The EuroCALL Review
Volume 31, No. 1, 2024, 51-60
ISSN: 1695-2618
http//doi.org/10.4995/eurocall.2024.18168

European Association for Computer-Assisted Language Learning

THE EUROCALL REVIEW

RECOMMENDED RESOURCE

Bilingual Viewer: A bilingual text generation tool

J. Mike Ertl, United Kingdom

mike.ertl@gmail.com

How to cite this article:

Ertl, J.M. (2024). Bilingual Viewer: A Bilingual Text Generation Tool. *The EuroCALL Review, 31(*1), 51-60. https://doi.org/10.4995/eurocall.2024.18168

Abstract

This article describes the website BilingualViewer.com; it discusses the technologies used to build it and the engineering challenges that were encountered. BilingualViewer.Com is an open and free application that uses web technologies to create Bilingual versions of articles and books. The website can be saved onto the homepage of a smartphone and used like an App (for reading and converting newspaper articles on the fly), in addition to being opened on a computer's browser to process intensive tasks such as whole book translations. The website will henceforth be referred to as *the app*. Bilingual text refers here to a piece of writing where sentences in two different languages are presented sequentially, one sentence in the language to be learned followed by the user's primary language. Bilingual texts are particularly useful for students who rely heavily on dictionaries when reading books and newspaper articles. The app helps to create bilingual texts from almost any electronic content.

A key feature from a copyright point of view is that the content is generated and stored locally on the device (unless added to an online library). Where a subscription is required to access newspaper content, the app generates a bilingual text article and stores it locally, so that the restricted content does not leave the device (i.e., it does not appear in any cloud server other than Google or Apple translation servers).

There is a large features list including switching on and off languages, getting the device to read out loud, saving a list of items on the device along with the currently read position, sending content to an eBook, merging articles and more.

A library of pre-translated books is provided in the app with the option of augmenting this with the user's own on-line content.

The app is available at https://www.bilingualviewer.com/

Documentation on the app can be found at https://bv-fc.netlify.app/about/

Keywords

Bilingual, Tool, Dual Language, Resource

1. Introduction

Regular reading in a target language leads to improvement. Bilingual content and an application that converts any text into this format improves on this further for the following reasons:

- Self-selected content is more pleasurable than miscellaneous homework articles.
- The first language acts in much the same way as stabilisers when learning to ride a bike; you are able to read things that would normally be beyond your ability. This opens up the possibility of reading some of the greatest literature ever written as part of your learning.
- Bilingual format articles can be printed out or sent to a static reading device such as an eReader, where you are less inclined to jump to other applications, and do not need to interrupt your flow to consult dictionaries. This means you can engross yourself in a story rather than treating the exercise as a homework task.

2. Technical issues relating to articles

2.1. Bilingual text format

The style and colour of the second-placed language was carefully selected so that it could be easily read or ignored. A number of shades of grey were tried (as they appear in eInk), in order to get something that is feint enough to not be read when on an adjoining sentence, but which can be read without squinting when needed.

By supplementing this with italics and larger than usual character spacing between languages, the first language pops out and can be determined without reading it, whilst the paragraph layout is maintained (but doubled in size). This is referred to as the Bilingual Format. This popping out is achieved by making each word in Language A appear different from that of Language B in multiple ways, so that it can be deduced instantly without needing to read any of the text. This is the opposite of Wally in the Where's Wally pictures. This subject been studied in detail, following a theory known as Feature Integration Theory (Treisman & Gelade, 1980).

Transition at the sentence-by-sentence level from one language into another was chosen. Whilst paragraph by paragraph would be the easiest to code, sentence-by-sentence avoids the problem of trying to locate the equivalent place in paragraphs.

There are two ways of presenting sentences, either one followed by the other with a large space between, or by starting a new line with each language transition. Since both have a place in language learning, the tool provides a button to present in either format, as well as a way to switch either language on or off. The advantage of sentences on a new line, especially for short sentences is that the two languages align and can be read

together. The advantage of separating by a large space is that paragraphing is maintained, and this is an important aesthetic in a book.

As the content is all HTML, CSS is the obvious choice for implementing these formats and options. By adding and removing CSS classes to a document, the App can change the format at an instant, since the browser itself is doing the rendering according to the CSS provided, and at high speed.

Since the content is going to be manipulated anyway, adverts etc. can be stripped out in much the same way as the iOS reader mode. Figure 1 shows the required format on a tablet, and Figure 2 shows how it appears on an e-reader.

Figure 1

Sentence-by-sentence and paragraph format for a newspaper article.

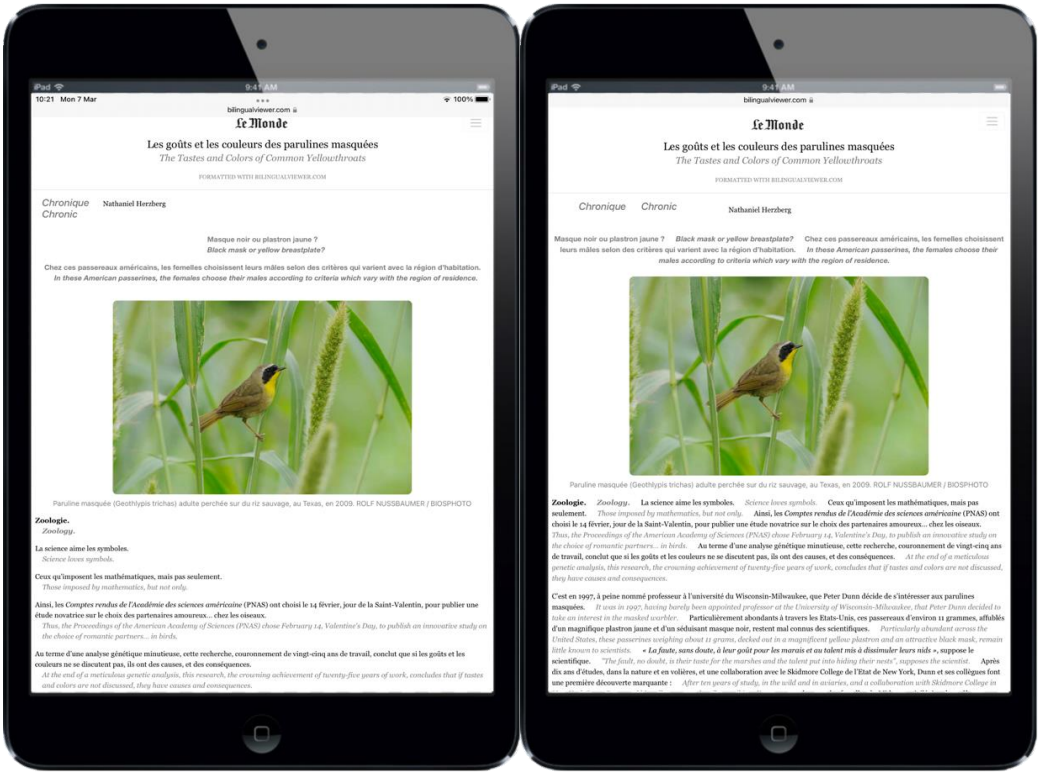
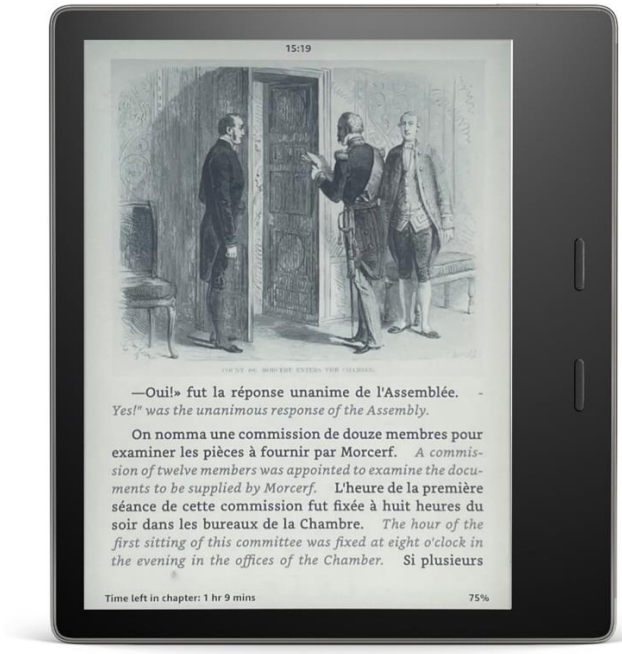


Figure 2

Greys and spacing on an eReader. Note the wide space between languages.



2.2. Attempts at bilingual viewing using existing browser tools

Browsers now have free access to translation tools, Google Translate and Apple Translate for example. The level of translation is at an advanced stage, with context sensitive phrases, idioms, and ambiguous words being translated extremely well. The translations appear to be made at the level of sentences which is ideal for the tool.

These translation tools are aimed at allowing a user to read websites as though they were written in their own language, however, they are not specifically designed for displaying both languages at once.

Without an app such as Bilingual Viewer attempting to read a newspaper article in both French and English might involve:

- Opening Safari twice on an iPad, opening the same page, and translating one of them. By scrolling with one thumb on each side there is a kind of bilingual effect.
- Another trick is to open an article, go into Reader Mode, come out of that mode, translate, and since Reader Mode is only calculated once, it is possible to toggle between Reader Mode and non-reader mode and effectively toggle between languages, with the downside of the styling changing completely.
- Highlight the sentence and select translate. It is not practical to do this for every sentence.

None of these options are satisfactory, or simple enough.

2.3. Adding speech

Browsers now come with the ability to read out sentences. It is slightly too cumbersome to use this feature for longer articles, since it is necessary to highlight text then select the Speak option. Furthermore, it can fail when meeting language transitions, with the wrong voice being used, sometimes comically. The App, on the other hand, makes use of the latest HTML functions to set the correct voice for the language before speaking, so that it can seamlessly switch from one voice to another while reading a paragraph containing

two languages. At the time of writing, iOS products work fine but Android can sometimes ignore the request to change language.

This made it necessary for the app to be able to read in either language or both in any order. It might be preferable to hear the sentence in the primary language first so that the understanding is already there, or alternatively to try to figure it out first and then be spoken to in the primary language, all the while selecting one language to see or possibly both. The option to read should be available at paragraph level, and be initiated with a single click on the paragraph when this setting is selected.

Note that some Android versions steal the click button on HTML documents and bring up a dictionary, again there are no such issues in iOS.

2.4. Converting an article into bilingual text format on the fly

The application allows a user to find an article of interest, e.g. from a newspaper subscription and convert it into a Bilingual version stored in the App at the click of a button.

In order to get the article from the webpage into Bilingual Viewer it is necessary to:

- Copy the html of the original article
- Translate the article
- Copy the html of the translated article
- Process the two html files to create a Bilingual html document

2.5. Processing and storing text on devices

From the outset this application had to be designed to work from a tablet or phone since this is the normal mode of reading articles for many people. Additionally, it had to be simple to use, therefore using Chrome Development tools to grab html was not an option. The processing and storing of the text needs to be done on the device itself from inside the paywall and not appear on servers. Bilingual Viewer is a piece of Javascript that runs on the device and places the generated content in the browser's local storage.

2.6. The bookmarklet

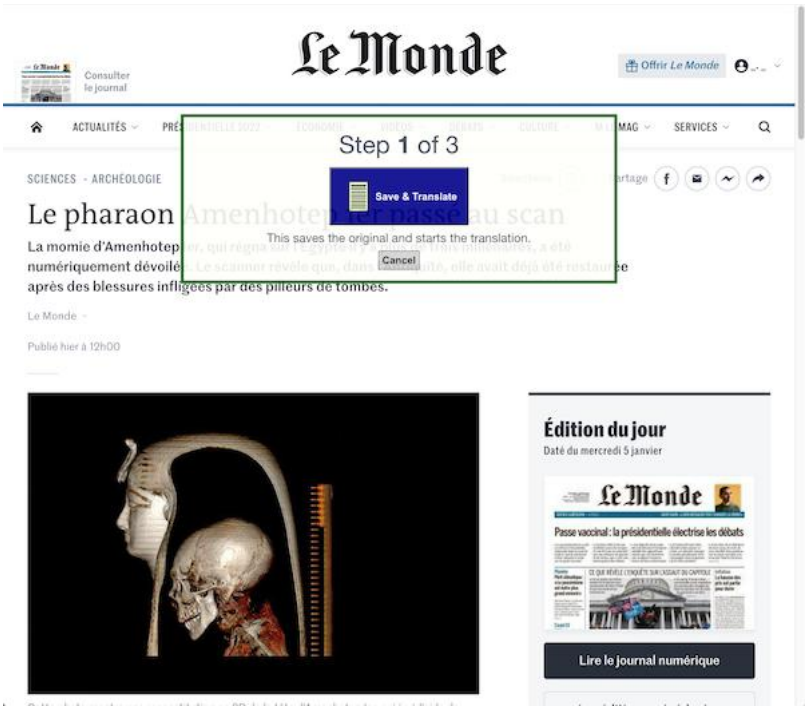
Because the starting point is inside a third party website, the first requirement is to insert Javascript into the code. This was achieved through the use of a Bookmarklet. This is a piece of Javascript that you can run by clicking on a bookmark. The bookmark contains a name as per usual but in place of the URL a chunk of Javascript is inserted. In IOS, unlike a desktop, you cannot simply drag a Bookmarklet onto the favourites bar, therefore, in the help screen I added instructions for manually adding the bookmarklet by copying and pasting the Javascript into the url field of the bookmark. In other words, if bookmarks are synchronised between your device and computer, it is only necessary to drag the bookmarklet on the front page onto the bookmark tabs.

Figure 3 shows the effect of clicking on a bookmark it while reading an article in Le Monde.

The act of pressing the bookmarklet causes the Bookmarklet Javascript code to be injected in the page:

Figure 3

Pressing the bookmarklet while viewing an article.



The bookmarklet code contains a pop up in the centre of the screen which guides the user through the process, as demonstrated by this French to English translation:

- Step 1) Click the large button to copy the French webpage into the buffer and inject Google Translate into the website. (Note the user has the option to ignore this and use a different translation service such as Apple Translate activated on the browser bar).
- Step 2) Wait for translation to start (it may require you to press Translate on the Google Translate bar at the top).
- Step 3) Scroll to the end of the article to ensure the whole page is translated.
- Step 4a) Press the large **Create Bilingual Article** button. This copies the translated article and appends it to the original article in the cut and paste buffer. It then jumps to BilingualViewer.com in the same browser.
- Step 4b) Alternatively, press the smaller **Copy Bilingual Article** button to do the same as 4a but allow the user to go to the Bilingual Viewer App which was saved as a Webapp on the home screen, rather than using the browser.
- Step 5) In BilingualViewer.com click the large **Paste** button to paste into the app, the app reads the buffer, creates a bilingual article, saves it to local storage as a Card on the Saved section, and shows the article in bilingual format.

In all there are 3 button presses and a scroll-to-end action.

Note that the bookmarklet has a limited task of gathering html, initiating the translation service and placing the two html documents (one for each language) in the cut and paste buffer. The complex task of converting into bilingual format is done in the app itself to keep the bookmarklet simple.

2.6. Processing in the main app

The main app, on receiving the pasted pair of HTML documents must find where the article starts in each document then match the equivalent sentences and create a new document

with html and css structures that enable all the features of the app to work. This can then be saved in local storage.

2.6.1 Analysing the webpage

As per the Reader Mode in Safari, it is necessary to find the article itself and ditch the menus and adverts. The article usually appears inside an <article> tag. However, having examined several newspapers this is not always the case. They have a mixture of tags such as <main>, etc. The code allows for several options, looking for each one in turn. For newspapers with a known method, it starts out with this one.

2.6.2 Sentence sorting

Once the main article was found, the HTML DOM structure of the main article part was converted into a two-dimensional array of paragraphs and sentences (also allowing for headers and images). Using a piece of regular expression, each paragraph is divided into sentences.

2.6.3 Creating a new page

Using the array of data, a new partial HTML document is constructed that makes use of CSS built into Bilingual Viewer that allows all the features to work such as language toggling, speech, paragraphing mode, colour choices etc. Keeping as much as possible dependant on CSS selection is by far the fastest way to change a website's look and feel, as it is all done inside the native browser, so the effects come into force immediately.

2.6.4 Saving the article

Articles currently being read are stored in browser local storage and shown as cards on the homepage. These show the first picture in the article along with the title. These cards can be rearranged, merged or deleted.

The app also allows the articles to be stored on the user's Dropbox cloud. This is because unlike iCloud, Dropbox can be integrated into a HTML page.

Books in the library are too large to fit into local storage (at least on a mobile device), therefore, when clicking on a book from the library, it is read online and only the title, author, position within the book, and url to the content are stored in local storage.

2.6.5 Bilingual Viewer file formats

The format of the ultimate file which has the extension .bv is the same format stored in local storage. It contains some metadata followed by the HTML for the bilingual article containing all the css classes and tag structure needed. It needs to be injected into a div in Bilingual Viewer, because this has both the CSS to render the text correctly and the Javascript functions that are called from within the document.

Note that there are stand-alone HTML versions of the books in the library which contain everything in one link for viewing without Bilingual Viewer. An option to produce this content is not available in the app at the time of writing, however, this is forthcoming.

3. Technical issues relating to books

The app also includes not only a selection of ready-made books in the Library section, but the tools necessary to make a bilingual book. Unlike an article a book is too large to fit in local storage so the options are:

- Send it to an e-reader, preferably converting it into epub format first.
- Place it on a static webserver in a cloud. The app allows personal libraries to be appended to the main library.

Because a .bv file format is difficult to edit, when converting a book an intermediate file format is used, known in the app as Language files. These are described in the help section of the app (https://bv-fc.netlify.app/about/#eg-lang-files).

In summary, these are text files containing a bare minimum of html tags which are kept on a separate line from the text itself for ease of editing and searching. Each language file is for one language. They are combined and processed to produce the final .bv file.

Note that a large novel can be converted and all the detected errors fixed in a couple of hours.

Full instructions are on the website itself, here some of the technicalities are discussed.

3.1 Error correction

Error correction is required. There are three types of common error:

- Translation tools such as Google Translate sometimes insert duplicates of sentences at the end of paragraphs.
- The regex that Bilingual Viewer uses to separate paragraphs into sentences can sometimes result in the number of sentences in a particular paragraph being different in one language compared to the other.
- When translating a book, if the scroll to the end is too fast, sections can be left untranslated.

In the Create page of Bilingual Viewer, when making a book there is an option to save an error file. This contains a list of where the app has detected the possibility of either of the first two above errors in a paragraph. This allows the user to correct them without needing to proofread to find them. The first type of error, duplicates, needs fixing first, since they create the second error type.

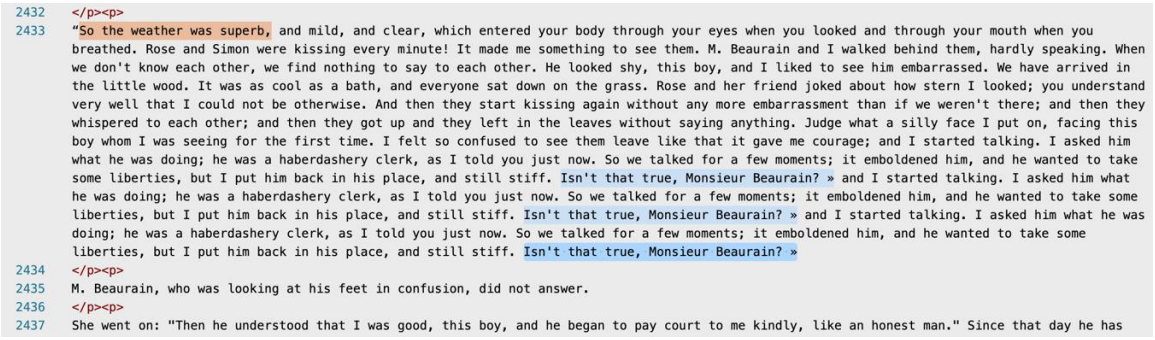
3.2 *Duplicate sentences*

Although not detected directly, these are picked up by assuming both languages will produce similar lengths of paragraphs, assuming a paragraph has more than around 200 characters, they should be within 90% of each other. The error file contains a list of paragraphs in each language where the software thinks duplicates may have been produced.

Visual Studio Code is ideal for fixing this issue, by highlighting the last few words all duplicates automatically show up. All you need to do is remove all words from just after the last occurrence of the duplicate to the end of the paragraph.

Figure 4

Duplicates highlighted in Visual Studio. The error file indicated a problem in the paragraph starting with "So the weather", the solution in Visual Studio is easy: Highlight the last few words in the paragraph and Visual Studio automatically highlights duplicates and deletes from the end of the first occurrence of "Monssiur Beaurain?" to the end of the paragraph.



Whilst it could be automated, the number of duplicates in even a large book is not too onerous, and deleting content in the original language was thought to be best left to manual intervention.

3.3 Sentence mismatches

When converting French Literature, Google Translate converts the French-style paragraphs of speech starting with a hyphen, into English by enclosing the speech in quotes and leaving the *he said* part outside. This has the effect of making the number of sentences in each language's paragraph different. This is an example of where matching sentences one to one goes wrong, so the app adds the paragraphs where it has detected this to the error file. The documentation details ways to fix these types of issues.

3.4 Him, Her, His

In French to English, Google Translate guesses the gender, this is because in French the pronoun often relates to the object rather than the subject. Again, using Visual Studio Code you can search for all instances of Him, Her or His by entering "Him|Her|His" in the regular expression box and correcting them manually one by one, although due to the usage of the second language as a guide to the meaning of the first language, it is probably overkill to spend too much time on this.

4. Features

The features and uses are described in the online documentation (https://bv-fc.netlify.app/about/#features)

Bilingual Viewer App features:

- Convert and read web articles in a bilingual format
- Store, sort and merge books and articles saving last read paragraph
- Convert any digital book into bilingual format (with DRM cut and paste or if this is restricted, OCR scan then cut and paste)
- Get device to read out loud a paragraph with one click in any language combination
- Automatic Contents page generation with the contents activated from the menu
- Adjust language(s) shown at any time with a single click
- Adjust viewing format instantly with a click
- As a teacher, create a library of content for students and share via the app
- As a student, by adding a url in settings the app can now contain content from the learning institution
- Send content to e-reader in preferred layout option by printing to pdf and sharing with e-reader app
- Encrypt and decrypt sensitive documents easily for sharing on the web
- Integrate with Dropbox for cloud storage of articles, with a button to generate a link to Bilingual Viewer for displaying the article, or for social media sharing
- Library sorting features including by size, with a visual indication of book size shown
- Night mode and other viewing modes

- Panoramic mode for reading articles on a phone, pictures fill height of device allowing left right scrolling to get a larger than life image feel
- Switch features on or off. Either have buttons for language changing, contents page etc. always on, or have an article only view, apart from a feint Home Icon and Menu Hamburger on the top right.

5. Use cases in a language learning context

This tool was intended to be used primarily to support existing learning methods, because it is capable of converting user selected content rather than that supplied by a tutor, it can replace existing reading for pleasure rather than being seen as additional homework.

The use cases for the app include:

- Daily reading of newspaper articles in bilingual format
- Daily book reading in bilingual format
- Occasional listening to aid aural comprehension (occasional as generated voices are good but not for long periods of time)
- Platform for teachers to curate both a book and article library for students
- Platform for students to receive bilingual content as part of their studies
- Share new translations of books and increase the library
- Share links to articles on social media from a smartphone (using Dropbox links)

Ethical statement

The tool includes some sample translated books originating from Project Gutenberg with kind permission from:

> Dr. Gregory B. Newby
>
> Chief Executive and Director
>
> Project Gutenberg Literary Archive Foundation http://gutenberg.org
>
> A 501(c)(3) not-for-profit organization with EIN 64-6221541

No potential conflict of interest was reported by the author.

References

Treisman, A.M., Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, *12*(1), 97-136.