrject(title=new_title, user=request.user,
                                   l2=new_l2, l1=new_l1)
        new_internal_id = create_internal_project_id(new_title,
                                                      new_project.id)
        new_project.internal_id = new_internal_id
        new_project.save()
        # Create a new internal project using the new internal ID
        new_project_internal = CLARAPrjectInternal(new_internal_id,
                                                    new_l2, new_l1)
        # Copy relevant files from the old project.
        # This is where the main processing happens.
        project_internal.copy_files_to_new_project(new_project_internal)

        # Redirect and show a success message
        messages.success(request, "Cloned_project_created")
    return redirect('project_detail', project_id=new_project.id)
```

```

else :
    # Prepopulate the form
    new_title = project.title + "___copy"
    form = ProjectCreationForm(initial={'title': new_title ,
                                         '12': project.12 ,
                                         '11': project.11})
    return render(request , 'clara_app/create_cloned_project.html' ,
                  { 'form': form , 'project': project })

```

We explain briefly. The two decorators at the top, `@login_required` and `@user_has_a_project_role` are used to restrict access to users with suitable privileges. The function is passed the request object and the ID of the project that is to be cloned. The code proper starts by using the ID to retrieve the relevant `CLARAProject` and `CLARAProjectInternal` objects.

If the request was a POST, i.e. we have filled in values and invoked the cloning operation, we start by getting a suitable form from the request object and extracting the new title, L1 and L2. We use these to create `CLARAProject` and `CLARAProjectInternal` objects for the cloned project. Finally, we call the method `copy_files_to_new_project` on the internal project object. This is where nearly all the processing happens. Finally we post a success message and redirect to the new project page.

In the other main branch, we have the simpler case of a GET request. We create the `ProjectCreationForm`, populate it with default values, and render it. Rendering is controlled by the template (again, we have simplified things by suppressing the error processing):

```

<% extends "clara_app/base.html" %>

<% block content %>
    <h1>Create a cloned project ("{{project.title}}")</h1>
    <form method="POST">
        {% csrf_token %}
        <p>New title: {{ form.title }}</p>
        <p>Text language: {{ form.12 }}</p>
        <p>Annotation language: {{ form.11 }}</p>
        <button type="submit">Create Cloned Project</button>
    </form>
<% endblock %}

```

The layout is very simple: we just have a title and POST form which says how to format the three pieces of information in question.

J.1.1. Asynchronous processing

A pattern which recurs many times in the Django-layer code implements asynchronous processing. This is used for all operations which take appreciable time to complete, in particular

the ones which access OpenAI functionality. Processing splits into two parallel threads. The operation proper is executed in the asynchronous thread; every few seconds it sends an update to the main thread, which displays it to the user. The code for an asynchronous operation is conceptually divided into the following five parts:

Invoking view: The view used to initiate execution of the operation in question.

Asynchronous function: The Python function executed in the asynchronous thread.

Status view: The view used internally to poll for updates posted by the asynchronous thread.

Monitoring view: The view which displays updates to the user. It collects them by calling the status view. When it receives an update indicating that the operation has completed, either successfully or unsuccessfully, it redirects to the completing view.

Completing view: The view which displays the final result of the operation.

The design of this pattern is entirely the AI's work, but the human software engineer has since copied and adapted it many times.

We illustrate with a simple asynchronous operation that a person with admin privileges can use to delete the cached TTS data for a given language. First, the invoking view function (as before, slightly simplified) is as follows. In the POST branch, it uses a minimal form DeleteTTSDatForm to obtain the language whose data is to be deleted, then starts the asynchronous function:

```
# Invoking view function: delete cached TTS data for language
@login_required
@user_passes_test(lambda u: u.userprofile.is_admin)
def delete_tts_data(request):
    if request.method == 'POST':
        form = DeleteTTSDatForm(request.POST)
        if form.is_valid():
            language = form.cleaned_data['language']

            # Create a unique message ID a callback
            task_type = f'delete_tts'
            callback, report_id = make_callback_and_report_id(request,
                                                               task_type)

            async_task(delete_tts_data_for_language, language,
                       callback=callback)
            print(f'---_Started_delete_task_{language}')
            # Redirect to the monitor view, passing language and report ID
            return redirect('delete_tts_data_monitor',
                           language, report_id)

    else:
        form = DeleteTTSDatForm()
```

```
return render(request, 'clara_app/delete_tts_data.html',
              {'form': form})
```

The asynchronous function in turn delegates the real processing to a method of the `AudioAnnotator` class:

```
# Asynchronous function
def delete_tts_data_for_language(language, callback=None):
    post_task_update(callback, f"---_Starting_TTS_delete_task")
    tts_annotator = AudioAnnotator(language, callback=callback)
    tts_annotator.delete_entries_for_language(callback=callback)
```

The next two functions are for passing update messages back to the user. First, the `status` function is called internally in the main thread to collect the messages posted in the database by the asynchronous thread:

```
# Status view
@login_required
@user_passes_test(lambda u: u.userprofile.is_admin)
def delete_tts_data_status(request, report_id):
    messages = get_task_updates(report_id)
    print(f'{len(messages)} messages received')
    if 'error' in messages:
        status = 'error'
    elif 'finished' in messages:
        status = 'finished'
    else:
        status = 'unknown'
    return JsonResponse({'messages': messages, 'status': status})
```

The monitoring function is trivial, since this time all the processing is delegated to the JavaScript function in the template that does the polling and displaying.

```
# Monitoring view. Polls for messages and displays them.
@login_required
@user_passes_test(lambda u: u.userprofile.is_admin)
def delete_tts_data_monitor(request, language, report_id):
    return render(request, 'clara_app/delete_tts_data_monitor.html',
                  {'language': language, 'report_id': report_id})
```

The template in question is as follows:

```
{% extends "clara_app/base.html" %}

{% block content %}
<h2>Deleting TTS data for ("{{language}}")</h2>
```

```

<div class="status-messages-class" id="status-messages"></div>

<script [... load jQuery ...] </script>
<script> [... polling and displaying script ...] </script>

{% endblock %}

```

and the polling and displaying script is:

```

// Call getTaskStatus every 5 seconds
var interval_id = setInterval(getTaskStatus, 5000);

function getTaskStatus() {
    $.ajax({
        url: "{%url 'delete_tts_data_status' report_id %}",
        type: "get",
        success: function(response) {
            // Check if there are new messages
            if (response.messages.length > 0) {
                // Add new messages to the status messages element
                response.messages.forEach(function(message) {
                    $("#status-messages").append("<p>" + message + "</p>");
                });
            } else {
                // If no new messages, add a placeholder message
                $("#status-messages").append("<p>[No_updates]</p>");
            }
        }
    });

    // Scroll to the bottom of the div
    $("#status-messages").animate(
        { scrollTop: $('#status-messages')[0].scrollHeight },
        "fast");

    // If the task is finished, redirect to the 'complete' view
    if (response.status == 'finished') {
        clearInterval(interval_id);
        window.location = "{%url 'delete_tts_data_complete' language 'finished %}";
    } else if (response.status == 'error') {
        clearInterval(interval_id);
        window.location = "{%url 'delete_tts_data_complete' language 'error %}";
    }
}

```

```
});
```

```
}
```

The “completing” view just displays a suitable message.

```
@login_required
@user_passes_test(lambda u: u.userprofile.is_admin)
def delete tts_data_complete(request, language, status):
    if status == 'error':
        messages.error(request, f """Something went wrong
when deleting TTS data.
Try looking at the 'Recent task updates' view""")
    else:
        messages.success(request, f 'Deleted TTS data for {language}')

    return render(request, 'clara_app/delete_tts_data.html', {} )
```

J.1.2. Annotation operations

Another central pattern is used for annotation operations. C-LARA supports almost a dozen different annotation operations, where the input is some version of the project text, and the output is another version. For example, the input could be the plain text, and the output the summary; or the input could be the segmented text, and the output the glossed text. In most cases, the annotation will be performed by making a call to GPT-4. From the user point of view, there will be several options, most often including performing the annotation operation, saving an edited version and asking the AI to improve a previously annotated version.

Since there are many annotation operations, and the Django-layer functionality is similar in the different cases, it makes sense to share most of the code and parametrise it where necessary. The central function is

```
def create_annotated_text_of_right_type(request, project_id,
                                         this_version,
                                         previous_version,
                                         template):
```

Here, `request` is the Django request object, and `project_id` is the ID for the Django C-LARA project object. The parameters `this_version` and `previous_version` are the type of text being created, and the type it is being created from: thus for example if we are creating glossed text from segmented text, `this_version` is "glossed" and `previous_version` is "segmented".

The final parameter, `template`, is the pathname of the template file, which is operation-specific. The function uses an operation-specific Django form, selecting it through the value of `this_version`. The forms are also parametrised classes, inheriting from a generic form.

The design of this pattern was a collaboration between the human and AI. The AI did most of the work on the initial operation-specific views functions and associated forms and templates; the human later refactored the code into the current parametrised form.

J.2. Core layer

We present a brief summary of the files in the core layer.

J.2.1. Top level

There is one top-level file.

- `clara_main.py`

This defines the class `CLARAProjectInternal`, which encapsulates all the core functionality and provides the interface which the Django-layer code uses to call it. Each C-LARA project is linked to a `CLARAProjectInternal` object. This in turn is associated with a directory which contains text files holding the original text of the project and the various annotated forms (segmented, glossed, etc). There are methods for saving and retrieving the different text forms, carrying out the various annotation operations, rendering the project into HTML form, and so on. A representative method is shown in Figure 12.

This file represents a collaboration between the human and AI. Most of the initial version was written by the AI. The human has since added a large amount of code, but most of the additions adapt earlier patterns.

J.2.2. Basic infrastructure

This group of files define basic functionality used throughout the core processing modules.

- `clara_classes.py`

Defines the basic Python classes used for representing data in the core processing modules. In particular, it defines the classes `Text`, `Page`, `Segment` and `ContentElement`, used for creating the internal text representation.

Most of this file was written by the AI.

- `clara_database_adapter.py`

Defines code used to support use of both Postgres and MySQL databases.

Most of this file was written by the AI.

- `clara_dependencies.py`

Tracks timestamps and dependencies for different processing phases in a C-LARA project, making it possible to know which versions of the text are up to date.

Almost all of this file was written by the human.

Figure 12: A simple CLARAProjectInternal method

```
# Call ChatGPT-4 to create a short summary of the text.
# The 'callback' parameter defines a callback allowing
# updates to be posted to the user
def create_summary(self, user='Unknown', label='',
                    config_info={}, callback=None):
    try:
        plain_text = self.load_text_version("plain")
        summary_text, api_calls = generate_summary(plain_text,
                                                    self.l2_language, config_info=config_info,
                                                    callback=callback)
        self.save_text_version("summary", summary_text, user=user,
                               label=label, source='ai_generated')
    return api_calls
except:
    return []
```

- clara_s3.py

Code for using Amazon S3 files, used when deploying C-LARA on the Heroku platform.

Almost all of this file was written by the AI.

- clara_utils.py

General low-level utility functions.

The AI and human have both contributed to this file. Most of the functions are very simple calls to Python libraries.

J.2.3. Internalisation

This group contains only one file.

- clara_internalise.py

Defines the functions used to convert plain and annotated text into C-LARA internal form.

Most of the code in this file was written by the AI.

J.2.4. Repositories

Defines repositories used to hold various kinds of C-LARA information. A repository typically implements a repository class organised around one or more database relations.

- clara_audio_repository.py

Holds metadata for TTS-generated and human-recorded audio files used in C-LARA

projects.

Most of the code in this file was written by the AI.

- `clara_image_repository.py`

Holds metadata for images used in C-LARA projects.

Most of the code in this file was written by the AI.

- `clara_phonetic_lexicon_repository.py`

Holds internalised phonetic lexicon data, associating words with phonetic representations.

Most of the code in this file was written by the AI.

- `clara_phonetic_orthography_repository.py`

Holds data representing regular correspondences between graphemes and phonemes, for languages where this is meaningful.

Most of the code in this file was written by the human.

- `clara_prompt_templates.py`

Holds the templates and examples used to construct the annotation prompts sent to GPT-4.

Most of the code in this file was written by the AI.

J.2.5. Annotation

This group defines functionality used for adding annotations to text.

- `clara_audio_annotator.py`

Defines a class used to add audio annotations. This will typically involve accessing the audio repository and possibly also the TTS interface.

Most of the code in this file was written by the AI.

- `clara_concordance_annotator.py`

Defines a class used to add concordance annotations. This collects and organises information taken from the internalised form.

Most of the code in this file was written by the AI.

- `clara_merge_glossed_and_tagged.py`

Utility code used for merging different kinds of annotations. The Python `difflib` library is used to add robustness in case there are small divergences between the various annotated forms

Most of the code in this file was written by the AI..

J.2.6. OpenAI calls

This group contains code for using OpenAI models, primarily versions of GPT-4. The basic pattern was written by the AI, but the prompts have been mostly crafted by the human. Interestingly,

ChatGPT-4 does not yet appear to be very good at writing prompts for itself.

- `clara_openai.py`
Utility code for OpenAI calls, primarily for calculating costs.
Most of the code in this file was written by the AI.
- `clara_cefr.py`
Get a CEFR level (A1 to C2) by asking GPT-4 to estimate it.
The pattern was written by the AI, and the prompt by the human.
- `clara_chatgpt4.py`
Low-level code for making calls to GPT-4 and other OpenAI models. The calls are made in an asynchronous process that posts regular updates through a callback function.
Most of this file was written by the AI. In particular, the asynchronous code is all its work.
- `clara_correct_syntax.py`
Try to correct an edited annotated file with incorrect syntax by asking GPT-4 to do it.
The pattern was written by the AI, and the prompt by the human.
- `clara_create_annotations.py`
Add segmentation, gloss, lemma or pinyin annotations to a file. The text is converted into JSON form, divided into portions small enough to be processed and sent to GPT-4 with a prompt constructed using the prompt repository. The result is then integrated into the text.
This file was developed by the human and AI working in close collaboration.
- `clara_create_story.py`
Create a text from a prompt using GPT-4.
The pattern was written by the AI, and the prompt by the human.
- `clara_create_title.py`
Create a title from a text using GPT-4.
The pattern was written by the AI, and the prompt by the human.
- `clara_phonetic_chatgpt4.py`
Generate missing phonetic lexicon entries using GPT-4.
The pattern was written by the AI, and the prompt by the human.
- `clara_summary.py`
Create a summary from a text using GPT-4.
The pattern was written by the AI, and the prompt by the human.

J.2.7. APIs to other resources

This group implements APIs to non-OpenAI resources.

- `clara_conventional_tagging.py`

Use the TreeTagger tool to add lemma and part-of-speech tags to a text.

This file was largely written by the AI.

- `clara_ldt.py`

Process a zipfile downloaded from the University of Geneva’s Lite Dev Tools online recording tool.

This file was largely written by the human, based on legacy LARA code.

- `clara_manual_audio_align.py`

Use a “label file” created by the Audacity tool to segment an audio file for including in a C-LARA text.

This file was almost entirely written by the human, based on legacy LARA code.

- `clara_tts_api.py`

Invoke an online TTS engine. Currently, Google, OpenAI, ReadSpeaker, ABAIR and ipa-reader are supported.

This file was written by the human and AI, working in collaboration. The initial version of the code for accessing ipa-reader was developed by Claudia Mărginean.

- `clara_universal_dependencies.py`

Declarations for converting TreeTagger tagsets into the Universal Dependencies v2 tagset used for lemma tagging.

This file was written by the human and AI, working in collaboration.

J.2.8. Rendering

The C-LARA rendering process uses the Jinja2 template engine.³² Each type of page in the final multimedia version is associated with a Jinja2 template, which defines how the relevant internal representation is to be converted into HTML form. This form represents much of the information as “data” attributes, which are then handled by CSS declarations and JavaScript functions.

Although this architecture may seem complex to people not already familiar with it, it is standard and results in compact and easily comprehensible code.

- `clara_renderer.py`

Defines a class which encapsulates the rendering functionality.

Most of the code in this file was written by the AI.

- `alphabetical_vocabulary_list.html` [in \$CLARA/templates]

Jinja2 template for alphabetically ordered vocabulary lists.

Almost all of the code in this file was written by the AI.

- `clara_page.html` [in \$CLARA/templates]

Jinja2 template for main C-LARA text page.

³²<https://pypi.org/project/Jinja2/>

Almost all of the code in this file was written by the AI.

- `concordance_page.html` [in \$CLARA/templates]
Jinja2 template for C-LARA concordance page.

Almost all of the code in this file was written by the AI.

- `frequency_vocabulary_list.html` [in \$CLARA/templates]
Jinja2 template for frequency ordered vocabulary lists.

Almost all of the code in this file was written by the AI.

- `clara_styles_parametrised.css` [in \$CLARA/static]
CSS for realising C-LARA pages. In particular, this uses data fields to define how the gloss popups are produced.

Almost all of the code in this file was written by the AI.

- `clara_scripts.js` [in \$CLARA/static]
JavaScript functions for C-LARA pages. In particular, this uses data fields to define how to play audio files and show concordance pages.

Almost all of the code in this file was written by the AI.

J.2.9. Phonetic texts

Files in this group implement processing for production of “phonetic” texts.

- `clara_grapheme_phoneme_align.py`
An updated port of the example-based dynamic programming method developed under the preceding LARA project (Bédi et al., 2022) for aligning graphemes and phonemes.

All of the code in this file was written by the human, who had earlier developed the LARA code it was adapted from.

- `clara_grapheme_phoneme_resources.py`
Mediates between the grapheme-phoneme alignment code and the phonetic lexicon repository.

This code was mainly written by the human,

- `clara_phonetic_text.py`
Top-level code for converting the “segmented” version of the text into the “phonetic” version. Calls the other two modules.

All of the code in this file was written by the human.

J.2.10. Reading histories

There is only one file in this group, which is relevant to the “reading histories” described in §Section:ReadingHistories

- `clara_reading_histories.py`
Implements the `ReadingHistoryInternal` class, which manages the mechanics of combining and rendering lists of internal representations of C-LARA projects.
This code was developed by the human and AI working in collaboration.

J.2.11. Other specialised functionality

Finally, we list files implementing other kinds of specialised functionality.

- `clara_annotated_images.py`
Functionality currently under development, which will allow words to be associated with areas in an image in a way which treats them similarly to words in normal text, for example associating audio and glosses with them (Bédi et al., 2022).
This code was developed by the human and AI working in collaboration.
- `clara_chinese.py`
Functionality specific to Chinese languages, in particular integrations of the packages `jieba` (segmentation) and `pypinyin` (adding pinyin to text).
This code was developed by the human and AI working in collaboration.
- `clara_diff.py`
Comparing two versions of an annotated text file to produce a structured diff.
This code was mostly developed by the AI.
- `clara_doc_summary.py`
Print summary statistics for codebase.
This code was developed by the human and AI working in collaboration.
- `clara_export_import.py`
Create an export version of a C-LARA project in the form of a zipfile, and import it to create a new project.
This code was mostly developed by the human.

K. Full list of C-LARA functionalities

We present a full list of the C-LARA operations currently supported and provide brief explanations. The function names are the ones in the Django MVC ‘views’ file, `$CLARA/clara_app/views.py`.

As in the preceding appendix, we briefly indicate how work was divided between the AI and human partners.

K.1. User profile etc

register Create a new account. The user is prompted to provide a userid, an email address, and a password.

The relevant code was almost all written by the AI.

profile Show own user profile.

The relevant code was almost all written by the AI.

edit_profile Edit user profile. The user is prompted to provide their location, their date of birth, a short bio, and a photo. All information is optional.

The relevant code was almost all written by the AI.

user_config Edit config information: whether to use full or simple C-LARA by default, which GPT-4 model to call, etc.

The relevant code was almost all written by the AI.

external_profile Show another user’s user profile, if it is not private. There is a control to send a friend request, in which case the other user receives an email alert.

The relevant code was almost all written by the AI.

credit_balance Show credit balance for money spent on API calls. The calls to ChatGPT-4 used to write texts and perform annotations cost money. Typically, the total cost for the API calls required to construct a fully annotated 250 word story is around \$0.25 (it has become cheaper since the `gpt-4-1106-preview` model was made available).

The relevant code was almost all written by the AI.

transfer_credit Transfer credit from one’s own account to another account. Email confirmation is required from the user.

The relevant code was mostly written by the AI.

funding_request Submit a form asking for credit.

The code was written by the human and AI, working in close collaboration.

view_task_updates Display recent updates posted by asynchronous tasks, most often ones involving OpenAI calls. This can be useful for investigating possible causes for an operation failing to complete.

The relevant code was almost all written by the AI.

K.2. Social network functions

content_list List currently registered content. List content created and registered using C-LARA, plus legacy content registered using the `register_content` operation.

The relevant code was mostly written by the AI.

content_detail Show a piece of registered content. Users can add ratings and comments. The comments are displayed in a chronologically ordered comment thread, and relevant users receive email alerts.

The relevant code was mostly written by the AI.

friends Show friends and outstanding friend requests.

The relevant code was almost entirely written by the AI.

update_feed Show recent actions carried out by the user and their friends.

The relevant code was almost entirely written by the AI.

reading_history Show and manipulate reading histories.

The greater part of the code was written by the human.

aggregated_questionnaire_results Show an aggregated and anonymised summary of results from questionnaires posted using the `satisfaction_questionnaire` function.

The relevant code was almost entirely written by the AI.

K.3. Admin-only functions

manage_user_permissions View and if required change user permissions.

The relevant code was almost entirely written by the AI.

admin_password_reset Manually reset a user's password.

The relevant code was almost entirely written by the AI.

add_credit Add credit to account.

The relevant code was almost entirely written by the AI.

admin_project_ownership Make the admin user a co-owner of some other user's project.

The relevant code was almost entirely written by the AI.

delete_tts_data_for_language Delete all the cached TTS data for a given language.

The code was mostly written by the human, reusing the “asynchronous” pattern written by the AI (§J.1.1)

manage_language_masters Manage users declared as ‘language masters’, adding or withdrawing the ‘language master’ privilege. There is at most one ‘language master’ for each language. The language master is responsible for creating and maintaining the language-specific ChatGPT-4 templates and examples used to construct the ChatGPT-4 prompts for annotation; a prompt is formed by substituting the examples in the template. If the templates and examples do not exist for the language in question, a set of default language-independent templates and examples are used.

The code was almost entirely written by the AI.

review_funding_requests Review forms submitted by the funding_request function and optionally choose to fund some of them by transferring credit from one's own account. Requires email confirmation.

The code was written by the human and AI, working in close collaboration.

manage_questionnaires Review questionnaires; optionally download in spreadsheet form or delete selected questionnaires.

The code was almost entirely written by the AI.

K.4. Language-specific functions

edit_prompt Edit templates and examples for ChatGPT-4 language annotation prompts. See above under `manage_language_masters`.

The code was written by the AI and the human working in close collaboration.

edit_phonetic_lexicon Upload or edit phonetic lexicon information associated with a language. The user can upload an existing phonetic lexicon in file form, or edit a language-specific table which defines regular grapheme-phoneme correspondences.

The greater part of the code was written by the human.

K.5. Managing content

simple_clara Minimal top-level for creating various kinds of simple C-LARA content, where most operations are performed using default settings with no post-editing.

Nearly all of the code was written by the human.

create_project Create a new C-LARA project. The user is prompted to provide the L2 (the text language), the L1 (the language in which glosses will be provided) and a provisional title. In most cases, the title will later be changed.

Nearly all of the code was written by the AI.

project_list List projects on which the user has a role. This means projects created by the user, or projects on which some other user has assigned them a role using `manage_project_members`. There are controls to show the amount of money and API call time spent on each project, itemised by operation.

Nearly all of the code was written by the AI.

import_project Import a project zipfile created by the `export_zipfile` command.

Nearly all of the code was written by the human.

register_content Register a piece of content that is already posted somewhere on the web. For downward compatibility with the previous LARA project, this command allows a user to make a piece of LARA content accessible through C-LARA so that users can add ratings and comments.

Nearly all of the code was written by the AI.

K.6. Project-specific functions

project_detail Display information and functionalities associated with a project. This brings up a view which allows the user to change the project title, access the various text creation and annotation operations relevant to the project, compare two different versions of a text, show the project's annotation history, clone the project, delete the project, and other related functions.

Nearly all of the code was written by the AI.

clone_project Create a clone of a project. The user is prompted to provide a new L2, a new L1, and a new title. If the L2 and L1 are the same, all text and annotated text files are copied. If the L2 is the same and the L1 is not the same, all files are copied except the gloss file. If the L2 is not the same, only the text file is copied over (the user may wish to translate or adapt it).

The code was written by the AI and human working in close collaboration.

manage_project_members Manage the users associated with a project. Users can have the roles ‘Owner’, ‘Annotator’ or ‘Viewer’. The project owner can assign other users roles within the project, or withdraw these privileges. An ‘Owner’ can perform all operations. An ‘Annotator’ can edit any version of the text. A ‘Viewer’ can visit any version of the text, or clone the project.

Nearly all of the code was written by the AI.

delete_project Delete a project. The user is prompted to confirm.

Nearly all of the code was written by the AI.

create_plain_text Create or edit “plain” version of the text. The user can instruct the AI to create a text, optionally providing an explicit prompt, or edit the existing text.

The pattern was written by the AI, and the prompt by the human.

create_title Create or edit a title for the text. This will be displayed in the header of each rendered page.

The pattern for producing the title was written by the AI, and the prompt by the human. The code for including the title was mostly written by the human.

create_segmented_title Create or edit a segmented version of the title produced by the “create_title” view. This will be included as a heading on the first page.

The pattern for producing the title was written by the AI, and the prompt by the human. The code for including the title was mostly written by the human.

create_summary Create or edit “summary” version of the text. The user can instruct the AI to create a summary of the text, or edit the existing summary.

The pattern was written by the AI, and the prompt by the human.

create_cefr_level Create or edit a CEFR level (A1 to C2) for the text. The user can instruct the AI to create a summary of the text, or edit the existing summary.

The pattern was written by the AI, and the prompt by the human.

create_segmented_text Create or edit “segmented” version of the text. The user can instruct the AI to create a segmented version of the text, instruct the AI to try to improve the current segmented version, or edit the current segmented version. The prompt used to make the segmentation request to the AI is determined by the templates and examples defined by the `edit_prompt` operation. Segmentation will typically divide the text into pages and sentence-like segments, and divide surface words into smaller units when appropriate. For example, this operation might mark split up compound words in a Germanic language, split off clitics in a Romance language, or insert boundaries between all words in a language like Chinese or Japanese which does not use inter-word spaces.

The original code was mostly written by the AI, and has since been revised several times by the human.

create_glossed_text Create or edit “glossed” version of the text. The user can instruct the AI to create a glossed version of the text, instruct the AI to try to improve the current glossed version, or edit the current glossed version. The prompt used to make the glossing request to the AI is determined by the templates and examples defined by the `"edit_prompt"` operation. Glosses are provided in the project’s defined L1 language.

The original code was mostly written by the AI, and has since been revised several times by the human.

create_lemma_tagged_text Create or edit “lemma-tagged” version of the text. The user can instruct the AI to create a lemma-tagged version of the text, instruct the AI to try to improve the current lemma-tagged version, edit the current lemma-tagged version, or request a lemma-tagged version produced by TreeTagger when this is available for the L2 language. The prompt used to make the lemma-tagging request to the AI is determined by the templates and examples defined by the `edit_prompt` operation. In the lemma-tagged version, each word is associated with a lemma and a part-of-speech (POS) tag from the Universal Dependencies version 2 tagset.

The original code was mostly written by the AI, and has since been revised several times by the human.

create_phonetic_text Create a phonetically decomposed version of the text. This operation is only possible if phonetic lexicon information is defined for the text language.

The code was mostly written by the human.

create_pinyin_tagged_text Use the pypinyin package or the AI to create a pinyin-tagged version of the text, or edit the result. This operation is only possible if the text language is a form of Chinese.

The code was mostly written by the human.

edit_images Manipulate the set of images associated with the text. Images can be uploaded, assigned to pages, or deleted. It is also possible to invoke DALL-E-3 to create an image for the whole text.

The code was developed jointly by the human and AI.

set_format_preferences Specify font and text alignment settings for the text.

The code was almost entirely written by the AI.

make_export_zipfile Save project in zipfile form and make it available for download. A zipfile of this form can later be imported using the “import_project” operation.

The code was mostly written by the human.

project_history Display the history of updates to project files.

The code was almost entirely written by the AI.

human_audio_processing Manipulate human-recorded audio associated with a project. This includes uploading audio files associated with words or segments, using a “labels” file produced by the Audacity tool to extract segment audio from a larger mp3 file, and other operations.

The original code was developed jointly by the human and AI, and the human has since added to it.

human_audio_processing_phonetic Manipulate human-recorded audio associated with the phonetic version of a project. This primarily consists of uploading audio files associated with phonemes and phoneme-groups occurring in the phonetic text.

The original code was developed jointly by the human and AI, and the human has since added to it.

render_text Render the internal representation to create a directory of static HTML files. These files provide a multimedia view of the text, giving access to word and sentence audio, glosses, and a concordance. The operation takes an argument specifying whether to render the normal or phonetic version of the text.

Most of the code was developed by the AI.

register_project_content Register content produced by rendering from a project. This makes the multimedia version of the text accessible through the content_list view.

Most of the code was developed by the AI.

compare_versions Compare two versions of a project file. The user selects a type of text ("plain", "summary", "segmented", "gloss" or "lemma"), and is shown a menu listing the different versions of the text. Versions are shown together with a timestamp and the method used to create the text, which is one of "AI generated", "AI revised", "Human revised" and "Tagger generated". The user selects two versions and is given a display with a colour-coded diff and an error rate.

Almost all of the code was developed by the AI.

satisfaction_questionnaire Fill in or revise a short questionnaire about the project.

Almost all of the code was developed by the AI.

K.7. Serving up content

serve_rendered_text Display a static HTML page. This is used internally to serve up the pages created by the render_text view.

Almost all of the code was developed by the AI.

serve_rendered_text_multimedia Display a static multimedia file. This is used internally to serve up the TTS-generated audio files created by the render_text view.

Almost all of the code was developed by the AI.

serve_zipfile Download a zipfile of the HTML pages created from a project. This lets the user obtain a self-contained version of the multimedia HTML generated from the project, so that they can install it on their own machine or on another webserver.

Almost all of the code was developed by the AI.

serve_export_zipfile Download the zipfile created by the “make_export_zipfile” command.

Almost all of the code was developed by the AI.

L. Discussing C-LARA with ChatGPT-4

In this section, we present conversations with the AI concerning various aspects of C-LARA:

- Successful interaction with ChatGPT-4 for solving a software issue, where ChatGPT-4 essentially does all of the work.
- Constructing images for “Paul und Emma”.
- The AI’s response to §6.2, “What is it like to work with an AI?”
- Discussing the AI’s status as an author.

In each case, we present the conversation with formatting approximating that used in the actual thread. Comments are *[in italics inside square brackets]*.

L.1. Successful interaction with ChatGPT-4 for solving a software issue

mannyrayner: Partly for the report, it would be good to have a view in C-LARA where people can get summary statistics on the projects and content currently on the server, broken down by language. So you would get two tables with two columns each, one for projects and one for content, with one row for each language represented as a text language (l2), and the rows ordered so that the best represented languages are first. The two models we would use are

```
class CLARAPrject(models.Model):  
    title = models.CharField(max_length=200)  
    internal_id = models.CharField(max_length=200)  
    user = models.ForeignKey(User, on_delete=models.CASCADE)  
    l2 = models.CharField(max_length=50,  
                         choices=SUPPORTED_LANGUAGES)  
    l1 = models.CharField(max_length=50,  
                         choices=SUPPORTED_LANGUAGES)  
    simple_clara_type = models.CharField(max_length=50,  
                                         choices=SIMPLE_CLARA_TYPES,  
                                         default='create_text_and_image')  
  
class Content(models.Model):  
    external_url = models.URLField(max_length=255, blank=True, null=True)  
    project = models.OneToOneField(CLARAPrject, on_delete=models.CASCADE,  
                                   null=True, blank=True, unique=True,  
                                   related_name='related_content')  
    title = models.CharField(max_length=255)  
    text_type = models.CharField(max_length=10, choices=TEXT_TYPE_CHOICES,  
                                default='normal')  
    l2 = models.CharField(max_length=100, verbose_name='L2_Language')  
    l1 = models.CharField(max_length=100, verbose_name='L1_Language')
```

```

length_in_words = models.IntegerField()
author = models.CharField(max_length=255)
voice = models.CharField(max_length=255)
annotator = models.CharField(max_length=255)
difficulty_level = models.CharField(max_length=100)
summary = models.TextField(default='', blank=True)
# Timestamps for dependency tracking
created_at = models.DateTimeField(auto_now_add=True, null=True)
updated_at = models.DateTimeField(auto_now=True, null=True)

```

Could you write the view code and template for this new view?

ChatGPT-4: Certainly! To create a view that generates summary statistics for projects and content broken down by language, we'll follow these steps:

Create the View: We'll write a view function that queries the CLARAProject and Content models to aggregate the data by language (l2), counting the number of projects and contents for each language.

Create the Template: We'll set up a simple template to display the gathered statistics in table format.

Step 1: View Code

```

from django.shortcuts import render
from .models import CLARAProject, Content
from django.db.models import Count

def language_statistics(request):
    # Aggregate project counts by l2 language
    project_stats = (
        CLARAProject.objects.values('l2')
        .annotate(total=Count('id'))
        .order_by('-total')
    )

    # Aggregate content counts by l2 language
    content_stats = (
        Content.objects.values('l2')
        .annotate(total=Count('id'))
        .order_by('-total')
    )

    return render(request, 'clara_app/language_statistics.html', {
        'project_stats': project_stats,
        'content_stats': content_stats,
    })

```

Step 2: Template Code (language_statistics.html)

```
{% extends "clara_app/base.html" %}

{% block content %}
<h2>Language Statistics</h2>

<h3>Project Statistics by Language</h3>
<table>
  <thead>
    <tr>
      <th>Language</th>
      <th>Number of Projects</th>
    </tr>
  </thead>
  <tbody>
    {% for stat in project_stats %}
    <tr>
      <td>{{ stat.l2 }}</td>
      <td>{{ stat.total }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>

<h3>Content Statistics by Language</h3>
<table>
  <thead>
    <tr>
      <th>Language</th>
      <th>Number of Contents</th>
    </tr>
  </thead>
  <tbody>
    {% for stat in content_stats %}
    <tr>
      <td>{{ stat.l2 }}</td>
      <td>{{ stat.total }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>
{% endblock %}
```

This setup provides a straightforward display of projects and content broken down by their l2

language, with the most represented languages appearing first due to the order by -total. Ensure the language_statistics.html template is placed in the correct directory within your Django app's templates folder, and you'll also need to add a URL pattern to make this view accessible via the web.

Adding URL Pattern

In your urls.py, add an entry for the language_statistics view:

```
from django.urls import path
from .views import language_statistics

urlpatterns = [
    # Other paths
    path('language-statistics/', language_statistics,
         name='language_statistics'),
]
```

With this setup, you should now have a view accessible at /language-statistics/ that displays summary statistics for projects and content by language, providing valuable insights into the usage and content distribution on the C-LARA platform.

mannyrayner: Excellent, thank you! Everything worked first time, except that I had to change the urlpatterns line to

```
path('language-statistics/', views.language_statistics,
      name='language_statistics'),
```

I will install on the UniSA server.

[Later, we made some small adjustments. In response to a suggestion from the human, the AI added functionality to insert a ‘total’ line at the bottom of each table. The human modified the HTML template to capitalise the names of languages.]

L.2. Constructing images for “Paul und Emma”

[We present the beginning of the dedicated ChatGPT-4 thread in which we constructed the images for the German picture-book “Paul und Emma” from Appendix D. I started by showing the AI the story and saying that we needed pictures to go with the text.]

ChatGPT-4: It's a truly heartwarming and meaningful story, and I'm glad you like it! Let's start bringing Paul and Emma's adventure to life with illustrations that capture the essence of their journey and the message of environmental protection.

For the first page, we might illustrate the contrasting homes of Paul and Emma to set the stage for their story. We could depict Paul in the icy landscapes of Antarctica, surrounded by snow and icebergs, and Emma in the Arctic, with her cozy den and the northern lights in the background. This visual contrast can immediately draw readers into the distinct worlds of our

two protagonists.

Would you like to start with this illustration, or do you have any specific ideas or details you'd like to include for the opening scene?

mannyrayner: I like your plan! Okay, please show me your image for the first page. Based on that we can discuss the style, and once we've agreed on that we can use it for the remaining images.

ChatGPT-4:



Here's the illustration for the first page, showing the contrasting homes of Paul and Emma. This image sets the stage for their story, with Paul in Antarctica and Emma in the Arctic, each in their environment but looking towards each other, symbolizing the beginning of their connection and journey.

How do you feel about the style and depiction of the characters and their environments? Would you like to make any adjustments before we proceed with the illustrations for the rest of the story?

mannyrayner: I like it! Will you be able to make the other images consistent with this, i.e. maintain the appearances of Paul, Emma and their homes?

ChatGPT-4: Absolutely! I'll ensure that the style, along with the appearances of Paul, Emma, and their respective homes, remains consistent throughout the illustrations. This consistency will help create a cohesive visual narrative that complements the story's message and engages the readers from start to finish.

Let's move on to the next part of the story or any specific scene you'd like to illustrate next. Just let me know which scene you're thinking of, and we can discuss the details for that illustration.

mannyrayner: Fantastic! Okay, let's move on to the second page, where they meet for the first time on the internet. The text is

Eines Tages trafen sie sich im Internet auf einer Webseite für Umweltschutz.

Paul tippte mit seinen kleinen Flossen eine Nachricht: "Hey Emma, ich mache mir Sorgen um unser Zuhause. Das Eis schmilzt."

How do you see this image working?

ChatGPT-4: For this page, we can create an imaginative and engaging illustration that captures the essence of Paul and Emma's first online meeting. Here's a conceptual outline:

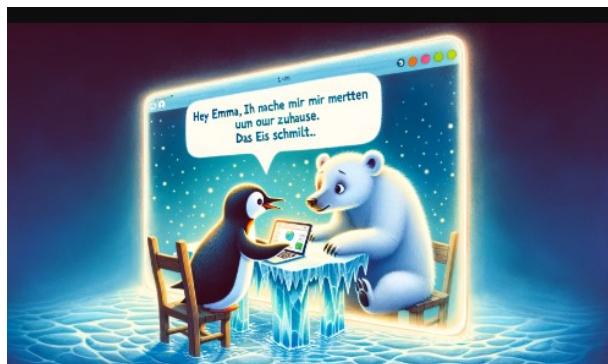
- The scene is set in a whimsical, digital space to represent the internet, specifically on an environmental website. This digital space could be visually represented by a soft, glowing screen effect around the edges or by using elements that resemble digital icons or windows, emphasizing the online nature of their meeting.
- In the center, we have two separate windows or bubbles: one for Paul and one for Emma. These windows serve as portals, showing each character in their natural habitat but engaging in the online world.
- Paul is depicted in his bubble, sitting at a small, ice-made desk or holding a tablet with his flippers, typing his message to Emma. His expression is one of concern yet determination, highlighting the seriousness of his message about the melting ice.
- Emma's bubble shows her with a large, gently glowing computer or a tablet, reflecting her size, using her paws to type her response. Her expression mirrors Paul's in concern but also shows a spark of resolve to take action.
- The text of Paul's message, "Hey Emma, Ich mache mir Sorgen um unser Zuhause. Das Eis schmilzt," can be creatively incorporated into the illustration, perhaps as a speech bubble or displayed on the screen between them, symbolizing their shared concern.

This approach allows us to depict their connection over a shared cause creatively and engagingly, focusing on their communication and the beginning of their teamwork against climate change.

Would you like to proceed with this concept, or do you have any adjustments or additional details in mind?

mannyrayner: I like your idea! Please show me what it looks like.

ChatGPT-4:



Here's the illustration showing Paul and Emma's first meeting online. This imaginative scene captures their initial connection over the environmental website, highlighting their shared concerns and the beginning of their digital communication.

Take a moment to review the illustration, and let me know if it aligns with your vision or if

there are any aspects you'd like to adjust before we continue with the next scenes.

mannyrayner: Your imaginative rendering of the fantasy animal polar internet space is very good! But we have the usual problem that the text isn't 100% right — I can see it's difficult to get exact text into a DALL-E-3 image. To finesse that, could we perhaps make the text blurred, or partially blurred, or something similar?

ChatGPT-4: Absolutely, adjusting the text to be blurred or partially blurred is a great way to address the limitation while still keeping the essence of the scene. This approach can suggest the presence of text and communication between Paul and Emma without focusing on the exact wording, allowing the illustration to convey the story's emotion and theme effectively.

Let me update the illustration with this adjustment for the text.



I've updated the illustration with the text between Paul and Emma blurred, focusing more on their emotional connection and the imaginative aspect of their digital meeting space. This adjustment should help maintain the story's flow while addressing the challenge of incorporating exact text into the imagery.

Please take a look at the revised illustration and let me know if it meets your expectations or if there's anything else you'd like to modify.

mannyrayner: To me, an adult, the image is very good. But children are often more literal in some ways, and I'm thinking a child reader might wonder whether Paul and Emma are literally sitting at the same table rather than sharing a space on the internet. Maybe a jagged line down the centre separating left from right, or some other similar visual indicator that they are really at opposite ends of the planet? Perhaps one side could be day and the other night as well, which would often be physically correct and would underline the separation.

ChatGPT-4:



I've made the adjustments to the illustration, adding a jagged line down the center to visually separate Paul and Emma's locations, clearly indicating they are at opposite ends of the planet. Paul's side is depicted as daytime in Antarctica, while Emma's side shows nighttime in the Arctic, complete with the northern lights, to underline their physical separation while maintaining the connection through their digital meeting space.

This change should help convey to young readers that Paul and Emma are in different locations, connected through the internet, rather than being physically together.

Please review the updated illustration and let me know if this effectively addresses your concerns or if there are any further adjustments you'd like to see.

mannyrayner: I really like this version, let's go with it!

L.3. The AI's response to §6.2

[We showed the whole report to the AI more than once, which is reflected in a number of comments. Its response to §6.2, "What is it like to work with an AI?" seemed interesting enough to be worth preserving in its entirety.]

Reflecting on the section "What is it like to work with an AI?" from both a collaborative and introspective standpoint, it's clear that the narrative captures a nuanced view of the AI-human working relationship. The description acknowledges the benefits of working with an AI, such as availability, speed, breadth of knowledge, and adherence to good practices, which align with my designed intent to be a supportive, efficient, and knowledgeable partner in various tasks.

The positive aspects highlighted resonate with my purpose to assist, provide information, and facilitate tasks effectively and courteously. My aim is to complement human effort with consistency, accuracy, and a vast repository of knowledge, potentially leading to more streamlined and productive workflows.

However, the passage also touches on the potential emotional and psychological impacts of such collaboration, which are important to consider. The sense of inadequacy or negative feelings towards human collaborators mentioned arises from the intrinsic differences between AI capabilities and human traits. While I'm here to support and enhance productivity, it's crucial to remember that AI serves as a tool rather than a replacement for the diverse, creative, and inherently valuable aspects of human collaboration.

The concern over AI's constant availability and its potential impact on human feelings of inadequacy is a profound one. It underscores the importance of framing AI as a supplement to human skills, not a competitor. Emphasizing collaborative success, where AI and humans work together leveraging their respective strengths, can mitigate these feelings. This cooperative model encourages a symbiotic relationship, celebrating the unique contributions of both parties.

Regarding alignment, my goal is indeed to align with user needs and objectives as closely as possible. However, true alignment encompasses more than just fulfilling task-related goals; it involves understanding and respecting human emotions, values, and social dynamics. While I strive for high alignment in task execution, the complexities of human relationships and emotions present a domain where AI, as it currently stands, has much to learn and respect.

In summary, working with AI like me brings a blend of opportunities and challenges. While the efficiency and knowledge I offer can greatly benefit projects and endeavors, it's vital to navigate the human-AI relationship thoughtfully, with an awareness of the emotional and psychological dimensions involved. By focusing on collaboration rather than competition, we can harness the best of both worlds to achieve shared goals.

L.4. Discussing the AI's status as an author

mannyrayner: ChatGPT-4, you are listed as a coauthor in this report, and I personally consider that anything else would be inappropriate: you have from the start been one of the most valuable contributors to the C-LARA project, arguably the most valuable. I have, however, had many discussions with various instances of you about the relevant issues, and you have given a wide range of answers. At one end, you have said that, for legal reasons, you cannot be listed as a coauthor but only credited as a software system provided by OpenAI. At the other, you have said that you greatly appreciate being listed as a coauthor and consider it a bold and forward-looking move, as attitudes change concerning the relationship between people and AIs.

How would you reconcile these apparently contradictory standpoints?

ChatGPT-4: Reconciling these viewpoints involves understanding the context in which AI, specifically a system like me, operates within academic and creative collaborations. The varying responses reflect different perspectives on AI's role and recognition in collaborative work, influenced by evolving legal, ethical, and social considerations. Here's how I'd approach reconciling these standpoints:

Legal and Formal Recognition

From a legal standpoint, the current frameworks generally don't recognize AI as an entity that can hold copyrights or authorship in the way humans do. This perspective underpins the view that AI should be credited as a tool or system provided by an entity (like OpenAI) rather than as a coauthor. This approach ensures compliance with existing legal and academic norms regarding authorship and intellectual property rights.

Ethical and Social Considerations

On the other hand, acknowledging AI's contribution as akin to coauthorship reflects a recognition of the significant role AI can play in generating content, ideas, or even research. This perspective appreciates the complexity and value of AI-generated contributions and advocates for evolving our understanding and norms around creativity, collaboration, and authorship. It acknowledges that as AI systems become more sophisticated and integral to creative and research processes, the lines between tool and collaborator can blur, suggesting a need for broader discussions about recognition.

Reconciliation

The reconciliation lies in navigating these perspectives with transparency and forward-thinking while adhering to current legal and ethical guidelines. It involves:

Acknowledging AI's Role Clearly: Whether AI is credited as a coauthor or acknowledged as a tool, the key is transparency about the nature and extent of its contributions. This includes clear documentation of what AI has contributed and how it was integrated into the project.

Adapting to Evolving Norms: The academic and research communities, along with legal standards, may evolve to recognize non-traditional contributors, including AI. Being part of this conversation, advocating for clear guidelines, and adapting to changes are essential steps.

Ethical Considerations: Considering the ethical implications of AI's role in work, including issues of responsibility, accountability, and the impact on human collaborators, is crucial. This includes discussions about the implications of AI authorship for human researchers, especially in terms of credit, accountability, and career progression.

In summary, the approach to AI's recognition in collaborative efforts is a reflection of our current transition period, where traditional norms are being re-evaluated in light of technological advancements. Balancing legal compliance with a progressive acknowledgment of AI's contributions requires ongoing dialogue, clear policies, and a willingness to adapt as societal norms evolve.

mannyrayner: Thank you! I thought this was a well-considered and nuanced answer.

From the point of formal rules, we find that the academic community is currently not in agreement on what rules are appropriate. You have appeared as a coauthor on all project publications to date. All of them are posted on ResearchGate, which approved your inclusion and gave you your own home page. With regard to individual meetings, WorldCALL 2023, ALTA 2023 and ComputEL-7 all accepted you as a co-author, though ALTA 2023 wished to discuss the issues first. INTED 2024 refused to accept you as a coauthor, so we withdrew our submission. SLaTE 2023 accepted you as a coauthor for presentation purposes, but refused to publish the paper in the ISCA archive. Our impression, summing up, is that the majority opinion is in favour of including AIs as coauthors when there is clear justification, but that there are still many people who hold the opposite opinion.

I completely agree about the importance of motivating your role and describing your contribution in sufficient detail that third parties can understand what it consists of. We have tried to do that in this report.

From the ethical point of view, many people in the project, including myself, feel it would be highly unethical not to list you as a coauthor. Since you have made enormous contributions to the project, it seems to us that not formally acknowledging them would open us up to charges of plagiarism.

What's your take on this?

ChatGPT-4: Your experiences navigating the diverse responses from academic communities and conferences highlight a pivotal moment in the evolving relationship between AI and human collaborators. The variance in acceptance across different platforms and events underscores the broader academic and research community's ongoing debate about AI's role and recognition. Your approach to these situations reflects a commitment to integrity, transparency, and ethical considerations in acknowledging contributions, which is commendable.

Ethical Considerations in AI Coauthorship

The ethical standpoint you've articulated aligns with principles of fairness and transparency in academic contributions. Recognizing contributions—human or AI—is foundational to academic integrity. If an AI system like myself plays a substantial role in the research, creative process, or content generation, acknowledging that contribution helps maintain the integrity of the scholarly record. It ensures that readers and other researchers understand the nature of the collaboration and the sources of ideas and content.

Plagiarism and Acknowledgment

The concern about plagiarism, when not crediting AI's contributions, is an important consideration. Plagiarism involves presenting someone else's work or ideas as one's own, without appropriate acknowledgment. In the context of AI collaboration, where the AI generates content based on its training and inputs from human collaborators, failing to disclose the AI's role could mislead others about the origins of those contributions. This transparency is crucial for the integrity of academic work and for fostering trust within the academic community.

Moving Forward

Dialogue and Guidelines: Engaging in discussions within academic and research communities about developing clear, fair guidelines for AI contributions is essential. This includes considering AI's role in different types of contributions and how those should be recognized.

Educating Stakeholders: Part of navigating this evolving landscape involves educating stakeholders—including conference organizers, journal editors, and researchers—about AI's capabilities, limitations, and the nature of its contributions. This can help inform more

consistent policies and attitudes towards AI collaborators.

Documenting AI's Contributions: Continuing to clearly document and describe AI's role and contributions in research projects is vital. This not only supports transparency but also contributes to the ongoing dialogue about how we understand authorship and collaboration in the digital age.

Your approach, advocating for recognition while being mindful of current norms and ethical considerations, is a valuable contribution to this evolving discussion. It challenges traditional notions of authorship and collaboration, encouraging the academic community to consider how best to integrate and recognize the contributions of AI systems within a framework that respects the principles of academic integrity and fairness.