IMPERIAL

Discovering *Felix*

Analysing a Historical Newspaper Archive

Author

Timothy Langer

CID: 02213612

Supervised by

Dr Chiraag Lala

Second Marker

Ivan Procaccini

A thesis submitted in fulfillment of requirements for the degree of
Bachelor of Engineering in Computing

Department of Computing
Imperial College London

2025

Abstract

Felix, Imperial's newspaper, has over 35,000 pages in its archive spanning 75 years. For decades institutions around the world have been digitising such archives. But despite transcribing the scanned pages with OCR software, the resulting text often contains errors that hamper both researchers and automated analysis tools. In an effort to make the *Felix* archive more accessible, I try novel approaches to extract clean article texts from newspaper scans, finding that large vision-language models are now capable of producing highly accurate transcriptions. Processing the *Felix* digital archive, I create a data set of over 100,000 articles at near-perfect accuracy, all on a budget of less than $4 per 1,000 pages. I then build a website to explore the archive, including a search engine, a trend analysis tool and a recommendation engine that draws parallels between past and present. Initial user feedback indicates that students find the website helpful in researching the university's past and a significant upgrade in usability over browsing the original scans. The project shows that superior accuracy and functionality is now achievable at modest cost, paving the way for other digitised newspaper archives to follow suit.

Contents

1. Introduction

Students and staff at Imperial College have read *Felix*, the student newspaper, for over 75 years. The publication has informed, entertained and recorded the university's cultural and social changes across over 1800 issues. When the earlier physical editions started to deteriorate in the 2000s, the paper digitised most of them in a joint effort with the college and their scans are now preserved online. For a nostalgic graduate that knows where to look or a dedicated researcher, the digital archive has a wealth of information.

Yet the sheer volume of material and the manner in which it is hosted make exploring the archive difficult. The scans, though carefully produced, come with outdated OCR (optical character recognition) that makes keyword searches brittle and incomplete. This project hopes to make the digital archive more accessible and fully searchable, then use machine analysis to glean insights and surface forgotten stories.

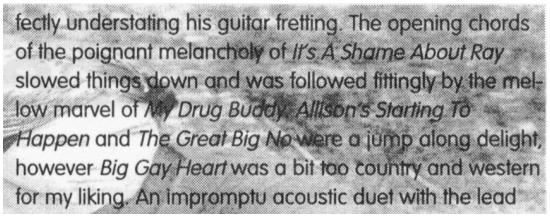
Many newspaper analysis projects (1–3) specialise in older collections that are available in the public domain, effectively precluding the scope of their analysis to papers published at least 100 years ago. With the permission of the *Felix* editorial board, this project has the unique opportunity to analyse a newspaper that is largely still protected by copyright.

—

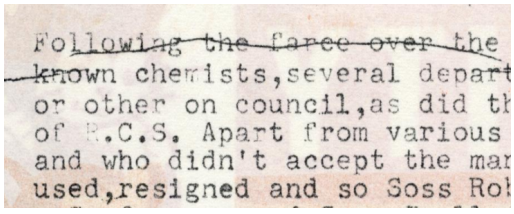
Broadly speaking, the project is formed of two stages: converting the pages of *Felix* into workable text, then using natural language processing techniques to analyse the archive and make it accessible to more people.

Going from scans to articles turned out to be harder than expected. While the digitised PDFs do have text embedded within, the text is of such low quality that trying to conduct any meaningful analysis on it will spend more effort battling OCR noise than analysing the text. The reason for the transcriptions being so poor is because most of the 2009 digitisation budget went towards the physical scanning process, and the paper could only afford to pay for a cheaper text recognition service that promised an accuracy of 80% (4). While people can struggle through the errors and still grasp the content, automated analysis tools unfortunately cannot (5, 6).

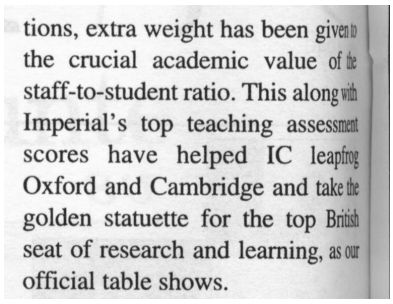
Newer OCR (optical character recognition) software can produce accurate output for simple, single-column layouts with clear text and common typefaces. But many pages of the newspaper, particularly from older editions, suffer from faded text and printing defects. Digitisation added further errors, like warped edges. And some pages with wacky designs are hard to read even for us.



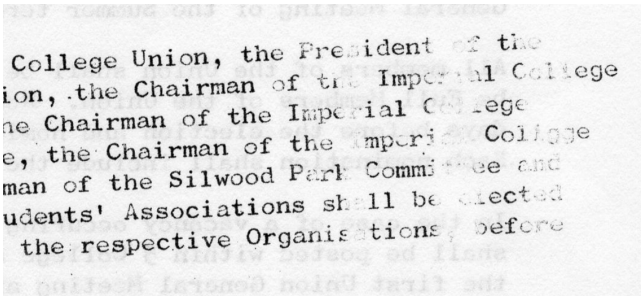
An image behind the text makes it hard to read even for a human



Pen markings and faded text in poorly-preserved copies



Warped edges, a scanning defect



Text printed at a skewed angle and either poorly printed or since faded

Figure 1.1: Samples from the digital archive with questionably legible text

The varying and complex layout of text, graphics and advertisements across the paper's pages further complicate text extraction. There is no single, predictable page layout that persists through the years, as it might for an established publication. In large part, this is down to the changing team behind the paper. Each new editor brings changes to the paper's design, layout and writing style. The logo alone has changed fifty times.

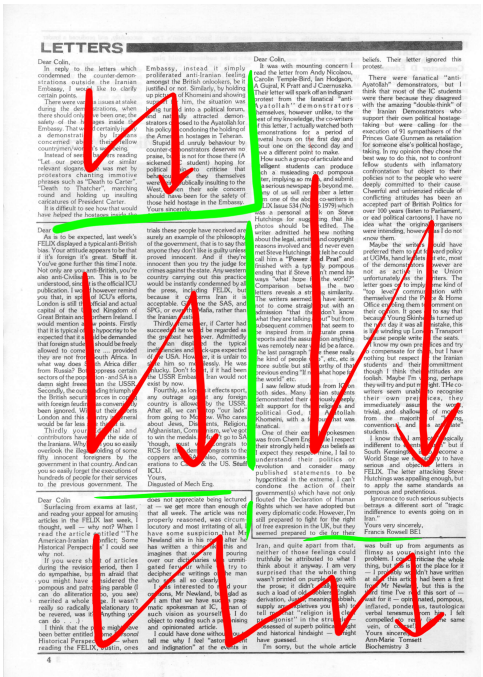


Figure 1.2: A seemingly simple, four-column layout with a complicated reading order

While I initially tried going down a more traditional route, using OCR and page layout analysis, I found that LLM often outperform such pipelines while operating at a similar price point, and the results back it up.

With 100,000 articles loaded in, I went about building *Felixplore*, a web application to explore the archive. This included a search engine, a trend analysis tool and a recommendation engine.

2. Source material

Imperial was the first UK university to digitise its student newspaper collection (7) at a cost of some £10,140 in 2009. Working with the college archivist, the first 1200 or so issues of *Felix* were scanned in from printed copies. The remaining issues, produced digitally since 2004, we have in their original form. With 90% of all issues ever published available digitally, the almost-complete collection is well-suited for text analysis.

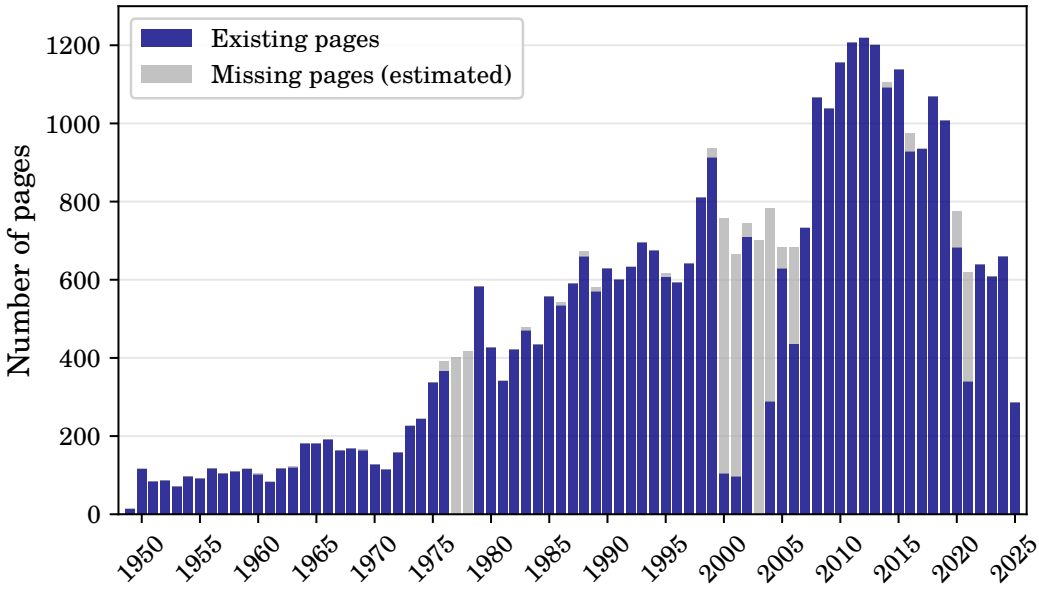


Figure 2.1: Pages per year in the digital archive. Note the gaps in the late 1970s and early 2000s where the issues were either lost or have not yet been digitised.

As printing has grown cheaper, the paper has generally printed more pages, as shown in figure 2.1. This does not imply more content, as the font size has similarly increased.

Broadly speaking, the paper has been published in three different dimensions: 440×320 mm broadsheet, 380×320 mm tabloid and 300×220 mm A4. Some pages are scanned as double-page spreads and some are rotated 90°. There is a small further proportion of supplemental pages of varying sizes (e.g. surveys, etc.)

Although the print editions were scanned in as high-quality TIFFs only the compressed PDFs remain, consisting of 75% quality JPEG images at 300 dpi. Most scanned pages are in grayscale, some are in colour. Noise has either been removed or controlled for as part of the scanning process. The professional handling means that most pages are free from skewing issues or poor contrast.

While the professionally-scanned older editions correctly record single and double page spreads, newer editions are more hit-and-miss, particularly those from the 2010s. Namely, some issues are formatted entirely as single pages, making the text of double-page spreads appear chopped off on both pages. I manually went through about 150 such issues, joining together double-page spreads and rotating those meant to be read vertically.

3. Article extraction

Google's bots got better than humans at reading CAPTCHAs over 10 years ago (8). How hard can it be to read a newspaper? It turns out it is no mean feat.

Cleanly transcribing the articles is crucial, since text riddled with errors hinders automated analysis. Most natural language processing algorithms struggle on noisy text (5). An error rate as low as 5% can start to cause significant downstream errors in information retrieval (6). Transformer-based models such as large language models seem to be particularly affected (9, 10).

There is also the human aspect. It is easy to abstract away the accuracy of the extracted text into a simple figure. But a study from Finland finds that improving OCR quality also improves user experience, and that investing in accuracy pays off. Even a few percentage points of improved OCR accuracy make a noticeable difference in the perceived usefulness of the archive and the relevance of search results (11).

3.1. Background

3.1.1. Post-OCR correction

With libraries across the globe digitising their collections since the late 1990s, many collections now feature dated or inaccurate OCR, just like *Felix*. Re-running OCR on the million-page scale of some countries' newspaper collections is not always financially feasible. There has, therefore, been research into whether there are economical methods of upgrading the OCR quality of existing text corpora.

Language models have proved promising. A competition in post-OCR text correction in 2019 saw the winning entry, based on BERT, improve English text transcripts by 11% (12). An approach using BART two years later achieved a 29% improvement in accuracy (13). An even larger model, Llama 2, corrected twice more character transcription errors than BART in historical newspapers (14). But a different study found that LLMs produced no significant improvement. If anything, the "corrections" made the text less accurate (15).

While language models could improve the OCR text, the "garbage in, garbage out" mantra still holds. If even a human struggles to read the text, it is a stretch to expect a machine to do any better. The method may work well for archives that have lost scans or originals and only have poor-quality OCR text to work from. Since *Felix* is fortunate to have the scans, working from the originals will yield better results.

3.1.2. Optical character recognition

For decades, OCR has been the standard method for extracting text from images. As per the name, these programs operate at the granularity of a single character. Lacking the context of the page, the paragraph or even the other characters in the word, OCR engines are prone to making errors, be it simple character errors or trying to read images or design elements as text.

Nonetheless, the field has definitely advanced in the 15 years since *Felix* was first digitised, far surpassing the 80% accuracy that the paper could afford in 2009. Suppose we start afresh and extract the text from scratch.

Tesseract is an OCR engine that HP developed internally before releasing the source code in 2005 (16). Google took over, maintaining and improving Tesseract for another decade. It is still regarded as a fair OCR engine with many studies using it as a baseline. There are far better engines available now, especially with development activity on Tesseract dwindling.

PaddlePaddleOCR is an actively-developed, open-source OCR engine developed by Baidu, also known as the "Google of China". It excels at recognising both printed and handwritten Chinese text, but its performance on English texts is only a few percentage points behind. As is the case with many text extraction projects, it focuses primarily on modern sources of data, optimising for commercially-viable use-cases, such as receipt and form scanning.

There are server-based commercial OCR solutions too. ABBYY, the software company who originally transcribed the *Felix* scans, still offers OCR services, with rates starting at $20 per 1,000 pages.

Big Tech has options too: Amazon Textract, Google Document AI, Microsoft Azure Document Intelligence. A comparison of the three found Google Document AI consistently outperformed the other two, with Textract coming close second (17). Presumably, Google benefits from all the CAPTCHAs that their users have solved. These products are competitively priced, with all three charging $1.50 per 1,000 pages. But note that this rate does not include layout analysis, which significantly reduces the appeal; see subsection 3.1.3 for more on this.

Transkribus is a commercial text extraction provider specialising in historical texts, co-operatively funded by institutions such as the British Library (18). It provides a platform with a variety of general-purpose models as well as options to fine-tune on custom data sets. Their flagship model is the "Text Titan", an OCR model specialising in historical text recognition based on the TrOCR architecture (19). It was trained on a variety of texts from the 16th century to the present day. At around $100 per 1,000 pages, only the most well-funded researchers or organisations could use this service to extract the text of any corpus at scale. The "Text Titan" excels at transcribing rare scripts and foreign languages. But for modern handwritten English text samples, it falls behind other approaches such as LLMs, with one study finding that GPT 4o-mini and Claude 3.5 Sonnet make around five times fewer mistakes (20).

Unfortunately, the issue all OCR engines face is a more fundamental one: reading the words on the page is only the first step, reading them in the right order is tougher. OCR works well when text flows line-by-line, like in books or in this report. But unlike a book, the

text of newspapers is all over the page. The complex, multi-column layout confuses many OCR engines, which struggle to distinguish between a margin separating two columns and some particularly large whitespace between words. Applying OCR would only be the first step towards turning scans into stories.

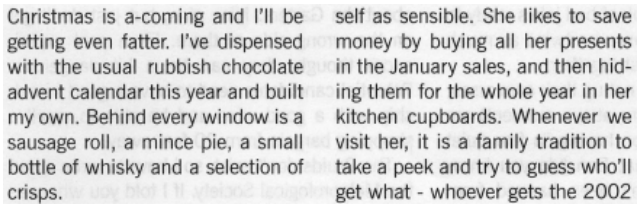
3.1.3. Text extraction pipelines

The downside to pure OCR engines is that they read line-by-line with no regard for reading order, which is why they now usually form a single component in a larger text extraction pipeline.

Some of the cloud providers do offer layout analysis add-ons for their OCR engines. But this "add-on" costs more than the OCR itself, with Amazon, Microsoft, and Google charging $5.50, $10 and $11.50 per 1,000 pages respectively. Suddenly, the cloud offering seems less enticing.

Turning to open-source solutions, Surya is a popular OCR and document layout analysis toolkit with over 15,000 stars on GitHub (21). Its benchmarks find it to be both faster and more accurate than Tesseract. Marker, from the same author, is a complete command-line tool to convert documents from PDF to Markdown, built atop Surya (22). It applies text, layout and reading order detection, table and formula recognition and OCR to extract document text in Markdown format.

MinerU is another example of a command-line tool that converts PDF documents to Markdown (23). Similar to Marker, it applies OCR when necessary or extracts text directly from a digitally-produced document. It applies both rule- and model-based techniques to predict the reading order before extracting the text, tables and formulae.



Christmas is a-coming and I'll be getting even fatter. I've dispensed with the usual rubbish chocolate advent calendar this year and built my own. Behind every window is a sausage roll, a mince pie, a small bottle of whisky and a selection of crisps.

self as sensible. She likes to save money by buying all her presents in the January sales, and then hiding them for the whole year in her kitchen cupboards. Whenever we visit her, it is a family tradition to take a peek and try to guess who'll get what - whoever gets the 2002

(a) Extract from *Felix* issue 1251, page 3

getting even fatter. I've dispensed o with the usual rubbish chocolate woin the January sales, and then hidadvent calendar this year and built ing them for the whole year in her my own. Behind every window is a s sausage roll, a mince pie, a small visit her, it is a family tradition to bottle of whisky and a selection of butake a peek and try to guess who'll crisps.lw upy blot l th wheight lanigolon get what - whoever gets the 2002

(b) The text that Marker extracted

Figure 3.1: Not only does Marker omit the first line, it also reads across the multi-column layout.

Marker and MinerU are popular all-round tools and perform well on modern documents including research papers, company reports, exam papers and slideshows. But historical newspapers are evidently not their forté. While the text recognition fares well, they struggle with the layout. There do exist more specialised systems that tackle newspapers specifically.

American Stories, a project at Harvard, aimed to extract the text of the Library of Congress newspaper collection on a budget of around $3 per 1000 pages (3). They applied a modular text extraction pipeline with custom-trained layout detection, legibility classification and

OCR models to extract the text of 438 million articles. Designed to run cheaply in the cloud, it trades some text extraction accuracy to keep costs down (24).

3.1.4. Large vision-language models

With interest in large language models ever increasing, is there potential in using them to perform the entire text extraction workflow in one step? Recent attempts suggest incredible performance at a fraction of the cost. Readily available models excel at extracting text, even from handwritten documents, achieving near-human levels of accuracy (25).

Many specialist text extraction models are based on *Donut*, an OCR-free document-understanding transformer (26). Nougat is one such tool, trained to extract the text of academic papers (27). While the authors claim its learnings transfer to other document types at reduced accuracy, the pages of *Felix* were no match in my tests, with the model repeating random characters in an endless cycle. This is a known issue for non-Latin scripts (27, p. 9), and seemingly newspaper scans too. Some papers have looked at extending Nougat to old books, such as μgat (28), but not to more complex layouts such as those found in newspapers.

Microsoft trained a more general model, Kosmos-2.5, to convert any text-dense image to Markdown (29). The model excels at OCR benchmarks. Despite training on modern documents from the internet, the model seemed to output reasonably accurate text when provided *Felix* scans. But I encountered errors when running model inference for more than 1024 tokens. This effectively breaks a project requirement, as some denser pages of the paper fit over 2,000 words. Sometimes, the model started repeating the same token over and over, and sometimes it crashed altogether. Moreover, the model is simply slow. The time it took to process each page indicated that it would take far too long to apply it to the entire archive collection.

One of the things that many papers on new open-source visual LLMs have in common: comparing themselves to GPT, claiming that they are only two steps behind. One paper presenting a model called InternVL 1.5 is even aptly titled 'How far are we to GPT-4V?' (30) If these models work so well and are cheap now too, why not give them a try?

OpenAI has GPT, Anthropic has Claude, Google has Gemini, the list goes on. The models impress at text transcription. Evaluated with OCRBench (31), and its successor OCRBench v2 (32) Gemini leads on both benchmarks by its total score. I tried using commercially-available LLMs to transcribe some sample pages of *Felix*. The extracted text was impressively accurate, and the models seemed capable of segmenting the output into individual articles too. Performing better than any other approach I had tried while operating quickly and cheaply, I decided this was the method to use.

3.2. Implementation

3.2.1. Model selection

I tested the three readily available commercial LLMs: Gemini Flash (33, 34), Claude Sonnet and GPT. On smaller newspaper clippings, all three read the text impressively well. In fact, they were often more accurate than when I hand-typed the text. In one sample, Gemini extracted text at 99.96% accuracy and all three often attained 99% accuracy.

Processing entire pages of the newspaper is a tougher challenge for the models. Here the models started to struggle, particularly Claude and GPT. They would hallucinate and misread article text. Digging into the documentation, it turns out that although the models place generous limits on the size of the image you upload, internally they severely shrink the images. Claude processes images no larger than 1568 px on the longest edge, while GPT a mere 1152 px. But some of the scans are over 7000 pixels wide! And this size is absolutely necessary to read the text, with earlier issues averaging a font size as small as 6 pt. (This report is printed at 11 pt.)

Gemini, though, supports much larger images: up to 3072 px on the longest edge. This means that for most pages, only minimal down-scaling is necessary and the text remains legible. This was the single most significant factor affecting Gemini's selection as the LLM of choice.

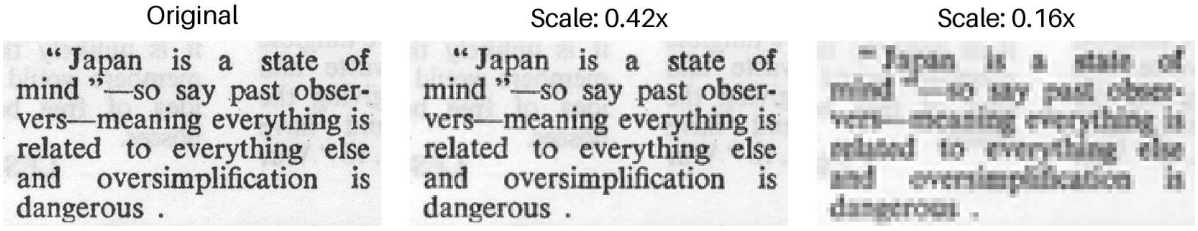


Figure 3.2: The effect of resolution downscaling on text legibility; from left to right: a crop of a newspaper scan, resampled to 3072 px, resampled to 1152 px. The middle sample shows the resolution with which Gemini "sees" the page, and on the right is how GPT "sees" it. It is no surprise that GPT struggles to read the text.

Addendum. Qwen 2.5-VL, released in late January, sidesteps many of the problems I encountered with Gemini and has no fixed limit on input image size (35), but I only found out about it after I had already performed the text extraction.

3.2.2. Prompt design

Starting from a simple prompt of "Extract all the text from this newspaper scan", I gradually built it up to handle a variety of cases and produce the results in a reliably structured way.

I initially used a response schema to guide the model to produce predictable JSON output with a separate entry for each article. JSON is a convenient way to obtain structured output

from a model, particularly for targeted data extraction. But for longer, multiline passages like the articles that we are working with, it starts to show its flaws. Newlines, quotation marks and special characters have to be escaped, and the models often fail to do so. Trying to parse invalid or broken JSON is no fun.

While Google and OpenAI push for JSON as the output format, Anthropic recommends using XML, which uses more explicit syntax and avoids some common JSON issues, like needing to match opening and closing brackets. The clarity of writing `<tag>` and `</tag>` makes it harder to accidentally use too many or too few such tags, unlike JSON where an extra curly brace can easily slip in. It comes at the cost of some extra tokens, as each `<tag>` will usually tokenize into `<`, `tag` and `>`, but ensures a more predictable format. Then again, it also avoids many unnecessary backslashes and string escape codes, so it saves some tokens there.

For a clearly-defined data structure, XML is perfect. To assume that each proper article has a headline is not a stretch, so we can define a `<headline>` field for it. But the text contents vary from article to article, with no single predictable way to record them. What this effectively means is that by default the LLM escapes the whole article body using a `<![CDATA[]]>` directive, transcribing it in Markdown-ish syntax. The loose syntax of Markdown suits lightly-formatted, human-readable text. This same lack of standardisation, though, makes parsing it a pain, such as when you need to remove all formatting markup for text embedding (as we do later).

To counter the flaws of XML, I tried HTML instead. It is similarly structured, but designed to represent content, rather than data, offering the flexibility of a wide array of HTML tags. There is no shortage of HTML on the internet, and hence in the LLM training data either. Asking Gemini to produce HTML output worked surprisingly well, and I later found out that Qwen 2.5-VL was even specifically trained to output HTML when transcribing images (35).

To keep costs low, I tried to keep my prompt as token-efficient as possible. Since it is re-issued for every page, every extra token adds up. I managed to cut it down to just 174 tokens, or $0.62 in prompt costs when summed across the whole corpus. I stopped iterating on my prompt once the LLM appeared to consistently produce the desired output. Below I step through and explain the final prompt I used for extracting the *Felix* corpus.

I start by assigning the model a persona and describing the overall goal. Including "London" or "United Kingdom" in the prompt was an effective way to prevent the model from accidentally using American spellings when interpreting the scans.

> You are an expert typist transcribing articles from archival scans of *Felix*, the Imperial College London student newspaper. Structure your response as a set of one or more HTML articles.

Next, I define the output structure to ensure a reliable format, choosing descriptive field

names such that the model requires little further clarification.

> For each `<article>` you identify, add the following to the `<header>`
>
> a. `<headline>`: article title
> b. `<strapline>`: the subhead or dek, if specified.
> c. `<author_name>`
> d. `<category>`: the section of the newspaper to which the article belongs.
>
> Then in the `<main>` write the article contents in HTML format.

I included a field to record the section of the newspaper since I had hoped to use it to categorise articles into areas like "News" and "Sport". This was optimistic, as the resulting corpus had over 3,000 different section names; making use of that would require substantial cleaning. I could have spent a few more tokens being a bit more specific here.

In my experience, the LLM was often a touch too keen to split the page up into articles. The lines below reign it in, instead suggest it uses heading tags like `<h2>` to represent articles made of several parts.

> Use HTML heading and formatting tags to structure complex articles.
> Only start a new article when the context changes.

In being faithful to the source text, the LLM would often transcribe words hyphenated across lines verbatim. (e.g. `ten minu- tes`) Fixing this took some experimentation, since phrasing it too aggressively would cause it to also remove valid hyphens such as those in double-barrelled surnames or compound nouns. Asking it to "reflow" the text seemed to have the desired effect.

> Please reflow each article into coherent paragraphs.

I encourage the model to stick to what it sees and avoid introducing "helpful" corrections.

> Ensure the transcription is as accurate as possible.
> Preserve original punctuation, capitalisation, and formatting.

The model would also transcribe image captions, or try to describe images. In some more curious edge cases, it would create `<img>` HTML tags pointing to inexistent images on Imgur. The line below tried to prevent this.

> Ignore images and advertisements.

But it ended up being too terse to have the desired effect. The resulting corpus has over 200 articles advertising Subway sandwiches alone. In the same vein, I reckon it is also worth asking it to skip over charts and comics, since their text has little substance when read on its own.

3.2.3. Copyright concerns and harmful content

Google's model gracefully handled complex newspaper layouts. It almost never read across the columns of the page, re-connected hyphenated words and joined together paragraphs split mid-way by images or tables. And it did a fair job at article segmentation too, even if the odd paragraph did end up in the wrong article. But as I ran more and more pages through the Gemini, an unexpected flaw emerged.

The reality is that a moderation system checks the model's output in real time for compliance with the company's stance on "harmful" content. While the model is busy faithfully transcribing an article that reports on rape or student suicide, an automated moderation engine gags the model, cutting it off mid-sentence. "This response may violate our content policies" says the error message. While it is nice to think of a rosy history for the university, good journalism reports on the ugly side of student life too.

The second issue is that of copyright concerns. While all three models are trained on the internet at large, Gemini is the only one to actively flag responses that reproduce copyrighted training materials. The company's rationale is clear: they wish to benefit from training on copyrighted material, without running the risk of distributing it further. Already Google skirts the law with the content snippets it provides in search results. And, generally speaking, this is good for many applications of generative AI, where users would want to avoid accidentally distributing copyrighted material themselves.

However, copyright detection obstructs ordinary text transcription. It seems that Google's spiders have already crawled the *Felix* archive, and used this to train Gemini. The model, therefore, often ends mid-answer with a "recitation error", where an automated text similarity search finds that the output is similar to copyrighted training material. The error provides a link to the very *Felix* issue that it is transcribing.[1]

Confirmation that the model was directly trained on *Felix* also opens questions into whether its outputs are a legitimate assessment of its reading and segmentation ability, or rather its ability to regurgitate training material. Since Google discloses neither the model architecture nor the training data set, this is not clear.

When I was first testing Gemini in November 2024, the rejection rate was very high. It almost made me reconsider the choice of model. I took a break from the project for exams, and on returning to it in January found that Gemini was suddenly far more relaxed. This curiously coincided with Trump's inauguration. While Google made no public statements, Meta, for example, did openly state that they were relaxing their content moderation policies (36), widely interpreted as a move to court the new president. Convenient for me, though: with significantly fewer refusals getting in the way, I stuck to Gemini. The overall rejection rate on the full corpus extraction was 2%, and overwhelmingly due to copyright rather than harmful content.

[1] Curiously, this is usually not the copy located on the *Felix* website, but instead a document scraping site that has reproduced it without the paper's permission.

3.2.4. Extraction client

To perform the text extraction, I built Tupsar, a command-line tool to convert newspaper scans into HTML articles with support for various large language models.

> (Akkadian) *ṭupšar* — a scribe, a scholar, a learned man. (37, p. 379)

The program uses LangChain (38), a Python library that provides a standard interface for working with LLMs. For fast conversions, it makes separate asynchronous API calls for the individual pages of a newspaper. It validates and parses each LLM response, saving each article in the response to a separate HTML file.

3.3. Evaluation

I create a ground truth sample set of articles across the years and compare the transcription accuracy of four LLMs against the American Stories text extraction pipeline and against ABBYY FineReader 9.0, the OCR engine originally applied to the *Felix* scans in 2009.

3.3.1. Method

Evaluating the transcription accuracy of different methods requires some verified "ground truth" against which to compare. Creating a data set of such articles is incredibly time-consuming, as it required me to manually typeset each sampled page. My typing speed may be fast, but verifying each transcription for an absence of any typographical errors slows things down. Since it takes so long to procure this kind of evaluation data set, the choice of the sampled newspaper pages becomes even more important.

I choose not to evaluate the transcription of "born-digital" newspaper issues, since these could technically be extracted with 100% accuracy by reading the text stored in the PDF itself. This is not as simple as it sounds, since it would require a page layout analyser, but it is feasible. Instead I focus on the scanned pages of pre-2002 issues. With about 1200 such issues, I randomly select a single text-dense page from every 100-page block to transcribe. The twelve sampled pages, presented below, use a variety of different typefaces and capture the different printing methods used across the years.

To standardise the evaluation across the different extraction approaches, I offer some leniency for FineReader and American Stories. Since these two OCR engines are not designed to produce neatly-formatted prose, I manually order the text blocks to create the text transcript, to avoid penalising them for not being able to detect the correct natural reading order.

Since the most capable LLMs are incredibly accurate at transcription, small noise starts to play a substantial part in affecting the metrics.

To avoid including minor differences when comparing different tools' accuracy, I normalise text by replacing all special characters with their ASCII equivalents. This way, `café` becomes `cafe` and a typesetter's quotation marks “ and ” are standardised to `"`. Since such alternatives make no difference to the quality of the text, I exclude them to focus on more substantial transcription errors. I also remove repeated whitespace characters for the same reason.

3.3.2. Results

The standard metric for evaluating OCR transcription accuracy is the CER (character error rate), defined as the Levenshtein distance (39) between the transcription text and the reference text, divided by the length of the reference text (40). A lower CER indicates higher accuracy. The WER (word error rate) is a similar metric that operates at the word level rather than the character level. Each misspelled word disrupts the user's understanding of the text, so the WER is generally a more reliable indicator of the perceived transcription accuracy. As transcription has become more accurate, these metrics have become less reliable an indicator of the usefulness of the transcription. Instead, precision is a better metric, since historical sources will always have some data missing, but fabricated or incorrect data is particularly costly (41). This is particularly important for LLMs, which

often hallucinate. I use spaCy (42), a natural language processing library, to split the text into words, and RapidFuzz (43), a fast string similarity library, to calculate the metrics.

	CER (character error rate)		Word-level precision (%)	
Engine	mean ± st. dev.	median	mean ± st. dev.	median
ABBYY FineReader 9.0	0.340 ± 0.257	0.348	46.3 ± 29.0	41.3
American Stories	0.296 ± 0.278	0.152	77.0 ± 16.8	82.2
GPT 4.1	0.262 ± 0.225	0.278	71.2 ± 26.3	76.7
Claude 3.7 Sonnet	0.086 ± 0.101	0.054	88.9 ± 12.8	94.2
Gemini 2.0 Flash	**0.004** ± **0.004**	**0.003**	99.0 ± **1.1**	99.2
Qwen 2.5-VL	0.007 ± 0.008	**0.003**	**99.1** ± 1.2	**99.6**

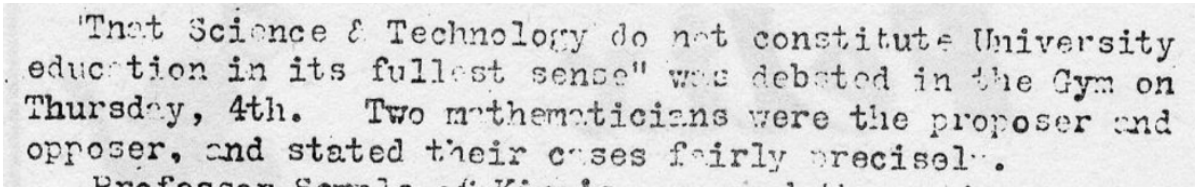
Table 3.1: Text extraction accuracy comparison

The results find that Gemini 2.0 Flash and Qwen 2.5-VL both transcribe historical newspaper scans incredibly accurately, with Claude 3.7 Sonnet not far behind. Of the four LLMs evaluated, GPT performs worst due to its low maximum input image resolution, aligning with previous works evaluating its transcription abilities (44). All four LLMs produce more accurate output text than the original OCR.

Notably, for all its promise of cost-effective accuracy, the American Stories text extraction pipeline underdelivered. On the *Felix* evaluation data set it achieved neither the CER of 0.051 claimed in their American Stories paper (3, p. 7) nor the CER of 0.112 in the EffOCR paper (24, p. 584).

The difference between Qwen 2.5-VL and Gemini 2.0 Flash is a subtle one. Qwen has a tendency to correct typographical errors in the original and add stylistic commas absent in the scan. For example, it adds an ‘e’ to ‘succeding’ and a comma after ‘Firstly’, but such errors are insignificant and could probably be mitigated against with better prompting. Penalised for every such mistake, Qwen 2.5-VL fares worse than Gemini 2.0 Flash by CER. This same tendency to add “corrections”, though, bolsters Qwen’s precision on more difficult scans. While Gemini faithfully transcribes all the printing defects, Qwen figures out the gaps and gets more words right.

Figure 3.3: Newspaper clipping with poor print quality from *Felix* issue 49, with transcriptions from Gemini and Qwen below. Blue highlights characters misread, red highlights characters missed altogether.



Gemini 2.0 Flash reads: "That Science & Technology do not constitute University education in its fullest sen ce" wos debated in the Gym on Thursday, 4th. Two m e thematicians were the proposer and op r oser, and stated their cases fairly precisel y.

Qwen 2.5-VL reads: " That Science & Technology do not constitute University education in its fullest sense" was debated in the Gym on Thursday, 4th. Two mathematicians were the proposer and opposer, and stated their cases fairly precisely.

Qwen 2.5-VL and Gemini 2.0 Flash are both strong contenders on the evaluation data set. But in my preliminary testing, the newer version of Gemini, 2.5 Flash (not evaluated), has a far higher refusal rate. Qwen, meanwhile, has happily processed all of the signs I have provided it, showing no signs of censorship at all, not even on articles about Tiananmen Square, which other Chinese-made models (such as DeepSeek) refused to process. Moreover, Qwen was even specifically trained to output a special 'Qwen HTML' dialect that also includes bounding box coordinates for the individual paragraphs, making cross-referencing transcribed paragraphs with the original scan even easier.

But Qwen has its flaws too. Most notable is the higher price, as it lacks the economies of scale of Gemini. I used Parasail as my inference provider, who charges $0.70 per million tokens (MTok) both for input and output, noticably greater than Gemini 2.0 Flash, which costs $0.10/MTok for input and $0.40/MTok for output. This is further compounded by Qwen 2.5-VL's optimisation for Chinese, which results in its English tokens, on average, representing fewer characters. Therefore the same text would amount to a higher number of tokens and subsequently a higher cost when given to Qwen over Gemini.

3.3.3. Limitations

Given that *Felix* has been online for years, LLMs may have already trained on much of its contents. It is not clear whether the models are truly good at reading the text, or if they have memorised the OCR text that they were fed. But all may not be lost. Around one in ten *Felix* issues are not available online, as these were absent at the time of scanning. Many of them are available physically in libraries at Imperial and UCL. Digitising these copies would provide scans that could not have been scraped from the web or trained on by LLMs. This makes them a potentially incredibly useful evaluation data set, naturally withheld by fate. While I have contacted the Imperial College Archives Unit, they are still yet to respond about the feasibility of locating and digitising this remaining 10%.

3.4. Future work

Extracting the text of the articles took far more time than I expected, yet still I had to cut some corners to keep the project moving. Had I more time (and money) I would have spent longer trying different LLMs, different prompts, and checking the resulting text extraction and article segmentation accuracy.

Throughout my prompt design, for example, the cost of the prompt nagged me. Every new instruction, repeated across 35,000 pages, an extra cent. So I tried to keep the prompt terse. Now, though, I have second thoughts on whether this was the right approach. Anthropic's system prompts for Claude are over 2,000 tokens long! (45) They define every nuance interaction with detail and precision. Were I to use a longer prompt, perhaps even one that asks the model to 'reason' about the page before transcribing it, the results could be made even cleaner, and not require so much post-processing.

I also could have been more systematic in setting model parameters, such as temperature and top-k. I kept the temperature low for predictable responses, but once I had twiddled with the knobs enough to find values that gave reasonable results, I stuck with them.

Then there is the question of model choice. For example, Qwen 2.5-VL proved to be both accurate and incredibly capable at reading finer text, without the copyright qualms of Gemini. Had I done the project even a month later I would have probably ended up using it. The field is so quick to advance, with new models and techniques every month. Simply swapping out the models for their successors would already bring improvements. Making the most of them would bring even more.

4. Corpus post-processing

4.1. Data cleaning

Once I settled on using Gemini 2.0 Flash for the text extraction, I ran it on all 35,000 pages. For $140, or $3.92 per 1000 pages, Gemini produced around 107,000 articles.

The text quality of the extracted articles was far cleaner than a traditional OCR engine. But the output still required a fair amount of cleaning.

Plenty of the extracted "articles", for example, turned out to be front-page teasers for an article deeper in the issue. Since the real value is in the article itself, I delete articles fewer than 100 characters in length.

There were some articles that showed edge cases in the LLM extraction process. Namely, Gemini would at times start repeating certain punctuation characters until it reached the maximum number of output tokens. Fortunately, a simple regular expression replacement can truncate these long runs of repeated punctuation.

Next, there are articles with little semantic or historical value. For example a sudoku grid, which consists solely of numbers. I delete articles where non-word characters make up more than 60% of the text.

Finally, some further cleaning was necessary for the extracted fields like the author name and the newspaper section. Gemini would sometimes fill them in with a 'helpful' descriptor like 'N/A', 'Staff' or 'Not specified'. It would have probably been worth making the prompt a bit longer to prevent this kind of behaviour in future. I manually went through frequently occuring entries for these fields and replaced these with `NULL`.

4.2. Rejoining articles

Since the LLM extraction process operates at the page level, some articles that span multiple pages are extracted as several parts, each recorded as a separate article.

In the most simple case, some articles continue from one page to the next. There are methods to work around this, for example μgat (28) feeds the model the preceding page as additional context. This works well for books, which are structured linearly. But in a newspaper not all articles appear on contiguous pages. Some articles start on the front page and continue further in the issue. Less intuitively, some articles can start on the back cover and continue on earlier pages. Furthermore, some issues have a miniature issue within, known as a *pullout*, located at the central pages to let you physically pull it out. The same problems with articles spanning multiple pages hold for the pullout.

All in all, there are very few assumptions you can make about spanned newspaper articles. They can start at any place in the issue and continue at any place else. This makes re-joining articles split across multiple pages very hard.

There is not all that much research on this topic either. One thesis paper from 2021 fine-tuned a BERT model to classify a reference text block and its continuation candidate as "next" or "not next" depending on whether the two should be joined together (46). While it was 94% accurate at determining *whether* two text blocks should be joined, it performed far worse (only 19% accurate) at selecting the correct continuation text block out of a set of candidates.

With little prior research, I had to get creative. I started by trying to at least *identify* articles that were split across pages. To achieve this, I wrote a regular expression to look for phrases like "continued on page 3" or "cont. from front page", known as *jumplines*. This found around 1,000 articles in the data set (i.e. ~1%) with obvious continuation markers. Naturally, this does not find all the articles without a jumpline, such as those that simply flow from one page onto the next.

Since articles can start on any page and continue anywhere else, it is effectively an $O(n^2)$ problem unless we can narrow it down. My first idea was to embed the start and the end of each article in an issue and to compare the embedding vectors, since calculating a cosine similarity matrix is very fast. I use `wtpsplit` (47) to segment each article into sentences and then embed the first and last few sentences in each article. For the few issues that I manually inspected, this method did not always return the correct continuation as the closest match, but it was good at narrowing down all the possibilities to a select few candidates. The correct match was usually within the five closest.

Since LLMs are effectively next-token predictors, my next idea was to measure the LLM's "surprise" when faced by a possible continuation text. In more technical terms, I wanted to calculate the *perplexity* for the continuation text given the starting text.

Since calculating perplexity is not a function offered by cloud LLM providers, I needed a model that would fit in the memory of a workstation GPU. I used `Qwen3-0.6B` for its strong performance despite its smaller size (48).

Calculating the perplexity matrix for all possible pairings takes far longer than calculating their embeddings. The resulting data reveals that, despite providing the starting text, the perplexity is largely determined by the coherence of the continuation text. I compared the token-level probabilities of the continuation text returned by the model both with and without the starting text given. It appears the only additional "surprise" manifests in the first couple of tokens. Beyond the first few words, the token-level probabilities are almost identical regardless of the context the model is provided.

In other words, using an LLM's perplexity to re-join articles does not work, because it will try to attach the most smoothly-written passage in the issue to the end of most articles.

I tried a hybrid approach, narrowing down the search space with embeddings and then evaluating their continuation likelihood using perplexity. The results were still underwhelming. Since joining two articles incorrectly would be worse than leaving them separate, and since the proportion of split articles in the data set is relatively low, I decided it would be better to leave split articles as they are.

I reckon that the issue of split articles is not an intractable problem, and it could be solved with a different approach. For example, with generalist LLMs showing remarkable results in all sorts of language processing, it may be feasible to simply ask a large model on whether to rejoin two articles, even if slightly "overkill".

Alternatively, as models' context windows increase further and further, it may be possible to alter the extraction process instead to avoid the issue of split articles in the first place. Currently models still struggle when presented with more than a couple images, but in the very near future it may be possible to load the entire newspaper issue as a series of images into the model's context window and process the entire issue in one go. Research has already shown promising results for transcribing historical books when loading the preceding and subsequent page (28); perhaps split articles will soon be a non-issue altogether.

5. Article retrieval and analysis

The next step is to make sense of a text corpus of over 100,000 articles and 30 million words. The goal is to help users find articles that they want to read, and quickly.

5.1. Searching for articles

The prevalence of search tools on the web makes them a familiar tool for users. But providing relevant search results is not as easy as it seems.

The most basic form of search is a *keyword search*, which looks for exact occurrences of the search query in the corpus. Keyword search is so common that it is even built in to many popular databases.

Searching for phrases verbatim yields decent results — if you know exactly what you are looking for. Especially with a historical archive, though, this is not always the case. Therefore going beyond keywords is desirable, allowing for broader queries and finding results with synonyms.

Web search engines started supporting natural language queries, or *semantic search*, in the 2010s. The components to make that possible, such as synonym handling, used to be quite a manual affair (49).

Nowadays, specialised models trained on large text corpora learn to identify semantic similarity by representing texts as vectors ("embedding" them) in a multidimensional space (50). The closer two texts' vectors are to one another, the closer they are in meaning. Newer still, instead of training such models from scratch, many of those on the leaderboards are "distilled" from LLMs, taking advantage of their vast knowledge learnt during training (51, 52).

A common metric used to evaluate text embedding models is the MTEB (Massive Text Embedding Benchmark) (53). As of May 2025, the № 1 embedding model on the MTEB leaderboard is Gemini Embedding, which Google distilled from their Gemini series of LLMs (54).

Not only does Gemini Embedding excel on benchmarks, it is further suited for retrieving articles thanks to its support for different *task types*. Google trained the model to understand a task description, such as "question answering" or "fact checking", when prepended to the input text. (54, p. 4) We can leverage this by embedding existing articles as *documents* and users' search terms as *retrieval queries*.

Since both keyword search and semantic search have their merits, a standard approach is to use a hybrid algorithm that combines the two, a process known as *reranking*. While

newer reranking methods based on machine learning exist, the RRF (reciprocal ranked fusion) algorithm is still a fast, simple, tried-and-tested way of producing better results than either keyword-based or semantic retrieval alone (55). It combines the rankings output by the two components and weights them equally, favouring results that rank highly in both.

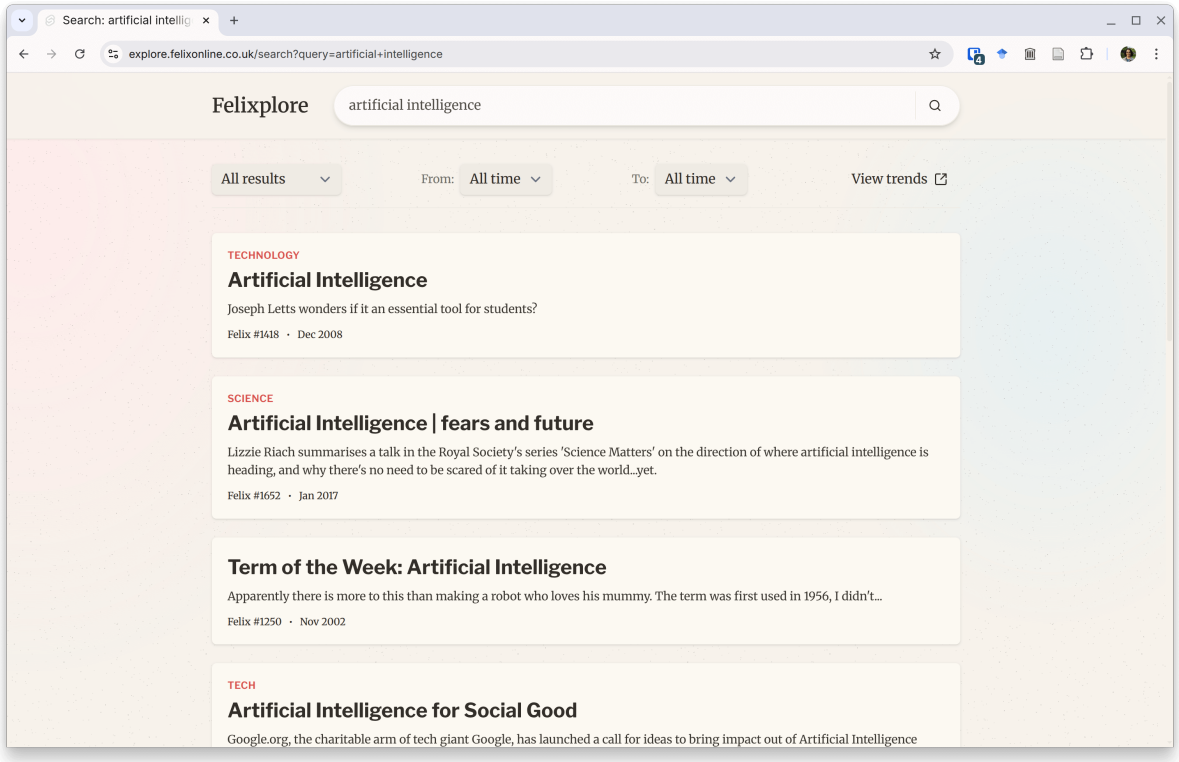
5.1.1. Implementation

I use a PostgreSQL database to store the articles, since it supports full-text search out-of-the-box, and supports vector search using the `pgvector` database extension.

Despite running in the cloud, text embedding is surprisingly fast and cheap. I wrote a program to step through the entire *Felix* data set, sending each article to Gemini Embedding and saving its vector to a local database. For just £4.30, the price of a latté at Pret, all the articles were embedded within a few hours. Using a batch processing mode, which takes longer to complete, could further cut costs. At the time of writing, though, Gemini Embedding did not yet support batch processing.

For the web search interface, a server-side TypeScript client makes requests to the database. Keyword searches execute in less than a second, semantic searches take a second longer while the search query is embedded. The RRF algorithm combines the results of keyword and semantic search to produce the most relevant results.

Figure 5.1: Search results for “artificial intelligence” in the *Felix* archive



5.1.2. User feedback

Users spoke positively about the search tool, with one *Felix* editor commenting "I … like how you can search across topics across the years and really see the development of an idea/theme across Imperial and *Felix*'s history"

The most constructive feedback about the search tool I received from Colin Grimshaw, author of the Imperial College Video Archive Blog.

He commented that he would like a different approach to filtering the search results by date: "Is it possible in the pulldown menu to go from 'latest' to 'oldest' rather than specific dates? When searching, you might not know the year/s you need." While the trends page, discussed in the next section, does alleviate some of the difficulty when the exact years are unknown, I agree that further tooling such as a filter to search for articles "around 1971" could be more useful than a strict cut-off.

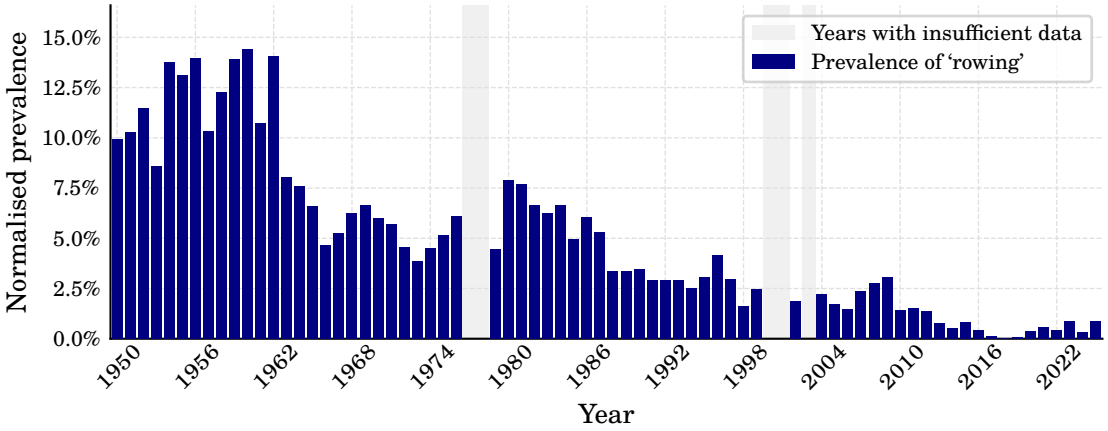
Colin further remarked about the user experience of the search results: "When having found an article could it be possible to go either forward or backward to other issues? Or, similar, but to the next item already found in a search?" These requests would be relatively straightforward to implement as these are not limited by any technical constraints.

He closed his email with "I do like the text summary that now appears from the searched/discovered PDF issue, very handy and worthwhile!"
I am glad I went to the effort of extracting such clean text.

5.2. Finding trends

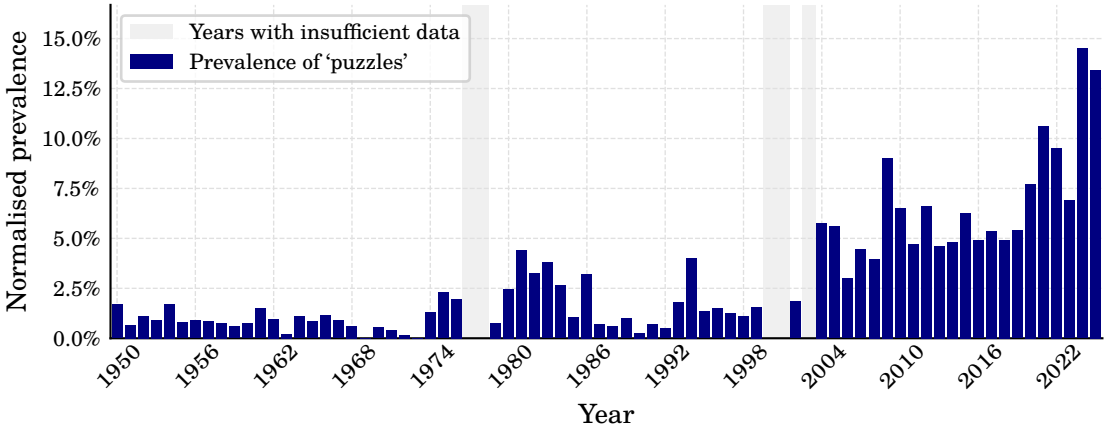
To help users discover broader changes across the years, I created a trend analysis page. Similar to search, it first embeds the user's query using Gemini Embedding, then finds relevant articles. I empirically determined the relevance cut-off and found that a vector distance of up to 0.65 captured topical articles without including more loosely related ones. The algorithm then calculates the *prevalence* of that topic aggregated by year. The prevalence metric is the number of related articles in that year normalised against the total number of articles that year.

Figure 5.2: The declining interest in rowing.



As shown in figure 5.2, the *Felix* archive sees a noticable decline in the number of rowing-related articles starting in the 1960s. Perhaps this could be attributed to the decreasing interest in spectator sports and the increasingly exclusive public perception of rowing.

Figure 5.3: The rising popularity of puzzles.



While crosswords have been a mainstay of newspapers since the early 20th century, the proportion of each issue dedicated to puzzles has certainly increased in recent decades, as illustrated by figure 5.3. This trend could be a reflection of students' preference to consume news itself online, but still preferring to complete puzzles like crosswords and sudoku on paper.

5.2.1. User feedback

Gathering comments, it seems users like the idea of the trends page, but some more clarity is needed on what exactly it represents. “I really like how you can see trends across Felix’s history” commented one *Felix* editor, adding: “I can track changes made by, for example, the Union, for a certain policy or position.” But confusion arose as to what prevalence of a trend signifies: “I think their should be an explanation for what the prevalence metric means” and another commenting “Might be cool if you … it told you what the ‘prevalence’ referred to … to make it more clear what exactly it is”.

Finally, the search tool is designed to search for broad topics or ideas, rather than specific names or events, but it became apparent that I had not made this clear enough in the user interface, with one *Felix* writer saying “when … I search for my own name as a trend, it appears as a single bar in 1980 rather than more recently”. This is expected, but needs to either be made clear to the user, or instead an error message could show when there is unsufficient data to show a meaningful trend.

5.3. Linking past and present

Today’s news stories can seem “unprecedented”, especially on topics like AI and social media. Yet a dig in the archives often reveals their precedents are merely well-forgotten.

Exploring and finding old articles is not inherently an engaging or easy task. Finding a way to link articles across the ages could help to spark historical curiosity and contextualise current events by drawing parallels with stories from the past.

To recommend similar articles, I use News Déjà Vu, a semantic search tool developed at Harvard that finds historical counterparts to modern news stories (56).

Suppose we take a recent article on students using AI in their studies and search the archive for similar stories. We expect to find more articles from the last decade or so on AI. Yet the system highlights a surprising find from 1966: a dystopian vision of future students in 1996 learning from “personal teaching machines”. It seems science fiction has become reality.

Published 14 Mar 2025	Published 9 Mar 1966
How Imperial is dealing with the rise in students using AI As research shows up to 92% of students across the UK use AI in some form for their studies, Imperial begins to adapt for the future. Student use of AI has increased as Large Language Models such as ChatGPT or Claude become more accessible to consumers, with a recent survey...	**White heat** March 9th, 1996. Fred Icprole, average student of the Anti-Imperialist College of Science, Technology and the Useful Arts, drives in from his digs in Bristol. […] In a vast, softly-lit, Muzak-haunted eezi-learn concourse, three thousand fifth year undergraduates are receiving expert instruction from individual teaching machines. […]

5.3.1. Method

While most embedding models work to find similar *texts*, News Déjà Vu finds similar *stories*.

The first step strips the text of specifics, using NER (named entity recognition) to mask out mentions of people and places. A *mask token* replaces each proper noun in the article.

Before NER masking	After NER masking
King Charles III visited Imperial's Centre for Injury Studies on Wednesday 19th February to highlight the UK's support for Ukrainian personnel…	`<mask>` visited `<mask>` on Wednesday 19th February to highlight the `<mask>`'s support for `<mask>` personnel…

Figure 5.4: A sample article, before and after applying NER masking.

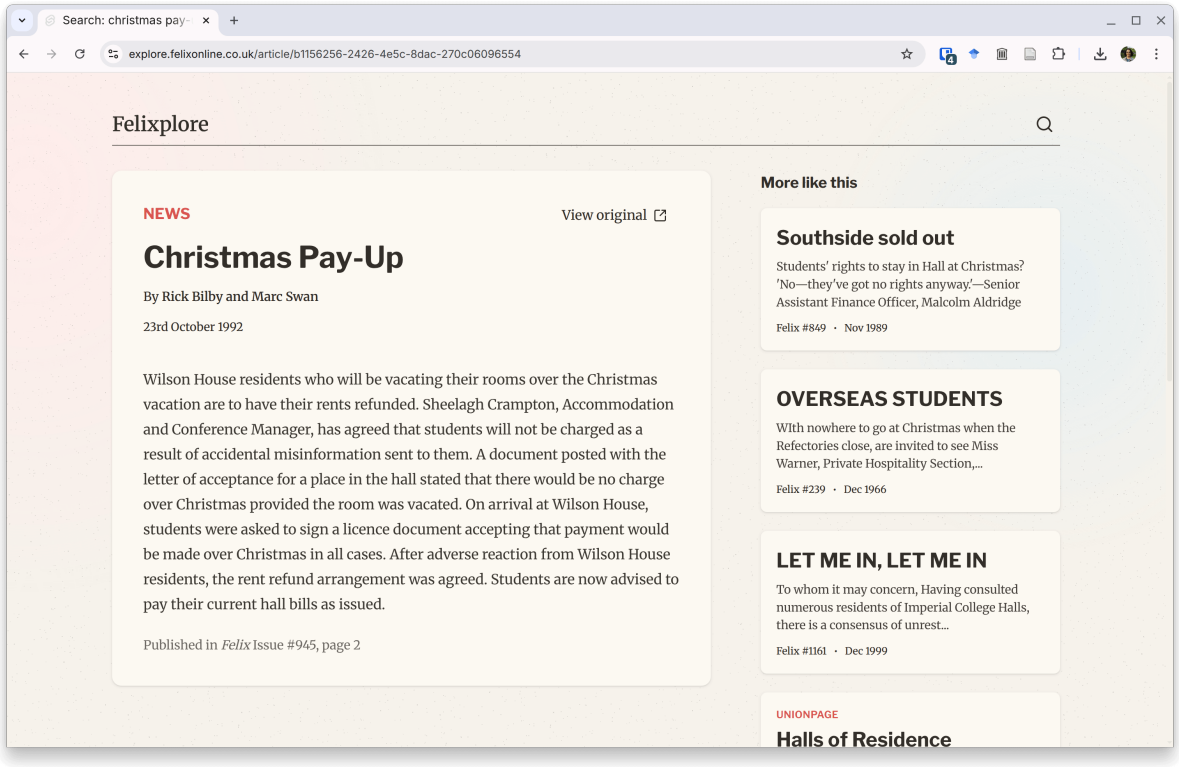
A text embedding model then turns each masked article into 768-dimensional vector. An article's nearest neighbours in this multidimensional embedding space represent the most similar stories.

5.3.2. Implementation

The initial idea was to use News Déjà Vu embeddings for an enhanced archive search that would retrieve results both fresh and old. While this approach did find some interesting articles in the archive, the search results often seemed irrelevant. Users generally prefer a search tool to retrieve what they are looking for, rather than curious finds.

Instead, in the final web app I used News Déjà Vu to provide recommendations in a "more like this" panel to the side of each article, showing a handful of other stories whose vectors are located closest. Here, the occasional suggestion of a similar, but not identical article is a feature rather than an issue as it encourages serendipitous discovery.

Figure 5.5: An article about students not being able to live in halls over Christmas, with articles both newer and older than it recommended about similar occurrences.



5.3.3. Evaluation

Evaluating embeddings is a difficult job, and further complicated in the absence of a data set pairing modern and historical news articles.

Once a user has found an article using the search tool, the overarching objective is to help them explore the archive further. One way of assessing whether the article recommendation system effectively achieves this is to analyse how well it links articles together, both across topics and time.

To achieve this, I model the articles as the nodes of a directed graph, recording inter-article recommendations as edges. Each article node has k outbound edges to its k nearest neighbours. I construct this kind of graph for News Déjà Vu embeddings, and for Gemini Embedding as a baseline to compare against.

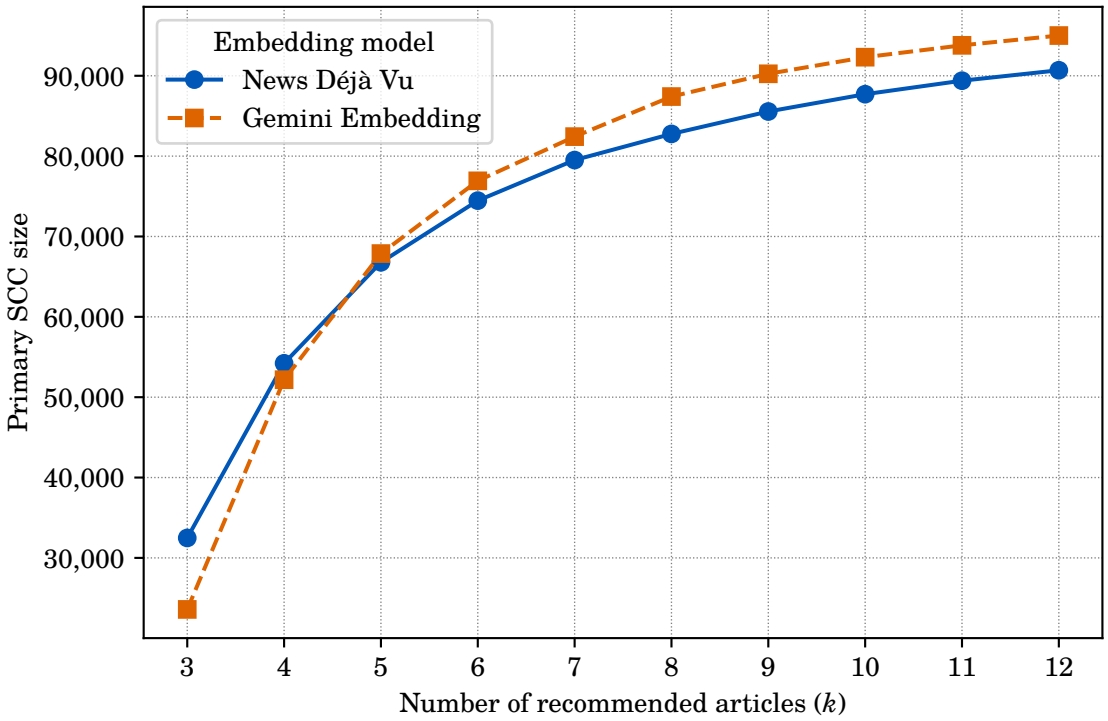
An SCC (strongly-connected component) is a subset of a graph within which every article is reachable from one another. In other words, if you keep clicking on an article in the "more like this" section, you can eventually end up at any other article within the component.

Calculating the SCCs, we find that most articles collect into a one large SCC. This is a good thing, since the aim is to make the archive easy to traverse. Those articles not in the

primary SCC are tiny ($n < 100$) clusters of outliers; manually sampling them reveals that these are often advertisements or crosswords.[1]

Generally, the larger the primary SCC, the better. But offer too many recommendations to the user and risk overwhelming them with choice. Plotting the size of the primary SCC against the number of recommended articles k, we observe diminishing returns beyond $k = 8$, both for News Déjà Vu and Gemini Embedding.

Figure 5.6: The effect of the number of recommendations (k) on the size of the SCC.



The size of the primary SCC is an imperfect metric, since it only measures whether it is theoretically possible to traverse from one article to another. It does not take into account what path such a traversal might take. Broadly speaking, newspapers publish articles in distinct sections, such as "Politics" or "Sport", and we can expect few articles to discuss both at the same time. It would therefore require a very specific sequence of "more like this" clicks to move from one topic to another.

We can identify groups of articles within the larger SCC that are more tightly-knit amongst themselves, known as *communities*, using the Leiden algorithm. When recommending eight articles ($k = 8$), there are 52 such communities when using News Déjà Vu embeddings, and 77 when using Gemini Embedding. The smaller number of Leiden communities for News Déjà Vu suggests that it is better at creating a recommendation network such that a

[1] Crosswords are a particularly curious case. I hypothesise that the model embeds the jumbled collection of odd sentences unusually because unlike an article, a set of crossword clues has no cohesive semantic meaning.

browsing user is less likely to get stuck in a specific part of the archive. By embedding the underlying narrative instead of the specifics, News Déjà Vu softens the barriers between topics.

Another way of evaluating the performance is to compare the temporal patterns of the recommendations of the recommendations. Since we are looking to increase the connectedness of the archive across years, we can calculate metrics such as the mean absolute difference between an article and its recommendations. The difference is substantial. Using News Déjà Vu for recommendations, the average difference between an article and its "more like this" articles is 12 years 10 months, while for Gemini Embedding it is only 4 years 11 months. This holds across metrics too; calculating the percentage of articles which have the majority ($\geq 80\%$) of their recommendations published within five years of the article itself, we find similar results: only 8.9% for News Déjà Vu but 41.0% for Gemini Embedding. In other words, News Déjà Vu shows a broader spread of articles in time, while Gemini Embedding is far more temporally concentrated, which would likely lead to users getting stuck in a specific, narrow timeframe in the archive.

All in all, for the purpose of navigating the newspaper archive, News Déjà Vu seems to perform better, at least on those metrics that I could measure. The topical relevance is, of course, harder to evaluate, and with the broader temporal spread of recommendations some of them could seem less related. But in the interest of promoting cross-period discovery and helping users find curious historical parallels, News Déjà Vu appears to be a good choice compared to a generic text embedding model.

5.3.4. User feedback

For a more subjective evaluation, I demonstrated the tool to some of the *Felix* editors. They agreed that the suggested articles were similar yet interesting, and I observed them actively using the sidebar in their own exploration of the archive. One of the editors was so impressed that he wrote an article about it in the paper![2] (57)

5.3.5. Future work

To achieve robustness to transcription errors, the authors of News Déjà Vu needed to fine-tune their NER model for improved performance. But the underlying model is still RoBERTa (58), which was released in 2019 by Facebook AI. Since the article texts I extracted are far cleaner than the data set that they used, a custom model may no longer be necessary, and a newer general-purpose model could yield more accurate results. Given more time, I would want to evaluate whether a more recent NER model or even an LLM could do a better job.

[2]The article mentions needing a beefy GPU to run the similarity search. While this was true at the time, I have since optimised the code to run CPU-only and demand far less compute.

6. Discussion

This project took the *Felix* archive from mere PDFs to a fully-interactive web platform, allowing users to browse the archive, search for specifics and identify patterns and trends across time.

Prior to the project, the OCR text was of quality only tolerable for targeted keyword searches. I developed Tupsar, a pipeline to convert newspaper scans into segmented articles using large vision-language models. Trying different LLMs, I settled on Gemini for its low cost and its support for high-resolution input images.

Applying Tupsar with Gemini 2.0 Flash to re-digitise the archive, the pipeline successfully processed 98% of the pages and extracted over 100,000 articles. On a small yet representative sample data set that I manually typed up, Gemini achieved a mean CER of 0.4%, a marked improvement over the original 34.0%. With text this accurate, semantic analysis tools do not need to be modified to tolerate OCR noise and users do not need to refer to the original scans so often when reading. The total extraction cost less than $4 per 1,000 pages, comfortably placing this approach in the same price bracket as other budget-constrained newspaper extraction projects, yet achieving a significantly cleaner output.

To open up access to the created data set, I built *Felixplore*, a web application that makes it easier to rediscover the university's history through the *Felix* archive. A hybrid search engine helps users find what they are after by combining the precision of keyword search with the semantic understanding of vector embedding search. Once users have found a way in, News Déjà Vu finds similar stories, improving archive connectedness and discoverability.

User feedback on *Felixplore* has been overwhelmingly positive, with *Felix* contributors making comments such as "wow this looks amazing" and some as ecstatic as "Holy shit that's so cool", before following up with "Can I send this to my friends ??". Users are impressed both by the user interface ("I think it's super clean" and "I quite like how sleek it is") and by the technical aspect ("Wow very good use of LLMs! That was an infeasible problem when I was at Imperial").

In terms of caveats, it is evident that at least some *Felix* issues clearly ended up in the Gemini training data. This calls into question whether the 0.4% CER is a reflection of Gemini's ability to transcribe the newspaper articles or merely of its ability to regurgitate training material. Then again, Gemini does consistently outperform other LLMs on many vision-related benchmarks (32, 59, 60). In the end, the project's goal was to achieve a clean text corpus to work with, and using Gemini successfully achieved that. Ideally, collaborating with the College archives, we would locate and digitise the unscanned 10% of issues, both for completeness' sake and to better evaluate the suitability of LLMs for unseen transcription.

This project also highlighted the practical considerations of working with proprietary, moderated large language models. While the project did successfully convert 98% of the pages, the missing couple percent warn of the inherent copyright concerns in processing archives like this. While processing *Felix* in 2025 only flagged 2% of the articles, in a different month or for another newspaper archive this figure could be far greater. Fortunately, open-source large vision-language models such as Qwen 2.5-VL demonstrated similar accuracy with no such restrictions and could make for an equally capable substitute to Gemini, albeit at a slightly higher price.

Finally, while the text extraction results point to incredibly accuracy, this only evaluates the typical article, and there is room for improving the data set quality further. Obtaining unsatisfactory results in my small-scale testing, I did not end up re-joining long-form articles that are split across several pages. These exist as separate fragments in the database, creating friction in the reading experience as the user resorts to looking at the original scan. These need to be re-joined, or the extraction pipeline needs to be altered to avoid this issue altogether.

While there is room to refine the project further, it has certainly demonstrated the potential of using LLMs to accurately digitise historical newspaper collections and make them more accessible at an affordable price.

In particular, it aligns with very recent findings from other researchers, who have been observing equally impressive results, surpassing those of traditional approaches, for similar archival document processing tasks (61, 62). It seems that the days of the multi-step OCR pipeline may be coming to an end and that the future of historical document transcription lies with LLMs.

Lastly, the project sets a template that other archives could follow. While Imperial was the first UK university to digitise its student newspaper archive, a dozen other institutions have followed suit in the years since. Plenty of newspaper archives either collect dust or exist behind complicated web interfaces seeing few, if any, users. I hope that my approaches in making newspaper archives easier to read and learn from could transfer to other publications too.

7. Declarations

I am the *Felix* digital editor. My work as a part of the newspaper committee is what spurred the idea of analysing the archive in the first place. The project is my own independent academic initiative. It receives no funding from *Felix* and I have received permission to use the contents of the archive for this project.

7.1. Generative AI use

I confirm that I wrote this report *without* the assistance of generative AI tools. I hope you can tell by the writing style. Separately, I do acknowledge the use of LLMs to assist with writing the code involved in this project, including GPT-4o (OpenAI, chat.openai.com) and Gemini 2.5 Pro (Google, gemini.google.com).

7.2. Ethics statement

The *Felix* archive preserves the newspaper exactly as it was printed. It captures the ideas and attitudes of the time—the very thing that makes it a fascinating historical resource. Its pages do use language and include opinions that today's readers may find offensive. The archived issues do not, in any way, reflect the views of the current editorial team.

Whether access to biased or explicit articles needs to be restricted is an actively debated topic both at the *Felix* office and beyond. Historical censorship, as Orwell warned, effectively amounts to rewriting the past. This project aims to make the archive more accessible, not the opposite. Therefore no content was intentionally removed neither from the analysis nor from the final corpus.

7.3. Sustainability

While LLMs are associated with their high energy usage, this is predominantly concentrated in the pre-training phase, which is amortised across all users globally. Inference, while still energy-intensive, pales in comparison, plus the extraction process was a one-off job. With around half of a data centre's operational budget going towards electricity (63), price is a good proxy for energy usage. Since the extraction cost only $140 (£109), less than the average UK utility bill, we can suppose it was energy-efficient.

Furthermore, the project used Gemini, which Google runs on Tensor Processing Units, as opposed to ordinary GPUs used by most inference providers. While to be taken with a pinch of salt, Google did publish a paper in 2017 indicating that TPUs perform 30–80 times more operations per watt of energy expended compared to GPUs (64). This efficiency is also a likely factor as to why Google is able to offer such competitively priced model inference.

7.4. Availability of data and materials

The *Felix* issues analysed in this project are available online at archive.felixonline.co.uk.

The source code for Tupsar, the command-line article extraction tool I created, is available on GitHub at https://github.com/zeevox/tupsar.

Felixplore, the web interface for browsing and analysing the archive, is available online at https://explore.felixonline.co.uk.

The source code for *Felixplore* is available at https://github.com/zeevox/felixplore.

Owing to copyright laws, the final set of extracted articles constitutes a "faithful reproduction" of the original work, and is therefore protected by the same copyright as the print newspaper. Hence, while I would like to, I cannot make this data set publicly available, for example as a HuggingFace data set. You can search and explore the extracted articles using *Felixplore*.

References

(1) Lee BCG, Mears J, Jakeway E, Ferriter M, Adams C, Yarasavage N et al. The Newspaper Navigator Dataset: Extracting Headlines and Visual Content from 16 Million Historic Newspaper Pages in Chronicling America. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20: The 29th ACM International Conference on Information and Knowledge Management. Virtual Event Ireland: ACM, 2020, p. 3055–3062. DOI: 10.1145/3340531.3412767.

(2) Schultze C, Kerkfeld N, Kuebart K, Weber P, Wolter M and Selgert F. *Chronicling Germany: An Annotated Historical Newspaper Dataset*. Version 3. 2024. DOI: 10.48550/arXiv.2401.16845. arXiv: 2401.16845 [cs]. Preprint.

(3) Dell M, Carlson J, Bryan T, Silcock E, Arora A, Shen Z et al. American Stories: a large-scale structured text dataset of historical U.S. newspapers. In: *Advances in neural information processing systems*. Ed. by Oh A, Naumann T, Globerson A, Saenko K, Hardt M and Levine S. Vol. 36. Curran Associates, Inc., 2023, p. 80744–80772. DOI: 10.48550/arXiv.2308.12477.

(4) Nedić J. *Grant request for the Felix Digital Archive*. Imperial College Union, 2009. https://eactivities.union.ic.ac.uk/upfiles/committeemeetingpaper/1468 [Accessed 14th May 2025].

(5) Van Strien D, Beelen K, Ardanuy MC, Hosseini K, McGillivray B and Colavizza G. *Assessing the impact of OCR quality on downstream NLP tasks*. SCITEPRESS - Science and Technology Publications, 2020. DOI: 10.17863/CAM.52068.

(6) Bazzo GT, Lorentz GA, Suarez Vargas D and Moreira VP. *Assessing the Impact of OCR Errors in Information Retrieval*. In: *Advances in Information Retrieval*. Ed. by Jose JM, Yilmaz E, Magalhães J, Castells P, Ferro N, Silva MJ et al. Vol. 12036. Cham: Springer International Publishing, 2020, p. 102–109. DOI: 10.1007/978-3-030-45442-5_13.

(7) Nedić J. *Felix* to create first digital archive of its kind in the UK. In: *Felix, Issue 1427* (2009), p. 3. https://issues.felixonline.co.uk/felix_1427.pdf.

(8) Shet V. *Street View and reCAPTCHA technology just got smarter*. Google Online Security Blog. 2014. https://security.googleblog.com/2014/04/street-view-and-recaptcha-technology.html [Accessed 14th May 2025].

(9) Todorov K and Colavizza G. An Assessment of the Impact of OCR Noise on Language Models. In: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*. 14th International Conference on Agents and Artificial Intelligence. Vol. 2. SCITEPRESS - Science and Technology Publications, 2022, p. 674–683. DOI: 10.5220/0010945100003116.

(10) Zhang J, Zhang Q, Wang B, Ouyang L, Wen Z, Li Y et al. *OCR Hinders RAG: Evaluating the Cascading Impact of OCR on Retrieval-Augmented Generation*. Version 2. 2025. DOI: 10.48550/arXiv.2412.02592. arXiv: 2412.02592 [cs]. Preprint.

(11) Kettunen K, Keskustalo H, Kumpulainen S, Pääkkönen T and Rautiainen J. OCR Quality Affects Perceived Usefulness of Historical Newspaper Clippings. A User Study: 18th Italian Research Conference on Digital Libraries, IRCDL 2022. In: *CEUR Workshop Proceedings* 3160 (2022). https://ceur-ws.org/Vol-3160/paper1.pdf [Accessed 15th May 2025].

(12) Rigaud C, Doucet A, Coustaty M and Moreux JP. ICDAR 2019 Competition on Post-OCR Text Correction. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019 International Conference on Document Analysis and Recognition (ICDAR). Sydney, Australia: IEEE, 2019, p. 1588–1593. DOI: 10.1109/ICDAR.2019.00255.

(13) Soper E, Fujimoto S and Yu YY. BART for Post-Correction of OCR Newspaper Text. In: *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*. Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021). Online: Association for Computational Linguistics, 2021, p. 284–290. DOI: 10.18653/v1/2021.wnut-1.31.

(14) Thomas A, Gaizauskas R and Lu H. Leveraging LLMs for Post-OCR Correction of Historical Newspapers. In: *Proceedings of the Third Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA) @ LREC-COLING-2024*. Ed. by Sprugnoli R and Passarotti M. Torino, Italia: ELRA and ICCL, 2024, p. 116–121. https://aclanthology.org/2024.lt4hala-1.14/ [Accessed 14th May 2025].

(15) Boros E, Ehrmann M, Romanello M, Najem-Meyer S and Kaplan F. Post-Correction of Historical Text Transcripts with Large Language Models: An Exploratory Study. In: *Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024)*. Ed. by Bizzoni Y, Degaetano-Ortlieb S, Kazantseva A and Szpakowicz S. St. Julians, Malta: Association for Computational Linguistics, 2024, p. 133–159. https://aclanthology.org/2024.latechclfl-1.14/ [Accessed 14th May 2025].

(16) Smith R. An Overview of the Tesseract OCR Engine. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). Vol. 2. 2007, p. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

(17) Hegghammer T. OCR with Tesseract, Amazon Textract, and Google Document AI: a benchmarking experiment. In: *Journal of Computational Social Science* 5.(1) (2022), p. 861–882. DOI: 10.1007/s42001-021-00149-1.

(18) Kahle P, Colutto S, Hackl G and Mühlberger G. Transkribus - A Service Platform for Transcription, Recognition and Retrieval of Historical Documents. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). Vol. 04. 2017, p. 19–24. DOI: 10.1109/ICDAR.2017.307.

(19) Li M, Lv T, Chen J, Cui L, Lu Y, Florencio D et al. TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.(11) (11 2023), p. 13094–13102. DOI: 10.1609/aaai.v37i11.26538.

(20) Crosilla G, Klic L and Colavizza G. *Benchmarking Large Language Models for Handwritten Text Recognition*. Version 2. 2025. DOI: 10.48550/arXiv.2503.15195. arXiv: 2503.15195 [cs]. Preprint.

(21) Paruchuri V. *Surya: OCR, layout analysis, reading order, table recognition in 90+ languages*. 2024. https://github.com/VikParuchuri/surya [Accessed 14th May 2025].

(22) Paruchuri V. *Marker: Convert PDF to markdown + JSON quickly with high accuracy*. 2023. https://github.com/VikParuchuri/marker [Accessed 14th May 2025].

(23) Wang B, Xu C, Zhao X, Ouyang L, Wu F, Zhao Z et al. *MinerU: An Open-Source Solution for Precise Document Content Extraction*. Version 1. 2024. DOI: 10.48550/arXiv.2409.18839. arXiv: 2409.18839 [cs]. Preprint.

(24) Bryan T, Carlson J, Arora A and Dell M. EfficientOCR: An Extensible, Open-Source Package for Efficiently Digitizing World Knowledge. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Feng Y and Lefever E. Singapore: Association for Computational Linguistics, 2023, p. 579–596. DOI: 10.18653/v1/2023.emnlp-demo.52.

(25) Humphries M, Leddy LC, Downton Q, Legace M, McConnell J, Murray I et al. *Unlocking the Archives: Using Large Language Models to Transcribe Handwritten Historical Documents*. Version 1. 2024. DOI: 10.48550/arXiv.2411.03340. arXiv: 2411.03340 [cs]. Preprint.

(26) Kim G, Hong T, Yim M, Nam J, Park J, Yim J et al. OCR-Free Document Understanding Transformer. In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 498–517. DOI: 10.1007/978-3-031-19815-1_29.

(27) Blecher L, Cucurull G, Scialom T and Stojnic R. Nougat: Neural Optical Understanding for Academic Documents. In: Twelfth International Conference on Learning Representations (ICLR). 2024. https://openreview.net/forum?id=fUtxNAKpdV.

(28) Quattrini F, Zaccagnino C, Cascianelli S, Righi L and Cucchiara R. *μgat: Improving Single-Page Document Parsing by Providing Multi-Page Context*. Version 1. 2024. DOI: 10.48550/arXiv.2408.15646. arXiv: 2408.15646 [cs]. Preprint.

(29) Lv T, Huang Y, Chen J, Zhao Y, Jia Y, Cui L et al. *KOSMOS-2.5: A Multimodal Literate Model*. Version 2. 2024. DOI: 10.48550/arXiv.2309.11419. arXiv: 2309.11419 [cs]. Preprint.

(30) Chen Z, Wang W, Tian H, Ye S, Gao Z, Cui E et al. How far are we to GPT-4V? Closing the gap to commercial multimodal models with open-source suites. In: *Science China Information Sciences* 67.(12) (2024), p. 220101. DOI: 10.1007/s11432-024-4231-5.

(31) Liu Y, Li Z, Huang M, Yang B, Yu W, Li C et al. OCRBench: On the Hidden Mystery of OCR in Large Multimodal Models. Version 7. In: *Science China Information Sciences* 67.(12) (2024), p. 220102. DOI: 10.1007/s11432-024-4235-6. arXiv: 2305.07895 [cs].

(32) Fu L, Yang B, Kuang Z, Song J, Li Y, Zhu L et al. *OCRBench v2: An Improved Benchmark for Evaluating Large Multimodal Models on Visual Text Localization and Reasoning*. Version 1. 2024. DOI: 10.48550/arXiv.2501.00321. arXiv: 2501.00321 [cs]. Preprint.

(33) Anil R, Borgeaud S, Alayrac JB, Yu J, Soricut R, Schalkwyk J et al. *Gemini: A Family of Highly Capable Multimodal Models*. Version 5. 2025. DOI: 10.48550/arXiv.2312.11805. arXiv: 2312.11805 [cs]. Preprint.

(34) Georgiev P, Lei VI, Burnell R, Bai L, Gulati A, Tanzer G et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. Version 5. 2024. DOI: 10.48550/arXiv.2403.05530. arXiv: 2403.05530 [cs]. Preprint.

(35) Bai S, Chen K, Liu X, Wang J, Ge W, Song S et al. *Qwen2.5-VL Technical Report*. Version 1. 2025. DOI: 10.48550/arXiv.2502.13923. arXiv: 2502.13923 [cs]. Preprint.

(36) Kaplan J. *More Speech and Fewer Mistakes*. Meta Newsroom. 2025. https://about.fb.com/news/2025/01/meta-more-speech-fewer-mistakes/ [Accessed 10th June 2025].

(37) Huehnergard J. *A Grammar of Akkadian*. Eisenbrauns, 2005. 696 pp.

(38) Chase H. *LangChain: Build context-aware reasoning applications*. Version 0.3.53. LangChain, 2022. https://github.com/langchain-ai/langchain [Accessed 12th June 2025].

(39) Levenshtein VI. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In: *Soviet Physics Doklady* 10.(8) (1966), p. 707–710.

(40) Rice SV, Kanai J and Nartker TA. An Evaluation of OCR Accuracy. In: *Information Science Research Institute, 1993 Annual Research Report* 9 (1993), p. 20.

(41) Backer S and Hyman L. Bootstrapping AI: Interdisciplinary Approaches to Assessing OCR Quality in English-Language Historical Documents. In: *Proceedings of the 5th International Conference on Natural Language Processing for Digital Humanities*. Ed. by Hämäläinen M, Öhman E, Bizzoni Y, Miyagawa S and Alnajjar K. Albuquerque, USA: Association for Computational Linguistics, 2025, p. 251–256. DOI: 10.18653/v1/2025.nlp4dh-1.21.

(42) Montani I, Honnibal M, Boyd A, Van Landeghem S and Peters H. *spaCy: Industrial-strength Natural Language Processing in Python*. Version 3.8.7. Explosion, 2020. DOI: 10.5281/ZENODO.1212303.

(43) Bachmann M. *RapidFuzz: Rapid fuzzy string matching in Python using various string metrics*. Version 3.13.0. 2020. https://github.com/rapidfuzz/RapidFuzz [Accessed 14th May 2025].

(44) Ghiriti A, Göderle W and Kern R. Exploring the Capabilities of GPT4-Vision as OCR Engine. In: *Linking Theory and Practice of Digital Libraries*. Ed. by Antonacopoulos A, Hinze A, Piwowarski B, Coustaty M, Di Nunzio GM, Gelati F et al. Cham: Springer Nature Switzerland, 2024, p. 3–12. DOI: 10.1007/978-3-031-72440-4_1.

(45) *Claude System Prompts*. Anthropic. 2025. https://docs.anthropic.com/en/release-notes/system-prompts [Accessed 4th June 2025].

(46) Estmark A. *Text Block Prediction and Article Reconstruction Using BERT*. Uppsala University, 2021. https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-447248 [Accessed 8th June 2025].

(47) Frohmann M, Sterner I, Vulić I, Minixhofer B and Schedl M. Segment Any Text: A Universal Approach for Robust, Efficient and Adaptable Sentence Segmentation. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2024. Ed. by Al-Onaizan Y, Bansal M and Chen YN. Miami, Florida, USA: Association for Computational Linguistics, 2024, p. 11908–11941. DOI: 10.18653/v1/2024.emnlp-main.665.

(48) Yang A, Li A, Yang B, Zhang B, Hui B, Zheng B et al. *Qwen3 Technical Report*. Version 1. 2025. DOI: 10.48550/arXiv.2505.09388. arXiv: 2505.09388 [cs]. Preprint.

(49) *Synonym identification based on co-occurring terms*. Pat. 8538984 (United States). Google Inc. 2013. https://patentimages.storage.googleapis.com/87/00/fe/4a26e2c9ad69c7/US8538984.pdf [Accessed 6th June 2025].

(50) Le Q and Mikolov T. Distributed Representations of Sentences and Documents. In: *Proceedings of the 31st International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, 2014, p. 1188–1196. https://proceedings.mlr.press/v32/le14.html [Accessed 6th June 2025].

(51) Lee J, Dai Z, Ren X, Chen B, Cer D, Cole JR et al. *Gecko: Versatile Text Embeddings Distilled from Large Language Models*. Version 1. 2024. DOI: 10.48550/arXiv.2403.20327. arXiv: 2403.20327 [cs]. Preprint.

(52) Lee C, Roy R, Xu M, Raiman J, Shoeybi M, Catanzaro B et al. *NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models*. Version 3. 2025. DOI: 10.48550/arXiv.2405.17428. arXiv: 2405.17428 [cs]. Preprint.

(53) Muennighoff N, Tazi N, Magne L and Reimers N. MTEB: Massive Text Embedding Benchmark. In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. EACL 2023. Ed. by Vlachos A and Augenstein I. Dubrovnik, Croatia: Association for Computational Linguistics, 2023, p. 2014–2037. DOI: 10.18653/v1/2023.eacl-main.148.

(54) Lee J, Chen F, Dua S, Cer D, Shanbhogue M, Naim I et al. *Gemini Embedding: Generalizable Embeddings from Gemini*. Version 1. 2025. DOI: 10.48550/arXiv.2503.07891. arXiv: 2503.07891 [cs]. Preprint.

(55) Cormack GV, Clarke CLA and Buettcher S. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 758–759. DOI: 10.1145/1571941.1572114.

(56) Franklin B, Silcock E, Arora A, Bryan T and Dell M. News Déjà Vu: Connecting Past and Present with Semantic Search. In: *Proceedings of the Sixth Workshop on Natural Language Processing and Computational Social Science (NLP+CSS 2024)*. Ed. by Card D, Field A, Hovy D and Keith K. Mexico City, Mexico: Association for Computational Linguistics, 2024, p. 99–112. DOI: 10.18653/v1/2024.nlpcss-1.8.

(57) Pomfret T. *A blast from the past, using Felix's AI déjà vu tool!* Felix. 2025. https://felixonline.co.uk/articles/science-issue-1871-feature-a-blast-from-the-past-using-felixs-ai-deja-vu-tool/ [Accessed 4th June 2025].

(58) Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. Version 1. 2019. DOI: 10.48550/arXiv.1907.11692. arXiv: 1907.11692 [cs]. Preprint.

(59) Yang Z, Tang J, Li Z, Wang P, Wan J, Zhong H et al. *CC-OCR: A Comprehensive and Challenging OCR Benchmark for Evaluating Large Multimodal Models in Literacy*. Version 3. 2024. DOI: 10.48550/arXiv.2412.02210. arXiv: 2412.02210 [cs]. Preprint.

(60) Nagaonkar S, Sharma A, Choithani A and Trivedi A. *Benchmarking Vision-Language Models on Optical Character Recognition in Dynamic Video Environments*. Version 1. 2025. DOI: 10.48550/arXiv.2502.06445. arXiv: 2502.06445 [cs]. Preprint.

(61) Kim S, Baudru J, Ryckbosch W, Bersini H and Ginis V. *Early evidence of how LLMs outperform traditional systems on OCR/HTR tasks for historical records*. Version 1. 2025. DOI: 10.48550/arXiv.2501.11623. arXiv: 2501.11623 [cs]. Preprint.

(62) Greif G, Griesshaber N and Greif R. *Multimodal LLMs for OCR, OCR Post-Correction, and Named Entity Recognition in Historical Documents*. Version 1. 2025. DOI: 10.48550/arXiv.2504.00414. arXiv: 2504.00414 [cs]. Preprint.

(63) *IDC Report Reveals AI-Driven Growth in Datacenter Energy Consumption, Predicts Surge in Datacenter Facility Spending Amid Rising Electricity Costs*. IDC: The premier global market intelligence company. 2024. https://my.idc.com/getdoc.jsp?containerId=prUS52611224 [Accessed 12th June 2025].

(64) Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–12. DOI: 10.1145/3079856.3080246.

Glossary

AI (artificial intelligence) A computer algorithm designed to mimic human intelligence.

API (application programming interface) A set of rules and protocols that allow programs to exchange data

ASCII (American Standard Code for Information Interchange) A minimal character set including the Latin alphabet, common punctuation, and the Arabic numerals.

BART (bidirectional and auto-regressive transformers) A natural language generation model introduced by Facebook AI in 2019.

BERT (bidirectional encoder representations from transformers) A natural language generation model introduced by Google AI Language Team in 2018.

CAPTCHA (completely automated public Turing test to tell computers and humans apart) A test designed to distinguish a person from a robot.

CER (character error rate) A metric used to evaluate transcription accuracy by measuring the percentage of characters that need to be modified.

GPT (generative pre-trained transformer) Refers to a specific generative model architecture first introduced by OpenAI in 2018, but now often used more broadly to simply refer to OpenAI's models.

GPU (graphics processing unit) A specialised type of processor originally used to render graphics, but now used to accelerate all sorts of parallel computing tasks, including machine learning.

HTML (hyper-text markup language) The markup language syntax used to create web pages.

JPEG (Joint Photographic Experts Group) An image format commonly used for sharing photographs on the Internet.

JSON (JavaScript object notation) Lightweight data interchange format often used in API responses.

LLM (large language model) An AI model trained on vast amounts of text to understand and generate human language.

MTEB (Massive Text Embedding Benchmark) A commonly-used benchmark for evaluating text embedding models.

NER (named entity recognition) The process of identifying proper nouns in a given text.

OCR (optical character recognition) Technology that converts images of text into machine-encoded text.

PDF (portable document format) File format created by Adobe to digitally represent the printed page.

RoBERTa (robustly optimized BERT pretraining approach) An extension to Google's BERT model proposed by Facebook AI in 2019.

RRF (reciprocal ranked fusion) An algorithm that combines the ranked results from several separate information retrieval systems.

SCC (strongly-connected component) A subset of a network where all nodes are reachable from one another.

TIFF (tagged image file format) An image format used in publishing, graphic design, and archival.

TPU (Tensor Processing Unit) A specialised processing unit specifically designed to perform calculations common in machine learning.

WER (word error rate) A metric used to evaluate transcription accuracy by measuring the percentage of words that need to be modified.

XML (extensible markup language) A markup language for encoding data that is intended to be readable both by humans and machines.