

Miteiru: Designing an Open-Source Subtitle-Immersion Player for Japanese, Mandarin, Cantonese and Vietnamese, Before and With AI Coding Assistants

Hocky Yudhiono

Independent researcher, Hong Kong SAR

`its@hocky.id`

8 October 2026

Abstract

Miteiru is an open-source desktop video player for learning Japanese, Mandarin, Cantonese and Vietnamese from the subtitles of real video. It segments each subtitle line into words, shows a reading over every word (furigana, pinyin, Jyutping) and a short gloss under it, and opens a dictionary entry when the learner clicks. Everything runs offline on bundled dictionaries; large language models are an optional layer that never touches the subtitle pipeline. This paper describes its design and architecture, and gives the rationale for twelve design decisions traced to the commits and issues where they were made. It argues that Cantonese needs a language mode of its own, not a Mandarin variant: in a screen of 22 YouTube videos found by Cantonese searches whose uploaders wrote Chinese subtitles, 15 were written in Standard Written Chinese and 6 in written Cantonese, so the words on screen often differ from the words spoken. Miteiru was built in two eras: 497 commits over 32 months written without AI coding assistants (December 2022 to August 2025), then 136 commits over 12 months written with them. In the second era the code grew from 9.7k to 33.6k lines and the test suite from nothing to 33 files, while the median commit grew from 13 to 89 changed lines. The paper reports these measurements, the system’s measured limits (Japanese long-vowel readings, Vietnamese segmentation), and its use: 3,983 release downloads and 117 GitHub stars, with no telemetry.

Keywords: language learning; subtitles; interactive captions; furigana; Jyutping; Cantonese; written Cantonese; AI-assisted software development

1 Introduction

Watching video in the language one is learning is popular advice, in the spirit of learning from comprehensible input (Krashen, 1985), and with target-language subtitles a well-supported one: subtitles in the spoken language help listeners learn to perceive a foreign accent, while subtitles in their own language get in the way (Mitterer and McQueen, 2009), and captioned video helps listening comprehension and vocabulary (Montero Perez et al., 2013; Vanderplank, 2016). The difficulty is the vocabulary. Television needs about 3,000 word families for 95% coverage of its words and about 7,000 for 98% (Webb and Rodgers, 2009), far beyond a beginner, and in Japanese and Chinese an unknown word cannot even be sounded out, because the script does not spell its pronunciation.

Interactive subtitles answer this by making every word a lookup. Kovacs and Miller (2014) showed that learners of Chinese who could hover over subtitle words for definitions could afterwards define more than twice as many new words as with dual subtitles, with no loss in comprehension or enjoyment. Browser extensions such as Language Reactor (Language Reactor,

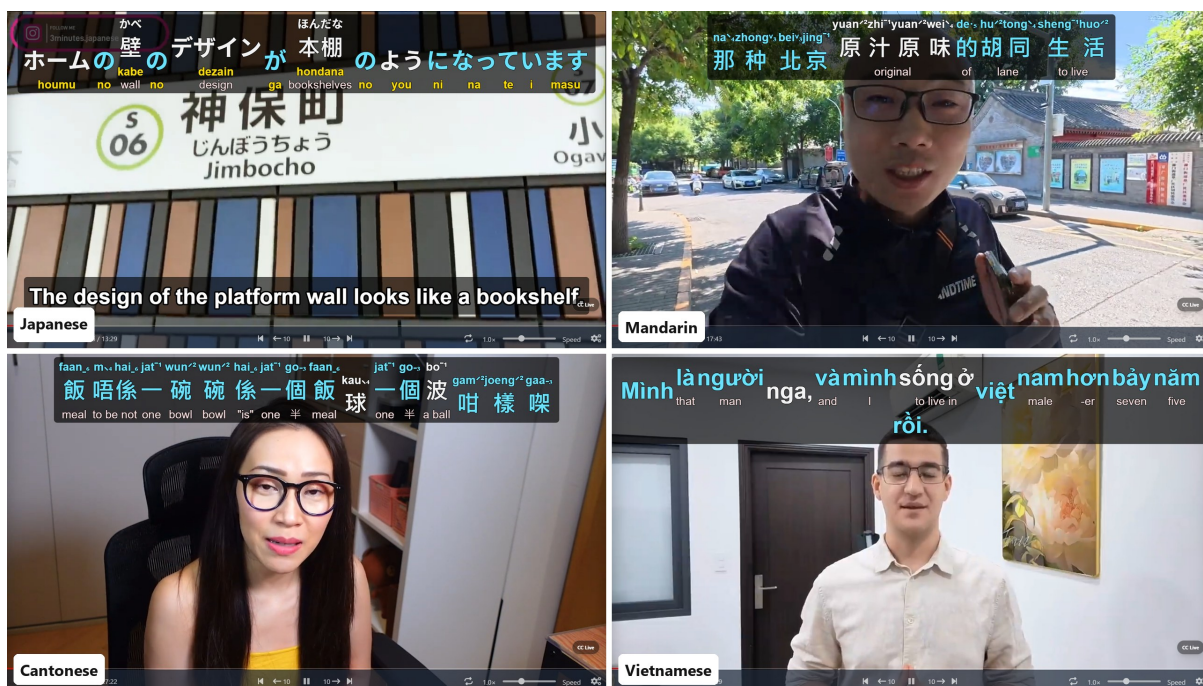


Figure 1: Miteiru 7.5.2 on one line in each language, each from a video whose uploader subtitled the speech in its own language. The primary track (top) is segmented, with readings above and glosses below; colour marks learning state (white: new; cyan: frequent in this video, Section 5.4). Japanese: furigana, romaji and an English secondary track. Mandarin and Cantonese: pinyin and Jyutping with contour tone marks (Figure 5). Vietnamese: glosses only, and two segmentation errors discussed in Section 5.3. Video frames: see Appendix A.

2026) and asbplayer (asbplayer contributors, 2026), and pop-up dictionaries such as Yomitan (Yomitan contributors, 2026), bring the idea to streaming sites and web pages.

Miteiru (*miteiru*, “watching” in Japanese) is a desktop player in this family (Yudhiono, 2026). It plays local files and YouTube videos with any subtitle file, and for four languages it segments each line into words, puts a reading above each word and a short gloss below it, colours words by how well the learner knows them, and opens a dictionary entry on click (Figure 1). It began in December 2022 as a Japanese player, added Cantonese a year later and Mandarin a month after that, and Vietnamese in 2025.

This paper is a systems paper with a short first-person account. Its contributions are:

1. **A system** (Sections 3–5): an open-source, offline subtitle-immersion player whose per-language pipelines share one rendering and lookup layer, with measured examples of where each pipeline succeeds and fails.
2. **A design rationale** (Section 7): twelve decisions, each with the alternative it rejected and the commit or issue where it was made.
3. **An argument that Cantonese is not a Mandarin setting** (Section 6), with a small screen of YouTube subtitles showing that most Cantonese videos are subtitled in Standard Written Chinese, so a learner often reads different words from the ones they hear.
4. **A longitudinal account of building it before and with AI coding assistants** (Section 8): 633 commits over 45 months, split at the month the author started using them.

The source, the screenshots’ inputs and every measurement’s data are public (Section 13).

2 Background and related work

Subtitles and vocabulary. Subtitle research distinguishes captions (same-language subtitles), translated subtitles and dual subtitles. Same-language subtitles help perceptual learning of foreign speech (Mitterer and McQueen, 2009) and, across studies, listening comprehension and vocabulary (Montero Perez et al., 2013). Dual subtitles trade form for meaning: in an eye-tracking study, bilingual subtitles beat captions for recognising word meanings but were worse for recognising word forms (Wang and Pellicer-Sánchez, 2022), and a study of the Netflix extension that became Language Reactor found dual subtitles helped vocabulary and gave the best listening comprehension (Dizon and Thanyawatpokin, 2021). Miteiru supports both: a primary track in the target language that is analysed, and an optional secondary track, usually a translation, shown as plain text.

Reading aids. For learners of Chinese, how the romanisation is shown matters. Pinyin presented at the same time as the character can interfere with learning the character (Chung, 2003); pinyin placed above the character works better than pinyin beside it, because it avoids split attention (Lee and Kalyuga, 2011a); and showing pinyin only for some words helps more experienced learners, an expertise-reversal effect, while beginners gain nothing from the reduction (Lee and Kalyuga, 2011b). For Japanese, the evidence on furigana is thin; an MA thesis found that furigana improved readers’ guesses at the meaning of unknown kanji words (Palmer, 2012). Miteiru follows the layout result (readings above, as ruby) and, for Chinese and Vietnamese, the expertise-reversal result (readings fade for known words, Section 5.4).

Interactive subtitle systems. Smart Subtitles (Kovacs and Miller, 2014) is the closest research system: hover-to-define subtitles for Chinese, evaluated against dual subtitles. Culbertson et al. (2017) had learners correct speech-recognition captions as a learning task. Practitioner tools fill the rest of the space: Language Reactor (Language Reactor, 2026) for Netflix and YouTube, asbplayer (asbplayer contributors, 2026) for any web video with a subtitle file, Yomichan and its successor Yomitan (FooSoft, 2026; Yomitan contributors, 2026) as pop-up dictionaries, Animelon (ani, 2026) as a streaming site with interactive Japanese subtitles, Migaku (Migaku, 2026) as a commercial suite, and Anisubber (ookii-tsuki, 2026), a player that shows Japanese subtitles with readings and which inspired Miteiru. Miteiru differs in three ways: it is a desktop player for local files, so it works with the subtitle files learners actually find (ASS fansubs, SRT, LRC lyrics); it treats Cantonese as a language of its own; and it works offline, with bundled dictionaries and no account.

Method. Miteiru was built partly for the author’s own study and partly for other learners. HCI calls design by a designer who is also the long-term user *autobiographical design* (Neustaedter and Sengers, 2012), and its tensions, between the designer’s needs and generality and between closeness and rigour, are well described (Desjardins and Ball, 2018). This paper uses the method lightly: the repository’s history is the main record, and the account of why something was built is checked against it. Mining a repository has its own perils, among them commits that are not units of work and histories rewritten by merges (Kalliamvakou et al., 2014); Section 8 uses the first-parent history of the main branch and says where the numbers are proxies.

AI-assisted development. Evidence on AI coding assistants points in several directions. In a controlled task, developers with GitHub Copilot finished 55.8% faster (Peng et al., 2023); in a randomised trial with experienced open-source developers working on their own repositories with early-2025 tools, mostly Cursor, they were 19% slower while believing they were faster (Becker et al., 2025). Across open-source projects that adopted Cursor, velocity rose sharply

but briefly, while static-analysis warnings and code complexity rose for good (He et al., 2026). Agent-authored pull requests are now common (Li et al., 2026; Watanabe et al., 2026), but commits are hard to attribute: counting a single signal such as a commit trailer undercounts agent commits by about thirty times (Khosravani and Mockus, 2026). Section 8 is one project’s record across the change, not a test of these findings.

3 Miteiru in use

A session starts on the home screen, where the learner picks one of five language stacks: Japanese with Kuromoji or with MeCab, Cantonese, Mandarin, or Vietnamese. The choice loads that language’s dictionaries and sets the tokenizer for everything that follows.

Watching. The learner drops a video file onto the window, or pastes a YouTube URL, then drops a subtitle file and chooses whether it is the primary track, analysed for learning, or the secondary track, shown for reference. A subtitle sitting next to the video with the same name, or the same name plus `.en`, loads automatically. For YouTube, Miteiru lists the video’s subtitle tracks with yt-dlp, manual tracks first and machine captions marked, and downloads the chosen one. The primary line appears at the top of the frame with each word’s reading above it and a short gloss below it (Figure 1); the secondary line appears at the bottom.

Looking up. Clicking a word opens a dictionary panel over the video (Figure 2a): the headword with its reading, dictionary senses, a pitch-accent contour for Japanese, a conjugation table for Japanese verbs and adjectives, stroke-order animations for single characters, the learner’s own notes, and an *Export to Anki* button. Clicking a Japanese word looks up its dictionary form, so *norimasu* opens *noru* “to ride”. Right-clicking cycles the word’s learning state, which changes its colour everywhere it appears.

Practising. Keys replay the current line (R) or shift either track by 100 ms ([and]); an auto-pause mode stops at the end of each line. A karaoke mode shows lyrics as three scrolling lines, and lyrics for a song can be searched on LRCLIB (tranxuanthang, 2026) by the video’s title. Live Captions (Section 4) feed the operating system’s own speech recognition into the same tokenized overlay, for audio with no subtitles.

Studying. The Learn page (Figure 2b) takes pasted text, one sentence a line, and renders each sentence with the same components as the video overlay. It adds a catalogue of 831 Japanese grammar points by JLPT level, and, with an OpenRouter key, AI translation in formal, neutral and casual registers, sentence analysis, and Anki sentence cards. A Flashcards page reviews saved words.



Figure 2: (a) Clicking *hondana* “bookshelf” in the Japanese line of Figure 1: JMdict senses, the reading, and an OJAD-style pitch-accent contour (*atamadaka*, accent on the first mora) from Kanjium data. (b) The Learn page’s analyzer and sentence list: pasted sentences are segmented and rendered like subtitles, with glosses above in this layout. The verb phrase *natte imasu* is one merged token.

4 Architecture

Miteiru is an Electron application (OpenJS Foundation, 2026) built with Nextron (saltyshiomix, 2026): a Node.js main process and a React renderer compiled as a static Next.js export (Figure 3). It ships for Windows, macOS and Ubuntu, and is licensed CC BY-NC-SA 4.0.

One analysis path. All tokenization runs in the main process behind a single function, `analyzeText`, which dispatches on the active language. The renderer calls it over IPC for each subtitle line; the Learn page calls it for pasted text; and a local HTTP server exposes it to other programs. The server binds to 127.0.0.1 on a port chosen by the operating system, requires a random token written to a registration file in the user’s data directory, and offers `/health`, `/analyze` and `/analyze-batch`. It cannot switch the language: a request that names a different language gets an error, so outside tools always see exactly what the window shows. The token data in Sections 5 and 6 were collected through this server.

Language plugins and assets. Each language is a plugin with dependencies: the Japanese, Mandarin and Cantonese plugins depend on a shared Han-character pack (KanjiVG and Make Me a Hanzi stroke data, WaniKani radicals). Plugins ship as asset packs bundled into the installer; on first use, each JSON dictionary is converted into a LevelDB database in the user’s data directory, so later lookups are prefix range scans on disk rather than searches in memory.

Subtitles. Subtitle files are decoded after detecting their encoding, which matters for the Shift-JIS, GBK and Big5 files common among older Asian subtitles, then parsed (SRT, VTT, ASS, LRC, and a JSON lyrics format). By default each file is first normalised: lines are sentence-cased, a step meant for all-caps English subtitles, and bracketed sound descriptions and speaker labels are removed. Each line ends 240 ms late and never overlaps the next, and the current line is found by binary search on start time.

Local media. Local video reaches Chromium through a custom `miteiru://` protocol that serves files with HTTP byte ranges, so seeking works on large files. Files Chromium cannot decode (HEVC, or several audio tracks) are probed with `ffprobe` and offered a remux or re-encode with `ffmpeg`; embedded subtitle tracks are extracted the same way.

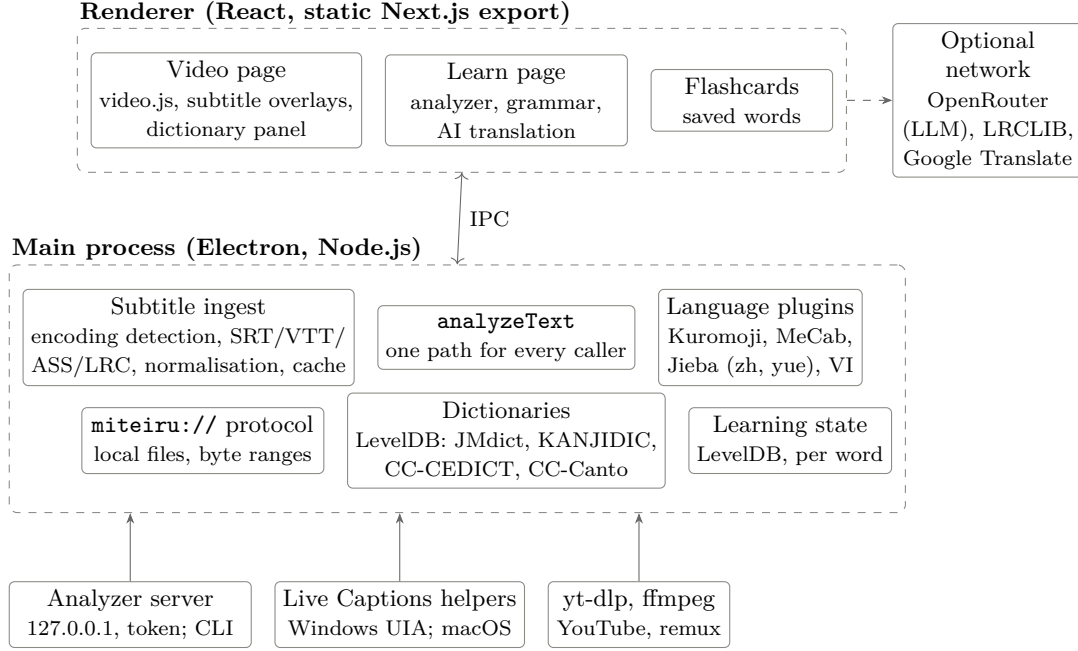


Figure 3: Miteiru’s architecture. Every tokenization, from subtitle lines, the Learn page, live captions or the local analyzer server, goes through one function in the main process. Network services are optional; the subtitle pipeline runs offline.

Live Captions. Rather than bundle a speech recogniser, Miteiru reuses the operating system’s. On Windows 11, a small .NET helper starts the built-in Live Captions app and reads its caption text through UI Automation every 80 ms; on Apple Silicon Macs running macOS 26, an Objective-C helper calls the system’s private live-transcription framework with the locale of the active language. Both helpers page the rolling transcript into subtitle-sized windows of at most 70 characters and print one JSON object per line, which the main process feeds into the ordinary subtitle path.

5 Four languages, four pipelines

The four languages differ in every respect that matters to a subtitle reader (Table 1): whether words are written with spaces, whether the script shows pronunciation, and what a reading aid has to show. Figure 4 shows each pipeline’s output on a real subtitle line.

5.1 Japanese

Japanese is segmented by `kuromoji.js` (takuyaa, 2026) through a small wrapper, with an IPADIC-style dictionary (MeCab project, 2026); MeCab (Kudo et al., 2004; Kudo, 2026) with any of its dictionaries, UniDic (Den et al., 2008; National Institute for Japanese Language and Linguistics, 2026) among them, is an option. Each token’s reading is converted to hiragana and then split over the word in at most three parts: leading kana, a kanji core, and trailing okurigana, each part keeping its own reading. So *toriatsukai* gets one reading over its kanji core and none over its last kana, and *okaasan* gets one only over the kanji for “mother” (Figure 4).

A second pass merges a verb with the auxiliaries and particles that follow it, using a deconjugator to check that the merged form still reduces to the verb’s dictionary form. *Taberarenakatta* “could not eat” becomes one token whose lemma is *taberu*. Because learning state and frequency are counted by lemma, every form of a verb shares one state, and clicking any form opens the dictionary entry. The dictionary panel adds a conjugation table generated from the lemma, with

Table 1: The four language stacks in Miteiru 7.5.2. Dictionary sizes are the bundled files’ entry counts.

	Japanese	Mandarin	Cantonese	Vietnamese
Script	kanji + kana	Han (Simplified or Traditional)	Han (Traditional, with Cantonese-only characters)	Latin with diacritics
Word boundaries	none	none	none	spaces between syllables, not words
Reading shown	furigana (hiragana), romaji	pinyin, 4 tones + neutral	Jyutping, 6 tones	none; gloss in its place
Segmenter	Kuromoji (takuyaa, 2026), or MeCab (Kudo et al., 2004)	Jieba (fxsjy, 2026; messense, 2026), Mandarin dictionary	Jieba, Cantonese dictionary	backward maximum matching
Readings from	tokenizer	pinyin-pro (zh-lx, 2026)	to-jyutping (CanCLID, 2026)	—
Dictionary	JMdict (Breen, 2004), 206,387 entries; KANJIDIC (Electronic Dictionary Research and Development Group, 2026)	CC-CEDICT (MDBG, 2026), 122,431 words	CC-Canto (Pleco Software, 2026) with CantoDict compounds, 95,399 words	VNEDICT (Denisowski, 2019), 54,345 headwords
Extras	lemma lookup, conjugation tables, pitch accent (9,255 words, Kanjium (mifunetoshiro, 2026)), 831 JLPT grammar points	Traditional to Simplified conversion	characters kept as written	—

a “when to use” note per form, and a pitch-accent contour in the Tokyo pattern that the NHK accent dictionary records (NHK Hōsō Bunka Kenkyūjo, 2016), drawn from Kanjium’s data (mifunetoshiro, 2026). Pitch accent is rarely taught, but training improves learners’ identification of it (Shport, 2016), and even advanced learners’ perception of it depends on their vocabulary (Goss and Tamaoka, 2019), which is an argument for showing it word by word.

The readings have one systematic error. Kuromoji’s pronunciation field writes a long vowel with a length mark, and converting that mark to hiragana after an *o*-row kana gives *u*: *ookii* “big” is shown as *oukii*, *toori* “street” as *touri*, and *Oosaka* as *Ousaka* (Figure 4). For most words the two spellings sound alike, since *ou* is also pronounced as a long *o*, but the furigana is misspelt, and the same reading feeds the pitch-accent lookup, which then misses those words. Using the dictionary reading for furigana, and keeping the pronunciation for romaji (where the particle *ha* should read *wa*), would fix it.

5.2 Mandarin

Mandarin is segmented by Jieba (fxsjy, 2026), through a Rust port (messense, 2026), with a dictionary of about 584,000 words that holds both simplified and traditional forms. Pinyin comes from pinyin-pro (zh-lx, 2026), called once per word, so polyphonic characters are resolved by the word they are in. Traditional subtitles are converted to simplified by default (a setting reverses it), since most learners’ dictionaries and textbooks are simplified.

Tones can be shown two ways (Figure 5). The default replaces the tone number with a small

Japanese Kuromoji, then okurigana alignment and verb merging; the lemma (below) is what a click looks up

ホーム	の	壁	の	デザイン	が	本棚	の	よう	に	なっています
		かべ wall		design		ほんだな bookshelves				→ なる

Ruby segments inside one token (dashed boxes); red: readings taken from Kuromoji's pronunciation field

とりあつか 取り扱い	かあ お母さん	た 食べられなかった	おう 大きい	おうさか 大阪
取り扱い	お母さん	merged; lemma 食べる	expected おお	expected おおさか

Mandarin Jieba, Mandarin dictionary; pinyin per word (pinyin-pro); Traditional converted to Simplified (OpenCC)

nà zhōng 那种	běi jīng 北京	yuán zhī yuán wèi 原汁原味	de 的	hú tòng 胡同	shēng huó 生活
		original	of	lane	to live

Cantonese Jieba, separate Cantonese dictionary; Jyutping per word (to-jyutping); characters kept as written

faan6 飯	m4 hai6 唔係	jat1 一	wun2 碗	wun2 碗	hai6 係	jat1 一	go3 個	faan6 飯	kau4 球	jat1 一	go3 個	bo1 波	gam2 joeng2 咁樣	gaa3 㗎
meal	to be not	one	bowl	bowl	is	one		meal		one		a ball		

Vietnamese backward maximum matching over syllables (VNEDict); the gloss takes the reading's place

Mình	là	người	Nga,	và	mình	sống	ở Việt Nam	hơn	bảy	năm	rồi.
	that	man	no match	and	I	life	in Vietnam	-er	seven	five	no match

Without the final full stop, matching from the right takes the longest entry:

bảy	năm rồi
seven	last year

(here: "seven years now")

Figure 4: Miteiru's own output on the subtitle lines of Figure 1, collected through its analyzer server. Boxes are tokens; small text above is the reading and below is the gloss the overlay shows. Red marks an error. Japanese readings come from Kuromoji's pronunciation field, which writes the long *o* of *ooki* and *Oosaka* as *ou*. The Vietnamese line keeps the punctuation attached to the syllable, and matching from the right picks *năm rồi* "last year" where the speaker means "seven years now".

contour glyph and the number, a level bar for tone 1, a rising stroke for tone 2, a dip for tone 3 and a falling stroke for tone 4; the alternative is standard pinyin with diacritics. Miteiru used contour marks for Jyutping first and extended them to Mandarin in August 2026; they draw each tone's pitch shape beside the syllable rather than over a vowel.

5.3 Vietnamese

Vietnamese has the opposite problem from Chinese and Japanese: spaces are everywhere, but they separate syllables, not words, and about 85% of word types have two or more syllables (Nguyen et al., 2018). Research segmenters use trained models (Nguyen et al., 2018; Vu et al., 2018); Miteiru uses backward maximum matching over syllables against VNEDICT (Denisowski, 2019), taking from the right end the longest run of syllables that is a dictionary headword. The written form already tells the learner how to pronounce a word, so the reading slot above the word holds its gloss instead.

The method finds most compounds, *ở Việt Nam* "in Vietnam" among them, and fails in three measured ways (Figure 4). Lookup is case-sensitive and punctuation stays attached to the syllable, so a comma or full stop hides the word. Greedy matching takes the longest entry even when the sentence means something else: *bảy năm rồi* "seven years now" becomes *bảy* plus *năm rồi* "last year". And the default normalisation, meant for all-caps English subtitles, lowercases every line, which turns *Việt Nam* into two separate dictionary misses; this is visible in Figure 1, where *nam* is glossed "male". The shortest-sense gloss shows the cost of having no context: *năm* is glossed "five" because the dictionary lists "five" before "year".



Figure 5: Tone display. Top: Mandarin with contour marks (the default). Middle: the same line with standard pinyin diacritics. Bottom: Cantonese Jyutping with contour marks for its six tones.

5.4 Display and learning state

Every language renders through the same component. A token is a nested ruby: the reading above the word, romaji below it for Japanese, and the gloss below or above the whole. Chromium left-aligns ruby annotations under some conditions, so Miteiru lays each ruby out with flexbox to centre the reading; and because `-webkit-text-stroke` breaks CJK ruby on Windows, the subtitle outline is drawn with eight to twenty-four text shadows instead.

Each word has a learning state: new, learning or known, cycled by right-click and stored per language and word. A fourth state is inferred: words that are among the most frequent in the current subtitle file (by default the top 30% of word types that occur more than once) and that the learner has not rated are coloured as frequent, so the words worth learning first stand out. For Chinese and Vietnamese the display follows the expertise-reversal result (Lee and Kalyuga, 2011b): a word marked known loses its reading, and new and frequent words always show their gloss. Japanese words change colour only; their furigana does not fade.

Speed. Tokenizing a whole subtitle file through the analyzer server, readings included but per-word glosses not, took a median of 53, 44, 44 and 43 ms for the Japanese, Mandarin, Cantonese and Vietnamese files of Figure 1 (211, 411, 356 and 426 lines; five runs each, Intel Core Ultra 7 265K). Loading a subtitle file is therefore dominated by the per-word dictionary queries for glosses, which run one by one and which this paper did not time.

6 Why Cantonese is its own language here

Miteiru’s Cantonese mode came before its Mandarin mode, and it was never a setting of it. The reasons are linguistic, orthographic and social.

A different spoken language. Cantonese and Mandarin are not mutually intelligible in speech. Cantonese has six tones to Mandarin’s four, different finals and initials, and its own grammar and core vocabulary (Matthews and Yip, 2011): its possessive particle, perfective marker, third-person pronoun, locative verb, negators and copula are different words, not different pronunciations of the Mandarin ones (Figure 6). It has its own romanisation, Jyutping (Linguistic Society of Hong Kong, 1993), its own corpora and tools (Luke and Wong, 2015;

Meaning	Spoken Cantonese, written	Standard Written Chinese, read in Cantonese	SWC read in Mandarin
We're going to eat roast goose today	我哋今日去食燒鵝 ngo5 dei6 gam1 jat6 heoi3 sik6 siu1 ngo2	我們今天去吃燒鵝 ngo5 mun4 gam1 tin1 heoi3 hek3 siu1 ngo2	wǒ men jīn tiān qù chī shāo é
He didn't come to see the film	佢冇嚟睇戲 keoi5 mou5 lai4 tai2 hei3	他沒有來看電影 taa1 mut6 jau5 loi4 hon3 din6 jing2	tā méi yǒu lái kàn diàn yǐng
watching (this app's name)	睇緊 tai2 gan2	在看 zoi6 hon3	zài kàn
Cantonese	嘅 咗 佢 喺 冇 啲 唔 係 嘢 睇		
	ge3 zo2 keoi5 hai2 mou5 di1 m4 hai6 je5 tai2		
SWC	的 了 他 在 沒有 些 不 是 東西 看		
in Mandarin	de le tā zài méiyǒu xiē bù shì dōngxi kàn		
in Cantonese	dik1 liu5 taa1 zoi6 mut6 jau5 se1 bat1 si6 dung1 sai1 hon3		
	poss. perf. he/she at not have some not is thing look		

Figure 6: Spoken Cantonese written down, against Standard Written Chinese (SWC) with the same meaning, as Miteiru reads them. Top: whole sentences, including the app’s own name, “watching”. Bottom: ten function and core words whose Cantonese form is a different word from SWC’s. Jyutping and pinyin are Miteiru’s output through its analyzer server, except the particle readings of the SWC perfective (*le*) and possessive (*de*), which it gives in isolation as *liao3* and *de*.

Lee et al., 2022), and its own dictionaries, though for language technology it remains under-resourced, colloquial and code-switched (Xiang et al., 2025); Miteiru uses CC-Canto (Pleco Software, 2026) with compounds from CantoDict, and a segmentation dictionary of 35,000 Cantonese words separate from the Mandarin one.

Two ways to write it. Cantonese speech can be written in two ways. Standard Written Chinese (SWC) is the written language of Hong Kong schools, newspapers and government; it is grammatically Mandarin, and Cantonese speakers read it aloud in Cantonese pronunciation. Written Cantonese writes down the spoken language, with characters that SWC does not use, and has grown into a vernacular with many, though not all, traits of a standard language (Snow, 2004, 2008). The two are not spellings of the same sentence. “We’re going to eat roast goose today” is *ngo5dei6 gam1jat6 heoi3 sik6 siu1ngo2* in speech; its SWC version uses a different pronoun, a different word for “today” and a different verb for “eat”, and read in Cantonese it comes out as *ngo5mun4 gam1tin1 heoi3 hek3 siu1ngo2*, which no one says (Figure 6).

Subtitles are usually SWC. Hong Kong subtitling is diglossic: SWC is the norm for subtitles, and written Cantonese appears for low-register speech such as swearing (Yau, 2012; Chen, 2004). To see what a learner meets on YouTube today, I screened videos found by five Cantonese-language searches (vlogs, cooking, street interviews) filtered to videos with subtitles. Of 37 videos, 27 had an uploader-made subtitle track in a Chinese language code; I downloaded the 24 under 30 minutes long and counted, in each, 26 characters that mark written Cantonese (the forms in Figure 6 and sentence particles) against 15 SWC forms (their counterparts and common function words). The score is the Cantonese share of all marker hits.

Fifteen of the 22 tracks were SWC and six were written Cantonese, four of those from one channel (Table 2); only one mixed the two. A full-time Cantonese teacher’s vlog, among others, was subtitled in SWC (3 Cantonese markers against 152 SWC ones). The screen is small and its searches are not a sample of YouTube, and I did not verify the spoken language of every video; it shows only that SWC subtitles over Cantonese speech are common enough that a Cantonese

Table 2: Register of uploader-made Chinese subtitles on 22 YouTube videos found by Cantonese searches (two of the 24 downloaded tracks had no marker hits). The scores are bimodal: a subtitler writes one register or the other.

Register (Cantonese share of marker hits)	Videos	Channels	Marker hits, median
Standard Written Chinese (≤ 0.03)	15	14	189
Mixed (0.28)	1	1	72
Written Cantonese (≥ 0.95)	6	3	266

mode must expect them.

What this means for a learner, and for Miteiru. With SWC subtitles, a learner of Cantonese reads one sentence and hears another. Miteiru’s Cantonese mode segments and reads whatever is written, so on an SWC line it shows the literary reading of each SWC word, *taa1* for “he” where the speaker says *keoi5*. That is correct as a reading of the text and wrong as a guide to the audio. Miteiru handles it in three ways today: it keeps Cantonese apart, so Cantonese-only characters, Jyutping and Cantonese compounds are always available; it keeps characters as written instead of converting them, as the Mandarin mode does; and its AI translation is told to write “written Cantonese in Traditional Chinese” and to prefer distinctly Cantonese wording over SWC, so a learner can produce the spoken form. Aligning an SWC subtitle with what was actually said, for example by showing a Cantonese live caption beside it, is open (Section 11).

7 Design decisions

Table 3 lists twelve decisions that shaped Miteiru. Each was checked against the repository: the date is the commit, release or issue where the decision shows.

Three of them need more than a table row.

Offline first, AI optional. Miteiru’s core was designed before large language models were practical for this job, and it stayed deterministic after they were. Segmentation, readings, glosses and dictionary lookup use tokenizers and dictionaries; an LLM is called only when the learner asks for a translation, an analysis, notes, example sentences or an Anki card, and only with the learner’s own API key. Where an LLM’s output meets the rest of the system, a symbolic tool bounds it: conjugation examples are accepted only for the exact forms the conjugation library produced, and when an LLM segments a sentence for an Anki card, Miteiru escapes the text and builds the ruby markup itself rather than accepting HTML from the model.

Anki instead of an SRS. Miteiru had its own spaced-repetition page from June 2024 until May 2025, when it was removed. Spaced repetition works (Cepeda et al., 2006), and good schedulers exist (Settles and Meeder, 2016; Ankitects Pty Ltd and contributors, 2026); building another one did not help learners who already used Anki. Miteiru instead writes Anki’s import format with deterministic note IDs, derived from the language and the normalised word, so exporting the same word again updates its note. A simple flashcards page remains for reviewing saved words in the app.

Borrowing the OS’s ears. Live Captions reuse the speech recogniser the operating system already has, which costs no download and covers whatever audio is playing, including streams and calls. The cost is that both integrations rest on interfaces Microsoft and Apple do not document for this use: reading another app’s window through accessibility APIs on Windows, and a private framework on macOS. Either can break with an operating-system update.

Table 3: Design decisions, the alternative each rejected, and where the history shows them. Era: **P** pre-AI (before September 2025), **A** AI-assisted.

	Decision	Instead of	Why	Evidence
P	A desktop player for local files	a browser extension over streaming sites	learners’ subtitles are files (ASS fansubs, SRT, LRC); offline; any codec Chromium plays	first commits, Dec 2022
P	Kuromoji built in, MeCab optional	requiring a MeCab install	MeCab had to be installed and located by hand; the issue asks for “better UX”	issue 32, PR 35, Jul 2023
P	Dictionaries bundled, cached in LevelDB	an online dictionary API	offline, private, fast prefix lookup	asset release, Aug 2023
P	Cantonese as its own language	a Mandarin setting	separate words, readings, dictionary and segmentation (Section 6)	Cantonese mode 19 Dec 2023; Mandarin 15 Jan 2024
P	Readings above every word, glosses below	a separate dictionary pane only	readings above the character beat readings beside it (Lee and Kalyuga, 2011a)	first releases
P	Learning state by colour; readings fade for known Chinese words	always-on readings	expertise reversal (Lee and Kalyuga, 2011b); frequent words stand out	learning state, Feb 2024 (issue 57)
P	Export to Anki; drop the in-app spaced repetition	building an SRS	Anki already schedules well (Settles and Meeder, 2016); the in-app SRS was removed as a “gimmick”	SRS Jun 2024 (issue 59), removed May 2025
A	Deterministic pipeline; LLMs optional and outside it	LLM tokenization or readings	offline, reproducible, free; an LLM never decides a reading in the subtitle path	AI features from Dec 2025
A	LLM output constrained by symbolic tools	free-form generation	example sentences must use a conjugated form from the table; ruby HTML is built locally from the LLM’s segments	v6.5.0, v7.4.3, Jul 2026
A	Reuse the operating system’s speech recognition	bundling an ASR model	no model download; same overlay for any audio; cost: undocumented interfaces	v6.0.0 Apr 2026; v7.5.1 Aug 2026
A	One analysis path, exposed on localhost	a separate API implementation	tools see what the window shows; the API cannot switch language	v6.1.0, Jun 2026
A	Deterministic Anki note IDs	new notes on each export	re-importing updates notes instead of duplicating them	v6.2.0, Jul 2026

8 Building it before and with AI coding assistants

Miteiru’s first commit came weeks after ChatGPT’s public release, but for its first 32 months it was written without AI coding assistants. The author began using them in September 2025, mainly Cursor (Anysphere, Inc., 2026). The history is therefore a natural record of one developer’s project across the change. This section compares the two eras.

Data. The analysis uses the 633 commits on the first-parent history of the main branch, from 18 December 2022 to 2 September 2026 (737 commits including merged branches). For each commit it counts changed lines in hand-written files, excluding the lock file, generated and bundled data, and images; code size is the line count of source files (TypeScript, JavaScript, Python, C#, Objective-C, CSS) at the end of each month. The era boundary is 1 September 2025, from the author’s account. The repository itself first shows AI involvement later: the first commit with a Cursor co-author trailer is dated 13 June 2026 and the first Cursor configuration file 11 July 2026, and only 36 commits carry the trailer. Trailers undercount agent commits (Khosravani and Mockus, 2026), so the boundary rests on the author’s account, with the repository’s traces as a lower bound.

What changed. The AI-assisted era wrote more code in less time (Table 4, Figure 7). In 12 months the code grew from 9.7k to 33.6k source lines, 3.5 times its size after 32 months of hand-written work. The commits became bigger rather than more frequent: the median commit changed 89 lines instead of 13, and the 90th percentile 970 instead of 156. The project also gained what it never had: a test suite (33 test files, 260 test cases, run in continuous integration), release notes for every version from 6.0.0, a JSON schema for its dictionaries, and escaping where dictionary and LLM text becomes HTML. Commit messages changed from terse labels (“Add video page”, “Update package”) to sentences with bodies: in August 2026 the median subject was nine words.

The feature mix changed too (Figure 7). The pre-AI era built the core a learner touches every minute: the player, Japanese segmentation and readings, the dictionary panel, verb merging, kanji pages, YouTube support, Cantonese and Mandarin, learning states and karaoke. The AI-assisted era added breadth around it: Vietnamese, Live Captions on two operating systems, the analyzer server, AI translation and notes, JLPT grammar study, conjugation tables, pitch accent, Anki decks for notes, sentences and grammar, and LRCLIB lyrics.

What the history cannot show. The comparison confounds AI assistance with everything else that changed. Activity had nearly stopped in the eight months before September 2025 (16 commits from January to August 2025), so the second era also measures renewed attention. Lines of code measure size, not value, and larger commits are partly a change of habit. The code written in the second era has not had time to show its maintenance cost, which is where the strongest evidence on these tools points (He et al., 2026; Liu et al., 2026). Several bugs found while preparing this paper sit in code from both eras (Section 10). What the record does show is a single developer’s project whose second year with AI assistance produced more than its first three without, while its core design decisions were made before.

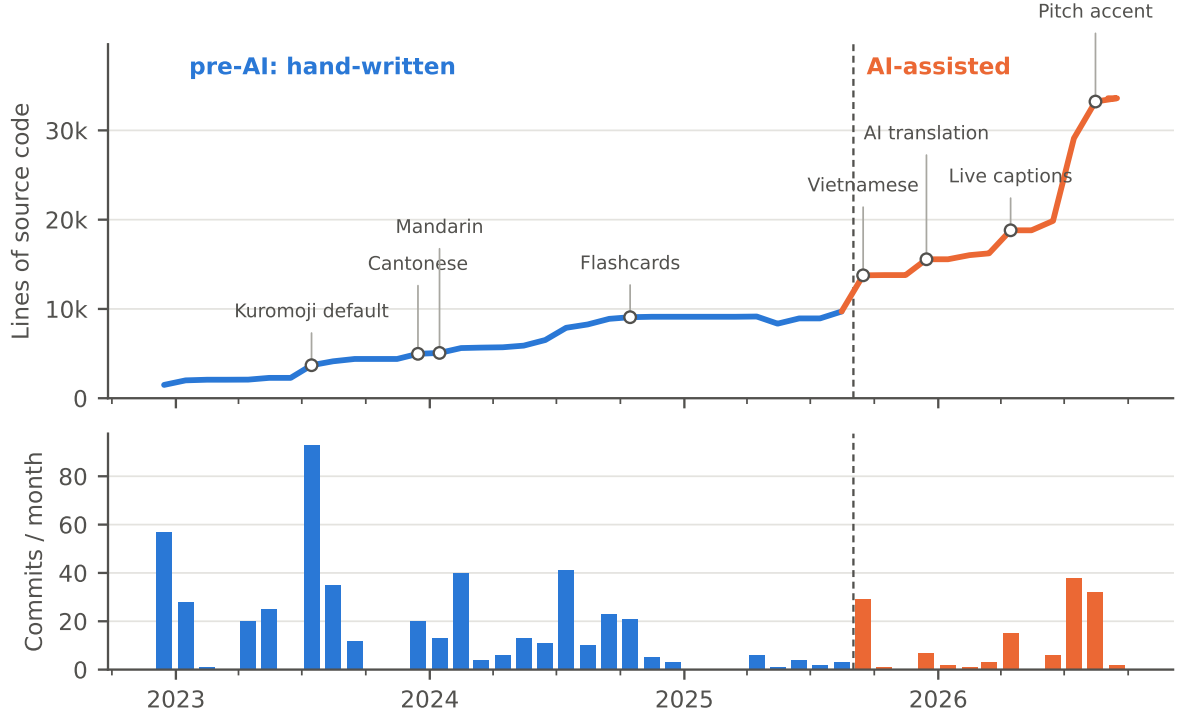


Figure 7: Code size at the end of each month (top) and commits per month (bottom), pre-AI in blue and AI-assisted in orange. The dashed line is September 2025. Labels mark when each language and some major features arrived.

Table 4: The two eras. Changed lines exclude the lock file, generated data and images.

	Pre-AI	AI-assisted
	Dec 2022 – Aug 2025	Sep 2025 – Sep 2026
Span (days)	959	364
Commits	497	136
Days with a commit	105	38
Commits per active day	4.7	3.6
Lines changed per commit (median / mean / p90)	13 / 57 / 156	89 / 308 / 970
Code lines added / deleted	19,003 / 9,296	32,880 / 9,059
Source lines at the era’s end	9,711	33,600
Test files at the era’s end	0	33
Commit subject, median words	3	4
Commits with a message body	3%	33%
Commits with a Cursor co-author trailer	0	36
Releases / release downloads	40 / 2,877	16 / 1,106

9 Deployment

Miteiru has no telemetry, so its use can be seen only on GitHub. As of 8 October 2026 the repository had 117 stars (1 given in 2022, 31 in 2023, 52 in 2024, 23 in 2025, 10 in 2026), 4 forks and 7 watchers. Its 56 application releases had been downloaded 3,983 times: 1,933 Windows, 1,543 Linux and 347 macOS assets, plus 160 other files; the separate language-asset release had 586 downloads. Release assets have been replaced over the years, so these are lower bounds. Releases from the pre-AI era account for 2,877 downloads and those from the AI-assisted era for 1,106; the later releases have had less time.

The issue tracker served mostly as the author’s own backlog: of 52 issues, 43 were opened by the author. Six other people opened the remaining nine, about build failures on Linux and Windows, YouTube subtitles, MKV playback, and Cantonese segmentation and dictionary sources, and one of them contributed a pull request that improved YouTube subtitles. The oldest open issue, a request to support Korean, has 13 comments.

10 Limitations

No learner study. This paper describes a system and its history; it does not measure learning. The design follows published results on subtitles and reading aids (Section 2), but whether Miteiru’s combination of them helps learners, and how much the fading readings help, is untested.

Pipeline errors. The errors in Figure 4 are systematic: Japanese long-*o* readings, Vietnamese punctuation, case and greedy matches, and glosses chosen by length rather than context. Japanese merges most verb forms but not all (*itta* “went” stayed two tokens). Glosses take the shortest dictionary sense, which is often not the sense in the sentence.

Defaults that remove text. Two subtitle normalisations are on by default and were designed for English subtitles. Sentence-casing lowercases every line, not only all-caps ones, which breaks Vietnamese lookups. Removing hearing-impaired annotations deletes text inside brackets, including the corner brackets that Japanese subtitles use for quoted speech.

Smaller defects found while preparing this paper. Running the released build on a fresh profile showed that the secondary track’s default style is the primary track’s, so translations overlap the primary line until the style is reset; conjugation tables list bare stems as separate forms; a dictionary entry with no senses is shown as an empty list rather than as missing; and the McCab path chosen in the interface is not passed to the tokenizer. None of them affects the measurements above.

The author is the designer, user and analyst. The account of why decisions were made is the author’s, checked against the repository but not independent of it (Desjardins and Ball, 2018). The era boundary is the author’s account; the repository’s own traces start later.

11 Future work

The most useful study is the one this paper lacks: learners using Miteiru against a dual-subtitle baseline, in the manner of Kovacs and Miller (2014), with a condition for fading readings. For Cantonese, the open problem is the gap between SWC subtitles and speech; showing the operating system’s Cantonese live caption beside an SWC subtitle, or aligning the two, would let a learner see both. Korean, the longest-standing request, needs a morphological analyser

and a reading aid that Hangul does not need for pronunciation but learners may want for Sino-Korean words. And the Japanese and Vietnamese errors above have direct fixes: the dictionary reading instead of the pronunciation field, and punctuation- and case-aware matching.

12 Conclusion

Miteiru is an attempt to make every word of real video a lookup, in four languages whose scripts hide pronunciation or word boundaries in different ways. Its main design choices are an offline, deterministic pipeline with AI kept to an optional layer, reading aids above the word that fade as words are learned, and a Cantonese mode that is not a Mandarin setting, because Cantonese video is often subtitled in a written language its speakers do not speak. Its history adds a small, concrete case to the evidence on AI-assisted development: the core design came from two and a half years of hand-written work, and the following year with AI assistance tripled the code and added tests, documentation and breadth. Whether that year’s code ages as well as the first is the next thing its history will show.

13 Availability

Miteiru’s source and releases are at <https://github.com/hockyy/miteiru> (Yudhiono, 2026). This paper’s source, its data (per-commit statistics, monthly code size, the analyzer’s token output for every example and its timings, and the Cantonese subtitle screen) and the sources of its diagrams are at <https://github.com/hockyy/miteiru-paper>.

Acknowledgements

The YouTube creators whose videos appear in Figures 1–5 are credited in Appendix A. The repository survey, measurements, screenshots and draft of this paper were prepared with Claude (Anthropic) under the author’s direction; the author checked the claims and is responsible for the content.

References

- Animelon. <https://animelon.com/>, 2026. Website for watching anime with Japanese and English subtitles for learners. Accessed: 2026-10-08.
- Ankitects Pty Ltd and contributors. Anki: Powerful, intelligent flashcards. <https://apps.ankiweb.net/>, 2026. Open-source spaced-repetition software (AGPL-3.0); source at <https://github.com/ankitects/anki>. Accessed: 2026-10-08.
- Anyosphere, Inc. Cursor. <https://cursor.com/>, 2026. AI coding editor and agent. Accessed: 2026-10-08.
- asbplayer contributors. asbplayer: Browser-based media player and Chrome extension for subtitle sentence mining. <https://github.com/asbplayer/asbplayer>, 2026. AGPL-3.0. Repository formerly at github.com/killergerbah/asbplayer. Accessed: 2026-10-08.
- Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein. Measuring the impact of early-2025 AI on experienced open-source developer productivity, 2025.
- Jim Breen. JMdict: a Japanese-multilingual dictionary. In *Proceedings of the Workshop on Multilingual Linguistic Resources*, pages 65–72, Geneva, Switzerland, 2004. COLING. URL <https://aclanthology.org/W04-2209/>.

- CanCLID. to-jyutping: Cantonese pronunciation automatic labeling tool. <https://github.com/CanCLID/to-jyutping>, 2026. GitHub repository (BSD-2-Clause). Accessed: 2026-10-08.
- Nicholas J. Cepeda, Harold Pashler, Edward Vul, John T. Wixted, and Doug Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132(3):354–380, 2006. doi: 10.1037/0033-2909.132.3.354.
- Chapman Chen. On the Hong Kong Chinese subtitling of English swearwords. *Meta*, 49(1): 135–147, 2004. doi: 10.7202/009029ar.
- Kevin K. H. Chung. Effects of pinyin and first language words in learning of Chinese characters as a second language. *Journal of Behavioral Education*, 12(3):207–223, 2003. doi: 10.1023/A: 1025560327860.
- Gabriel Culbertson, Solace Shen, Erik Andersen, and Malte Jung. Have your cake and eat it too: Foreign language learning with a crowdsourced video captioning system. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW ’17)*, pages 286–296, New York, NY, USA, 2017. ACM. doi: 10.1145/2998181.2998268.
- Yasuharu Den, Junpei Nakamura, Toshinobu Ogiso, and Hideki Ogura. A proper approach to Japanese morphological analysis: Dictionary, model, and evaluation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*, pages 1019–1024, Marrakech, Morocco, 2008. European Language Resources Association (ELRA). URL <https://aclanthology.org/L08-1535/>.
- Paul Denisowski. VNEDICT (Vietnamese-English dictionary). http://www.denisowski.org/Vietnamese/vnedit_readme.html, 2019. Release of 15 February 2019 (55,483 entries); CC BY 3.0. Accessed: 2026-10-08.
- Audrey Desjardins and Aubree Ball. Revealing tensions in autobiographical design in HCI. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS ’18)*, pages 753–764, New York, NY, USA, 2018. ACM. doi: 10.1145/3196709.3196781.
- Gilbert Dizon and Benjamin Thanyawatpokin. Language learning with Netflix: Exploring the effects of dual subtitles on vocabulary learning and listening comprehension. *Computer-Assisted Language Learning Electronic Journal (CALL-EJ)*, 22(3):52–65, 2021. URL <https://callegj.org/index.php/journal/article/view/352>.
- Electronic Dictionary Research and Development Group. The KANJIDIC project. https://www.edrdg.org/wiki/KANJIDIC_Project.html, 2026. Kanji dictionary file; project begun in 1991. Accessed: 2026-10-08.
- FooSoft. Yomichan: Japanese pop-up dictionary extension for Chrome and Firefox. <https://github.com/FooSoft/yomichan>, 2026. Archived GitHub repository; succeeded by Yomitan. Accessed: 2026-10-08.
- fxsjy. Jieba: Chinese word segmentation. <https://github.com/fxsjy/jieba>, 2026. GitHub repository (MIT licence). Accessed: 2026-10-08.
- Seth J. Goss and Katsuo Tamaoka. Lexical accent perception in highly-proficient L2 Japanese learners: The roles of language-specific experience and domain-general resources. *Second Language Research*, 35(3):351–376, 2019. doi: 10.1177/0267658318775143.

- Hao He, Courtney Miller, Shyam Agarwal, Christian Kästner, and Bogdan Vasilescu. Speed at the cost of quality: How Cursor AI increases short-term velocity and long-term complexity in open-source projects. In *Proceedings of the 23rd International Conference on Mining Software Repositories (MSR '26)*, pages 181–193, New York, NY, USA, 2026. ACM. doi: 10.1145/3793302.3793349.
- Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*, pages 92–101, New York, NY, USA, 2014. ACM. doi: 10.1145/2597073.2597074.
- Arsham Khosravani and Audris Mockus. Detecting AI coding agents in open source: A validated multi-method census of 180 million repositories, 2026.
- Geza Kovacs and Robert C. Miller. Smart subtitles for vocabulary learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*, pages 853–862, New York, NY, USA, 2014. ACM. doi: 10.1145/2556288.2557256.
- Stephen D. Krashen. *The Input Hypothesis: Issues and Implications*. Longman, London and New York, 1985.
- Taku Kudo. MeCab: Yet another part-of-speech and morphological analyzer. <https://taku910.github.io/mecab/>, 2026. Source: <https://github.com/taku910/mecab>. Accessed: 2026-10-08.
- Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain, 2004. Association for Computational Linguistics. URL <https://aclanthology.org/W04-3230/>.
- Language Reactor. Language reactor. <https://www.languagereactor.com/>, 2026. Browser extension, formerly “Language Learning with Netflix”. Accessed: 2026-10-08.
- Chee Ha Lee and Slava Kalyuga. Effectiveness of different pinyin presentation formats in learning Chinese characters: A cognitive load perspective. *Language Learning*, 61(4):1099–1118, 2011a. doi: 10.1111/j.1467-9922.2011.00666.x.
- Chee Ha Lee and Slava Kalyuga. Effectiveness of on-screen pinyin in learning Chinese: An expertise reversal for multimedia redundancy effect. *Computers in Human Behavior*, 27(1): 11–15, 2011b. doi: 10.1016/j.chb.2010.05.005.
- Jackson L. Lee, Litong Chen, Charles Lam, Chaak Ming Lau, and Tsz-Him Tsui. PyCantonese: Cantonese linguistics and NLP in Python. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6607–6611, Marseille, France, 2022. European Language Resources Association. URL <https://aclanthology.org/2022.lrec-1.711/>.
- Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. AIDev: Studying AI coding agents on GitHub. In *Proceedings of the 23rd International Conference on Mining Software Repositories (MSR '26)*, pages 1029–1033, New York, NY, USA, 2026. ACM. doi: 10.1145/3793302.3797249.
- Linguistic Society of Hong Kong. Jyutping Cantonese romanization scheme. <https://lshk.org/jyutping-scheme/>, 1993. Scheme designed and proposed by the LSHK in 1993. Accessed: 2026-10-08.

- Yue Liu, Ratnadira Widayarsi, Yanjie Zhao, Ivana Clairine Irsan, Junkai Chen, and David Lo. Debt behind the AI boom: A large-scale empirical study of AI-generated code in the wild, 2026.
- Kang Kwong Luke and May L.-Y. Wong. The Hong Kong Cantonese Corpus: Design and uses. In Benjamin Tsou and Oi Yee Kwong, editors, *Linguistic Corpus and Corpus Linguistics in the Chinese Context*, number 25 in Journal of Chinese Linguistics Monograph Series, pages 312–333. The University of California, Berkeley, CA, 2015.
- Stephen Matthews and Virginia Yip. *Cantonese: A Comprehensive Grammar*. Routledge Comprehensive Grammars. Routledge, London and New York, 2nd edition, 2011. ISBN 9780415471312.
- MDBG. CC-CEDICT. <https://cc-cedict.org/wiki/>, 2026. Chinese-English dictionary, continuation of CEDICT (started by Paul Denisowski in 1997); CC BY-SA 4.0. Download: <https://www.mdbg.net/chinese/dictionary?page=cc-cedict>. Accessed: 2026-10-08.
- MeCab project. mecab-ipadic (version 2.7.0). <https://taku910.github.io/mecab/>, 2026. IPA dictionary for MeCab, built from the IPA dictionary and IPA corpus; distributed with MeCab (<https://github.com/taku910/mecab/tree/master/mecab-ipadic>). Accessed: 2026-10-08.
- messense. jieba-rs: The jieba Chinese word segmentation implemented in Rust. <https://github.com/messense/jieba-rs>, 2026. GitHub repository (MIT licence); Node.js binding via @node-rs/jieba (<https://github.com/napi-rs/node-rs>). Accessed: 2026-10-08.
- mifunetoshiro. Kanjium: The ultimate kanji resource. <https://github.com/mifunetoshiro/kanjium>, 2026. Includes pitch-accent data (accents.txt); CC BY-SA 4.0. Accessed: 2026-10-08.
- Migaku. Migaku. <https://migaku.com/>, 2026. Language-learning app and browser extension. Accessed: 2026-10-08.
- Holger Mitterer and James M. McQueen. Foreign subtitles help but native-language subtitles harm foreign speech perception. *PLoS ONE*, 4(11):e7785, 2009. doi: 10.1371/journal.pone.0007785.
- Maribel Montero Perez, Wim Van Den Noortgate, and Piet Desmet. Captioned video for L2 listening and vocabulary learning: A meta-analysis. *System*, 41(3):720–739, 2013. doi: 10.1016/j.system.2013.07.013.
- National Institute for Japanese Language and Linguistics. UniDic. <https://clrd.ninjal.ac.jp/unidic/en/>, 2026. Electronic dictionary for Japanese morphological analysis; GPL-2.0 / LGPL-2.1 / BSD triple licence. Accessed: 2026-10-08.
- Carman Neustaedter and Phoebe Sengers. Autobiographical design in HCI research: Designing and learning through use-it-yourself. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*, pages 514–523, New York, NY, USA, 2012. ACM. doi: 10.1145/2317956.2318034.
- Dat Quoc Nguyen, Dai Quoc Nguyen, Thanh Vu, Mark Dras, and Mark Johnson. A fast and accurate Vietnamese word segmenter. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 2582–2587, Miyazaki, Japan, 2018. European Language Resources Association (ELRA). URL <https://aclanthology.org/L18-1410/>.
- NHK Hōsō Bunka Kenkyūjo, editor. *NHK Nihongo Hatsuon Akusento Shin Jiten*. NHK Shuppan, Tokyo, 2016. ISBN 9784140113455. [NHK Japanese pronunciation and accent dictionary, new edition]. In Japanese.

- ookii-tsuki. Anisubber: Learn Japanese from anime and Japanese drama. <https://ookii-tsuki.github.io/Anisubber/>, 2026. Source: <https://github.com/ookii-tsuki/Anisubber>. Accessed: 2026-10-08.
- OpenJS Foundation. Electron: Build cross-platform desktop apps with JavaScript, HTML, and CSS. <https://www.electronjs.org/>, 2026. Source: <https://github.com/electron/electron> (MIT licence). Accessed: 2026-10-08.
- Joy A. Palmer. The effect of furigana on lexical inferencing of unknown kanji words. MA thesis, Brigham Young University, Provo, UT, 2012. URL <https://scholarsarchive.byu.edu/etd/3235>.
- Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of AI on developer productivity: Evidence from GitHub Copilot, 2023.
- Pleco Software. CC-Canto: A Cantonese dictionary for everyone. <https://cantonese.org/about.html>, 2026. Cantonese-English dictionary designed to be used alongside CC-CEDICT; CC BY-SA 3.0. Accessed: 2026-10-08.
- saltyshiomix. Nextron: Next.js + Electron. <https://github.com/saltyshiomix/nextron>, 2026. GitHub repository (MIT licence). Accessed: 2026-10-08.
- Burr Settles and Brendan Meeder. A trainable spaced repetition model for language learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1848–1858, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1174.
- Irina A. Shport. Training English listeners to identify pitch-accent patterns in Tokyo Japanese. *Studies in Second Language Acquisition*, 38(4):739–769, 2016. doi: 10.1017/S027226311500039X.
- Don Snow. *Cantonese as Written Language: The Growth of a Written Chinese Vernacular*. Hong Kong University Press, Hong Kong, 2004. ISBN 9789622097094. doi: 10.2307/jj.40271861.
- Don Snow. Cantonese as written standard? *Journal of Asian Pacific Communication*, 18(2): 190–208, 2008. doi: 10.1075/japc.18.2.05sno.
- takuyaa. kuromoji.js: JavaScript implementation of Japanese morphological analyzer. <https://github.com/takuyaa/kuromoji.js>, 2026. GitHub repository. Accessed: 2026-10-08.
- tranxuanthang. LRCLIB: Free synchronized lyrics service. <https://lrclib.net/>, 2026. Server source: <https://github.com/tranxuanthang/lrclib>. Accessed: 2026-10-08.
- Robert Vanderplank. *Captioned Media in Foreign Language Learning and Teaching: Subtitles for the Deaf and Hard-of-Hearing as Tools for Language Learning*. Palgrave Macmillan, London, 2016. doi: 10.1057/978-1-137-50045-8.
- Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. VnCoreNLP: A Vietnamese natural language processing toolkit. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 56–60, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-5012.
- Andi Wang and Ana Pellicer-Sánchez. Incidental vocabulary learning from bilingual subtitled viewing: An eye-tracking study. *Language Learning*, 72(3):765–805, 2022. doi: 10.1111/lang.12495.

- Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E. Hassan. On the use of agentic coding: An empirical study of pull requests on GitHub. *ACM Transactions on Software Engineering and Methodology*, 35(10):1–24, 2026. doi: 10.1145/3798166.
- Stuart Webb and Michael P. H. Rodgers. Vocabulary demands of television programs. *Language Learning*, 59(2):335–366, 2009. doi: 10.1111/j.1467-9922.2009.00509.x.
- Rong Xiang, Emmanuele Chersoni, Yixia Li, Jing Li, Chu-Ren Huang, Yushan Pan, and Yushi Li. Cantonese natural language processing in the transformers era: A survey and current challenges. *Language Resources and Evaluation*, 59(2):1747–1773, 2025. doi: 10.1007/s10579-024-09744-w. Published online 8 June 2024.
- Wai-Ping Yau. Power, identity and subtitling in a diglossic society. *Meta*, 57(3):564–573, 2012. doi: 10.7202/1017080ar.
- Yomitan contributors. Yomitan: Pop-up dictionary browser extension for language learning. <https://github.com/yomidevs/yomitan>, 2026. Successor to Yomichan; GPL-3.0. Documentation: <https://yomitan.wiki/>. Accessed: 2026-10-08.
- Hocky Yudhiono. Miteiru. <https://github.com/hockyy/miteiru>, 2026. Open-source Electron video player for learning Japanese, Mandarin, Cantonese and Vietnamese; repository created 2022-12-05. Project site: <https://miteiru.hocky.id>. Accessed: 2026-10-08.
- zh-lx. pinyin-pro. <https://github.com/zh-lx/pinyin-pro>, 2026. Chinese-to-pinyin conversion library (MIT licence). Accessed: 2026-10-08.

A Video sources

The screenshots use frames and subtitle tracks from four public YouTube videos, chosen because their uploaders subtitled the speech in the language spoken, which was checked against YouTube’s metadata where it records the speech language. Titles are transliterated.

- Japanese: Akane-teki Nihongo Kyoushitsu, “[Tokyo Vlog] Jinbouchou de ohirugohan o tabete hon’ya ni ikimashita” (lunch in Jimbocho and a bookshop), 18 September 2025, https://www.youtube.com/watch?v=_e-uQ0tIEMY. Japanese and English tracks.
- Mandarin: Da Qi, “Beijing Caochang hutong you difang 0 youke” (a Beijing hutong with no tourists), 12 September 2026, <https://www.youtube.com/watch?v=y0H1npFoAHk>. Chinese track.
- Cantonese: McFatty Mak Fa Tin, “Hainan chicken rice (easy rice-cooker version), with Cantonese subtitles, Ep32”, 9 May 2021, <https://www.youtube.com/watch?v=E2gaC2a7C3A>. Written-Cantonese track (yue-HK).
- Vietnamese: Anh Tay Bao Minh, “Nguoi Viet tot voi toi den muc toi khong hieu noi” (Vietnamese people are so kind to me I can’t understand it), 6 October 2026, <https://www.youtube.com/watch?v=CoUJ406wer0>. Vietnamese track.