Saving Time, Saving Money

Or, how to do metadata curation using LLMs locally.

Aerith Y. Netzer,

January 1, 2025

Abstract

In the past century, the well-documented consolidation of the academic publishing industry created economic conditions that incentivize universities to publish their own academic journals. While these journals are necessary and good, they are not as well-funded or well-staffed as corporate players. These preconditions necessitate university libraries and presses to use all resources at their disposal to make their publishing operations run at peak efficiency. A large opportunity cost incurred by the university-owned publisher is machine-actionable metadata creation, leading to a cropping of for- and non-profit players in the industry to provide these services. We present three insights [there is a better word than insights here] - that reference metadata curation using LLMs is possible, that we can run these LLMs on local hardware, and we end with some tutorials for replication at other university-owned presses.

Background and Motivation

University-owned journal-publishing operations operate under far tighter economic constraints — direct and opportunity — and therefore must solve the same problems of corporate academic publishers with a fraction of the resources available. One of these problems is reference metadata, i.e. machine-actionable references that are then used to count citations of articles. The act of capturing, counting, and using citations accurately allows for funding agencies, universities, and publishers to make data-driven decisions for funding allocation, allows for reviewers to validate the research of a manuscript, and allows for faster literature review. Here, we evaluate the use of local large language models to curate the metadata with minimal human intervention.

An Example

The workflow for our university—a medium-size, elite university in the Mid-west United States—consists of receiving manuscripts from authors in a Microsoft Word file format. We then use pandoc [@PandocIndex] to transform this Word document to a markdown file format, from which we can build PDF and Web versions from a single source. But due to author unwillingness to use plaintext markup formats such as LaTeX or Markdown, we must recreate the bibliography. Previously, this meant looking up each source, adding them to a Zotero[@ZoteroYourPersonal] library, and then exporting the biblatex file for use as metadata in the web version of the article. This would allow for services such as Google Scholar and Web of Science to scrape the metadata and count citations for the cited articles. This allows for researchers conducting literature reviews to find articles easier and faster, and allows for easier cross-checking for dubious claims. The present system, can automate this labor-intensive machine-actionable metadata creation process. With the advent of Large Language Models (LLMs), we can create systems to parse out the plaintext citations in an article, pass it to a Large Language Model, and output a machine-actionable metadata citation entry.

Limitations and Concerns

This analysis is — by necessity — is limited to works that appear in the crossref API, creating a bias in the dataset against older works and academic monographs. While this limits the usefulness of this analysis to publishers whose specialty lies within fields where citations are limited to recent works (such as the physical sciences), future work can and should include plain-text citations of historical, non-digital, and non-academic works.

Along with the rapid growth in users of Large-Language models, so have concerns over the ecological sustainability of LLM technology.[@dingSustainableLLMServing2024] [@chienReducingCarbonImpact2023] Most of these concerns, however, can be alleviated with the use of "small" models such as those provided by Ollama. Further, there are concerns about the validity of Large-Language models, especially concerning their propensity to hallucinate. However, in combination with validity checkers such as bibtexparser and human review, we are confident enough in this system to be used in production of our journals.[@JournalBulletinApplied] Future work in this area should include building scalable, verifiable workflows that do not necessitate human oversight.

Methodology

Data Collection

Data was collected via the CrossRef API @bartellRESTAPI. We sampled a DOI and randomly selected a plain-text citation style from the following list:

1. Chicago Author-Date
2. Elsevier-Harvard
3. Ecoscience
4. APA
5. MLA
6. IEEE
7. Council of Science Editors

Using the CrossRef API, we pulled the BibTex Citation, the Plaintext Citation, and the DOI to create a dataset for our analysis. [Table 1] presents the variables and their descriptions.

Table 2: Citation Metadata

Variable	Description
DOI	The Digital Object Identifier of the requisite work.
BibTeX Citation	Metadata about the work in BibTeX format.
Plain Text Citation	The cited work in a given citation style.
Plain Text Citation Style	The style in which the plain text citation is given.

To fairly analyze the performance of each language model on a broad set of citation formats, each variable was randomly assigned one of the above citation styles and returned by the CrossRef content negotiation API endpoint. Using this random assignment of citation formats, we achieved a roughly even distribution of each citation style in the dataset.

Analysis

All language models were tested using the Ollama@OllamaOllama2025 toolkit using the Quest@QuestHighPerformanceComputin supercomputing cluster at Northwestern University, running in a singularity container.@kurtzerSingularityScientificContainers20 Testing of all models took 14 hours to complete on two NVIDIA A100 Graphical Processing Units, one node with eight cores, and 128 gigabytes of memory.[1] [2] All code was written in python using an Anaconda environment to aid in reproducible deployments of this code.

We used the plain text citation given by the CrossRef API as a ground truth to which the model would aspire to. We prompted each model with the same text:

> You are a professional citation parser.
>
> Given the following plain text citation:

[1]The script used to create the SLURM job can be found in the GitHub repository mentioned in the "Data and Code Availability" section.

[2]Thank you to Kat Nykiel at Purdue University for her assistance in building and deploying the singularity container.

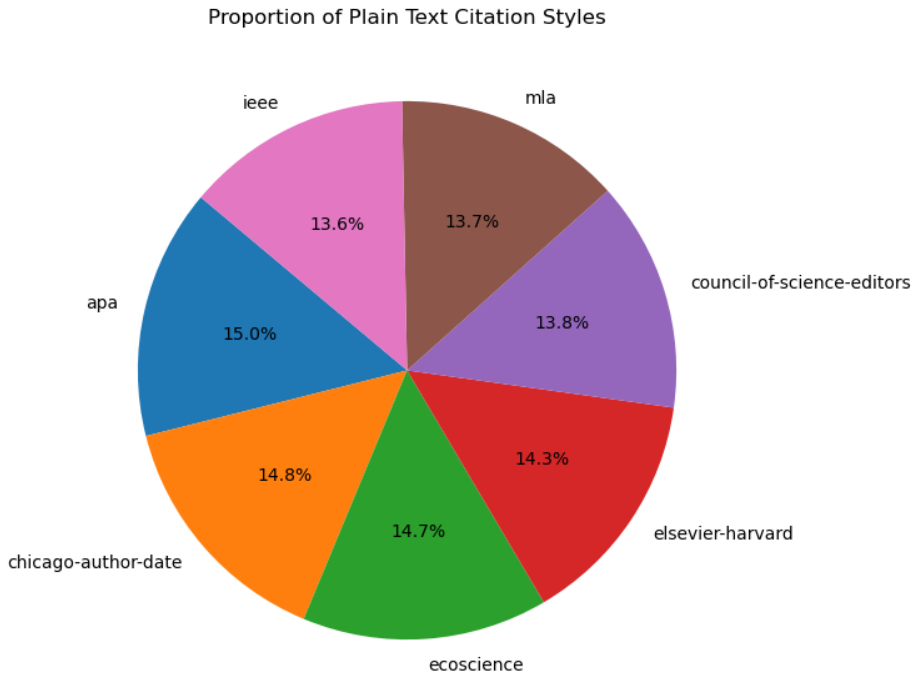


Figure 1: Pie Chart demonstrating the proportion of each citation style present in the dataset

{plain_text_citation}

Please convert this citation into a structured BibTeX entry. Include all relevant fields such as author, title, journal, volume, pages, year, etc.

Output only the BibTeX entry, and nothing else. Do not include any explanations, preambles, or additional text.

The following models were prompted:

codegemma:2b [@codegemma_2024]
codegemma:7b [@codegemma_2024]
llama2:7b [@touvronLlama2Open2023]
llama3.3:70b [@llama3modelcard]
llama3:8b [@llama3modelcard]
mistral [@Mistral]
starcoder2:3b [@li2023starcoder]
tinyllama [@zhang2024tinyllama]

The following variables were saved to the output file of the model:

Variable	Description
Model	The model being tested. One of the eight models listed above.
PlainTextCitation	Maps to plain text citation field table 1.
TimeToGeneration	Time taken to generate the entry.
ActualBibTeX	BibTeX entry retrieved from Crossref.
TotalFields	The number of BibTeX fields being compared in generated and "ground truth" entries.
Matching Fields	The number of fields that have a match in both the generated and "ground truth" entries.
Percentage Match	$\frac{\text{Total Fields}}{\text{Matching Fields}}$

This generated 8 CSV files of approximately 3,000 lines each. Each row corresponds to a single DOI. These files were used for analyzing the efficiency and effectiveness of each model.

Results

Model Effectiveness

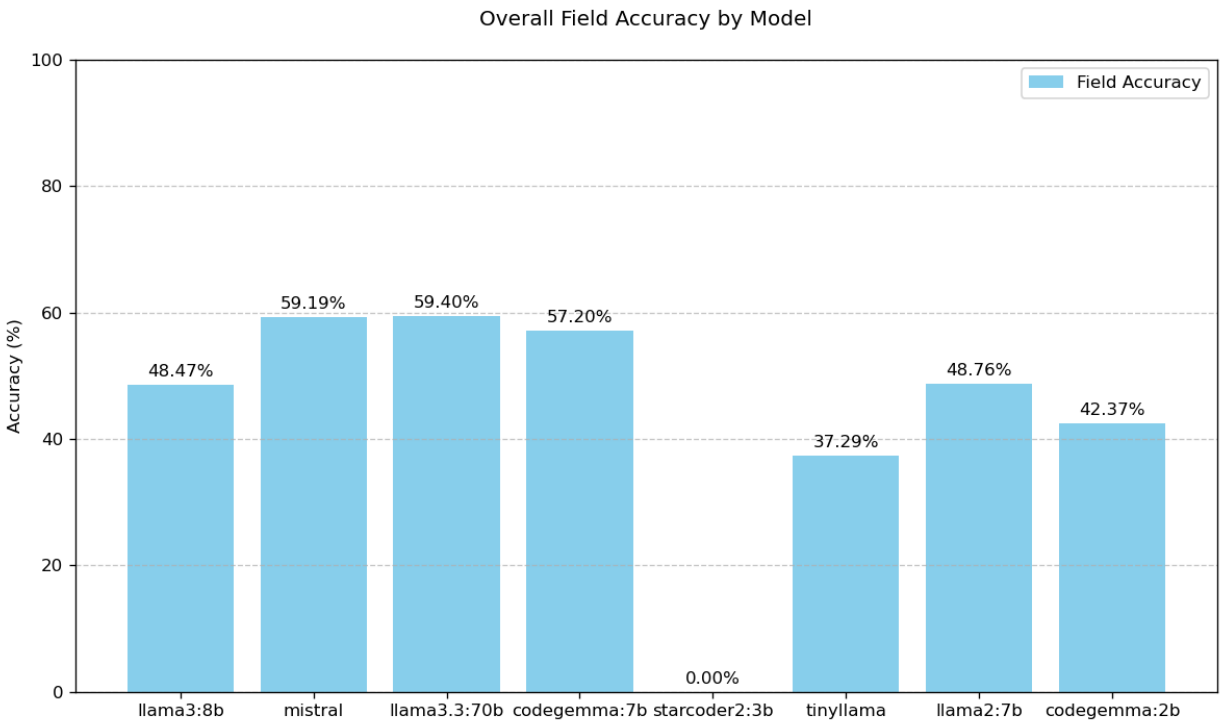
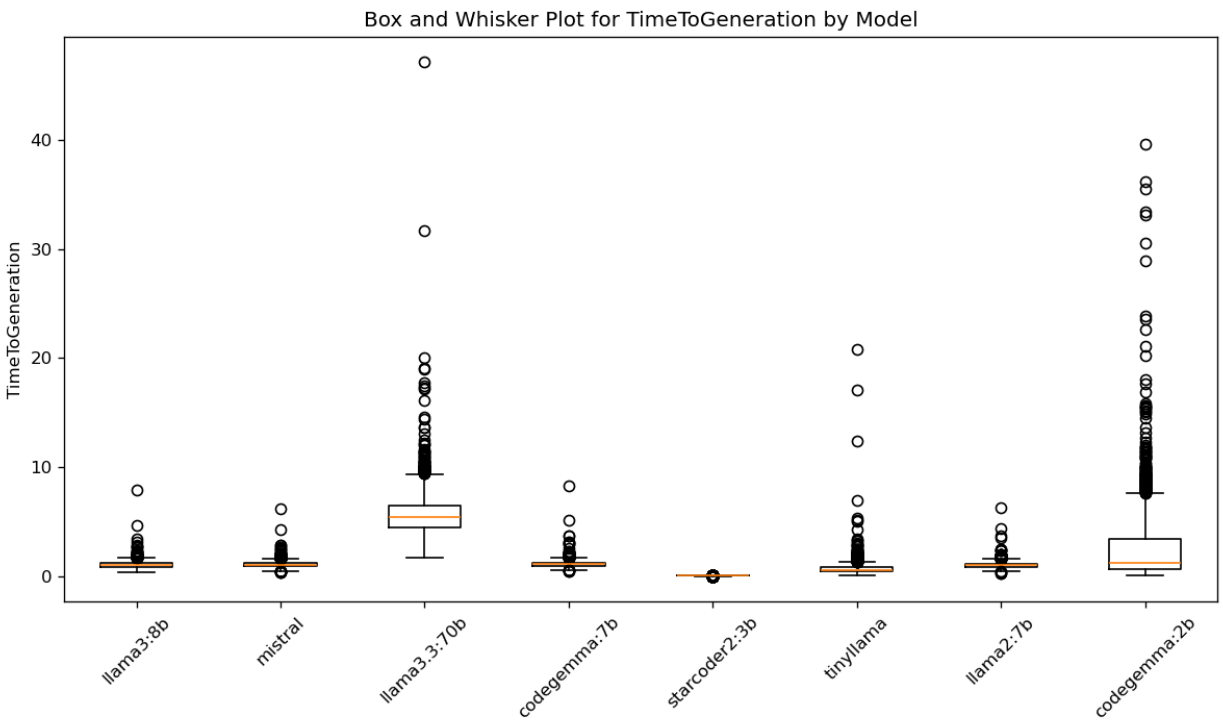


Figure 2: Per Field Accuracy and Valid JSON of the Model

Unsurprisingly, llama3.3:70b, the most advanced and largest model of the chosen models, performed the best. Further, starcoder2:3b failed to create any valid BibTeX entries, whereas every other model created valid BibTeX for every citation.


Box and Whisker Plot for TimeToGeneration by Model
TimeToGeneration
0
10
20
30
40
llama3:8b
mistral
llama3.3:70b
codegemma:7b
starcoder2:3b
tinyllama
llama2:7b
codegemma:2b

Model	Median Time to Generation (seconds)	Standard Deviation (seconds)
codegemma:2b	1.18	3.24
codegemma:7b	1.08	0.29
llama2:7b	0.96	0.27
llama3.3:70b	5.45	1.90
llama3:8b	1.00	0.31
mistral	1.00	0.27
starcoder2:3b	0.00	0.00
tinyllama	0.57	0.64

As every model except for starcoder2:3b created valid BibTeX perfectly, we are primarily concerned with and the accuracy of the fields. In this discussion, the *validity* of the BibTeX simply means that if the BibTeX can be parsed without errors, then the BibTeX is valid. However, a well-formed BibTeX entry can be *valid* but *incorrect*. Meaning that the entry can be parsed, but the data in the entry is wrong. But llama3.3:70b generated the most *accurate* BibTeX entries. However, we should not take this that the model was necessarily *incorrect*, but was just different from how Crossref represented the field. Mistral and Codegemma, though, are very close behind, especially with their parameter sizes (and therefore cost of compute) much lower than llama3.3:70b, it may be economical for some publishing operations to use smaller models, decreasing their cost, while keeping parity with the accuracy of the model. Trading a .2% reduction in accuracy for, on average, a 5x faster computation is an effective strategy for this use case.

Per-Field Accuracy

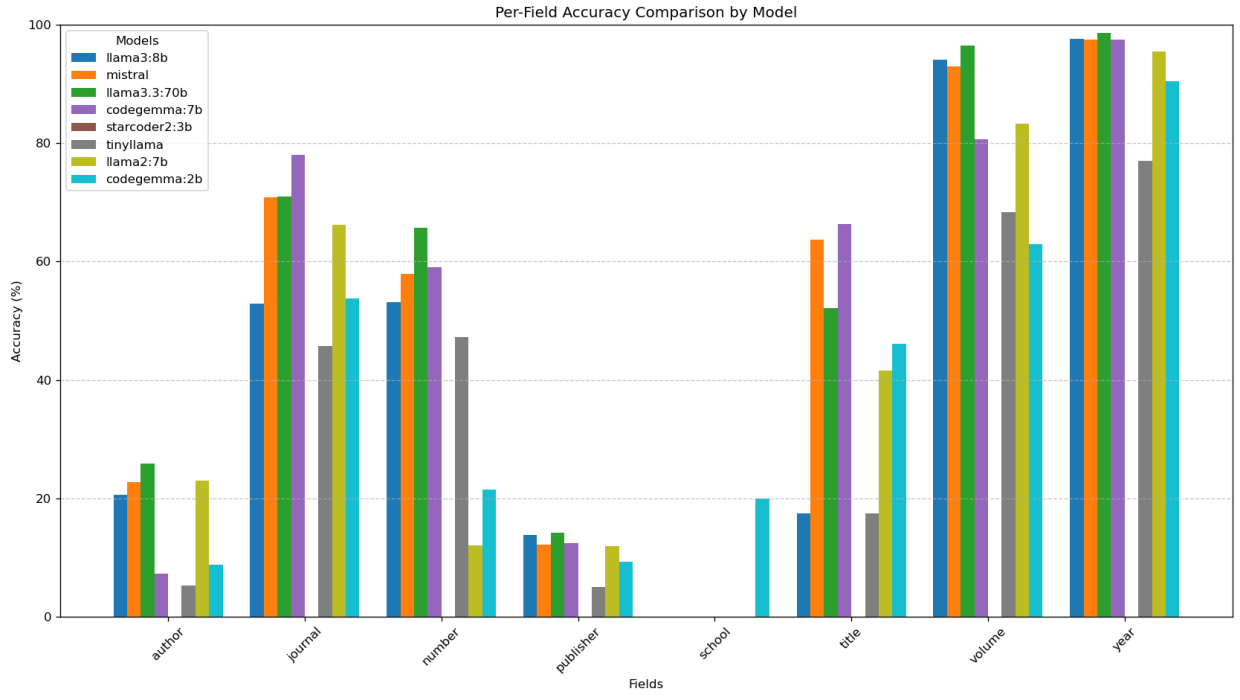


Figure 3: Per-field accuracy by model

Data and Code Availability

Bibliography