NBER WORKING PAPER SERIES

A PRACTITIONER'S GUIDE TO USING LARGE LANGUAGE MODELS AND GENERATIVE AI IN ECONOMIC HISTORY

Andreas Ferrara

Working Paper 35374
http://www.nber.org/papers/w35374

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
June 2026

I am grateful to Humoyun Abdumavlon, Omer Ali, Sascha O. Becker, Stephan Heblich, Sharun Mukand, Max Steinhardt, Patrick A. Testa, Gabby Toborg, and Sebastian Garcia Torres for helpful discussion and comments. Replication files for all worked examples in this guide are available at: https://doi.org/10.3886/E249897V2 The views expressed herein are those of the author and do not necessarily reflect the views of the National Bureau of Economic Research.

NBER working papers are circulated for discussion and comment purposes. They have not been peer-reviewed or been subject to the review by the NBER Board of Directors that accompanies official NBER publications.

© 2026 by Andreas Ferrara. All rights reserved. Short sections of text, not to exceed two paragraphs, may be quoted without explicit permission provided that full credit, including © notice, is given to the source.

A Practitioner's Guide to Using Large Language Models and Generative AI in Economic History

Andreas Ferrara

NBER Working Paper No. 35374
June 2026
JEL No. C55, C8, N0

ABSTRACT

Large language models (LLMs) are lowering the entry barriers to working with exciting data sources that used to require strong data science skills, such as handwritten ledgers, text, images, or sound recordings. This guide provides an introduction for researchers who are new to LLMs. It sets out a step-by-step workflow for turning a research idea into working code and data, and describes the four main ways of interacting with an LLM: the chat window, editor-integrated assistants, agentic coding tools, and the API. It then works through the decisions a practitioner meets in sequence, beginning with whether an LLM is the right tool and whether the data are allowed to be sent to one, then how to select models, write prompts, manage context limits, and control costs, and finally how to validate, reproduce, document, and correct LLM-generated measures in regression settings. A review of recent research shows how these tools already extract, link, harmonize, and classify historical data at scale. Four worked examples with replication files illustrate the use of LLMs. They classify emotions in paintings, link census records without names, measure newspaper salience and sentiment around the 1882 Chinese Exclusion Act, and score the emotional delivery of Franklin D. Roosevelt's wartime speeches. The guide also condenses the workflow, the best-practice recommendations, and the preparation of replication packages into summary tables and checklists to aid applied economists.

Andreas Ferrara
University of Pittsburgh
Department of Economics
and NBER
a.ferrara@pitt.edu

Link to replication files is available at https://doi.org/10.3886/E249897V2

1 Introduction

Large language models (LLMs) and generative AI (GenAI) are increasingly being used by researchers for various tasks ranging from data extraction to theoretical modeling, coding, and empirical analysis, among others.[1] Economic history is a field that stands to gain from this new technology due to its data-intensive nature. Newspapers (Beach and Hanlon, 2023), yearbook photographs (Voth and Yanagizawa-Drott, 2024), geospatial records (Abramitzky et al., 2025), and even paintings (Gorin et al., 2025) are just some of the creative data sources economic historians have used to shed new light on the past and to answer important economic questions. A key constraint was the cost of being able to work with such data. Aside from the data collection itself, working with text or images used to mean hiring a specialized research assistant, collaborating with a computer scientist, or spending years acquiring knowledge in machine learning, computer vision, natural language processing, and coding. LLMs have the potential to reduce such entry barriers, enabling researchers without these specialized skills or resources to realize their full creativity when answering their research questions.

This potential is already visible in recent work which uses LLMs for a multitude of tasks. One strand turns historical sources into structured data, including county vehicle-registration tables (Bäcker-Peral et al., 2025), German patent records (Griesshaber and Streb, 2025), printed city directories (Greif et al., 2025), handwritten archives (Humphries et al., 2024), and reports containing a combination of tables and narratives of the Freedmen's Bureau (Chyn et al., 2024). A second harmonizes economic measures across time and place, standardizing occupations (Dahl et al., 2026), dating the adoption of technologies (Asirvatham et al., 2026), coding the rules of colonial guilds (Griesshaber and Ogilvie, 2025), and indexing censorship in early-modern England (Grajzl and Murrell, 2025). A third reads meaning out of text at scale, from the political speech of firms (Ottonello et al., 2024) and legislative debate (Giommoni et al., 2026), to the content of life histories (Lagakos et al., 2025), and the values carried in folklore (Michalopoulos, 2025). Other work extends beyond text analysis, classifying the emotion in paintings (Gorin et al., 2025), the vocal tone of central bankers (Gorodnichenko et al., 2023), and the themes in the songs of refugees (Logothetis et al., 2022). Many other works have likely benefited from the coding, proofreading, or logical capabilities of LLMs.

Even though LLMs can greatly assist economic historians, they can also harm their work if these tools are not applied properly. An LLM will attempt almost any task and respond with confidence, even when the response is wrong or incomplete, potentially without the researcher noticing that this is the case. Currently, LLMs are simply not suitable for certain tasks. They can hallucinate references (Topaz et al., 2026), generate incorrect historical information by inserting cultural biases (Celli and Spathoulas, 2025) or artifacts from data which they were trained on (Crane et al., 2025), reason poorly about geography (Van de Weghe et al., 2025), display systematic behavioral biases in economic decision tasks (Bini et al., 2026), and cannot be relied on to produce a valid causal design (Kıcıman et al., 2024). Even without such issues, an LLM-generated variable can still be problematic. A measure built by a model, the sentiment of a newspaper or the emotion in a painting, for instance, is a noisy proxy for what it claims to capture. Such errors are often systematic. Changing the prompt or the model can change the size and even the sign of a regression coefficient (Ludwig et al., 2026; Yin et al., 2026), while the usual

[1] Generative artificial intelligence is the umbrella term that includes models that can generate audio, video, text, images, and code. Large language models are a subset of GenAI models, specifically designed to generate text. For simplicity, this guide will mostly refer to LLMs and specify in the text when the analysis extends beyond textual data or output.

standard errors may not flag the bias (Battaglia et al., 2025). A further risk is more conceptual and can affect researchers directly. Leaning on the model for too much of the thinking erodes the judgment that academic work depends on. If researchers ask the same questions of the same tools and adopt the same LLM-provided answers, then the field will lose the heterogeneity of thought on which it has always thrived. This is why parts of the work belong with researchers, especially when it comes to writing, thinking, planning, and *researching*.

This is not to say that economic historians should not use these tools. To the contrary, many researchers will benefit from this technology but may not have yet explored it much. Doing so from scratch via trial and error can be daunting, time consuming, and frustrating. This guide seeks to provide an introduction to those who are new to using LLMs and GenAI for their research work. It is not an encyclopedia that contains every possible bit of knowledge about LLMs and many of the details will be outdated in a few months due to the fast pace this technology progresses at. But the more conceptual points on how to get started with LLMs, what considerations need to go into an LLM-based workflow, and what pitfalls to avoid should be more durable.

This guide begins with sketching a workflow for new LLM practitioners to show how this technology can potentially be applied in a gainful but also critical way. Specifically, Section 2 sets out a step-by-step workflow for new practitioners and the four ways of working with an LLM via i) a browser chat window, ii) the coding assistant inside an editor, iii) agentic models that can work with files on the researcher's computer directly, and iv) application programming interface (API) calls for processing larger amounts of tasks or data. It then provides a worked example which demonstrates how the proposed workflow can enable a researcher to analyze emotions in paintings. This is the setting of a recent study by Gorin et al. (2025) who extract emotions from paintings from 1400 onward to test whether they reflect major events such as wars, technological progress, or food insecurity, among others. Their work relies on an elegant two-step convolutional neural network (CNN), an approach that requires machine-learning expertise that many applied economists do not readily have. The applied example here shows how the same task can be achieved via the proposed LLM workflow even without such specialized coding knowledge. Conceptualizing, collecting the data, as well as validation and testing nonetheless remain crucial and time-intensive steps that cannot be skipped.

Section 3 provides the practitioner's guide for researchers who have to confront the detailed questions and pitfalls they will encounter when implementing a full LLM-based workflow in their projects. The three self-contained subsections discuss issues pertaining to the setup, implementation, and validation. Setup considerations include whether an LLM fits the task, and data sensitivity, which determines if the data are even allowed to be sent to an LLM. Implementation requires choosing a model, writing and refining the prompt, i.e., the instructions the LLM receives, working around the memory limits that degrade model performance, and managing the costs of running a large job. The validation discussion is centered around checking LLM output against hand-coded data, reproducibility and what to disclose in the paper and replication package, and how measurement error in an LLM-generated variable can impact statistical analysis. None of this has to be read at once. A reader new to the material can work through the three parts in order, while someone who wants only one of them can go straight to that subsection. The section ends with a table that collects the main recommendations as well as an appendix table that provides a checklist for producing replication packages involving LLM-based work.

Section 4 provides a review of what economic historians have already done with LLMs in recent

work to showcase their utility. The most immediate use is data extraction. An LLM can read sources that resist ordinary optical character recognition (OCR), like scanned tables and handwritten ledgers, straight into structured data. Bäcker-Peral et al. (2025) do this for county vehicle-registration tables. Their pipeline cuts critical parsing errors from around 40 percent to under 1 percent, and the downstream results match the human-coded benchmark. The same models also harmonize historical measures, standardizing them across decades and languages. Asirvatham et al. (2026) use LLMs to build a series that did not exist before. Classifying some 37,000 technologies, they show that the gap between a technology's invention and its spread shrank about tenfold over the industrial age. Other work quantifies what historical actors said and meant, from legislative debate (Giommoni et al., 2026) to the life stories of people (Lagakos et al., 2025). There are other types of data a transcript or a table cannot provide. The tone of a voice or the mood of an image carries economic information too, and LLMs are starting to put such sources within reach of researchers who lack specialized data science skills to extract such data.

The guide also provides several worked examples with replication files to showcase the usefulness of LLMs. Two short demonstrations cover data extraction: i) a frontier model transcribes a 1930 newspaper article that defeats conventional OCR, and ii) turns a handwritten 1950 census page directly into a data table. The first of the larger examples revisits census linking but with an important caveat. Due to licensing restrictions, the models were given all census variables except for the names. The models were asked to perform a name-blind matching using data for 10,000 Connecticut men who were previously linked from 1900 to 1910 using the IPUMS historical identification keys (HIK) by Ruggles et al. (2025). Despite this major disadvantage, the LLMs managed to achieve a match rate ranging between 43 and 62 percent based on all other variables in the census, which compares to a lower bound of 4.3 percent achieved via a simple joinby merge using similar linking variables, as well as a 93 percent agreement rate achieved by the Census Tree Project links (Price et al., 2021). In this exercise, the agentic versions of ChatGPT-5.5 and Claude Opus 4.8, i.e., the models with direct access to the author's files, did best. A second example measures how American newspapers wrote about Chinese immigrants around the 1882 Chinese Exclusion Act (Long et al., 2024). Sending all 524,458 pages to a frontier model would have cost too much. A preprocessing step narrowed the corpus to the 11,011 pages that mentioned Chinese people at all, which were then sent to a batch API. Salience and hostility both rose in the weeks before the act, then fell once it passed. The last example scores the emotion in Franklin D. Roosevelt's wartime speeches. The Day of Infamy address stands out as the most intense, an emotional signal a written transcript leaves out. Each example comes with its prompts, data, and code. A reader can take whichever example is closest to their own problem and adapt it to a new source.

Beyond those examples, the guide contributes in two further ways. The first is conceptual. Section 2 gives new users a concrete workflow for turning an idea into working code and data, and Section 3 develops the reasoning behind each step, from the first setup choices to the final checks on the output. The second is to the literature. Broad surveys already map what generative AI and deep learning offer economists (Korinek, 2023; Dell, 2025), and a growing body of econometric work studies how to use and correct variables that a model has generated (Ludwig et al., 2026; Battaglia et al., 2025), including methods for harvesting historical data with these tools (Moulton and Severen, 2025). This guide complements the data-source guides economic historians already rely on, such as the newspaper guide of Beach and Hanlon (2023), and carries that tradition over to LLMs by organizing the material around the decisions a practitioner has to make when working with these new tools.

2 A Workflow for New LLM Practitioners

Economic history offers a vast array of exciting data sources to study, from newspaper text to yearbook photographs or even voice recordings of Federal Reserve chairs (Gorodnichenko et al., 2023). As discussed in the introduction, working with such non-standard data used to be subject to substantial entry barriers in the form of specialized personnel or years of methods training.[2] Many historical questions have been left untouched not because they were uninteresting but because the methods were out of reach. The recent rise of LLMs and GenAI has essentially leveled these entry barriers.

This section describes a generic workflow for how researchers can use LLMs to unlock such non-standard data sources or econometric methods not currently in the researchers' toolkit. For someone without prior knowledge of LLMs, a first starting point should be to learn about the different LLMs out there, their capabilities, pros, cons, and costs.[3] Aside from gaining a basic understanding of the options, researchers should check access, e.g., if their university provides a subscription to a model vendor, and how or whether they will be charged for it. Once a given LLM is chosen, achieving the desired result is a little more complex than simply asking the model to perform the task at hand. In fact, the LLM itself may not necessarily be used for the analysis of the data. While the LLM can potentially analyze the data too, there are many other tasks that it can help with, such as conceptualization, brainstorming, or coding, among others. Either way, instead of submitting a single question or task, an iterative approach with sufficient details at each step is likely to yield noticeably better results. Table 1 provides a stylized step-by-step workflow for a scenario where a researcher wants to be able to analyze a non-standard data source. This broadly involves communicating the task, evaluating the LLM's proposed options, understanding the chosen option, implementation, refinement, and validation before proceeding with the at-scale application.

The same style of workflow can be applied in several different ways. For instance, a researcher could ask the LLM to set up a simple theoretical model to justify an empirical result. They could upload an Econometrica paper to the LLM and ask it to code up the estimator or test statistic in R or Python, provide simulation code, or ask for an explanation of the mechanics or assumptions in more plain terms.[4] Appendix Table A.1 provides a few more examples of use cases for economic history research and applied work in general. Treating the LLM not just as a cheap and hard-working research assistant, but as a colleague to ask for advice and to challenge oneself will unlock more of this new technology's potential. This also highlights an important shift in the skill set of applied economists, which has potential implications for how we train graduate students going forward. Before the AI era, knowing a programming language by heart was a skill. As this skill is eroded by LLMs and GenAI, more emphasis is going to be placed on the novelty of the research question (though this has always been the case), the creative use of new data sources, and on *critically* using LLMs to implement the data collection, data management, and empirical work, while asking the LLM the right questions, double-checking its answers, and ensuring

[2]Non-standard in the sense that they are not in a typical tabular format that can be easily read by statistical software and that they require additional work beyond digitization and transcription.

[3]For instance, some are better at coding, some are cheaper for certain tasks, and so on. It would be desirable for the purpose of this guide to provide an overview table with this type of information. However, the rapidly changing technology in this sphere would outdate any such table before the paper even goes into print. Readers should consult current model documentation or independent benchmarks for up-to-date capabilities and pricing structures.

[4]Many methods exist theoretically but were never made accessible to the broader research community due to a lack of time or coding skills. An entrepreneurial researcher somewhere will eventually realize that they can make a name for themselves by using LLMs to provide easy-to-use implementations of these methods as a public good.

that the research results are correct.[5]

Table 1: A Workflow Sketch for an Iterative LLM Interaction

Step	Action
0	Research available LLMs and learn about their pros, cons, pricing structure, and capabilities. Usually done once and updated as new models become available.
1	Once a suitable model is chosen (such as Claude, ChatGPT, Gemini, or others), open a session and describe what you would like to do. This should accurately convey the task at hand but also the data, ideally with some examples or data snippets. Ask for multiple options that would be suitable to tackle this task.
2	After the LLM provided the options, ask: "What are the relative pros and cons of each? What should I look out for? Which would be best suited to my scenario?" The model should be able to articulate why a given method is more or less suitable for your task.
3	Based on this answer, pick the preferred method and ask: "Please explain this to me in detail. How does it work? What are the assumptions and potential issues in my application?" The goal is to understand the method well enough to evaluate whether it is appropriate to the task, but also to later communicate it to seminar audiences and in the paper.
4	Request implementation: "Please write Python code that implements this method on the attached sample". When working with larger data sets, doing this at a smaller scale with a representative pilot sample first may be preferred as testing and troubleshooting will be faster.
5	Do not accept the first version uncritically. Ask the model to double-check whether the code works as intended, whether it is optimized for computational time and memory usage, and whether performance could be improved through parallelization or related techniques. The LLM will provide code that is likely going to run but it is likely not going to be the most efficient (or correct) version. Challenge it on this.
6	Run the code, inspect a sample of the output, and feed it back to the model: "Please check the attached output. Does this look correct? Is there anything I should be concerned about?"
7	Validate against a ground-truth test data set if available: "How would I validate that this worked correctly? Here is an annotated sample produced by my research assistants, who hand-coded this test data set. How can I use it to test my output?"
8	If validation succeeds, ask for refinements or extensions, or proceed to implementation at scale. If not, return to Step 6 and continue iterating on the code. A worthwhile idea to pursue is to return to Step 2 and make the model challenge the overall idea as well as the process: "Based on what we have developed, does what I want to do make sense? Are there any issues with my approach, hypothesis, data, or methods that I should revise?" Receiving a critical outside opinion early in the process is valuable but the LLM will not necessarily provide this input unless prompted.

That latter point is fundamentally important because a real danger of the workflow sketched out in Table 1, and the use of LLMs more broadly, is the temptation to outsource too much of the researcher's own thinking process, to rely too much on the LLM, and to not question the output. At a personal level, and for lack of a more benign term, intellectual laziness can impact creativity and potentially lead to mistakes. Even though LLMs perform at a high level when it comes to coding and other tasks, they do make mistakes. Without proper checks and validation, this can become a problem, such as the recent rise of fake citations in papers which will reduce trust in academic work in general (Topaz et al., 2026). For

[5]The arrival of early digital computers offers a historical parallel. Research output and direction shifted within years, precisely in the fields where computation had been the binding constraint (Aldighieri and Malpassi, 2026).

the profession, the danger is that we may lose diversity of thought. If everyone relies on LLMs, gets and follows the same advice, then this will homogenize what we collectively produce. While Section 3.1.1 deals with tasks that LLMs cannot reliably perform, one task that should stay with the researcher even where the model is capable deserves to be highlighted at this stage: writing. Writing forces researchers to think, evaluate, and frame what they are doing. Making a conscious effort to retain a creative and critical mind, as well as knowing how but also when and when not to use the LLM, is a potential path to avoid such issues. There is much reason to be optimistic about this technology and its potential to generate better research, open up new data sources, improve coding and replicability, among other benefits, as long as it is used responsibly.

2.1 Technical setup

Before diving into an application of the sketched workflow, a brief note on the technical setup is in order. Most readers will be familiar with the interactive chat interface provided by LLMs such as ChatGPT, Claude, or Gemini. There are other options and modes for using LLMs, each with a different purpose and varying degrees of ability, which are briefly described below. In order of appearance, these include the classical chat window, integrated development environments (IDEs), agentic models, and API models.[6] These four options are best read as a set of potential choices rather than a ranking. Empirical projects benefit from using several of them at different stages or for different tasks. For instance, a researcher might think through a method or question in a browser chat window, write the analysis code with help from an editor-integrated assistant, build the data pipeline with an agentic coding tool that has access to the project folder on their local machine, and then run that pipeline across an archive using the API. Picking the right tool is a question of matching the mode of interaction to the task. One point that should be noted is that this overview reflects the state of these tools at the time of writing. In the future, there will be new ways to interact with these LLMs, names of models and companies may change, capabilities and pricing will shift, and so on.

The chat window: The browser-based chat interface is the option most readers likely already know. ChatGPT, Claude, and Gemini all offer a web page where the user types a message and the model responds. This is where most economists first encountered LLMs, and for many research-related tasks this is a good starting point. The workflow described above runs comfortably in a chat window for any project that fits within a single conversation, and the next section will showcase this in a worked example. The model can read pasted or uploaded code, suggest fixes, explain methods, draft prose, and discuss the design of an analysis.

The chat interface has grown beyond the single-conversation model that early users will remember. Both Claude and ChatGPT, for instance, now support project-specific folders into which the researcher can upload files. The model has access only to files in that folder and can work with them in its own sandboxed execution environment online. A single project folder can accommodate multiple chats, where each can be devoted to a different task but with shared access to the uploaded materials and conversations. Adding an `AGENTS.md` file into the folder is a useful coordination device. This is a simple text file that supplies the project's main context, any rules that the LLM is supposed to follow, or current results to every chat that the researcher opens inside the project, such that work in a new conversation does not start from scratch. Since LLMs have limited memory, it is best practice to have

[6]The alphabetic glossary in Appendix Table A.2 further defines and explains these and other LLM-related terms.

one chat per task. The binding constraint of the chat interface, even in its project-folder form, is the number and size of file uploads. These limits are restrictive enough that a serious empirical project, with archival scans, transcribed corpora, or large data files, will quickly exceed them. The model also cannot reach files on the researcher's own machine, so the workflow remains one of uploading rather than connecting. For initial exploratory purposes, uploading a smaller sample file is typically a more viable starting point.

IDE-integrated assistants: Many researchers are familiar with code editors, such as VS Code, RStudio, PyCharm, or Cursor. IDE-integrated assistants are now available within some of these and add the LLM's capabilities to the editors in two main ways. The first is inline completion, where the model proposes the next line or block of code as the user types, and the user accepts or rejects it with a keystroke. GitHub Copilot was one of the first widely used tools of this type. The second is an editor-side chat panel, where the user can ask the model questions about the file currently open, with that file automatically being included in the conversation. Cursor and similar agentic IDEs go further and let the model edit across multiple files inside the editor. This mode is useful once a researcher is actively writing analysis code. The IDE-integrated LLM is continuously aware of what the user is working on, and allowing direct access to the user's files on their own computer resolves both the copy-paste friction and the file restrictions of the web browser-based models. For conceptual exploration or for tasks that go substantially beyond coding, the chat window can be a more convenient option. In this sense, the two modes are complements with exploration and ideation happening in the browser, and implementation being done via the IDE editor.

Agentic coding assistants: An agentic coding assistant is a program that lets an LLM read and write files on the researcher's own computer, execute code, and decide on its own which steps to take to carry out a task. The distinction from the previous two options is that the model is not just suggesting code or responding to questions about an open file. It is acting on the project with access to the same file system the researcher works in with greater autonomy.[7] For project-scale empirical work, this is the option that turns the LLM from a smart conversation partner into a research assistant who can actively work with the researcher's files and data. Tools in this category include Claude Code, OpenAI Codex, and Gemini CLI. Agentic models started out as command-line programs that ran inside the terminal, which was not the most user-friendly setup. Companies have recognized this as a potential barrier to adoption, hence the major products now also ship as downloadable desktop applications that look much closer to a familiar chat interface than to a command line. Installation is simple as it only requires downloading the corresponding installer for the user's operating system and signing in. Some require additional installations, such as Git, but the overall setup costs are relatively small.

Working with these agentic tools via the GUI is fairly intuitive. The user opens the desktop app and points the LLM at a project folder on their computer. A chat panel is displayed on one side of the window while side panes can show the project's file tree, an editor for spot edits, an embedded terminal for the user's own commands, or a preview of any output the agent generates. The user types a request in natural language, such as "Read the file `painting_metadata.csv`, show me the first ten rows, and produce a table with summary statistics." The agent reads the file, executes any code it needs to inspect

[7] While there are frightening stories out there of agentic models having wiped entire databases, the more common mode is to restrict model access to only the project folder or a subdirectory. It is nonetheless recommended to store copies of original data files outside of this directory.

the data, and responds in the chat. When the agent wants to change a file, it shows the proposed change as a message that the user can accept or reject. Permission settings let the cautious user review every action before it runs, or let an experienced user grant the agent more autonomy. The latter opens more complex tasks, such as "Look at all newspaper `.txt` files in folder X, check their encoding, and then summarize all text encoding schemes you have encountered. Then store this list in a new `.csv` file."

Agentic models come closest to delegating work to a research assistant who then takes the relevant steps toward the end product with little or no further input. Such delegation can extend to entire data products.[8] Agents are also not restricted to a single session. Recent versions support running several sessions in parallel on the same project, each in its own working copy of the code.[9] A researcher can have one session cleaning data while another debugs a figure or runs regressions. More elaborate setups, where one agent's output is checked or challenged by another, require deliberate configuration and are best understood as multi-agent pipelines.

API models: The fourth option is qualitatively different from the first three. The chat window, the IDE assistant, and the agentic desktop app are mostly tools for developing the data collection or analysis. The API is the tool for executing that task at scale. Instead of typing into a chat window, the user writes Python or R code that sends a prompt to the API model, receives the response, and generates the desired output file. Other options to specify are the behavior and version of the model to be used, which allow for replicable results to some degree. For instance, the researcher may tell the model what role it is supposed to take, "You are a research assistant extracting structured data from 19th-century newspaper articles", the degree of creativity in the output (the so-called "temperature", see Appendix Table A.2), and the model version, which can be fixed, such as `gpt-4o-2024-05-13`.[10]

Concretely, the API is what a researcher uses when the same prompt needs to run across thousands or millions of documents or files. Classifying every article in a 200-year newspaper corpus, extracting structured fields from every page of a digitized ledger series, or generating embeddings for an archival collection are all tasks where the chat window is unusable and the desktop app would be impractical. The workflow looks different from the interactive modes. The user writes a script that loops over the corpus (in the case of text analysis or extraction of data from text), calls the API for each document, parses the response, and writes the results to a data file. The user then initiates the API job, walks away, and inspects the output when it finishes. Since such at-scale tasks can become expensive fast, cost discipline becomes an important concern. A main advantage of API-based usage is that the costs for input and output tokens for large-scale batch submissions tend to be much lower. Batch submissions write a single `.json` or `.jsonl` file, wherein each row is a request that specifies the model, parameters, request, and file to be analyzed. Example code for batch submissions to a hosted API are provided for the worked example in Section 4.4. Model selection, cost management, data sensitivity, and validation are discussed in more detail in Section 3.

[8] For example, Afonso et al. (2026) use AI agents that search public sources on a loop to assemble a documented cross-country data set at the cost of a standard LLM subscription.

[9] Avoiding clashes among agents is achieved via dedicated file structures, for instance, using Git work trees, which let separate sessions edit the project without interfering and merge their changes later.

[10] Notice that while this version fix generates somewhat replicable results, models can still be permanently retired at a later point, meaning that researchers lose the ability to replicate the results from that particular model.

2.2 An example of the workflow sketch in practice

To illustrate the workflow sketch in Table 1 as well as the chat-window and API-based uses of LLMs more concretely, consider the following example. Suppose a researcher wishes to extract emotions from a large collection of paintings they have downloaded from ArtEmis (Achlioptas et al., 2021). This is the setting of a recent paper by Gorin et al. (2025) who study the relationship between emotions extracted from 630,000 paintings from the year 1400 onward and world events related to wars, technological progress, and agricultural shocks, among others. Further assume that the researcher for this example has no knowledge of computer vision or classification of image data, but they have already done their basic LLM research regarding access and cost described above, meaning that step zero of the workflow has already been completed. Appendix A reports an abbreviated version of the chat with Claude Opus 4.7 in extended thinking mode, i.e., the model's reasoning setting, which allocates more compute to working through complex prompts.[11] This is a reduced version of the workflow which omits the validation steps 7 and 8. This omission costs little because the validation of new data products and measures is a well-established skill for economic historians. The purpose is to provide a minimally worked example that highlights how someone without specialized machine learning knowledge of computer vision can make such data accessible to themselves via the use of LLMs. What should also be noted is that a project like that of Gorin et al. (2025) requires a substantial amount of time to acquire, clean, and then link the data on paintings, painters, and historical events, in addition to the time spent on the conceptualization, training, and verification of the model and output. Their paper provides a nice example of how this is done in a comprehensive way. Just because the LLM can enable researchers to conduct the analysis as demonstrated below, this does not imply that research becomes a trivial task.

The first prompt asked the model to analyze the emotional content of four attached sample paintings. These were four images taken from the eight total images of Figure 3 in Gorin et al. (2025). The remaining four were held out for later classification. The prompt also specified the emotions over which the paintings should be ultimately scored (contentment, amusement, excitement, awe, fear, anger, sadness, and disgust), as well as a request to suggest statistical methods for how these emotions can be extracted. The model immediately realized that the eight requested emotions correspond to the classifications provided by the ArtEmis annotators. It then provided four options: a zero-shot vision-language model (VLM),[12] a fine-tuned image classifier using ArtEmis data as training input, a CLIP (Contrastive Language-Image Pretraining) zero-shot classifier, as well as a hybrid approach. The second option of using a fine-tuned image classifier trained on the ArtEmis data is essentially the approach taken by Gorin et al. (2025). Claude flagged this as too heavy to use for only eight images and pointed toward the lighter zero-shot route instead.

The second prompt requested further explanation of each option, their relative pros and cons, and suitability for the task at hand. Based on this discussion, the zero-shot VLM option was selected. For the VLM option, Claude correctly highlighted that no training data were needed (unlike for the original paper, which used a training sample of almost 80,000 hand-annotated paintings), but also balanced this discussion with the cons regarding reproducibility and cost. Prompt three then asked for an intuitive ex-

[11] The full conversation, input files, as well as the resulting Python code and output files can be downloaded from https://doi.org/10.3886/E249897V2. Anthropic released Opus 4.8 partway through this project; earlier examples were not re-run on the newer model.

[12] Zero-shot refers to LLM classification without providing specific examples. See the alphabetic glossary of frequently used LLM terms in Appendix Table A.2.

planation of how a VLM works at a conceptual level. Prompt four requested the Python implementation, which relied on a ChatGPT-4o vision model via the OpenAI API, and an open-weight model, namely OpenCLIP's Vision Transformer ViT-B/32 trained on LAION-2B.[13] This example highlights the complementarity between different LLM uses described in the previous section: the chat window aided with conception and coding-related challenges, while data for the analysis were submitted to a vision LLM (GPT-4o) via a looped API call.

Prompt five for the code audit and efficiency gains was skipped since only eight images were analyzed and since keeping the appendix conversation concise was a priority. After the Python code had executed, prompt six requested a review of the initial trial for which three of the eight images were classified. Claude correctly argued that three images did not provide a particularly useful sample size for a test run, but it did not find any other red flags with the initial output. The results from the classification of all eight images are reported in Figure 1 where panel a displays the original images taken from Gorin et al. (2025), and panel b reports the classifications for each image from the two models. The ChatGPT-4o model classified seven out of eight emotions correctly as judged by comparing its classifications with those provided by the ArtEmis annotators. It misclassified image h as sadness when the ArtEmis-assigned label was disgust. The CLIP model missed image c (excitement, misclassified as amusement) and image f (anger, misclassified as disgust).

Figure 1: Classifications of Emotions in Paintings by Method

Panel a: Images from Gorin et al. (2025)

(a) Contentment (b) Amusement (c) Excitement

(d) Awe (e) Fear (f) Anger

(g) Sadness (h) Disgust

Panel b: GPT and CLIP classifications

Fig.	GPT	CLIP	Match
a	contentment	contentment	both
b	amusement	amusement	both
c	excitement	amusement	GPT only
d	awe	awe	both
e	fear	fear	both
f	anger	disgust	GPT only
g	sadness	sadness/disgust	both
h	sadness	disgust	CLIP only

Notes: Panel a displays the images from Figure 3 in Gorin et al. (2025), which had the highest emotion scores for each emotion as assigned by the ArtEmis annotators. Panel b reports the emotion classifications from the ChatGPT-4o vision model and OpenCLIP's Vision Transformer ViT-B/32 for each of the images in panel a, as well as whether either or both models correctly classified the respective emotions. For image g, the CLIP model returned a tie between sadness and disgust; since sadness is the ArtEmis label, the tie is counted as a correct classification. The data files, prompts, and code used for this exercise can be found at https://doi.org/10.3886/E249897V2.

The point is not to evaluate these models based on the classification of eight images, which are also the easy cases because they were examples of paintings with the highest score for each emotion

[13] LAION-2B (Large-scale Artificial Intelligence Open Network) is a publicly available data set of 2.3 billion image-text pairs, which was released in 2022.

as assessed by the annotators. The main purpose of this exercise was to showcase how the workflow in Table 1 could be applied to enable someone with little or no knowledge of machine learning or computer vision to conduct a study like that of Gorin et al. (2025). Their paper built a custom two-step convolutional neural network (CNN) based on an EfficientNetV2-S encoder pretrained on ImageNet, which was then fine-tuned on almost 80,000 hand-annotated paintings,[14] with a regression head that mapped the 1,280 encoded features into a 9-dimensional probability vector of the emotions to be classified, i.e., the eight emotions listed above plus a ninth "something else" category used by the ArtEmis annotators. The readers to whom the previous sentence meant little or nothing are a key target audience for this guide. For a skilled machine learner, the implementation of this two-step CNN model from conception to deployment, including the data work, would have taken several weeks, if not months. The Python code to generate the classifications in panel b of Figure 1 via Claude using the above workflow took a few minutes. This does not mean that a CNN was not a good choice. In fact, using a CNN provides a reproducible model that can be fine-tuned and adapted by the user, and it can be deployed at scale. The latter point is important because the original study needed to classify around 630,000 paintings. The approach taken here would have struggled with this because the ChatGPT-4o VLM processed the test run and the final eight image classifications, eleven in total, for $0.04. At this per-image cost, the 630,000 paintings would have come to roughly $2,290. Submitting the images via OpenAI batch submission halves this cost. The CLIP model would have been a viable zero-cost, reproducible alternative, but potentially with a higher error rate.

3 A Practitioner's Guide for Working with LLMs

The previous section has given new practitioners a road map for how to get started with LLMs, followed by a worked example to illustrate this. This section now details additional considerations that need to go into the planning, execution, and verification process, depending on the data and question at hand. The topics are discussed in the order a practitioner is likely to encounter them and are organized into three broader subsections. The first deals with setup decisions, such as whether an LLM is the right tool for the task at all, data sensitivity issues, and how to navigate them. The second subsection covers the operational choices and potential issues related to the actual implementation, including model selection, prompting, context windows and performance degradation, and cost control. Once the LLM has generated the desired output, the third subsection deals with validation, issues pertaining to reproducibility and what to disclose in the paper and replication package, and what to look out for when using LLM-generated measures in a regression setting, including potential biases and currently available corrections. The section concludes with an overview table of important topics and recommendations for new LLM practitioners that can be used as a checklist.

3.1 Setup considerations

3.1.1 Decide if an LLM is the right tool

Before tackling a problem with an LLM, it is worth noting that these models are not reliable for certain tasks. This is problematic because the LLM itself will try to perform these tasks and report output, but

[14] A brief technical note is that the CLIP approach is similar in the sense that it relies on image embeddings as the two-step CNN did, with the main difference being the encoder used.

the output in many cases will be flawed. The list below provides examples of such tasks.

1. *Literature search*: Current LLMs are not good at summarizing the literature in a field or work related to a particular research topic. Many references tend to be hallucinated. A recent study by Topaz et al. (2026) audited 2.5 million biomedical papers and found that fabricated references appear in published work at surprisingly high rates, and that the rate has accelerated from mid-2024 through early 2026. More recent models tend to hallucinate less, but they still do not manage to produce literature reviews that accurately summarize the state of a research topic. Better options are to use Google Scholar, EconLit, JSTOR, references in recent papers on a topic, and direct conversation with colleagues who know the literature.

2. *Looking up historical facts or data*: An LLM may not actually know the 1934 unemployment rate in Ohio, or how many enslaved people lived in Virginia in 1860.[15] The LLM will produce a confident answer. This answer has a non-trivial probability of being wrong, and such errors are harder to catch by the researcher without validation data. Crane et al. (2025) evaluate frontier LLMs on historical macroeconomic data series and document two types of errors. These include mixing up data from different sources, in which the model conflates first-print data with later revisions; or within sources, in which the model conflates data for past and future reference periods. Both errors are invisible to a researcher who does not already know the correct answer.

3. *Spatial reasoning and geocoding*: Frontier models currently do not reason well about space. Asking an LLM to convert a list of historical addresses into coordinates, or to identify which county a township belonged to in a given year will produce output that looks structured but contains errors that are hard to detect (Van de Weghe et al., 2025). The standard geocoding pipeline, in which addresses are matched against a reference data set such as GeoNames, the U.S. Geological Survey's Geographic Names Information System, or a historical equivalent, gives reproducible and verifiable results. This is something current LLMs do not deliver. For now, a preferable workflow is to have the LLM write Python or R code that calls established geocoding libraries against accurate reference data, and to keep the actual spatial reasoning in those libraries rather than in the model. One place LLMs can help at this stage is in harmonizing or cleaning place name strings before geocoding, where the task is closer to text normalization than to spatial reasoning.

4. *Generating causal designs*: An LLM can be a useful conversation partner when a researcher has already done the work of formulating candidate research designs. This includes the trade-offs, required assumptions, the data, and the institutional and historical context. Providing these pieces of information to the LLM directly and asking it to challenge the designs can highlight potential issues the researcher had not anticipated. What the LLM should not be asked to do is to generate the design. Asking the LLM for a good instrument, for instance, is potentially problematic because identifying a workable instrument requires the kind of engagement with the specific setting and data that the model is not doing, even when given background information (Kıcıman et al., 2024).

The list above is not exhaustive and some of these cautionary examples may change in the future as LLMs evolve. If uncertain as to whether a model can reasonably perform a task, a researcher should do

[15] Models with web search or other retrieval tools can look up historical statistics more reliably than models relying on parametric memory. The remaining concern is source quality. The canonical sources for pre-1960 economic statistics are not always the top result for a given query, and the model has no reliable way to judge whether a returned source is authoritative.

an online search to learn more about LLMs and their current performance for the task at hand. Alternatively, they may collect a small validation data set against which the model output can be compared, or bluntly ask the model whether it can reasonably perform the task. This last option should be used carefully because models are not always reliably aware of their own shortcomings.

3.1.2 Data sensitivity

Another key setup question is whether a researcher is *allowed* to send their data to an LLM. For most economic historians, this is usually not a question about human subjects, as IRB oversight is rare given the historical nature of the data.[16] The oftentimes more relevant constraints are data use agreements, the contracts a researcher signs with the data provider that govern what can be done with the data, to whom they can be disclosed, and where they may be stored. These agreements were drafted before LLMs existed and are often not explicit about what counts as "disclosure," which puts the burden of interpretation on the researcher.

An example is the full-count U.S. census with names, available through IPUMS under a special agreement that prohibits the export or disclosure of names and other string variables outside the licensed environment. A researcher who pastes a batch of records into a chat window, or who writes a Python script that loops over the data and sends each record to a hosted API, is likely violating this prohibition. The data leave the researcher's machine, transit the vendor's infrastructure, and are processed on the vendor's servers.[17] Whether the vendor retains the data afterward is a separate question. The transmission itself is a disclosure to a third party. A cautious interpretation is that if the agreement forbids exporting strings or other data, then hosted LLMs or API submissions are not an option. Other historical sources have similar restrictions. Major newspaper databases such as ProQuest, newspapers.com, or Readex are licensed under terms of service that limit redistribution and often prohibit automated bulk extraction. There are at least two potential solutions to the issue of data sensitivity when using LLMs, which are described below.

Enterprise licenses: One option is to see whether a researcher's university has already negotiated an enterprise contract with a vendor.[18] These contracts commit the vendor not to store data, prompts, or outputs beyond the duration of the request, and not to use any of these for future model training. Researchers should consult with their institution's IT and research offices to ensure that adequate licenses are in place to safeguard sensitive data. One important detail is that zero data retention is a commitment about what the vendor does after receipt, while a typical data use agreement is about whether the data may be disclosed at all. Whether the agreement and the enterprise contract together satisfy both questions is a case-specific judgment that the research office is better positioned to make than the researcher. The institutional contract may also constrain which models are available, so the researcher may not be choosing between Claude, ChatGPT, and Gemini freely so much as choosing among whichever vendors their institution has licensed with.

Self-hosting: When the data use agreement prohibits the use of hosted LLMs or API calls entirely, another alternative is to run an open-weight model on hardware the researcher controls, i.e., their own

[16] Exceptions are projects involving living participants, such as oral histories with surviving respondents, or recent administrative microdata with personally identifiable information about living individuals.

[17] The vendor refers to Anthropic for Claude users or OpenAI for ChatGPT users, for example.

[18] Examples include OpenAI Business and Enterprise contracts, Google Cloud Vertex AI, Microsoft Azure OpenAI Service, and Anthropic's Claude for Enterprise.

computer or protected university server. This essentially means to download an LLM directly after which it can be used without online access. In this case, the data never leave the environment covered by the agreement. In practice, the model weights are downloaded once,[19] loaded into memory on a local graphics processing unit (GPU), and called from local code. Several families of capable open-weight models are available, such as Llama, Qwen, Mistral, and DeepSeek, among others. Open-weight models are usually not as advanced as proprietary frontier models such as Claude or ChatGPT. For tasks that economic historians care about, including data extraction from text and basic image classification, the gap can be small enough to be acceptable. For complex multimodal tasks like transcribing handwritten ledgers,[20] for instance, the gap is larger and may matter for the project's feasibility. Self-hosting also carries a reproducibility advantage, since open-weight models can be archived and rerun in perpetuity, a point to be returned to in Section 3.3.2.

3.2 Implementation choices and potential pitfalls

3.2.1 Model selection

Once a researcher knows which models are available given their data and institutional constraints, the next question is which model to actually call. The workflow in Table 1 can help narrow the choice. For instance, the example in Section 2.2 pointed toward a vision model for image classification. Vision capability is needed for any task involving handwritten documents, photographs, maps, scanned tables, or other visual content the researcher does not want to OCR first. The next choice is which tier of model should be deployed, meaning the degree of model complexity. More complex models tend to be slower and more costly than less complex models. This is particularly true for reasoning models, which are suited for more difficult tasks. Using such a model on a simple digitization task would be overkill. Models can also be chained into an LLM pipeline. For instance, consider a text classification task whereby simpler, less-costly models filter out the obviously irrelevant cases, followed by more complex models that can work on the resulting smaller, pre-screened data set but with more compute and reasoning power.

Model complexity: The first-order distinction is between frontier models and the cheaper variants offered by the same vendor.[21] Frontier models are the state-of-the-art proprietary models. They tend to have the highest cost per token and the best accuracy on difficult tasks, and they handle complex multimodal inputs well. Mid-tier models are smaller or older versions from the same vendor, often marketed explicitly for high-volume work. They are typically an order of magnitude cheaper. For straightforward extraction and classification tasks, they are often good enough that the accuracy gap to frontier models is negligible. Empirical projects in economic history do not always need a frontier model for every document or data set. Defaulting to the most expensive option without considering cheaper and more

[19] Training LLMs from scratch is extremely expensive and usually costs several million dollars. Models available online, e.g., via the model-sharing platform Hugging Face, provide the pretrained weights from existing models and tweak these for specific purposes. The "weights" are coefficients (in machine learning lingo) from a very large neural network called a *transformer* with billions of coefficients. This is what the acronym GPT (Generative Pre-trained Transformer) refers to.

[20] A multimodal model is an AI system that can process, understand, and generate multiple types of data, e.g., text, images, audio, and video.

[21] For example, the current frontier model by OpenAI is ChatGPT-5.5 Pro. Mid-tier variants include ChatGPT-5.4 mini and ChatGPT-5.4 nano. Older versions, such as ChatGPT-4o or ChatGPT-3.5 turbo, are also still available, but may be retired at some point. Notice that this name list is only accurate at the time of writing.

appropriate models is an easy way to waste research money. Performance also differs across languages, scripts, and historical orthography, with English-language material typically having the best coverage. For sources in other languages or with archaic spelling, validation samples described in Section 3.3.1 should be drawn from that same material rather than from English-language equivalents.

A recent additional option is to choose between standard models and reasoning models. Reasoning models allocate additional compute to work through a problem step-by-step before responding. They are slower per query and more expensive as the thinking process often consumes billable tokens. The cost premium is justified when the task requires composing several inferences (an extraction that depends on a prior judgment about context, or a classification that hinges on a chain of conditions), or when a standard model is producing confidently wrong answers on cases the researcher can verify. For high-volume single-step tasks of the form "does this document mention X, yes or no," reasoning models are typically too powerful. The paintings example in Section 2.2 used Claude Opus 4.7 in extended thinking mode, which is a reasoning variant. That choice was deliberate given the conceptual step-by-step application of the workflow described in Table 1. A quick test is to submit a small hand-coded validation sample to both a more complex and a simpler model, and base the decision on their respective error rates.

Chained models: Based on the complexity of the task and the amount of data, another choice is whether to use one model or several. A useful pattern for high-volume tasks is the cheap-filter/expensive-classifier setup, which combines a small model and a frontier model in sequence.[22] Suppose the goal is to identify historical newspaper articles that report lynchings in a given corpus. The naive alternative is to simply search all pages for the word "lynching" and inspect the matches. This is problematic for two reasons. Historical newspaper OCR has high error rates, so a meaningful share of relevant articles contains the search term in a misspelled form that no plain text search will catch. The term itself is also not a reliable proxy for the topic. It was rarely used to describe extra-legal mob executions prior to the Civil War and instead became more widespread in the 1880s. A page with the search term may also contain opinion pieces or articles about court cases, among others. A two-stage LLM pipeline handles such problems. A small model reads each article and flags anything plausibly on topic, with a prompt tuned for high recall rather than precision.[23] A frontier model then reads the flagged subset under a more specific prompt, potentially with examples, and produces the final labels.

Fine-tuned models: For repeated, well-defined classification or extraction tasks with labeled training data, a fine-tuned small model can dominate a frontier LLM on accuracy, speed, and cost. Fine-tuning means taking a pretrained model and continuing to train it on a labeled data set specific to the researcher's task, which adjusts the model's behavior toward that distribution of inputs. Having said that, depending on the complexity of the task, fine-tuning a model can require several thousand hand-coded examples. If fine-tuning is feasible, using an open-weight model is often the only choice because its weights are freely downloadable and can be retrained locally. Open-weight models were first described in Section 3.1.2. Some commercial vendors offer fine-tuning, though the specific capabilities change frequently.[24] Two

[22] More complex chains are possible. For instance, a cheap lower-tier model may be used to pre-screen data, followed by a mid-tier, and a high-tier model. Other options are to increase the prompt complexity, add examples, or a mix of these strategies.

[23] In machine learning lingo, recall refers to the share of actual positive cases that the model correctly identifies. It is also referred to as the true positive rate (TPR) or sensitivity. Precision measures how many of the model's positive predictions were correct.

[24] At the time of writing, OpenAI supports fine-tuning on select ChatGPT-4o and o4 mini models, with retirement announced

implementation details are worth mentioning, both of which make fine-tuning tractable on consumer hardware. First, fine-tuning does not require retraining all of the model's billions of weights; a method called Low-Rank Adaptation (LoRA) can instead add small trainable layers to the original weights, which modify model behavior in the final classification step. Second, quantization reduces the numerical precision of the stored weights by rounding them, which lowers memory requirements and enables LLMs to run locally on consumer-grade graphics cards. These two tools make fine-tuning tractable on consumer hardware, and the steps and code required to implement them can be worked out via an LLM conversation following the workflow in Table 1.

An example use case in economic history is the standardization of occupational codes. Dahl et al. (2026) introduce OccCANINE for this purpose. Their fine-tuned open-weight LLM maps free-text occupational descriptions to HISCO codes at 96 percent accuracy. The authors did not fine-tune a frontier LLM but a much smaller pre-existing model called CANINE. This open-weight base LLM is small enough to fine-tune on a workstation and to be run on a laptop, and it tokenizes inputs at the character level rather than considering whole words. This is what makes the encoder robust to OCR errors, which garble the word "farmer" into "frmter" or "farmar", for instance. Fine-tuning then teaches the model the actual distribution of historical occupational strings via 15.8 million description-code pairs from 29 sources in 13 languages. The training data were assembled from existing research projects in which historians had already hand-coded their own sources. This is the kind of labeled corpus that makes fine-tuning possible in the first place. The architecture itself is a standard encoder-decoder where CANINE encodes the input string into a numerical representation, and a decoder head translates that representation into one or more HISCO codes (a description can map to several codes, such as the example "lives of fishing and farm work" which yields codes for both farmer and fisherman). On the resulting model, classifying 10,000 occupational descriptions takes a few minutes on a laptop GPU, at no marginal API cost, with replicable outputs given the same inputs. A frontier LLM called per record would likely be slower, less accurate, and substantially more expensive.

3.2.2 Prompt engineering

A prompt is a specification of a task or question in the same way one would ask a colleague or direct a research assistant. Instead of coding or knowing a programming language by heart, prompt engineering is a new skill introduced into the skill set of researchers. Asking the right questions in the right order, and refining them to improve the output, is the core of working effectively with an LLM. The discussion below focuses on the opening prompt, whether submitted through an API call or as the starting point of a chat or agentic session. The same principles, components, and pitfalls apply to follow-up prompts in a multi-prompt conversation, as sketched in Table 1.

A well-engineered prompt typically contains the following components:

- *Task definition:* This might seem obvious but actually requires a surprising amount of thoughtfulness. A model, like a research assistant, will perform only as well as the researcher defines the task. The prompt has to specify the request, the expected output format, how to handle edge cases, and any relevant background the model needs to understand the data or setting. The trap to avoid

for early 2027. Anthropic does not currently offer direct fine-tuning. Google offers fine-tuning of Gemini models through Vertex AI on the Enterprise Agent Platform.

is the implicit task definition. For instance, a prompt like "classify this article as relevant or not relevant to lynching" works only if "relevant" has been defined. A model without clear instructions and detail will apply its own arbitrary choices and may or may not communicate these to the researcher. The output specification is equally important, including the desired output in terms of the data to be collected (e.g., a simple yes or no classification, a score on a defined scale, or a continuous measure) as well as the desired file formats. Before submitting a prompt to an LLM, a useful test is whether the researcher would be willing to hand the instructions to a new research assistant without further explanation. If not, it is not a finished prompt.

- *Examples:* Including a small number of worked examples in the prompt, sometimes called few-shot prompting, is the most effective tool for resolving the ambiguity that the task definition cannot fully account for. A few examples drawn from the researcher's hand-coded validation sample, covering both easy and ambiguous cases, are typically sufficient. A broader array of easy and more ambiguous cases will give the LLM an overview of what to look out for. More examples are not always better though. Examples that look too similar to each other can narrow what the model treats as a positive case instead of broadening it. The case where examples can even hurt is when they are inconsistent with the written task definition. A good practice is to write the task definition first, then add examples to clarify cases the definition leaves ambiguous, and to revise the definition rather than the examples when a problem surfaces in the validation step.

- *Classification rubrics:* For classification or rating tasks, a rubric specifies the categories or scale points and gives a one-sentence definition of each. It is important that every category has a definition. These need to be non-overlapping, and the rubric needs to include an "uncertain" or "none of the above" option. The option to opt out matters because forcing a choice with no escape valve produces confident-looking labels on cases that should have been flagged for review. The researcher then loses the ability to distinguish cases the model handled well from cases it handled poorly. If the researcher cannot write a rubric a research assistant could apply, the task is not yet well-defined enough to give to an LLM either.

- *Period vocabulary:* Prompts written in modern terms will systematically miss historical usage. Section 3.2.1 gave an example where searching for "lynching" in newspaper data misses articles before the term was widely used. The fix is to write the period vocabulary explicitly into the task definition, ideally with examples drawn from the actual data. Sometimes it can be instructive to ask the LLM first about period-specific vocabulary, but the answer should be checked by the researcher and it may not be a complete list.

- *Justifications:* Asking a model to provide a brief one-sentence answer for why it classified a given observation into a category serves two purposes. First, justifications give the researcher something to double-check, which catches cases where the model produced a defensible label for a random or non-sensible reason. Second, with reasoning or thinking models, requesting a justification can improve classification accuracy because it forces the model to articulate its reasoning before providing an answer. Notice that this is different from asking the model to quantify its own uncertainty about responses or labels via an additional variable, which typically does not work.[25]

[25]For instance, asking for an uncertainty classification from one to five is not going to work. LLMs cannot accurately self-

Figure 2 provides an example of an annotated first prompt to start a model on a data collection task using the prompt engineering points discussed above, with the data context and the output specification, which are both part of the task definition, highlighted separately. The only component not shown is the justifications request. In practice, this is a single sentence added to the output specification.

Figure 2: Annotated Example of Prompt Components

Extract information on lynchings from a large database of 2 million historical newspaper pages between 1860 and 1900.

The texts are in .json format and may contain OCR transcription errors. The data are stored in "C:/User/Project/Data". Save the output to "C:/User/Project/Data/Output" as a .csv file called "lynchings.csv".

The output file should have the following columns:

- date: date of the lynching
- location: location of the lynching
- newspaper: name of the newspaper
- page: page number
- file: .json file name

Definition of a lynching: an extra-legal mob killing of a person.

Do not classify as a lynching: opinion pieces, court reports about lynching cases, legislative debates, or other texts that do not directly describe a lynching event.

Relevant terms in the corpus may include: lynching, mob justice, mob killing, summary execution, mob violence, and related period-appropriate phrases. The term "lynching" itself was rarely used for extra-legal executions before the 1880s, so attend to the underlying event rather than the specific terminology.

The attached file1.json and file2.json are examples of texts that describe a lynching as defined above. The attached file3.json and file4.json are examples of texts that do not.

	task definition		data context		output definition
	rubric		period vocabulary		examples

Notes: Prompt for extracting lynching events from a historical newspaper corpus. Colored highlights identify the components discussed in Section 3.2.2. The example assumes the prompt is run through an interactive coding-agent session rather than a single API call.

3.2.3 Context windows and performance degradation

One issue that many practitioners learn about over time is that LLMs have a finite memory and that model responses become significantly worse if a chat has grown too long. This is because models can only retain a certain amount of information, which is constrained by the *context window*. It includes the system prompt, the conversation history, any attached documents, written code, and the model's own previous responses. All of these count toward a token limit. Current frontier models advertise context windows in the hundreds of thousands to millions of tokens. This may sound like a lot, but a model writing thousands of lines of R code can exhaust the limit quickly and performance tends to degrade well before the stated limit (Paulsen, 2026). Some chat interfaces also apply a compaction step when a conversation approaches the limit, summarizing earlier turns to free up space. The problem is that

assess the uncertainty in their responses. Chen et al. (2026) document this formally: self-declared confidence is biased, whereas uncertainty measures built from the model's token-level probabilities, available via the API, do predict accuracy.

this greatly limits the model's memory and the quality of its subsequent responses as it loses access to the original text of earlier messages and works based on the summary instead. Hitting the compaction step is the typical cue to start a new chat. The main reason for this recommendation is that model performance degrades as the context fills, even before the nominal limit is reached (see Liu et al., 2024; Du et al., 2025). This shows up in several ways. Earlier instructions get summarized in ways that are not transparent to the user, so a rubric established twenty messages ago may no longer be reliably applied. Data outputs stored in spreadsheets can become less consistent, with format slips appearing in places they did not earlier in the session. The model loses track of the structure of long inputs, and a 100-page document attached to a prompt is going to quickly fill up the context window.

When planning, thinking through, coding, analyzing, and refining a research workflow, a single chat session is often not going to be enough. A useful strategy is to dedicate a new chat for each task in this entire process instead of trying to tackle everything at once. Especially in chat window and agentic environments, this leads to much improved results. A potential worry with spreading the work across many chats, organized in a project folder with meaningful titles, is that these chats may live in isolation. For dedicated project folders, this is not the case as chats can access each other's information. A project log, sometimes named `AGENTS.md` (briefly mentioned in Section 2.1), addresses this concern. This plain-text document is loaded at the beginning of each chat and can provide a compact overview of the main previous decisions made by the researcher or what happened in other chats to coordinate them. It can also specify desired behavior, such as avoiding em dashes and filler phrases, or limiting response lengths.

3.2.4 Cost control

One issue to be mindful of is that the costs of using LLMs can grow quickly for certain tasks. This is particularly true for agentic and API use of high-level models.[26] LLMs charge per token, where input and output tokens are billed separately. More complex and advanced models cost more per token, which makes the model-selection question of Section 3.2.1 also a cost question. For small jobs, like the example analysis of emotions in paintings showcased in Section 2.2, this is a minor constraint. At the scale of the 630,000 paintings analyzed by Gorin et al. (2025), the example demonstrated how costs can rise quickly. A first step is to look up the input and output token costs for the chosen model and to project the total cost using a smaller test sample.

Aside from choosing a smaller model, the other option is to be strategic about how to submit the data. This is highly task-specific. To make things more concrete, consider a researcher who wishes to study anti-Chinese sentiment before and after the Chinese Exclusion Act using historical newspaper data. Full-text historical newspaper data typically come as OCR-processed text files, such as `.json` or `.txt` formats, from sources like newspapers.com or Chronicling America (Beach and Hanlon, 2023). These text files typically contain the OCR of an entire newspaper page rather than individual articles.[27] The first step would therefore be to understand which pages mentioned Chinese people and then the textual sentiment towards this group. A researcher who wants to bluntly analyze all 20 million pages from Chronicling America using ChatGPT-5.5 via the OpenAI API, for example, can expect to pay north

[26] In chat environments, costs surface as usage limits instead. The model will simply tell the user that the limit has been reached and when it will reset, which marks the end of the current session for a few hours.

[27] An exception is the American Stories data set of Dell et al. (2023), which provides article-level texts.

of $600,000.[28] Obviously, this is an approach that few researchers, if any, would take, but it is instructive to show the worst-case cost dimension. For the example at hand, a more practical approach would be the following:

1. Restrict the newspaper corpus to a certain time window before and after the Chinese Exclusion Act, e.g., the 12 months before and after May 6th, 1882.

2. Pre-screen pages with a simple word search. Pages that do not mention Chinese people at all will likely be irrelevant. Notice that the period vocabulary mentioned in Section 3.2.2 is important here. This will miss some pages where such vocabulary is misspelled, and one should check with a random sample of pages how much of a problem this is. Alternatively, a more computationally intensive fuzzy word search can be applied.

3. Among the remaining pages, only keep, say, the 50 words before and after a given Chinese-related term. Another common pre-processing step is to remove stop words from text.[29]

This approach is going to greatly reduce the number of tokens that need to be analyzed. It will also yield better results given that the time period is more focused and the model does not get distracted by other content. Using a text window around each Chinese-related term cuts out advertisements and all other articles on the page. To then determine whether a given text snippet is related to the Chinese Exclusion Act, one can use the model chaining approach described in Section 3.2.1 where a cheap model flags relevant texts and a more complex model determines the sentiment classifications. Notice that the specific parameter choices in this example, such as the 12-month window or the 50-word text radius, are not a fixed rule but an illustration of how costs could be reduced. The appropriate values depend on the setting and on what the analysis is supposed to achieve; the worked example in Section 4.4, for instance, uses a three-month window and 30-word snippets. As always, testing such an analysis pipeline before deploying it at scale is advisable.

Lastly, several smaller tweaks also matter. Prompt caching, offered by most vendors, bills repeated long prompts at a discount when the same system prompt or set of few-shot examples is reused across many calls. For a workflow that sends a 2,000-token prompt to 200,000 documents, this can produce substantial savings. Batch APIs, which return results within a 24-hour window rather than immediately, typically offer cost reductions of about 50 percent and often finish sooner depending on server demand.

3.3 Validation and reproducibility

3.3.1 Validation data

Many applied economists have heard of a training data set. In the formal ML sense, a hand-classified data set is split into i) a training portion the model gets trained on, and ii) a test portion in which the trained model's predictions are compared to the true labels to estimate its performance. LLMs are already pretrained models, hence the researcher does not control this process.[30] A *validation data set*

[28] This assumes an average of 6,000 tokens per page for a simple yes/no output, not counting the prompt. At the time of writing, the specified model charges $5 per 1 million input tokens and $30 per 1 million output tokens. For 20 million pages, this results in approximately 120 billion input tokens, which at $5 per million tokens yields a total of $600,000, before counting output tokens or the prompt itself.

[29] These are such commonly used words (such as *this*, *the*, *an*, etc.) that they do not provide any predictive power.

[30] The exception is fine-tuned models which were described in Section 3.2.1.

in LLM-assisted research is also a small, hand-coded data set. But instead of training a model, the validation data set is used to check the LLM's responses or classifications. Unlike an ML model, the LLM does not receive these labels and they exist only so the researcher can compare the machine's answer to a known correct one.

The validation sample has three separate purposes. First, it can be used to evaluate a model's performance. If a researcher has a validation data set that was independently hand-coded by two trained research assistants and the model gives the same or at least similar answers, then this is a useful performance benchmark. Notice though that it might matter who the research assistants are. An all-male team trying to hand-code issues regarding gender discrimination in historical texts may introduce a cultural bias into this validation sample (Celli and Spathoulas, 2025). Second, the validation data can be used for choosing among a set of candidate prompts. After each candidate prompt has classified the validation data, the one with the lowest error rate becomes the preferred choice for the entire data set. Third, models can be compared in a similar fashion as prompts. The choice among the model tiers from Section 3.2.1, e.g., between a cheaper model and a frontier one, or between a standard model and a reasoning one, can be based on performance in the validation sample. There is one very important caveat to this approach. The first task collides with the second and third. If the validation data set is used to select the appropriate prompt and model, then using the same data set for estimating the LLM's error rate becomes circular and inflates the performance metrics. This is akin to fitting a machine learning model on the training data and evaluating its fit on the same data set. To avoid this, one should set aside a part of the validation data that is not used for prompt and model selection, and use that subsample to generate the accuracy for the final model.

Another consideration is how large this validation data set should be. There is no formula for this because the answer depends on the type of variable or object to be classified or predicted by the LLM, the researcher's budget, and the necessary level of confidence one wishes to achieve in the validation tests mentioned above. Dahl et al. (2026) recommend validating at least 100 hand-coded observations in their occupational-coding setting and reporting the achieved accuracy. Variables with rare categories need larger validation data sets, because 100 draws may contain only a handful of the positive cases that carry the information. Construction of the validation data also matters. A random draw from the data gives an unbiased picture of average accuracy but spends most of its observations on easy cases the model was never going to miss. Stratified sampling, which oversamples the harder-to-classify categories, buys more information per hand-coded item, at the cost of an accuracy figure that has to be reweighted back to the overall data to be representative. A random sample is a good baseline but is worth combining with a second stratified sample aimed at evaluating the harder cases.

A related consideration is the metric used to judge performance. Overall accuracy is a misleading yardstick when the category of interest is rare. In a corpus where one page in a thousand reports a lynching, a classifier that labels every page as irrelevant is 99.9 percent accurate and yet finds nothing. For rare categories, precision and recall, introduced in Section 3.2.1, are the more informative measures and should be reported alongside accuracy. The appropriate balance between the two also differs across the stages of a chained pipeline: the cheap screening model should be tuned for high recall so that few true cases are lost, while the final classification stage should deliver high precision so that the resulting data set is not diluted with false positives.

One final caveat is that public benchmarks may be part of a model's training data. Labels and linked

samples that are freely available online, such as published crosswalks or the linked census samples used in Section 4.2, may have been ingested during pretraining, in which case the measured accuracy can overstate how the model would perform on truly new material. A validation sample hand-coded by the researcher and not published before the study is immune to this concern. The problem is that researchers typically cannot know what data a model has previously seen.

3.3.2 Reproducibility and version drift

LLM-assisted empirical work has two reproducibility problems that conventional empirical work does not. First, the same prompt sent to the same hosted frontier model on the same day can produce slightly different outputs across calls. Second, the model the researcher called today may not exist in the future. Even if it still exists, the vendor may have re-trained it on new data, which affects the output even though the researcher has not changed the prompt, the data, or anything else on their end.

The first problem stems from how hosted LLM inference is implemented. Temperature is the parameter that controls randomness in the model's token sampling.[31] At temperature zero, the model deterministically picks the highest-probability token at every step of formulating a response or classification. In principle, this should produce identical outputs across calls, but in practice a hosted model does not. The reason is not that the model contains randomness at temperature zero but that floating-point arithmetic is not associative: the order in which a long series of numbers is summed can change the result in its final decimal places, and a small enough change can flip which token the model picks next. On a server, a request may be grouped into a batch with other users' requests, and the model's low-level operations are not invariant to the size and composition of that batch. The same prompt can therefore have its numbers summed in a slightly different order depending on what else the server was processing at the time, and once one token changes the rest of the response can diverge from there. That is why, for a hosted model, temperature set to zero does not behave like the random seed researchers set in R, Python, or Stata, where fixing the seed makes results exactly reproducible. Some APIs expose a seed parameter for LLM inference, but despite this the researcher does not control how requests are batched on the server, hence outputs can still vary across calls.

The second problem comes from how vendors maintain and continue to train their models. The model behind a given API call can change without notice, either through model updates or changes in model versions. The same API call to ChatGPT-5.5 or Claude Opus 4.8 over a year may route to different underlying models. Older versions also get deprecated whenever a vendor decides to retire them. A paper that depends on a specific model version may not be reproducible a year later, regardless of how carefully the prompt and all steps were documented. Many vendors offer dated or versioned model identifiers that are intended to remain stable rather than being silently updated or re-trained. For instance, specifying `gpt-4o-2024-05-13` as the model in an API call to OpenAI will be replicable up to the minor serving-level variation described above, or until ChatGPT-4o is permanently retired.

Several active choices by the researcher can help. None fully solves the problem but at least reduces the issue to some extent:

1. Pin the specific model identifier (e.g., `gpt-4o-2024-05-13`, `claude-opus-4-7`) and the date of access in the methodology and replication materials. The model name alone is not suffi-

[31] Notice that the temperature setting is mostly an API-level option. Chat interfaces typically do not expose it, and some newer API and agentic models no longer allow it to be changed either.

cient as the underlying model weights can change without a corresponding renaming of the model itself.

2. Archive the actual model outputs together with the prompts and code. Since rerunning the prompt against a hosted model is not guaranteed to reproduce the original outputs, the outputs themselves can at least be used to make downstream analysis replicable.

3. Where the API provides the option to set the temperature parameter, set it to 0. This reduces variation even though it does not eliminate it.

4. If long-run reproducibility is the goal, self-hosted open-weight models offer by far the most control. They are a full solution to the disappearing-model problem, since the weights are a static file the researcher can store and re-use indefinitely. They also largely resolve the run-to-run variation: because the researcher controls the whole setup rather than a shared remote server, the same prompt can be made to return the same output on the same machine.[32] This is reliable on the researcher's own machine but is not a perfect guarantee, and a replicator who re-runs the model on different hardware should not assume identical results. The trade-off, discussed in Section 3.1.2, is that open-weight models often lag the strongest frontier models in capability.

For hosted frontier models, these are partial solutions that get outputs close to reproducible without making them fully so, and self-hosting is the most complete alternative rather than a perfect one. For a replication package that relies on a hosted model, this residual variation is an imperfection that data editors will have to tolerate.

3.3.3 Replication packages and disclosure

The reproducibility limits described above raise a practical question: what should go into the paper and its replication package when a study relies on LLM-generated data or measures? A useful principle is that a reader should be able to see exactly what the model was asked, what it answered, and how well it performed, even if rerunning the model is not possible. For the paper and its appendix, this means reporting the exact model identifier and the date of access, the temperature and other relevant settings, the size of the validation sample, and the accuracy achieved on the held-out portion described in Section 3.3.1. When licensed or otherwise sensitive material was processed, the paper should also state which route made this permissible, for instance an enterprise contract with zero data retention (see Section 3.1.2). The figure and table notes accompanying the worked examples in this guide provide templates for such statements.

The replication package itself should contain the full prompts, including system prompts, rubrics, and few-shot examples, all code for the pre-processing, API calls, and analysis, the archived raw model outputs, and the hand-coded validation data together with a description of who coded them and how. Archiving the raw outputs is the single most important step. Rerunning the prompts against a hosted model is not guaranteed to reproduce them, while archived outputs make every downstream result mechanically replicable. Lastly, journals are increasingly adopting disclosure policies for generative AI,

[32] Achieving this in practice means running the model with greedy decoding (always taking the highest-probability token rather than sampling), keeping the hardware and software versions fixed, and turning on whatever deterministic options the inference software offers. The reason it is still not a perfect guarantee is that the low-level numerical calculations a GPU performs can differ slightly across different hardware, drivers, or software versions.

and data editors are formulating expectations for AI-assisted empirical work. The current policies of the target journal should be checked before submission, but a package assembled along the lines above should satisfy most of them by construction. Appendix Table A.3 condenses this subsection into a checklist that can be consulted when assembling the package.

3.3.4 How LLM-generated measures can bias regression results

A common application of LLMs in economic history is to generate a measure based on text, images, or audio for later use as a variable in a regression analysis. This includes measures of sentiment toward a group in a newspaper corpus, the emotion in a painting as in Section 2.2, or how conservatively a politician speaks, among others. Something to keep in mind is that the resulting variable is not perfect. When economic historians construct new variables from archival data, they pay close attention to where the data came from, what the data potentially measure poorly or omit, the inherent biases, coverage, and so on. The same reasoning needs to be applied to what comes out of LLMs but also to what goes in. The LLM itself will generate the measure but it will not by default caution users about the potential issues unless prompted to do so.

Consider the previous example of a researcher who wants to build an LLM-generated county-level measure of anti-Chinese sentiment (sentiment_c) from nineteenth-century newspapers and asks whether that sentiment predicts later violence against Chinese residents (viol_c). Ideally, they would regress

$$\text{viol}_c = \alpha + \beta \text{sentiment}_c^* + X_c'\gamma + \epsilon_c \tag{1}$$

where sentiment_c^* is the true sentiment, but in practice they only have the LLM-generated proxy. An important question is how the LLM was instructed to measure sentiment, e.g., as a binary variable, a discrete score, or a more continuous measure, and how this choice relates to the unobserved true sentiment. The prompting can change results in important ways. Ludwig et al. (2026) provide an example where measures created from different prompts lead to drastically different regression results. Similarly, Yin et al. (2026) show that LLM-generated occupational exposure scores can vary substantially across models, with downstream estimates changing in magnitude, sign, or significance.

The reason is that the LLM-generated measure is a noisy proxy for the true underlying measure, such that $\text{sentiment}_c = \text{sentiment}_c^* + \eta_c$, where η_c captures measurement error introduced by the LLM and the broader measurement pipeline. This includes cases where the model misclassified a text, compressed a complex concept into a user-requested binary or ordinal variable, inherited bias from newspaper coverage,[33] or accumulated errors when text-level labels were aggregated into a county-level index, leading to so-called index error (Battaglia et al., 2025). If the error is classical, it will lead to the typical attenuation bias of $\widehat{\beta}$.[34] Here, the error is likely not random. A model that received a prompt without period-specific vocabulary and definitions (discussed in Section 3.2.2) will miss terms like "coolie labor" or "the Chinese question." If such systematic errors correlate with the controls or the structural error term ϵ_c, then measurement error is no longer classical and will bias the coefficient on the LLM-generated variable in unpredictable ways. Another insight from Battaglia et al. (2025) is that this problem may not be visible

[33] Beach and Hanlon (2023) discuss important data quality checks in the context of newspaper data.

[34] Notice that measurement error in binary and bounded variables, such as scores or percentages, is never classical by construction, and can lead to attenuation bias in OLS and inflation bias in IV regressions (Bingley and Martinello, 2017). See Ferrara et al. (2024) for measurement error in the newspaper context.

from the usual standard errors. They may still capture sampling uncertainty around the estimand for the LLM-generated variable, while the confidence interval is centered on a biased coefficient.

3.3.5 Potential tests and corrections

Practitioners should keep issues relating to sample selection and measurement error in mind when using LLM-generated measures, but this is not to say that such measures should not be used. For the LLM-specific error sources, a few potential corrections and tests already exist. Given the rising popularity of LLMs in empirical work, econometricians will likely pay close attention to this and will develop additional ways to test for and overcome such issues. Readers are encouraged to stay up to date with the literature.

As of now, two kinds of fixes exist. One tests whether the measure is trustworthy before it is used, and the other corrects the regression for whatever error potentially remains. Asirvatham et al. (2026) focus on the first. They compare GPT-generated measures against more than a thousand tasks for which human labels already exist and find that the model's scores usually track the human ones. Another contribution they make is the provision of a test of whether the model is reading the text or guessing from context. To implement this test, a researcher has to take a sample of documents and strip out the words that carry the attribute being measured. Then they rerun the model on the stripped text. To give a concrete example, if a rating of how pro-environment a political speech is barely falls after every environmental sentence is removed, the model was not reading the speech's content. It was inferring the rating from something correlated with it, such as the speaker's party, age, or other characteristics. These test procedures come as an open-source Python library, GABRIEL, that runs the same checks on a researcher's own data in a few lines of code. Ludwig et al. (2026) and Yin et al. (2026) show that changing the prompt, model, or output measure can change downstream estimates, which suggests another useful diagnostic. Re-generate the LLM-based measure while varying the prompt, model, and output measure (e.g., binary, discrete, and continuous versions of the LLM-generated variable), and show robustness of the regression to such changes. Invariance in the coefficient of interest here lends credibility to the LLM-generated measure. This applies not only to variables generated from text, but also to those based on images, audio, or video.

The second kind of fix corrects the regression once the measure is built. Ludwig et al. (2026) use the validation sample discussed in Section 3.3.1 for this purpose. Their recommendation is to hand-code a small random subset and compare the model's labels to the hand codes on it. The gap between the two estimates the bias, which is then subtracted from the full-sample result. Because the model's labels still carry information, the corrected estimate can come out more precise than one built from the hand-coded subset alone. Battaglia et al. (2025) supply the formulas and standard errors for this correction, and a version that estimates the measure and the regression together, in their `ValidMLInference` package. A last point worth reiterating is that the validation sample may not itself be unbiased, depending on who has collected it (Celli and Spathoulas, 2025).

3.4 Summary of recommendations for practitioners

To conclude the section, Table 2 provides a summary of recommendations and key points to consider as new practitioners develop LLM-based workflows for their economic history research projects. Appendix Figure A.1 provides a visual summary of the section as a flowchart.

Table 2: Summary of Key Topics and Recommendations for LLM Practitioners in Economic History

Stage / topic	Recommendation
Panel A. Setup considerations	
Is an LLM the right tool?	• Avoid LLMs for literature searches or summaries. Use Google Scholar, EconLit, JSTOR, recent papers, and conversations with colleagues instead. • Avoid LLMs for information on historical facts or settings (if you do, double-check answers). • Avoid LLMs for spatial reasoning or geocoding. Have it write Python/R that calls established libraries (GeoNames, USGS GNIS), and keep the spatial logic in those libraries. • Avoid asking an LLM to generate a causal design or propose an instrument. A safer use is to stress-test designs you have already formulated. • When it is uncertain whether a task is feasible for an LLM, test against a small hand-coded sample rather than trusting the model's self-assessment.
Data sensitivity	• Read the data use agreement, IRB rules, and consult the research office if in doubt. Treat any transmission of data to a hosted API as potential disclosure to a third party. • Do not paste or loop restricted records (e.g. IPUMS full-count names, ProQuest, newspapers.com, Readex) through hosted APIs. • Check whether your institution holds an enterprise contract with zero data retention; have the research office confirm the agreement and contract jointly permit the use. • Where hosted APIs are foreclosed, self-host an open-weight model (Llama, Qwen, Mistral, DeepSeek) so the data never leave the controlled environment.
Panel B. Implementation choices	
Model selection	• Match model capabilities to the task (e.g., use a vision model for handwriting, photographs, maps, or scanned tables). • Match model complexity to the task (e.g., use a mid-tier model for a simple digitization task instead of an expensive frontier model). • Reserve reasoning models for multi-step or more complex tasks. • Choose between tiers and models on error rates measured on a hand-coded sample. • For high-volume jobs, chain a cheap first-round model with a frontier classifier run only on the screened subset. • For repeated, well-defined tasks with thousands of labeled examples, consider fine-tuning a small open-weight model (cf. OccCANINE) instead of calling a frontier LLM per record.
Prompt engineering	• Define the task explicitly including the job, output format, edge-case handling, and background. Smell test: would you hand it to a new RA with no further explanation? • Add a few worked examples spanning easy and ambiguous cases; keep them consistent with the task definition and not near-duplicates of one another. • For classification, supply a rubric with non-overlapping, individually defined categories and always include an "uncertain / none of the above" option. • Write period-appropriate vocabulary into the prompt; asking the model to list historical terms first is a cheap pre-step (double-check historical accuracy). • It makes sense to ask a model for justifications of its classifications to spot-check labels. Do not ask the model to self-quantify its uncertainty.
Context windows	• Dedicate a separate chat to each task rather than tackling everything in one session. • Start a new chat once the interface compacts or summarizes; performance degrades well before the nominal token limit. • Coordinate chats through a project folder and a plain-text project log (e.g. `AGENTS.md`).

(Table 2 continued)

Stage / topic	Recommendation
Cost control	• Project total cost on a small test sample, using the model's input and output token prices, before running at scale. • Cut tokens. E.g., for textual data, restrict the corpus by period, pre-screen with keyword search, keep only a text window around the target term, and remove stop words. Pre-analysis brainstorming with the LLM about how to reduce data dimensionality can help. • Use prompt caching for repeated prompts and batch APIs (about 50 percent cheaper).
Panel C. Validation and reproducibility	
Validation data	• Build a hand-coded validation sample and use it to evaluate accuracy and to select prompts and models. • Hold out a separate subsample for the final accuracy estimate. Do not use the same validation data to refine prompts/model versions and to evaluate accuracy. • Oversample rare or hard categories via stratified sampling and reweight to the full data. • Be mindful of who hand-codes, since coder composition can introduce cultural bias into the labels.
Reproducibility	• Fix the exact model identifier and the date of access. • Archive the actual model outputs alongside the prompts and code, since rerunning is not guaranteed to reproduce them. • Set temperature to 0 where the API allows it (this reduces but does not eliminate variation, and is not a random seed). • For long-run reproducibility, use self-hosted open-weight models, whose weights are static files you can preserve.
Replication and disclosure	• Report the exact model version, access date, settings, validation sample size, and held-out accuracy in the paper; state how sensitive data were handled (e.g., enterprise version with zero retention). • Ship full prompts, code, archived raw model outputs, and the hand-coded validation data in the replication package. • Check the target journal's generative-AI disclosure policy before submission.
Measurement bias and fixes	• Treat the LLM-generated variable as a proxy carrying likely non-classical measurement error, and remember the usual standard errors may not reveal the resulting bias. • Verify the model reads content rather than guessing from context (e.g. the word-removal test in the GABRIEL library). • Show robustness by re-generating the measure across prompts, models, and output types (binary, discrete, continuous). • Correct the regression with a hand-coded random subset to estimate and net out the bias (e.g. the `ValidMLInference` package).

Notes: Condensed recommendations from Section 3, ordered as a practitioner is likely to encounter them: setup decisions, implementation choices, and validation and reproducibility. The list is not exhaustive and some items may date as the technology evolves.

4 Applications and Potential for LLMs in Economic History Research

This section discusses the potential uses for LLMs and how the recent literature has already applied this new technology for different tasks relevant to economic historians. This includes the large-scale extraction of data from historical sources that are not yet or not easily machine-readable, the merging of data sets without unique identifiers, or the harmonization of economic measures across time and space. This is followed by more analysis-based discussions of text analysis, classification, and sentiment analysis,

as well as more non-standard data sources including images, audio, or video. This is to showcase the broad applicability of LLMs to research in economic history, and hopefully some of these examples will inspire research based on such novel data sets that are now more readily accessible to researchers.

4.1 Using LLMs to generate structured data from historical sources

The most immediate use of LLMs in economic history is to convert historical sources into structured data. The most interesting data sources tend to be newly unearthed archival data and data that have not already been machine-read. This includes scanned tables, handwritten ledgers, printed directories, patent registers, administrative reports, maps,[35] forms, and other unstructured texts. Researchers typically had to transcribe such material by hand, hire research assistants, outsource digitization to data-entry companies, or build custom OCR and machine-learning pipelines. LLMs lower the fixed cost of this step because they can read printed and handwritten text as well as images, interpret layouts, and return structured output from documents that are problematic for conventional OCR methods. An example is given in Figure 3, where a newspaper article (panel a) was digitized via a traditional Python tool, namely Tesseract OCR (panel b), while the same image was digitized by ChatGPT-5.5 in panel c. The LLM not only avoided the standard OCR errors, but also retained the textual structure, which can be of interest for analysis or measurement purposes. At scale, frontier models like the one used in panel c are often too powerful and costly. Simpler models tend to do as well with clean scans but at lower cost. Other tools such as Amazon Textract have also adopted LLM integrations to improve OCR processing for larger digitization tasks. One general word of caution is that the LLM will provide supposedly clean output text, even when the source is in such poor condition that it is barely legible. Unless otherwise prompted about the image or OCR quality, the LLM may generate text that is actually not there but that fits into its next-token prediction scheme. It is advisable to validate the output against the original images or scans for a random subset to see how problematic such behavior is.

Several recent papers show how quickly this frontier is moving. Bäcker-Peral et al. (2025) use a multimodal LLM pipeline to digitize historical county-level vehicle-registration tables and benchmark the resulting panel against human-validated data. The LLM pipeline is substantially cheaper than outsourcing, cuts critical parsing errors from around 40 percent to well below 1 percent, and yields downstream empirical results that are statistically indistinguishable from those obtained with the human-coded data. Griesshaber and Streb (2025) extract more than 300,000 German patent entries from archival image scans printed in varying multi-column layouts and historical fonts, reporting a pipeline several hundred times faster and cheaper than research assistants. The relevant information here is not plain text but is embedded in page structure, columns, headings, typography, and repeated visual patterns. A conventional workflow would separate this into OCR, layout parsing, post-correction, and field extraction. A multimodal LLM can perform several of these steps jointly, or at least reduce the engineering needed to connect them. Similarly, Greif et al. (2025) benchmark multimodal LLMs on historical German city directories across the three steps a digitization pipeline would normally tackle separately, namely machine-reading the text, fixing the errors in that raw output, and sorting each entry into separate spreadsheet columns. They find that one model can handle all three. City directories are a rich micro-geographic source in their own right; Albers and Kappner (2023) document their perks and pitfalls and

[35] While the elements can be digitized by the LLM, the caveats and issues regarding spatial reasoning mentioned in Section 3.1.1 apply.

Figure 3: OCR and LLM Transcription of a Historical Newspaper Article

Panel a: Original image

STOCK MARKET CRASH RECOVERY

John N. Willys Tells President Hoover That Millions of Cars Will Be Made Soon

Washington, Feb. 5. —Effects of the recent stock market crash will be entirely overcome within 30 or 60 days and American automobile production, while somewhat less than in 1929, will yield manufacturers as large profits as last year, John N. Willys of Toledo, Ohio, nationally known motor manufacturer, told President Hoover today.

Willys said he had originally estimated that 4,000,000 automobiles would be manufactured this year but that now he is confident this would be increased by at least 500,000 and possibly 1,000,000 cars.

Panel b: Tesseract OCR

```
STOCK MARKET -
GRASH RECOVERY
Johi N. Willys Tells President:
| Hoover That Millions of
= cf POM a -
'Washington, Feb, 5.  -Effects
of the receft stock market crash
iwill be entirely overgome within
130 or 60 days and American auto-
|what less than in 4929, will yield
[manvfactuters ag jarge profits |
as last yeat, yn cand of To-
letto, Ohio, natioua}ly known mo-
tor magufaeturer, told President |
py oder.  fy
estimated that 'see automo-
biles would, be manufactured this
year but that now:  he is confident
this would:  be Increased by .at
O80 eee 5 Ue ee.  |
```

Panel c: LLM transcription

```
STOCK MARKET
CRASH RECOVERY

John N. Willys Tells President
Hoover That Millions of
Cars Will Be Made
Soon

Washington, Feb.  5.-Effects
of the recent stock market crash
will be entirely overcome within
30 or 60 days and American auto-
mobile production, while some-
what less than in 1929, will yield
manufacturers as large profits
as last year, John N. Willys of To-
ledo, Ohio, nationally known mo-
tor manufacturer, told President
Hoover today.

Willys said he had originally
estimated that 4,000,000 automo-
biles would be manufactured this
year but that now he is confident
this would be increased by at
least 500,000 and possibly 1,000,-
000 cars.
```

Notes: Panel a displays the original newspaper screenshot from The Washington Democrat (Feb. 5th, 1930) from newspaperarchive.com, downloaded on May 31st, 2026. Panel b shows the digitized version generated by Tesseract OCR 5.4.0 in Python. Panel c displays the text transcribed from the image by ChatGPT-5.5 with high reasoning effort in the chat window. The enterprise version of the model was used, which does not retain the submitted data (see Section 3.1.2). All data, code, and prompts for this figure are included in the replication package at https://doi.org/10.3886/E249897V2.

develop a semi-automated digitization workflow.

The same logic applies to handwritten sources, which have been particularly challenging for OCR software and used to require specialized routines and algorithms.[36] Figure 4 provides an example of a handwritten census page from the 1950 Census of Population (panel a), which was transcribed and transformed into tabular format by ChatGPT-5.5. The LLM extracted the information accurately, and also realized that the farm column continued vertically with the *yes* answer for all individuals on the sheet. On a larger and more representative scale, Humphries et al. (2024) evaluate LLMs on eighteenth- and nineteenth-century handwritten documents and find that they can outperform specialized handwritten-text-recognition software, particularly when used to correct an existing transcription. The model need not be perfect on every page to be useful. A first pass accurate enough for targeted human review, or a correction layer on top of an existing OCR or handwritten text recognition (HTR) system, already cuts the labor required to make a source usable. The same correction ability extends to degraded or poorly scanned images.

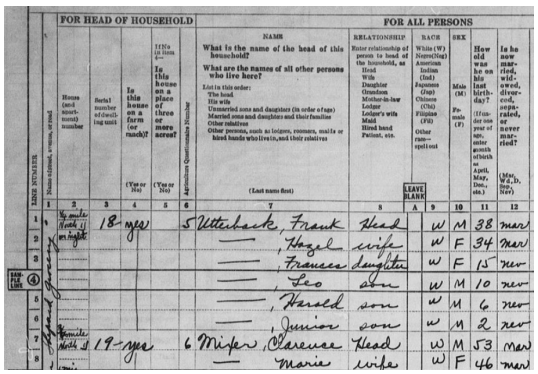
Recent papers are already making use of this new technology to extract data from historical sources in larger quantities. Bessa Ribeiro (2026) develops automated extraction pipelines using Claude, Gemini via Vertex AI, and OCR tools to turn digitized Rand McNally Bankers Directory pages and related banking sources into machine-readable bank-level data for the Great Depression era. The tabular data are then validated against city/county matching and balance-sheet verification. Chyn et al. (2024) use LLMs to read the Freedmen's Bureau records, which combine structured tables with free-text narratives to generate a data set of "outrages," the era's term for acts of violence against freedpeople after the Civil War. The model produces a first version that is then reviewed by human annotators.

In practice, the workflow for such projects follows the iterative pattern of Table 1 and the imple-

[36] See for instance the handwritten name (HANA) database for offline handwritten text recognition by Dahl et al. (2023) to improve linking methods faced with transcription errors in handwritten names.

Figure 4: Digitizing Handwritten Census Records with an LLM

Panel a: Original 1950 census manuscript



Panel b: Extracted structured data

Line	Farm	Name	Rel.	Race	Sex	Age	Mar.
1	Yes	Utterback, Frank	Head	W	M	38	Mar.
2	Yes	Utterback, Hazel	Wife	W	F	34	Mar.
3	Yes	Utterback, Frances	Daughter	W	F	15	Never
4	Yes	Utterback, Leo	Son	W	M	10	Never
5	Yes	Utterback, Harold	Son	W	M	6	Never
6	Yes	Utterback, Junior	Son	W	M	2	Never
7	Yes	Myers, Clarence	Head	W	M	53	Mar.
8	Yes	Myers, Marie	Wife	W	F	46	Mar.

Notes: Panel a shows a snippet of a handwritten census enumeration page obtained from the official census archives website (https://1950census.archives.gov, downloaded on June 1st, 2026). Panel b shows the LLM-transcribed data from the handwritten image for select variables that were requested in the prompt. The LLM used to transcribe the data was ChatGPT-5.5 with high reasoning effort in the chat window. The data files, prompts, and code used for this exercise can be found at https://doi.org/10.3886/E249897V2.

mentation guidance of Section 3, and so need not be restated in full here. A few points are specific to extraction. Schema design carries more weight than in a typical classification task, because the structure of the output is itself a research decision. A researcher has to decide and define in the LLM prompts which fields to record, how to encode missing or illegible entries, and how to represent one-to-many relationships. The prompt-engineering and validation steps of Sections 3.2.2 and 3.3.1 apply, and large production runs are best suited for the batch API and cost control ideas discussed in Section 3.2.4.

4.2 Merging historical data sets without unique identifiers

Merging data sets without unique identifiers is a commonly encountered task in economic history and applied economics. This includes linking data sets that only contain company names, addresses, or county and town names, among others. The most well-known example is census linking, which has a strong set of tools developed by the literature ranging from probabilistic matching to the supervised machine learning behind the Census Tree Project (Price et al., 2021). Linked census data continue to provide the foundation for much current research, typically matched with other outside data to study the specific question at hand (Abramitzky et al., 2025). LLMs are potentially useful tools for merging historical data sets as they can detect patterns across fields and variables that the researcher themselves may not have considered as matching variables.

The following example shows this with the canonical census linking task but with an important twist. The LLMs were not supplied with individuals' names, key variables for any census linking procedure. Since names from the full-count census cannot be submitted to a hosted LLM, the public-use census versions from IPUMS (Ruggles et al., 2024) were used.[37] The sample consists of 10,000 men from Connecticut who had already been linked by IPUMS from 1900 to 1910 (Ruggles et al., 2025). These individuals are identified via the historical identification key (HIK) variable. This linked sample acts as a quasi-ground truth data set. The sample size was chosen to reduce computational times and costs for

[37] Submissions were made via enterprise versions of OpenAI's ChatGPT and Anthropic's Claude models which do not store or retain any of the data. These were more readily accessible via the author's institution rather than through the NBER server which hosts the full-count census files with names.

this example. Without the names, any correct match must then come from age, birthplace, household composition, or any of the 171 total variables available in the census in both years.

The 1900 and 1910 data were then randomly sorted, before being sent to ChatGPT-5.5 and Claude Opus 4.8 with the request to link the two data sets. Table 3 reports the results from this name-blind census linking exercise. The first column reports the agreement between IPUMS and Census Tree Project links,[38] while the second column shows the unique matches achieved via a joinby merge based on birth year, race, county, state, birthplace, mother's birthplace, father's birthplace, occupation, industry, and city. This resulted in 9,258 and 429 matches, respectively, that agreed with the HIK links. The CTP and joinby match results provide a useful upper- and lower-bound case against which the LLM linking performance can be benchmarked. Columns 3 and 5 report the chat-based links produced by ChatGPT and Claude, which achieved 5,131 and 4,313 links that agreed with the HIK links. When using the agentic environments in columns 4 and 6, OpenAI Codex and Claude Code generated 5,728 and 6,243 matches that agreed with the IPUMS links. With direct access to the data and to the local Python distribution, the agentic models managed to use a broader range of variables to match on. The models were explicitly prompted not to just build a traditional census linking pipeline, which would have heavily relied on names, and to use all other available variables instead.

Table 3: Name-Blind Census Linking with LLMs

	CTP	Joinby	ChatGPT-5.5		Claude Opus 4.8	
			Chat	Codex	Chat	Claude Code
	(1)	(2)	(3)	(4)	(5)	(6)
Matches	9,258	429	5,131	5,728	4,313	6,243
Share	0.93	0.04	0.51	0.57	0.43	0.62

Notes: Results from name-blind census linking of 10,000 Connecticut men who had been successfully linked by IPUMS (Ruggles et al., 2025) between 1900 and 1910. These are the public-use files taken from the full count census without names (Ruggles et al., 2024). Column 1 reports the agreement between the IPUMS HIK (historical identification key) matches and the Census Tree Project (CTP) links provided by Price et al. (2021). Notice that while the CTP used to nest the IPUMS Multigenerational Longitudinal Panel (MLP) links, the MLP links have since been updated, which is why the CTP no longer matches perfectly, with an agreement rate of around 93 percent. Column 2 reports the unique matches from a joinby merge between the 1900 and 1910 files using birth year, race, county, state, birthplace, mother's birthplace, father's birthplace, occupation, industry, and city. The main question of interest is how well LLM-based models can link census records if no names are given. Columns 3 and 4 report the name-blind linking done by ChatGPT-5.5 with high reasoning effort in the chat (col. 3) and in an agentic model using Codex (col. 4). The exercise was repeated in columns 5 and 6 using Claude Opus 4.8 with extended thinking via the chat (col. 5) and an agentic model using Claude Code (col. 6). The respective OpenAI (ChatGPT) and Anthropic (Claude) enterprise versions were used. The data files, prompts, and code used for this exercise can be found at https://doi.org/10.3886/E249897V2.

Obviously, no researcher would use such a name-blind linked data set in their actual work. Match rates of 43 to 62 percent are far from sufficient for that. The more important takeaway is a different one. Even in the absence of names, those match rates are fairly high compared to the deterministic match in column 2, and probably better than what any human research assistant would have accomplished in the same setting. The main point was to demonstrate that LLMs can be potentially useful for merging historical data sets without a common identifier by exploiting a broader set of variables and patterns in the data that a human annotator would not have chosen or would have missed. Lastly, the assessment here was made possible because of the HIK links. For any real-life linking exercise trying to merge two data sets without unique identifiers, a hand-linked validation sample of the kind described in Section 3.3.1 would be necessary.

[38] The Census Tree Project links were originally built to include links made by IPUMS. The IPUMS links have since been updated, which is why the match rate with the CTP links is not a hundred percent.

4.3 Standardizing historical economic measures

Another broad use of LLMs is to harmonize historical data across time and space. Variables such as occupation, industry codes, education, or product types are available either as free text, as numerical codes that differ across schemes, or as a mix of the two. These are just some of the issues that researchers encounter when working with sources from varying countries or time spans. For empirical work, however, it is essential to keep measures as constant and comparable as possible.

LLMs can help with such standardization. The first option is to use a fine-tuned model as described in Section 3.2.1. The example given in that section was the OccCANINE classifier developed by Dahl et al. (2026). To achieve the standardization of occupational codes, the model trains on the actual distribution of historical occupational strings and reads them character by character. This allows the model to cope with the OCR errors and historical spellings that a modern-vocabulary prompt would miss. The LLM potentially also covers many languages and source schemes at once if it is trained on multilingual corpora and occupational descriptions. Once trained, it codes records quickly and at very little cost on a local machine, with results that can be reproduced even with larger data sets. This option is suitable when there are many labeled description-code pairs and the volume is large enough to justify the training cost as opposed to hiring a team of research assistants. Having said that, research assistants also require training and the LLM has the advantage of being internally consistent.

The second option skips training and prompts a hosted frontier model directly. It trades the reproducibility and consistency of a fixed classifier for flexibility, potentially higher performance, and lower fixed costs. This is an appropriate choice when the amount of data to be harmonized is low, or when the data layout or the desired target coding scheme are unique enough that no pretrained classifier exists. Several recent papers take this route. Ohler (2026) uses Claude to sort free-text occupational titles into status classes. These classes then serve as a socioeconomic-status measure to study fertility and the inheritance of status in pre-industrial Germany. Long et al. (2024) study the economic effects of the 1882 Chinese Exclusion Act. They find that the act slowed growth in the western United States well into the twentieth century. In an additional exercise, they use ChatGPT-4 to sort industries into tradable and non-tradable to estimate heterogeneous effects. Vidart (2026) digitizes International Correspondence Schools records and links students to census records to study correspondence education, occupational mobility, and selection into high school in early twentieth-century America. She uses ChatGPT and Claude to read the courses offered by these schools and assigns each one a schooling level, such as middle school or technical high school. The courses carry no standard grade label of their own, hence the models place them on a common educational scale.

Another interesting application is to build consistent measures over time that did not exist before. Asirvatham et al. (2026) use their own LLM pipeline to classify text about roughly 37,000 technologies. They document that the lag between a technology's invention and its adoption fell about tenfold over the industrial age, from roughly fifty years to roughly five. Liu et al. (2025) use LLM-generated task descriptions for U.S. Census occupations by decade, combine them with patent-text embeddings, and join the resulting technology-exposure measures to census occupation-employment data. Because occupational definitions shift over time, they emphasize comparability limits and focus much of the analysis on the post-1900 period. Griesshaber and Ogilvie (2025) do the same for institutions, turning colonial guild ordinances into comparable measures of human-capital requirements and product-quality regulation across Mexico and Peru. For robustness, they show that running the same model at temperature

zero reliably replicates the regression findings, even though temperature zero alone does not make LLM output fully deterministic (see Section 3.3.2). These examples showcase how LLMs can be used to build empirical measures with the property of comparability across time, space, source, and country.

A few caveats are specific to such standardized measures. First, harmonization can lose information or context. Forcing historical categories into a modern coding scheme can remove distinctions that mattered economically at the time. This is why U.S. census users occasionally resort to the original occupational strings instead of the harmonized occupational classification provided by IPUMS, for example. Second, different models might generate different standardizations. Yin et al. (2026) show that LLM-generated occupational-exposure scores can diverge significantly across models. The divergence is large enough that the same subsequent difference-in-differences estimate can change in magnitude, or flip sign, depending only on which model produced the measure. This relates to the measurement-error problem discussed in Section 3.3.4. Third, the model can stop reading the source and start drawing on its own training knowledge. When it assigns a code by reading the string in a given text, then that is standardization and it is what the researcher wants. The risk is that the model instead generates a value the source never contained, using the information the model had absorbed in pretraining. This is a real concern given that models, such as ChatGPT, are trained on a substantial amount of text that OpenAI has extracted and scraped from the internet and other sources.[39] This potentially includes online archives such as HathiTrust. The difficulty for researchers is that they cannot know what has been previously ingested by a frontier model during training.

Sometimes, this can be an advantage. Grajzl and Murrell (2025) build an annual index of print censorship in early-modern England. Censorship is inherently difficult to quantify as the relevant evidence is scattered across many qualitative sources. They therefore have the model assemble the index, combining its implicit historical knowledge from pretraining with hundreds of facts from secondary sources provided by the authors. The measure then leans on what the model absorbed in training, not only on the sources it was given. To check robustness, they repeat the index-building exercise and the empirical analysis with Claude 4, ChatGPT-4o, and Gemini 2.5 Pro to account for the potential issue that different models may generate different indexes. The alternative indexes are positively but not perfectly correlated with the original, while the paper reports that the main empirical findings are unchanged. A final point regarding the topic of harmonizing measures concerns the validation step. Even if researchers can reasonably generate a hand-coded validation data set, this benchmark may itself not be neutral. Celli and Spathoulas (2025) find that humans and models alike bring cultural assumptions to historical interpretation, so the "correct" codes a model is scored against can themselves encompass the annotators' priors.

4.4 Text classification, sentiment, and historical narratives at scale

Text analysis is by now well established in economics as diaries, newspapers, reports and correspondence can provide a unique window into the economic conditions but also the mindset or feelings of people at the time. A significant literature has quantified salience, emotions, and sentiment, or extracted information from such sources. LLMs can further reduce the barriers to entry for working with text as

[39] As mentioned earlier, LLMs are in essence very large neural networks. The larger these networks become, the larger the training data sets needed to train them well. Training refers to estimating the model weights that are used by the LLM to generate responses to later prompts.

data, either by supplying the relevant code to conduct the preparation and analysis steps as per the workflow sketched in Table 1, or by directly analyzing the data. Recent examples in the literature that have used LLMs for such purposes include Ottonello et al. (2024) who detect whether a statement expresses a political opinion. Giommoni et al. (2026) classify historical legislative speech, and Chyn et al. (2025) score the ideological content of government reports. Ferrara et al. (2025) estimate the quality of U.S. Civil War captains using a leader value added fixed effects approach. To validate this measure, they show that captains with better fixed effects were more often described as a leader, hero, or brave in their postwar biographies, and confirm this with a separate ChatGPT classification of whether each biography portrays the captain as a good leader. Lagakos et al. (2025) recover the meaning people attach to their lives from over 1,300 personal life-history narratives. Similar methods have been used to extract cultural values in folklore (Michalopoulos, 2025) and forms of social unrest in administrative records (Keller et al., 2024). That last case still uses supervised NLP of the kind an LLM could perform as well. LLMs can also run in the opposite direction, turning structured records into narrative text. For instance, McLean et al. (2024) generate life-course biographies from Australian convict records and find the output fluent but in need of human review for bias and stereotyping.

The worked example for this section measures the salience of, and sentiment toward, Chinese immigrants in U.S. newspapers around the 1882 Chinese Exclusion Act, whose economic effects Long et al. (2024) study. The data contain 524,458 newspaper pages from states in the Pacific and Mountain census divisions for the three months before and after passage of the Chinese Exclusion Act on May 6th, 1882. These were obtained from newspaperarchive.com. Since submitting over half a million pages to a hosted frontier model would be expensive, the example incorporates some of the data preprocessing and cost control steps discussed in Section 3.2.4. In particular, a Python script first performed a fuzzy match on period-specific vocabulary referring to Chinese people,[40] resulting in 11,011 pages that were flagged. The selected texts were then stripped of stop words. After that, the 30 words before and after each Chinese-related term were retained as the text snippets that were ultimately submitted to ChatGPT-5.4 mini via the OpenAI batch API.[41] The code also illustrates to readers how many files and prompts can be collected as a batch into a `.json` file for API batch submission. Submitting batches instead of sending many individual requests can save up to 50 percent of the costs for input and output tokens. The code also shows how the output can be retrieved and returned to the researcher as a structured tabular data set, which can then be analyzed. The exercise does no validation and draws no causal conclusion.

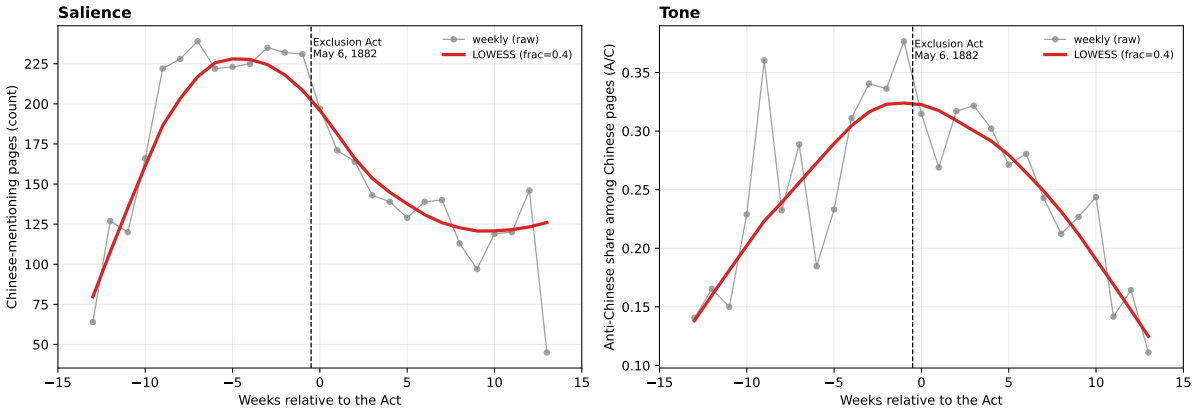
The results from this worked example are shown in Figure 5. Panel a documents that salience built up in the months before the Chinese Exclusion Act, peaking a few weeks before passage, where salience is defined as the raw weekly count of flagged pages that the model classified as concerning Chinese people.[42] Panel b shows that negative sentiment regarding Chinese people was also on the rise. Both measures fell after the act passed. A more granular empirical analysis is feasible in which a researcher could geolocate these newspapers to the county level and perform a difference-in-differences analysis

[40] The fuzzy word match raises the probability of including false positives in the final data set, i.e., pages that were flagged but did not include information about Chinese people, but it can reduce the rate of missed true positives by dealing with OCR errors in the OCR-processed historical texts.

[41] As with the census linking exercise in Section 4.2, the enterprise version of the model was used, which does not store or retain the submitted data beyond the duration of the request (see Section 3.1.2).

[42] An ideal measure would normalize this count, for instance using the weekly number of pages containing a neutral term such as "Monday" as the denominator, as recommended by Beach and Hanlon (2023), to avoid changes in corpus coverage or seasonality driving this pattern.

Figure 5: Salience and Tone in Newspapers Before and After the Chinese Exclusion Act



Notes: Weekly newspaper attention to the Chinese around the Chinese Exclusion Act (signed May 6th, 1882; dashed line), three months before and after. OCR-robust fuzzy matching on period terms for Chinese people (including slurs and racist tropes) screened 524,458 pages in the Mountain (369,471) and Pacific (154,987) census divisions and flagged 11,011 pages, each classified by ChatGPT-5.4 mini as concerning the Chinese and, if so, anti-Chinese, neutral, or sympathetic. Salience is the raw weekly count of pages concerning the Chinese (C_t); tone is the anti-Chinese share among them (A_t/C_t). Grey markers are raw weekly values; the red curve is a LOWESS fit (bandwidth 0.40). The input data files, prompts, and code (incl. the data pre-processing and GPT batch submissions) used for this exercise can be found at https://doi.org/10.3886/E249897V2.

using an above-/below-median Chinese-population in 1880 indicator as the treatment. In this case, newspaper-specific issues relating to coverage would need to be taken into account (Beach and Hanlon, 2023). Other caveats carry over from earlier sections. Model performance depends on the quality of the OCR and article segmentation behind it (Dell et al., 2023). Newspaper corpora are selected, and that selection shapes any measure built from them; hence, the result describes the corpus rather than public opinion at large (Beach and Hanlon, 2023). van Binsbergen et al. (2024) measure newspaper sentiment over almost two centuries and report that LLMs were infeasible at full corpus scale. That is the cost problem the preprocessing in the worked example provided here was designed to solve.

4.5 Non-standard data: Images, audio, songs, and videos

The initial example from Gorin et al. (2025) and their analysis of paintings has showcased how creatively economists can use different types of data that do not necessarily appear to be *data* prima facie. The combination of data creativity and some serious coding and machine learning skills was key to unlocking such data. As Section 2.2 has argued, the rise of LLMs has shifted this skill distribution toward the creativity side. This section highlights other potential use cases for LLMs and non-standard data sources, such as audio, video, songs, folklore and tales, among others, to inspire researchers to think outside the box and to uncover more such exciting data uses.

There are several recent papers that fit this description, even if they do not use LLMs or GenAI to find, extract, and analyze their data sets. What many of them have in common though is that the data themselves tend to be a major contribution. Voth and Yanagizawa-Drott (2024) measure conformity and style change using more than 14 million U.S. high-school yearbook portraits. Their paper relies on a tailored image-processing pipeline, including trained models for style attributes, rather than off-the-shelf vision-language prompting. This illustrates both the value of image archives and the kind of highly specialized data work that newer multimodal models can open up to more researchers wishing to analyze such data. Gorin et al. (2025), the painting-emotion study behind the worked example of Section 2, treats

visual material as economic measurement in the same way. Spinaci et al. (2025) give direct evidence that multimodal models can classify cultural-heritage imagery at scale. The main takeaway from these examples is that visual data can hold social, political, religious, and economic information, even if it is not immediately obvious.

The same lesson applies to audio, which carries information that no textual transcript preserves. What a central banker said can be read from the minutes, but not how it was said. Gorodnichenko et al. (2023) show that the tone matters for monetary policy and markets, based on the 7-38-55 rule posited by Mehrabian (1971). It states that 7 percent of meaning is conveyed by the spoken word, 38 percent by tone of voice, and 55 percent by body language. Gorodnichenko et al. (2023) combine speech-emotion recognition with BERT-based text analysis to study nonverbal monetary-policy communication. They classify the vocal tone of Federal Reserve chairs during post-FOMC press-conference sessions and control for the hawkish or dovish content of policy texts using BERT embeddings. They find that a more positive vocal tone raises stock prices and affects inflation expectations and exchange rates, suggesting that non-textual features of communication move financial markets independently of policy actions and textual content.[43] Another interesting example that could have used audio-based methods is provided by Logothetis et al. (2022). They trace the ancestry of Greek singers who were related to Greek refugees who had to flee Turkey during the 1923 population exchange between the two countries. Using textual data of scraped song lyrics, Logothetis et al. (2022) show that singers who had refugee ancestors were more likely to sing about topics relating to impoverishment, slums, hardship, or injustice. Similar ideas apply to video, which may be somewhat less applicable to economic history. But as time progresses, a larger period for which video is a potential data source will become part of economic history. The tools for such sources are maturing quickly; zero-shot LLM rerankers can already search large audio and video archives by topic (Qian et al., 2024).

Lastly, this section provides a worked example that applies the workflow sketched in Table 1 to analyze the speeches by President Franklin D. Roosevelt (FDR) in the roughly three months before and after the attack on Pearl Harbor. Audio files for the 16 speeches given by FDR between September 1941 and March 1942 were obtained from https://www.fdrlibrary.org/. Claude Opus 4.7 was prompted to develop the workflow, explore potential data extraction and analysis tools, and to generate the Python code to perform the voice analysis.[44] The Day of Infamy speech on December 8, 1941, i.e., the day after the attack on Pearl Harbor, was analyzed individually and then together with the other speeches in the roughly three months before and after combined for comparison.

The emotional delivery in FDR's December 8, 1941 address to Congress was scored using audEERING's `wav2vec2-large-robust-12-ft-emotion-msp-dim`, a pretrained speech emotion recognition (SER) model. It is a deep neural network, pretrained on unlabeled speech and then fine-tuned on the MSP-Podcast corpus of human-rated podcast clips. The model outputs three continuous dimensions of affect on a $[0, 1]$ scale, i) valence (negative-to-positive), ii) arousal (calm-to-excited), and

[43] Similarly, Kakhbod et al. (2026) use LLMs to recover the Fed's real-time attribution of inflation to demand and supply pressures from FOMC meeting records, but without the audio component. Howes et al. (2026) hand-code FOMC transcripts from 1966 to 1990 to characterize members' policy preferences, a large-scale manual coding effort of exactly the kind LLMs now stand to accelerate.

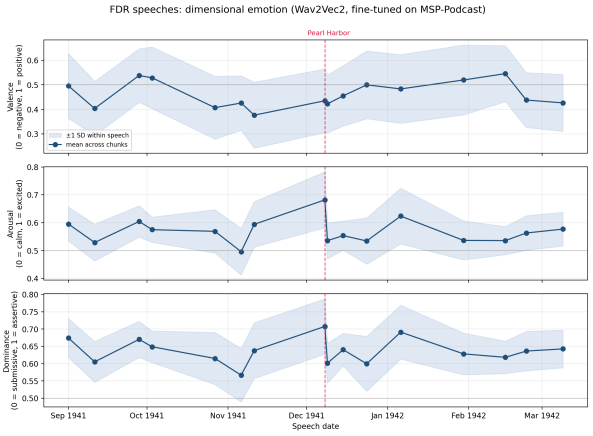
[44] This is also a computationally interesting example. The analysis was performed with a GPU instead of using the conventional CPU computing to which most researchers are accustomed. This reduced the analysis time from 144 minutes on the CPU to 8 minutes on the GPU. The analysis was performed on an NVIDIA GeForce RTX 5090 graphics processor and compared to the same analysis using an Intel Core Ultra 9 285K CPU with 24 cores.

iii) dominance (submissive-to-assertive). The voice data were partitioned into overlapping five-second windows that were each scored by the model. This produced a time series of the three dimensions across the speech. The same procedure was then applied to all of FDR's 16 recorded speeches over the sample period. For the cross-speech comparisons, the window-level predictions of valence, arousal, and dominance were averaged with within-speech standard deviations capturing variability across the address.

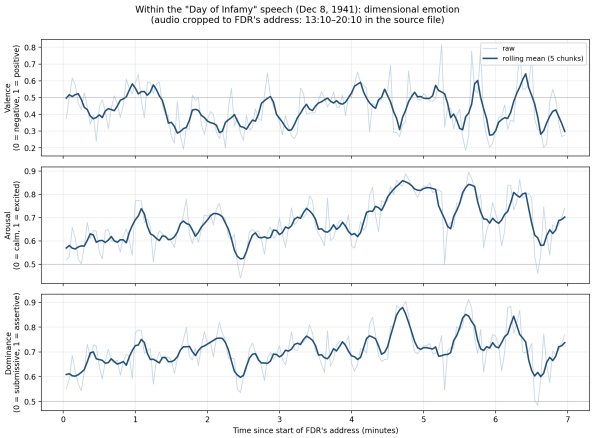
Figure 6 shows the within- and between-speech comparison measures. The December 8 address is a clear outlier in the corpus as its mean arousal and mean dominance are the highest of any speech in the six-month period, with both dimensions returning to baseline within days (panel a). Valence on December 8 is more in the middle of the corpus distribution. The within-speech measures in panel b zoom into the December 8 speech, where arousal and dominance rise over the course of the address and peak between minutes 4:45 and 6:30. This is where FDR enumerates the Japanese attacks and builds to his closing request for a declaration of war. Valence is noisier and shows no clear trend, though some dips coincide with the early framing of the attack.

Figure 6: Analyzing FDR Speeches Before, During, and After Pearl Harbor

Panel a: Speeches between Sept. 1941 – March 1942



Panel b: Day of Infamy Speech (Dec. 8, 1941)



Notes: Panel a shows mean valence, arousal, and dominance for each of FDR's 16 recorded speeches between September 1, 1941 and March 9, 1942, with shaded bands giving ± 1 standard deviation across windows within each speech. The vertical dashed line marks the Day of Infamy address on December 8, 1941. Panel b shows the within-speech time series for that address (cropped to 13:10 to 20:10 in the source file). Light traces are raw window-level scores and dark traces are five-window rolling means. The audio comes from the FDR Presidential Library. Each speech was cropped to FDR's actual address, partitioned into overlapping five-second windows, and scored window-by-window with audEERING's `wav2vec2-large-robust-12-ft-emotion-msp-dim`, a deep neural network fine-tuned on the MSP-Podcast corpus. The model outputs valence (negative-to-positive), arousal (calm-to-excited), and dominance (submissive-to-assertive) on a $[0, 1]$ scale. The data files, prompts, and code used for this exercise can be found at https://doi.org/10.3886/E249897V2.

5 Conclusion

Looking ahead, the use of LLMs in economic history research is likely to continue growing rapidly. Thus far, applications have been concentrated in a few areas: the extraction of structured data from historical sources, the harmonization of measures across time and place, and the classification of mostly English-language text. There is clearly scope to broaden beyond these. Much of the historical record consists of material that used to be uneconomical to process, from handwritten ledgers to maps, paintings, and sound recordings, so the set of questions that can now be asked affordably is considerably larger than

the set that has been asked.

A number of directions for future work seem particularly promising. One is the move beyond Anglo-American sources. Model performance in other languages and historical scripts keeps improving, and many archives outside the English-speaking world have never been systematically exploited because transcription costs were prohibitive. Another direction is the non-standard data discussed in Section 4.5. The voice of a politician, the mood of a painting, or the imagery of a propaganda film carry economic information that no transcript preserves, and the worked examples in this guide indicate that such sources are now within reach of researchers without specialized pipelines.

The infrastructure around these tools is also still taking shape. The econometrics of LLM-generated variables is young, and additional tests, corrections, and validation standards are likely to follow (Ludwig et al., 2026; Battaglia et al., 2025). Journal and data-editor policies for AI-assisted empirical work are similarly in flux, which makes the documentation and archiving practices described in Section 3.3.3 all the more valuable. Many of the technical details in this guide will age quickly as models, prices, and interfaces change. The workflow itself should age more slowly: match the tool to the task, validate the output against hand-coded benchmarks, archive what cannot be regenerated, and keep the questions, the judgment, and the writing with the researcher.

The next frontier in this technology pertains not only to the models themselves but also to the underlying computational architecture. One important development is quantum computing, which promises to expand the set of computational problems that can be tackled at scale (Aizpurua et al., 2026). Unlike conventional computers, which process information through bits that take values of either zero or one, quantum computers use qubits that can exist in superpositions of both states, allowing them to represent and manipulate information in fundamentally different ways. This does not mean that quantum computers will replace classical computers for every task, but it does suggest that some computationally demanding problems may eventually become much more tractable. The first economics working papers using quantum computing have already begun to appear (e.g. Fernández-Villaverde and Hull, 2023), and quantum-enhanced LLMs and GenAI systems may become part of the next stage in the development of these tools. The broader point is that LLMs are not a finished technology: their future capabilities will depend not only on better models, but also on the computational infrastructure that makes new forms of analysis feasible.

References

Abramitzky, Ran, Leah Platt Boustan, and Adam Storeygard, "New Data and Insights in Regional and Urban Economics," *NBER Working Paper No. 33561*, 2025.

Achlioptas, Panos, Maks Ovsjanikov, Kilichbek Haydarov, Mohamed Elhoseiny, and Leonidas Guibas, "ArtEmis: Affective Language for Visual Art," *CoRR*, 2021, *abs/2101.07396*.

Afonso, Santiago, Sebastian Galiani, Ramiro H. Gálvez, and Raul A. Sosa, "Deep Research on a Loop: Using AI Agents to Construct Economic Datasets," *NBER Working Paper No. 35188*, 2026.

Aizpurua, Borja, Sukhbinder Singh, Augustine Kshetrimayum, Saeed S. Jahromi, and Roman Orus, "Quantum-Enhanced Large Language Models on Quantum Hardware via Cayley Unitary Adapters," *arXiv:2605.05914*, 2026.

Albers, Thilo N. H. and Kalle Kappner, "Perks and Pitfalls of City Directories as a Micro-Geographic Data Source," *Explorations in Economic History*, 2023, *87*, 101476.

Aldighieri, Pedro and Franco Malpassi, "Technological Breakthroughs and the Progress of Science: Evidence from Early Computers," *Working Paper*, 2026.

Asirvatham, Hemanth, Elliott Mokski, and Andrei Shleifer, "GPT as a Measurement Tool," *NBER Working Paper No. 34834*, 2026.

Bäcker-Peral, Verónica, Vitaly Meursault, and Christopher Severen, "Can LLMs Credibly Transform the Creation of Panel Data from Diverse Historical Tables?," *arXiv:2505.11599*, 2025.

Battaglia, Laura, Timothy Christensen, Stephen Hansen, and Szymon Sacher, "Inference for Regression with Variables Generated by AI or Machine Learning," *arXiv:2402.15585*, 2025.

Beach, Brian and W. Walker Hanlon, "Historical Newspaper Data: A Researcher's Guide," *Explorations in Economic History*, 2023, *90*, 101541.

Bingley, Paul and Alessandro Martinello, "Measurement Error in Income and Schooling and the Bias of Linear Estimators," *Journal of Labor Economics*, 2017, *35* (4), 1117–1148.

Bini, Pietro, Lin William Cong, Xing Huang, and Lawrence J. Jin, "Behavioral Economics of AI: LLM Biases and Corrections," *NBER Working Paper No. 34745*, 2026.

Celli, Fabio and Georgios Spathoulas, "Cultural Biases of Large Language Models and Humans in Historical Interpretation," *arXiv:2504.02572*, 2025.

Chen, Hui, Antoine Didisheim, and Luciano A. Somoza, "Out of the Black Box: Uncertainty Quantification for LLMs via Conditional Probabilities," *NBER Working Paper No. 34965*, 2026.

Chyn, Eric, Kareem Haggag, and Bryan A. Stuart, "Inequality and Racial Backlash: Evidence from the Reconstruction Era and the Freedmen's Bureau," *NBER Working Paper No. 32314*, 2024.

_ , _ , and Christian Maruthiah, "Ideology in Government: Evidence from the Office of Indian Affairs and the Assimilation Era," *NBER Working Paper No. 34415*, 2025.

Crane, Leland D., Akhil Karra, and Paul E. Soto, "Total Recall? Evaluating the Macroeconomic Knowledge of Large Language Models," *Finance and Economics Discussion Series No. 2025-044*, 2025.

Dahl, Christian M., Torben S. D. Johansen, Emil N. Sørensen, and Simon F. Wittrock, "HANA: A Handwritten Name Database for Offline Handwritten Text Recognition," *Explorations in Economic History*, 2023, *87*, 101473.

Dahl, Christian Møller, Torben Johansen, and Christian Vedel, "Breaking the HISCO Barrier: Automatic Occupational Standardization with OccCANINE," *arXiv:2402.13604*, 2026.

de Weghe, Nico Van, Lars De Sloover, Anthony G. Cohn, Haosheng Huang, Simon Scheider, Renée Sieber, Sabine Timpf, and Christophe Claramunt, "Opportunities and Challenges of Integrating Geographic Information Science and Large Language Models," *Journal of Spatial Information Science*, 2025, *30*, 93–116.

Dell, Melissa, "Deep Learning for Economists," *Journal of Economic Literature*, 2025, *63* (1), 5–58.

_ , Jacob Carlson, Tom Bryan, Emily Silcock, Abhishek Arora, Zejiang Shen, Luca D'Amico-Wong, Quan Le, Pablo Querubin, and Leander Heldring, "American Stories: A Large-Scale Structured Text Dataset of Historical U.S. Newspapers," *arXiv:2308.12477*, 2023.

Du, Yufeng, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton

Wells, Roy Schwartz, Eliu A. Huerta, and Hao Peng, "Context Length Alone Hurts LLM Performance Despite Perfect Retrieval," *arXiv:2510.05381*, 2025.

Fernández-Villaverde, Jesús and Isaiah J. Hull, "Dynamic Programming in Economics on a Quantum Annealer," *NBER Working Paper No. 31326*, 2023.

Ferrara, Andreas, Christian Dippel, and Stephan Heblich, "Frontline Leadership: Evidence from American Civil War Captains," *NBER Working Paper No. 34057*, 2025.

_ , Joung Yeob Ha, and Randall P. Walsh, "Using Digitized Newspapers to Address Measurement Error in Historical Data," *Journal of Economic History*, 2024, *84* (1), 271–306.

Giommoni, Tommaso, Gabriel Loumeau, and Marco Tabellini, "Extractive Taxation and the French Revolution," *NBER Working Paper No. 34816*, 2026.

Gorin, Clément, Stephan Heblich, and Yanos Zylberberg, "State of the Art: Economic Development Through the Lens of Paintings," *NBER Working Paper No. 33976*, 2025.

Gorodnichenko, Yuriy, Tho Pham, and Oleksandr Talavera, "The Voice of Monetary Policy," *American Economic Review*, 2023, *113* (2), 548–584.

Grajzl, Peter and Peter Murrell, "Innovation Under Suppression: Censorship's Effect on Cultural Production in Early-Modern England," *SSRN Working Paper No. 5382070*, 2025.

Greif, Gavin, Niclas Griesshaber, and Robin Greif, "Multimodal LLMs for OCR, OCR Post-Correction, and Named Entity Recognition in Historical Documents," *arXiv:2504.00414*, 2025.

Griesshaber, Niclas and Jochen Streb, "Multimodal LLMs for Historical Dataset Construction from Archival Image Scans: German Patents (1877–1918)," *arXiv:2512.19675*, 2025.

_ and Sheilagh Ogilvie, "Transplanting Craft Guilds to Colonial Latin America: A Large Language Model Analysis," *CEPR Discussion Paper No. DP20556*, 2025.

Howes, Cooper, Marc Dordal i Carreras, Olivier Coibion, and Yuriy Gorodnichenko, "How Monetary Policy Is Made: Lessons from Historical FOMC Discussions," *NBER Working Paper No. 34632*, 2026.

Humphries, Mark, Lianne C. Leddy, Quinn Downton, Meredith Legace, John McConnell, Isabella Murray, and Elizabeth Spence, "Unlocking the Archives: Large Language Models Achieve State-of-the-Art Performance on the Transcription of Handwritten Historical Documents," *arXiv:2411.03340*, 2024.

Kakhbod, Ali, Amir Kermani, and Bernardo Maciel, "In the Fed's Mind," *NBER Working Paper No. 35016*, 2026.

Keller, Wolfgang, Carol H. Shiue, and Sen Yan, "Mining Chinese Historical Sources at Scale: A Machine Learning-Approach to Qing State Capacity," *NBER Working Paper No. 32982*, 2024.

Kıcıman, Emre, Robert Osazuwa Ness, Amit Sharma, and Chenhao Tan, "Causal Reasoning and Large Language Models: Opening a New Frontier for Causality," *arXiv:2305.00050*, 2024.

Korinek, Anton, "Generative AI for Economic Research: Use Cases and Implications for Economists," *Journal of Economic Literature*, 2023, *61* (4), 1281–1317.

Lagakos, David, Stelios Michalopoulos, and Hans-Joachim Voth, "American Life Histories," *Working Paper*, 2025.

Liu, Huben, Dimitris Papanikolaou, Lawrence D. W. Schmidt, and Bryan Seegmiller, "Technology and Labor Markets: Past, Present, and Future; Evidence from Two Centuries of Innovation," *NBER Working Paper No. 34386*, 2025.

Liu, Nelson F., Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics*, 2024, *12*, 157–173.

Logothetis, Vassilis, Stelios Michalopoulos, and Elias Papaioannou, "Songs of Refugees," *Presentation*, 2022.

Long, Joe, Carlo Medici, Nancy Qian, and Marco Tabellini, "The Impact of the Chinese Exclusion Act on the Economic Development of the Western U.S.," *Harvard Business School Working Paper No. 23-008*, 2024.

Ludwig, Jens, Sendhil Mullainathan, and Ashesh Rambachan, "Large Language Models: An Applied Econometric Framework," *Annual Review of Economics*, 2026, *18*, 283–316.

McLean, Mark A., David Andrew Roberts, and Martin Gibbs, "Ghosts and the Machine: Testing the Use of Artificial Intelligence to Deliver Historical Life Course Biographies from Big Data," *Historical Methods: A*

Journal of Quantitative and Interdisciplinary History, 2024, *57* (3), 146–162.

Mehrabian, Albert, *Silent Messages: Implicit Communication of Emotions and Attitudes*, Belmont, CA: Wadsworth Publishing, 1971.

Michalopoulos, Stelios, "Ethnographic Records, Folklore, and AI," *NBER Working Paper No. 33700*, 2025.

Moulton, Daniel and Christopher Severen, "Harvesting Historical Data with LLMs," *Federal Reserve Bank of Philadelphia Research Department, 2025 Q4*, 2025.

Ohler, Johann, "Selection and Evolutionary Growth in pre-Industrial Germany," *LSE Economic History Working Paper No. 390*, 2026.

Ottonello, Pablo, Wenting Song, and Sebastian Sotelo, "An Anatomy of Firms' Political Speech," *NBER Working Paper No. 32923*, 2024.

Paulsen, Norman, "Context Is What You Need: The Maximum Effective Context Window for Real World Limits of LLMs," *arXiv:2509.21361*, 2026.

Price, Joseph, Kasey Buckles, Jacob Van Leeuwen, and Isaac Riley, "Combining Family History and Machine Learning to Link Historical Records: The Census Tree Data Set," *Explorations in Economic History*, 2021, *80*, 101391.

Qian, Mengjie, Rao Ma, Adian Liusie, Erfan Loweimi, Kate M. Knill, and Mark J. F. Gales, "Zero-Shot Audio Topic Reranking Using Large Language Models," *arXiv:2309.07606*, 2024.

Ribeiro, Mariana Bessa, "Branch Banking Regulations and Financial Stability: A Cross-State Analysis of Depression-Era Banking," *Working Paper*, 2026.

Ruggles, Steven, Matt A. Nelson, Matthew Sobek, Catherine A. Fitch, Ronald Goeken, J. David Hacker, Evan Roberts, and J. Robert Warren, "IPUMS Ancestry Full Count Data: Version 4.0," Minneapolis, MN: IPUMS 2024. Dataset.

_ , Nesile Ozder, Catherine A. Fitch, Matthew Sobek, Julia A. Rivera Drew, J. David Hacker, Jonas Helgertz, Cheyenne Lonobile, Matt A. Nelson, Evan Roberts, and John Robert Warren, "IPUMS Multigenerational Longitudinal Panel: Version 2.0," Minneapolis, MN: IPUMS 2025. Dataset.

Spinaci, Gianmarco, Lukas Klic, and Giovanni Colavizza, "Benchmarking Vision-Language and Multimodal Large Language Models in Zero-Shot and Few-Shot Scenarios: A Study on Christian Iconography," *arXiv:2509.18839*, 2025.

Topaz, Maxim, Nir Roguin, Pallavi Gupta, Zhihong Zhang, and Laura-Maria Peltonen, "Fabricated Citations: An Audit Across 2.5 Million Biomedical Papers," *Lancet*, 2026, *407*, 1779–1781.

van Binsbergen, Jules H., Svetlana Bryzgalova, Mayukh Mukhopadhyay, and Varun Sharma, "(Almost) 200 Years of News-Based Economic Sentiment," *NBER Working Paper No. 32026*, 2024.

Vidart, Daniela, "Learning by Mail: The Impact of Correspondence Schools in Early 20th Century America," *NBER Working Paper No. 35147*, 2026.

Voth, Hans-Joachim and David Yanagizawa-Drott, "Image(s)," *Working Paper*, 2024.

Yin, Michelle, Hoa Vu, and Claudia Persico, "How (un)Stable Are LLM Occupational Exposure Scores? Evidence from Multi-Model Replication," *NBER Working Paper No. 35110*, 2026.

Appendix

List of Figures

List of Tables

A LLM Workflow Worked Example

This appendix provides a worked example applying the workflow introduced in Table 1 to the setting of Gorin et al. (2025), i.e., the emotional classification of paintings using images from ArtEmis (Achlioptas et al., 2021). The model used for the conversation was Claude Opus 4.7 with extended thinking, i.e., the model's extended-reasoning setting, which allocates more compute to working through complex prompts. The sample images used for the analysis were taken from Figure 3 in Gorin et al. (2025), which shows eight images that represent each emotion with the highest emotion scores provided by the ArtEmis annotators. This example of the workflow discussed in Section 2 is a minimal approach without the validation steps 7 and 8, as well as with reduced challenging of the model, the proposed options, code, and output. This is due to the length of the conversation. The reported prompts and responses are a condensed version of the entire conversation, where omissions are marked by [...]. The entire conversation and input files, as well as the resulting Python code and output files produced can be accessed at https://doi.org/10.3886/E249897V2. The main purpose of this appendix is to provide a quick overview for an applied example of the workflow introduced in Table 1, highlighting how even limited effort can make non-standard data sources (such as images) accessible to researchers via the use of LLMs.

Researcher prompt (Step 1: introducing the task)

I would like to extract the emotional content from a series of paintings. I have these as png files. I would like to measure how much of a given emotion is being conveyed by each painting. The emotions I am looking for are: contentment, amusement, excitement, awe, fear, anger, sadness, and disgust. I am uploading a few example images for your reference such that you can understand what I am working with. Can you suggest some statistical models or methods that can do what I want to achieve?[a]
(Please treat this chat as if we had never spoken, i.e., pretend like you know nothing about me or my previous projects)[b]

[a] The original prompt has 4 images attached.
[b] The parentheses were inserted because Claude responded initially with previous knowledge from other projects.

LLM response

Good news: your exact eight emotions are the ArtEmis taxonomy. Achlioptas et al. (2021) built ArtEmis specifically around amusement, anger, awe, contentment, disgust, excitement, fear, and sadness (plus a ninth "something else" you can ignore), and the data set contains 439K emotion attributions on 81K paintings from WikiArt. That's almost a free lunch for your problem — you have a ground-truth data set and pretrained models in the literature using the identical label set. Let me walk through the realistic options.

Option 1: Zero-shot with a vision-language model (probably your best starting point). Send each PNG to GPT-4V, Claude, or Gemini with a prompt asking for a 0–100 rating on each of the eight emotions, with a brief justification. GPT-4 has been shown to closely approximate human emotion ratings under zero-shot conditions on validated affective image databases like GAPED, and GPT-

4V exhibits strong multimodal emotion understanding, sometimes surpassing supervised systems. [...]

Option 2: Fine-tuned classifier trained on ArtEmis. This is the publication-defensible option. You'd take a vision backbone (CLIP image encoder, ViT, or ResNet) and train a multi-label head on ArtEmis.

Option 3: CLIP zero-shot. Encode each painting with CLIP's image encoder, encode prompts like "a painting that evokes [emotion]" with the text encoder, and compute cosine similarities. Cheap, no API costs, fully reproducible. Lower accuracy than VLMs on art specifically but useful as a sanity-check baseline or for very large corpora.

Option 4: Hybrid (image → caption → text emotion classifier). Have a VLM produce an emotionally-grounded description, then run that text through a text emotion classifier (e.g., GoEmotions-fine-tuned models on HuggingFace). Decomposes the problem and lets you swap components, but adds error compounding.
[...]

Researcher prompt (Step 2: understanding the pros/cons of each option)

What are the relative pros and cons for each of the four options you presented? What is the most suitable option for my goal of being able to classify possibly hundreds of paintings into the above mentioned emotions?

LLM response

Let me break each down and then give you a concrete recommendation.

Option 1 — Zero-shot VLM (GPT-4V, Claude, Gemini)
Pros: No training data required; handles diverse styles and periods well (a big deal given your four examples span 500 years); produces continuous per-emotion ratings naturally (which matches your stated goal of measuring how much of each emotion, not just the dominant one); returns interpretable text explanations alongside scores, which strengthens your downstream analysis; trivial to iterate on prompt design; fastest path from PNGs to data.

Cons: Costs API credits (cheap at your scale — roughly $5–25 total for 500 images); not fully deterministic unless you set temperature=0 and pin the model version; known positive-emotion bias in some VLMs requiring calibration; reproducibility depends on the provider keeping the model available.
[...]

Researcher prompt (Step 3: understanding the chosen option)

How does a VLM work? Can you explain the details to me? I am not a computer scientist but I would like to better understand the mechanics behind a VLM and appreciate what these models are

doing at a conceptual level.

LLM response

Happy to walk through this. The core ideas are not actually that hard once you see the trick — VLMs are built on a fairly elegant insight that I think you'll find intuitive given your econometrics background.

The central insight: turn everything into vectors in the same space. A VLM's job is to "understand" images and text together. The way it pulls this off is by converting both modalities into the same mathematical object — a long list of numbers, called a vector or embedding, typically 768 to 4096 dimensions. Once an image and a sentence both live in the same vector space, you can do meaningful operations on them: compute similarity, combine them, ask the model to generate text conditioned on an image, and so on. The whole architecture is in service of this trick.

Step 1: Turning an image into a sequence of vectors. A modern VLM treats an image somewhat like a paragraph of "visual words." It chops the image into a grid of small square patches — say, 16×16 pixels each — and runs each patch through a small neural network that produces a vector summarizing what's in that patch. A 224×224 image becomes a sequence of 196 patch vectors. This module is called a Vision Transformer, or ViT. Crucially, the model also adds positional information so it knows which patch came from where in the image. After this step, your painting is no longer pixels — it's a sequence of vectors, each one a numerical summary of one small region.
[...]

Researcher prompt (Step 4: request implementation)

Can you write Python script that can implement this approach? I have an API key for ChatGPT.

LLM response

Here's what the script does and a few notes on the choices I made.

Pipeline structure. Three independent rating methods run per image, results merged into one CSV with eight scores per method (24 emotion columns total) plus the raw handcrafted features and the ChatGPT-4o justification text. After processing, it prints and saves a second CSV of pairwise Spearman correlations between methods, per emotion — that's your contamination diagnostic.

Method 1 (ChatGPT-4o). Uses temperature=0 and seed=42 for near-deterministic outputs, response_format=json_object to guarantee parseable responses, and the prompt explicitly tells the model to rate emotions independently rather than picking a dominant one. Retries with exponential backoff on transient errors.
[...]

Step 5, the code audit for correctness and efficiency gains, was skipped here, both because only eight images were being classified and to keep the appendix conversation concise.

Researcher prompt (Step 6 (first pass): inspecting sample output)

This is what the output looks like for the first 3 images. Does this look okay to you? Any red flags?

```
Found 3 images.
[1/3] painting1.png
[2/3] painting2.png
[3/3] painting3.png
Shape: (3, 34)
======================================================================
Cross-method agreement (Spearman correlation per emotion)
Interpretation: agreement => robust to method choice / contamination
======================================================================
emotion          GPT~CLIP   GPT~hand  CLIP~hand
--------------------------------------------------
contentment         0.500     -0.500      0.500
amusement           0.500      0.500     -0.500
excitement          1.000     -0.500     -0.500
awe                 0.500     -0.500     -1.000
fear               -0.500     -0.500      1.000
anger               0.866      0.866      1.000
sadness             0.000     -0.866     -0.500
disgust             0.866     -0.866     -1.000
```

LLM response

The pipeline ran cleanly — script works, files are being written, agreement diagnostics are being computed. But there is one big red flag in the output, and it's a methodological one rather than a software one.

The Spearman correlations are meaningless at N=3. Look at the values: 1.0, 0.5, -0.5, -1.0, 0.866, -0.866, 0.0. Those aren't your data telling you anything — those are essentially the only values Spearman can return with three observations. With three paintings and no ties, there are only 3! = 6 possible rank permutations, so ρ can only take values in 1.0, 0.5, -0.5, -1.0. The 0.866 and 0.0 values appear when there are ties in the rankings. In other words, the entire agreement table reflects sample size, not signal.

This isn't a bug. It's just that the diagnostic was designed to be informative at N=50 or 100, not at N=3. As a sanity check on the pipeline, your run is fine; as a measurement of cross-method agreement, those numbers shouldn't be interpreted.
[...]

Researcher prompt (Step 6 (second pass): checking output from classifying all 8 paintings)

Here is what the emotion scores look like. What should I take away from this? Which image corresponds to which emotion? [output csv attached]

LLM response

This is a really nicely designed validation set — it looks like you've curated one painting per emotion in the order I listed them, which lets us test whether the model recovers the intended emotion. Let me work through what's there.

Mapping (GPT's top-scoring emotion per painting):

Painting	Apparent intent	GPT top	GPT score	CLIP top	Match?
1	contentment	contentment	85	contentment	✓ both
2	amusement	amusement	85	amusement	✓ both
3	excitement	excitement	85	amusement	✓ GPT, near-miss CLIP
4	awe	awe	80	awe	✓ both
5	fear	fear	85	fear	✓ both
6	anger	anger	40	disgust	✓ GPT weak, miss CLIP
7	sadness	sadness	90	sadness/disgust	✓ both
8	disgust	sadness	50	disgust	miss GPT, ✓ CLIP

GPT correctly identifies the intended emotion in 7 of 8 cases as the top score. CLIP gets 6 of 8. Across the two methods together, every intended emotion is correctly identified by at least one method.

[...]

Additional Tables and Figures

Table A.1: Examples of LLM Use Cases in Historical Research

Purpose	Example questions
Understanding a method in a paper	• How did the authors implement this particular estimator? • Can you explain the main assumptions to me in very basic terms? • Can you implement this estimator in Python and write simulation code so I can understand this better?
Challenging your own empirical work	• Here is my setting [insert description], what potential issues can you foresee with this? • What would be an argument for an exclusion restriction violation for my IV approach? • Did I estimate my standard errors correctly or do I need something else here?
Final touches on the manuscript	• Can you check that my table and figure notes are consistent with the text? • Carefully read each section [attach sections as single PDFs]. Are there flaws or internal inconsistencies in my chain of logic? • Please look for typos or grammar errors. Do not change the text, only flag what is strictly incorrect.
Brainstorming ideas	• Here are the results and robustness checks I have [upload]; what other robustness or sensitivity checks should I consider? • I would like to study the Great Depression. Which archives and data repositories would be a good start? • Here is a data set that I just collected. Have a careful look and give me some ideas for what kinds of topics I could study with this.
Optical Character Recognition	• Please turn this image of text into a machine-readable format. I need this as a .txt file. • Can you clean up my messy OCR text? • Can you extract these tables from the PDF and store them as separate CSV files?
Code documentation	• Here are all my Python files. Can you write code documentation that i) outlines what each script is doing, ii) provides a list of sources used for my data, and iii) generates a dictionary of variable definitions?
Data inspection	• Attached is my data set and codebook in CSV format. Please carefully study it and tell me if there are any red flags, undocumented top coding or missing values, or other patterns I should be aware of.
Mapping	• I am sending you a CSV file with 1,000 observations, two variables, and latitude-longitude coordinates. Please go online, retrieve a U.S. county and U.S. state shapefile, then plot variable X at the state level and variable Y at the county level.
Non-standard econometric methods	• Here is a CSV file with geocoded observations and my variable of interest, Y, and time T. What network econometric methods are there to assess the geographic diffusion of Y over time? How can I plot a network graph?

Table A.2: Alphabetic Glossary of LLM Terms

Term	Definition
Agent / Agentic system	An LLM equipped with tools (file system access, code execution, web browsing, database queries) and the ability to take multi-step actions to complete a task. Distinct from chat-based interaction, where the model only produces text and the researcher executes any actions. *Shifts more autonomy to the model and correspondingly raises validation requirements. Common implementations include Claude Code, Codex, and IDE-integrated tools like Cursor.*
API (Application Programming Interface)	The programmatic interface through which a model is accessed, as distinct from the chat window. *Enables automation, batch processing, structured outputs, and integration with research workflows. Most large-scale empirical work requires API access rather than the chat interface.*
Batch API	A discounted, asynchronous mode of API access in which many requests are submitted together and returned within a longer time window, typically 24 hours. *Substantially cheaper than real-time API calls and suitable for large extraction tasks where latency does not matter.*
Caching (prompt caching)	A feature offered by some vendors that stores parts of a prompt across calls, reducing cost for repeated tasks that share a common preamble. *Relevant for high-volume extraction tasks where the same instructions are applied to many inputs.*
Chain-of-thought prompting	Instructing the model to articulate its reasoning step by step before producing a final answer. *Often improves accuracy on tasks requiring multi-step reasoning, at the cost of additional tokens.*
Closed model / closed-weight model	A model whose parameters are not publicly available and which can only be accessed through a vendor's API or interface (GPT-4, Claude, Gemini). Contrasts with open-weight models.
Context window	The amount of text, measured in tokens, the model can consider at once. Includes the system prompt, conversation history, any documents provided, and the model's own response. *Performance often degrades as the context window fills up, motivating the practice of starting fresh sessions for distinct tasks.*
Embedding	A numerical vector representation of text (or images, audio) that captures semantic content. Two pieces of text with similar meaning have embeddings that are close together in vector space. *Enables similarity search, clustering, and measurement tasks distinct from classification; useful, for example, for tracking how the meaning of a word shifts across decades of historical text.*
Few-shot prompting	Providing examples of input-output pairs within the prompt itself, before asking the model to perform the task on new input. *Often dramatically improves accuracy without requiring any model training. Contrasts with zero-shot prompting.*

(Table A.2 continued)

Term	Definition
Fine-tuning	Further training of a pre-existing model on a researcher's own labeled data to specialize it for a particular task. Distinct from prompting, which uses an off-the-shelf model. *Worthwhile for high-volume specialized tasks where prompting reaches a quality ceiling, but requires technical setup and labeled training data.*
Frontier model	The most capable model from a major vendor at a given time. *Costs substantially more per token than mid-tier alternatives and may be overkill for routine tasks. The frontier shifts every few months as vendors release new models.*
Hallucination	Output that is plausible-sounding but factually wrong: fabricated citations, invented historical facts, non-existent sources. *The dominant failure mode for knowledge-retrieval tasks. Particularly insidious in historical work because errors do not have verifiable referents the way fake citations do.*
Inference	The act of running a trained model to generate output, as distinct from training the model in the first place. *Most practitioners only do inference; training is the vendor's job. Pricing is denominated in inference cost.*
MCP (Model Context Protocol)	An emerging open standard for connecting LLMs to external tools, data sources, and services. *Relevant for researchers building or using agentic systems that connect to databases, archives, or analysis software.*
Model card	Vendor-published documentation describing a model's capabilities, training data, intended uses, and known limitations. *The first place to look when evaluating whether a model is suitable for a task.*
Model deprecation	Vendor retirement of older model versions, after which they can no longer be called. *A central reproducibility concern: a paper relying on a specific model version may become impossible to replicate when the vendor discontinues that version. Motivates archiving raw model outputs as part of the replication package.*
Model version	The specific identifier of a model snapshot (e.g., `gpt-4o-2024-05-13`, `claude-opus-4-7`). *The same nominal model name often refers to different underlying models over time. Reproducibility requires recording the exact version used, not just the model family.*
Multimodal model	A model that accepts inputs beyond text; typically images, increasingly audio, and in some cases video. *Necessary for historical sources that are not purely textual, including handwritten ledgers, maps, photographs, and audio recordings.*

(Table A.2 continued)

Term	Definition
Open-weight model	A model whose parameters are publicly downloadable and can be run locally (the Llama, Qwen, and Mistral families). Distinct from "open-source" in the strict sense, as training data and code are often not released even when weights are. *Matters for data sensitivity (no data leaves the researcher's system), reproducibility (models can be preserved indefinitely), and cost (no per-token charges, though hardware costs apply).*
Prompt	The instruction given to the model. Practitioners often use this loosely to mean both the user-facing query and the full input including system instructions, examples, and data. *Prompt engineering is the practice of refining prompts to improve output quality.*
Prompt injection	A security concern in which untrusted input (e.g., text scraped from a document or webpage) contains instructions that override the researcher's intended prompt. *Relevant when LLMs process external content the researcher has not vetted.*
RAG (Retrieval-Augmented Generation)	Augmenting model inputs by first retrieving relevant documents from an external corpus and then including them in the prompt. *The standard solution for grounding a model in a specific archival or domain-specific corpus when training or fine-tuning is not feasible.*
RLHF (Reinforcement Learning from Human Feedback)	A training procedure in which human evaluators rank model outputs, and the model is adjusted to produce more highly ranked responses. *Explains some of the model's stylistic tendencies (hedging, refusing certain requests, producing balanced answers) that may not be optimal for research use.*
Seed	A parameter that, when supported and combined with low temperature, makes model outputs more reproducible across runs. *Imperfect: even with a fixed seed, model outputs may not be perfectly deterministic due to infrastructure-level non-determinism.*
Structured output	Constraining the model to return data in a specific format, typically JSON conforming to a researcher-defined schema. Sometimes implemented via "function calling" or "tool use." *Essential for extraction tasks because it eliminates parsing failures.*
System prompt	Instructions given to the model that condition its behavior across an entire conversation, usually invisible to the chat-window user but explicit when using the API. *Can substantially alter output style, format, and content; should be logged for reproducibility.*
Temperature	A parameter controlling randomness in the model's output. Low temperature (near 0) produces more deterministic, repeatable output; high temperature produces more varied output. *For research measurement, low temperature is usually preferred.*

(Table A.2 continued)

Term	Definition
Token	The unit in which models count input and output, roughly corresponding to a short word or word fragment. *Both context-window limits and pricing are denominated in tokens. As a rough rule, English text averages about 0.75 words per token, or roughly 1.3 tokens per word.*
Tool use / function calling	A capability in which the model decides when to invoke researcher-defined functions (running code, querying a database, calling another API) and how to use their outputs. *The underlying mechanism for agentic systems.*
Training cutoff	The date after which events are not represented in the model's training data. *Relevant for prompts about recent events, but also for historical work because models may have weaker knowledge about the deep past relative to recent decades.*
Zero-shot prompting	Asking the model to perform a task without providing any examples of input-output pairs in the prompt. Contrasts with few-shot prompting. *Zero-shot performance is the relevant baseline for evaluating whether providing examples actually helps.*

Table A.3: Replication Package Checklist for LLM-Assisted Research

Component	Items
Paper and appendix	☐ Exact model identifier(s) (e.g., `gpt-4o-2024-05-13`) and date(s) of access.
	☐ Settings that affect output: temperature, top-p, the maximum output-token limit (e.g., `max_tokens`), seed, reasoning or thinking mode, and output format.
	☐ Validation sample size, how and by whom it was hand-coded, and accuracy on the held-out portion (precision and recall for rare categories).
	☐ How licensed or otherwise sensitive data were handled (e.g., enterprise version with zero data retention, or a self-hosted open-weight model).
	☐ Which parts of the analysis and code were LLM-assisted, following the target journal's generative-AI disclosure policy.
Prompts	☐ Full text of every prompt in its final form, including system prompts, classification rubrics, few-shot examples, and period vocabulary lists.
	☐ Prompt variants used for robustness checks, if any.
	☐ The project log (e.g., `AGENTS.md`) if one coordinated the work.
Code	☐ Pre-processing code (corpus restriction, keyword screens, text windows, stop-word removal).
	☐ API-call or batch-submission scripts, including the code that parses model responses into data files.
	☐ Analysis code from the LLM-generated measure to every table and figure in the paper.
	☐ The software environment: language and package versions (e.g., a requirements file), with hardware notes where relevant (e.g., GPU use).
	☐ A README with the order of execution, approximate runtimes and API costs, and a mapping from scripts and output files to the paper's tables and figures.
Data and model outputs	☐ The archived raw model outputs for every call used in the paper. This is the single most important item, since rerunning the prompts is not guaranteed to reproduce them.
	☐ The hand-coded validation data together with the coding instructions given to the coders.
	☐ The final analysis data sets built from the raw outputs.
	☐ For restricted source data, access instructions in place of the data themselves. Do not include licensed material in the package.
	☐ For fine-tuned or self-hosted models, the weights or a pointer to the archived version (e.g., a Hugging Face identifier), plus the fine-tuning data and code where applicable.
Final check	☐ A reader without access to the model vendor can reproduce every number in the paper from the archived files alone.
	☐ A note states which steps cannot be rerun exactly (model nondeterminism and deprecation) and what is archived to compensate.

Notes: Checklist for assembling a replication package for research that uses LLM-generated measures, data, or code. Items condense Sections 3.3.2 and 3.3.3; the validation items are described in Section 3.3.1 and the data-sensitivity routes in Section 3.1.2. Not every item applies to every project.

Figure A.1: A Practitioner's Workflow for Using LLMs in Economic History (Section 3)