Accepted for publication in the International Journal for Humanities and Arts Computing

Cost-Effective Machine Learning for Automatically Processing Bibliographic Metadata

Samuel J. Huskey (University of Oklahoma)

Abstract

Many Digital Humanities projects involve tedious and repetitive tasks take time away from the higher-level tasks further down the pipeline that require intelligent decision-making. When funding is available, the tedious and repetitive tasks are often assigned to research assistants, but when funding is scarce, those tasks tend to create bottlenecks that either impede progress or halt it altogether. This paper argues that artificial intelligence and machine learning tools and techniques are worth exploring as cost-effective, accessible solutions to these problems. The Digital Latin Library project provides a case study through its experiments with fine-tuning pretrained transformer language models to process noisy bibliographic metadata. The results show that the models have potential for accelerating this tedious task. But the experiments also had an unexpected, yet positive outcome: the models revealed gaps in the catalog's coverage, helping to focus the efforts of the human experts working on the project.

Keywords: Artificial Intelligence, Machine Learning, workflow automation, metadata

1. Introduction

One of the goals of the Digital Latin Library project (DLL) is to catalog all editions of Latin texts available in any digital format on the internet.[1] During the grant-funded planning and implementation stages of the project, a team of Latinists, librarians, and computer scientists developed a metadata model for cataloging editions, created authority and work records, and launched a robust content management system (CMS) for working with the data.[2] Team members also built custom web scrapers and crawlers to seek out and retrieve metadata for Latin texts, allowing us rapidly to accumulate tens of thousands of records of texts in different formats. The project was awash in data, and the future looked bright for the DLL Catalog.[3]

And then the grant period came to an end, along with funding for student workers, graduate assistants, developers, consultants, and other team members.[4] Until the next successful grant application, the DLL, like so many DH initiatives, would have to proceed with a skeleton crew (i.e., me) and existing resources. Consequently, processing all the accumulated metadata, let alone any additional metadata the project might acquire, and adding it to the catalog took lower priority to more existential issues for the project, like migrating from an outdated server to a new one and staying on top of updates and upgrades to the catalog's CMS—not to mention managing the DLL's other major initiative of publishing born-digital critical editions in partnership with three learned societies through the *Library of Digital Latin Texts*.[5] Even with adequate time and funding, processing those metadata records would not be at the top of the agenda, because the remaining tasks for processing the metadata, necessary though they may be, are tedious, repetitive, and boring. But that is precisely where Artificial Intelligence and Machine Learning (AI/ML) technology is supposed to excel, so it seemed worthwhile to find out if it could be a cost-effective method for alleviating the project's metadata bottlenecks.

This report describes the techniques, procedures, and resources I have used to fine-tune a pretrained transformer language model and to harness other AI/ML tools to assist with reconciling the bibliographical metadata for thousands of editions of Latin texts with the authority and work records in the DLL Catalog. Although some challenges related to the Latin language are unique to this project, normalizing large and noisy bibliographic metadata is a significant, complex problem common to most disciplines. Accordingly, it seems beneficial to share the early results of these machine learning experiments, particularly in a special issue devoted to AI and the Digital Humanities. It also seems timely to report on potentially cost-effective solutions for Digital Humanities projects when funding is scarce.

If the tone of this piece seems casual, that is deliberate. I want to reduce the intimidation factor that many experience with technical topics like machine learning. There is no doubt that the technical aspects of machine learning are intimidating, but I hope to demonstrate that developments in tools and resources make it possible for amateurs like me to achieve good results. Second, writing in the first person is also a way of ensuring that the human in this project is not just 'in the loop' but in control of how AI/ML technology is used as a tool for research in the humanities.

2. Background

Despite the widespread adoption of standards and ontologies for bibliographic metadata, their implementation necessarily reflects the languages and cultures of individuals and organizations.[6] In particular, the various forms of authors' names and the titles of their works found in catalogs and collections pose significant problems when it comes to reconciling individual items with local authority and work records. The Roman poet known in Latin as Publius Vergilius Maro is a good example. Not only is he known in English as both 'Vergil' and 'Virgil', a glance at his record in the Virtual International Authority File (VIAF) shows 36 distinct variations of his name in use at international research libraries.[7] The titles of his

works are similarly varied. In Latin, the titles *Bucolica* and *Eclogae* are used interchangeably to refer to his collection of ten pastoral poems known in English as both *Bucolics* and *Eclogues*.[8] Complicating the issue is the fact that Calpurnius Siculus, Nemesianus, Petrarch, and Boccaccio (each with their own multiple variant name forms) wrote collections of poems in Latin with the same or similar titles.

But individual editions of these works almost never bear simple titles like *Bucolica* or *Eclogae*. For example, Calpurnius Siculus and Nemesianus' pastoral poems often appear together in volumes with such elaborate titles as *M. Aurelii Olympii Nemesiani Eclogae IV et T. Calpurnii Siculi Eclogae VII ad Nemesianum Carthaginiensem, cum notis selectis Titii, Martelli, Ulitii, et Petri Burmanni integris* (Anonymous 1774), or more obscure titles like *Venatici et Bucolici Poetae Latini: Gratius, Nemesianus, Calpurnius* (Barth 1613), or simply *Calpurnii et Nemesiani Bucolica* (Schenkl 1885). They also appear in collections with the catch-all title *Poetae Latini Minores* (e.g., Baehrens 1881). Indeed, editions of most authors' works have similarly generic titles like *Opera Omnia* ('Complete Works') or *Carmina* ('Poems').

Disambiguation of similar or even identical titles can be achieved by associating individual records with a catalog's authority and work records. Each entity in the DLL Catalog has a unique identifier (e.g., A4830 for Virgil; W3894 for his *Eclogae*) to aid in the disambiguation. Matching individual editions to these entities ensures that they will be included in the DLL Catalog's author pages, which aggregate all records associated with a particular author.[9] It also increases the likelihood that records will be among the results of a faceted search based on an author's name or the title of a work.[10] In short, reconciling new records with authority and work records is an essential step in fulfilling the DLL Catalog's mission as a finding aid for Latin texts.

Reconciling one or two records does not take much time; manually reconciling thousands and thousands of records, however, is a Sisyphean task. Even if the project had

funding for a team of research assistants to do the work, it is worth asking whether such mechanical, repetitive work is a good use of human effort. The work of research assistants should be meaningful for them; ideally, they will be collaborators, not mere data-entry workers. Second, the task is boring and repetitive, and bored workers are more prone to making mistakes. In this case, their mistakes and errors can cause records to appear with the wrong results or, worse, to disappear from the search index entirely.[11] Finally, the variety and complexity of some records requires more scholarly expertise and judgment than any one research assistant is likely to have.

That is why my guiding principle has been to 'automate the boring stuff,' bearing in mind that computers excel at repetitive, rule-based tasks.[12] In the past, when I considered automating this reconciliation process, I experimented with deterministic and probabilistic (i.e., "fuzzy") matching techniques, but the results were underwhelming. Those algorithms could accurately reconcile the obvious matches, but so could I. Instead, I wanted something that would reliably and correctly process the thousands of records that were not obvious matches. Recent advances in AI/ML technology led me to try new experiments. The rest of this article will detail my approach and the results I was able to achieve.

3. Assumptions

If a human can see that 'Ovid' and 'Publius Ovidius Naso' refer to the same author, then a machine should be able to recognize that relationship, too. Similarly, a machine should be able to distinguish Vergil's (or Virgil's) *Bucolica* from Calpurnius Siculus' *Bucolica*, if it has access to sufficient data for disambiguation purposes. The essence of machine learning, after all, is identifying patterns in data.

My assumptions, then, are that a machine learning model can be fine-tuned to identify variations of authors' names and titles of works in metadata records from external catalogs with entities representing those authors and works in the DLL Catalog; that the

model will perform this task accurately and more rapidly than a human; and, finally, that the development and implementation of this model will be feasible without access to paid developers, license fees, or special equipment.

4. The Datasets

4.1. Training

Although much progress has been made with 'single-shot' and 'few-shot' training, in which an existing model accepts one or a small number of examples of the inputs and desired outputs and attempts to extrapolate a pattern and apply it to an entire dataset, the ideal is still to work with large amounts of pre-labeled data. 'Large' being a relative term, I started with the records the DLL Catalog had accumulated by the end of the most recent funding period: 3,233 authority records and 5,263 work records.[13] The next two subsections describe the steps to create sets sufficiently large and varied for the learning process.[14]

4.2. Author Reconciliation Training Set

Beginning with a CSV file with one row per author and one column each for every variant name form and identifier, I reshaped the data so that each row contained one variant form, the corresponding authorized form, and the DLL ID, expanding the original 3,233 rows to 27,290. The DLL Catalog team had made the decision to include in the authority records only a subset of the variant name forms available in VIAF records and other sources, so an obvious next step was to scrape the unused variant name forms from VIAF.[15] Since some authors (e.g., Virgil) naturally have far more variant name forms than others (e.g., Giovanni Battista Vico), I achieved a more even distribution by programmatically creating some artificial variants for authors with fewer than eight genuine variants.[16] In addition to evening out the numbers of rows per DLL ID, the artificial variants also enhanced the model's ability to account for misspellings in new, unseen data. In an extreme example, this helped the

model correctly identify ‘Ta Cito Cayo Cornelio’ (an actual misspelling in the test data) as ‘Cornelius Tacitus’. Augmenting the data also corrected the model’s original tendency to identify Pliny the Elder as Pliny the Younger. The final augmented dataset for training the author reconciliation model contained 34,876 rows of variant name/DLL ID pairings.

4.3. Greek-Latin Classification Training Set

Preliminary examination of the test data (see below) revealed the presence of numerous editions of works in languages that do not use the Latin alphabet, but that bear titles translated into Latin to facilitate discovery in catalogs.[17] The majority of these were editions of ancient Greek texts, so I decided to fine-tune a separate model to winnow out as many of them as possible, since they would otherwise unnecessarily complicate the reconciliation process.

I assembled a list of authors of ancient Greek texts from data publicly available from the Thesaurus Linguae Graecae (TLG).[18] A database kept by the Perseus Catalog provided the VIAF ID numbers for most of these authors.[19] I used the VIAF ID numbers to scrape variant name forms from VIAF records, with one author (identified by VIAF ID and DLL ID) per row and one variant per column. Finally, I reshaped the rows to produce a dataset with two columns, ‘Name’ and ‘Label’, with one name form per row. The value in the ‘Label’ column for the Greek authors was, of course, ‘Greek’.

The author reconciliation dataset (described above) provided records labeled ‘Latin’ to yield a dataset of 40,458 author names labeled with either ‘Greek’ or ‘Latin’. Latin authors outnumbered Greek authors by more than two to one, but since the number of authentic Latin authors will normally be greater than the number of Greek authors in any collection of metadata records to be processed for the DLL, the imbalance was acceptable.[20]

5. Equipment

For cleaning and formatting the datasets, I worked on the 2024 MacBook Pro laptop provided to me by my home institution, the University of Oklahoma. Although my computer's M4 Pro GPU, 24GB RAM, and storage capacity are sufficient for fine-tuning runs, I also purchased a Google Colab Pro subscription ($9.99 per month) to have access to a hardware accelerator emulating the more commonly available NVIDIA A100 Tensor Core GPU. I also paid for a subscription to OpenAI's ChatGPT Plus ($20 per month) for hints and help with coding (see below, 'Understandability and Reproducibility'). I used the freely available Microsoft product VS Code for running Python scripts and performing basic data cleaning and preparation.

6. Models

My first experiments with basic recurrent neural network (RNN) architectures were disappointing.[21] Their accuracy probably could have been improved with more time, development, and expertise, but the point of this experiment was not to develop more effective RNN's, but to leverage existing AI/ML technology to save time and effort. Moreover, transformer models have become the standard for text-to-text and text-classification applications, owing to both their relative simplicity and their superior performance according to standard benchmarking. Thus, switching to fine-tuning a pre-trained transformer model made sense.

The model I selected is a descendant of the original Bidirectional Encoder Representations from Transformers (BERT) language representation model.[22] BERT has enjoyed wide-spread adoption in AI/ML research and development. As of this writing, Hugging Face lists over 35,000 repositories devoted to BERT or variants of it.[23] The DistilBERT base multilingual (cased) model seemed to be well-suited for this project for three reasons.[24] First, it was trained not just on English, but on 104 languages, including

Latin.[25] Since much of the information in the test dataset for this project is in languages other than English, having a model capable of handling different languages was of utmost importance, which is why I did not select the Latin-only LatinBERT model.[26] Second, as the name indicates, the model has been 'distilled' from the larger BERT model so that it is not only smaller and faster, but also cheaper to pre-train.[27] All three of those comparative adjectives are attractive when resources are scarce. The third reason is that the BERT models are openly available, well-documented, and easy to fine-tune because the Hugging Face API has pipelines specifically for them.[28]

7. Fine-tuning

Although BERT models are pre-trained, they must be fine-tuned for specific tasks. For this project, I fine-tuned two instances: one for identifying authors as 'Greek' or 'Latin' (a binary classification task); and another for reconciling variant author names with their authorized name forms (a multi-class classification task).

Thanks to the Hugging Face API, the code for the former is not much different from the code for the latter. The Hugging Face's 'Trainer' class automatically adjusts for a binary or a multi-class classification task according to the number of labels in the dataset. I also used the 'train_test_split()' method in Hugging Face's 'Datasets' class to divide the datasets into training (70 per cent), validation (10 per cent), and testing (20 per cent) sets. To ensure even distribution of the labels in both instances, I also used the 'stratify_by_column' parameter, and I used the 'seed' parameter to promote reproducibility. To save time and energy, I deployed the 'EarlyStoppingCallback' method from the Hugging Face 'Transformers' class to halt training after three epochs with no improvement in the assessment metrics, which were computed using Scikit-Learn's 'metrics' package.[29]

8. Results

I ran the fine-tuning scripts for both the Author Reconciliation and the Greek-Latin Identification models multiple times so that I could compare the results over several runs. The Author Reconciliation model achieved an average f1 score of 0.9518 and an average accuracy score of 0.9529 in training and, respectively, 0.9475 and 0.9489 in evaluation. The Greek-Latin Identification model achieved an average f1 score of 0.9659 and an average accuracy score of 0.9650 in training, with identical average f1 and accuracy scores of 0.9677 in evaluation.

The slightly better scores of the Greek-Latin Identification model can be explained by the fact that its binary classification task was far simpler than the Author Reconciliation model's multiclass classification task. Ideally, each run would have the same results, but even though I controlled as many variables as possible, some variation in results is to be expected across platforms, versions, equipment, and, as in this case, training runs in a single environment.[30] I describe below ('Understandability and Reproducibility') what I have done to mitigate this problem.

9. Testing

Of course, the real test, is in how well the models perform on data from an external source. Immediately after training, I tested the Author Reconciliation model's inferencing ability on a small sample of three authors (Julius Caesar, Marcus Tullius Cicero, and Livy). It assigned the correct DLL ID's to each. Similarly, I submitted some Greek and Latin author names (Juvenal, Unknown, Caesar, Cicero, Homer, Hesiod, Aristotle) to the Greek-Latin Identification model. It assigned the label 'Greek' or 'Latin' correctly in each case.

For more extensive testing, I downloaded records from the HathiTrust Digital Library (https://www.hathitrust.org/), which holds over 18 million digitized items, many in the public domain. Several languages and cataloging methods are represented in the collection, which

makes it ideal for testing models designed to reconcile variant names of authors and titles with the canonical records in the DLL's catalog. To obtain a diverse enough sample while also limiting the results to works primarily written in Latin, I searched for items of format 'Book' in language 'Latin' with forms of *liber*, *carmen*, or *opera* in the title field. That yielded 24,799 records. Preprocessing of the test data to remove duplicates reduced the number of test records to 14,923.[31]

9.1. Greek-Latin Identification

I tested the Greek-Latin Identification model first since it is the first step in the pipeline for processing metadata records. It reliably and accurately filtered out the Greek authors from the test data, and it did so at a rapid rate, finishing its run in 134 seconds. My manual review of the output showed that it accurately removed 1,477 rows from the original 14,923. Since I want to be sure not to omit any Latin translations of Greek works, the records removed during this step are saved for manual review. In general, this model will always err on the side of caution and label records as 'Latin' to maximize the chances that a human will be the final judge when there is any uncertainty. Accordingly, the model outputs three CSV files: one each for records identified as 'Greek' or 'Latin' and one with all records combined.

9.2. Author Reconciliation

After the obviously Greek records had been removed, the next step in the pipeline is to reconcile the authors in the remaining records with the correct authority record in the DLL Catalog. The Author Reconciliation model completed its work in 132 seconds. I sorted the model's output according to the model's confidence score and manually reviewed each record—a painstaking but necessary test. Although the model did not have 100 per cent confidence about any of its inferences, my review found that it was reliably accurate—and more effective than the simple deterministic matching or fuzzy matching methods I

previously tried—when its confidence was greater than or equal to 99.9957 per cent, which was true for 3,876 out of 13,446 metadata records.

That result needs some context, since it is underwhelming at first glance. The number of unique author names in the 13,446 records is much lower: 4,779. The majority (4,016) of the unique author names appear only once in the test data. Indeed, my manual review shows that a substantial number of those authors are relatively obscure and thus less likely to have an authority record in the DLL Catalog. Without considering the large number of works attributed to ‘unknown’ (611), the most frequently occuring name forms (normalized by removing punctuation and capitalization) in the test data represent some of the most well-known authors in all of Latin literature: cicero marcus tullius (496 rows), horace (341), livy (297), virgil (252), tacitus cornelius (246). In contrast, the majority of the names with fewer rows are more obscure: nakatenus wilhelm, ludwich arthur, christ johann friedrich, each appearing in only two rows apiece.

Considering that only 1,472 unique names appear more than once in the dataset, the model’s successful identification of 615 unique authors, or 41.7 per cent, is still an underwhelming result, but it is better than the deterministic and probabilistic algorithms that I previously used, which identified 491 (33.4 per cent) and 526 (35.7 per cent) authors, respectively.

As for the 58.3 per cent of authors the model did not reliably identify, that says more about gaps in the DLL Catalog than it does about the model’s accuracy. Indeed, most of the authors not reconciled by the model were not reconciled for a simple reason: they do not have authority records in the DLL Catalog, so there is nothing for them to be reconciled *to*. The reason is simple: almost all individuals in this unreconciled group are authors of Medieval or Neo-Latin works, categories currently underrepresented in the catalog—and naturally so. The canon of Classical Latin authors is closed, so it is easy to have comprehensive coverage. But a comprehensive list of Medieval Latin authors is more

difficult to establish because there are so many of them; indeed, the list of Neo-Latin authors is asymptotic because they are not as well documented and because people are still writing and publishing works in Latin. In light of that, an unexpected, positive feature of the model is its ability to identify authors as candidates for potential addition to the catalog. After those authority records have been added and augmented with variant author name forms, the model will be retrained with a view to increasing its coverage.

9.3. Titles

Reconciling titles to the DLL Catalog's work records has proved to be a problem of an entirely different category, but one that might yield to another type of machine learning solution. Indeed, I have had some promising results using the LatinCy model to apply some Natural Language Processing methods to title strings.[32] Experiments with Retrieval Augmented Generative AI models have also lead me to think that they could become a useful part of this automation pipeline. In any case, since reconciling titles manually is one of the most time-consuming tasks, it will continue to be an area of active research. One thing is clear, however: the author reconciliation model's output will be an important factor, since the accurate identification of an author will help to limit the number of works represented in a given record's title field. That is, if the model identifies the author as Virgil, only Virgil's *Bucolica* (not Calpurnius Siculus' or Nemesianus') is a possible match for *Bucolica* in the record's title field.

10. Understandability and Reproducibility

As recent publications have demonstrated, it is of utmost importance not only for DH researchers to understand their own work, but also for that work to be available to other researchers for evaluation and replication.[33] To those ends, I have endeavored to adhere to both the so-called 'good enough practices in scientific computing' and guidelines specific to

AI/ML research, including making verbose comments in my code, using open-source software, tracking the project's development with a distributed version-control system (Git), and posting my code, datasets, and models in openly available repositories.[34]

I wrote 'my code' in the last paragraph, but that needs some explanation. A classicist by training, I am by no means an expert programmer, but I have been working with Python and other languages for the better part of two decades on various projects. I mention this background to introduce another way AI impacts the project under discussion here.

For the most part, I am a self-taught programmer. I read books on coding, consult the internet, attend workshops, and ask friends for advice.[35] The iterative, trial-and-error process of development and debugging often consumes time that I could spend writing the books, articles, and grant proposals that a classicist would normally be expected to write. Nevertheless, the effort has advanced various initiatives related to the DLL, so it has been time well spent.

Yet the gulf between writing procedural scripts in Python and developing machine learning models is vast, and I freely admit that I made use of chatbots and coding frameworks during this project to save time not only on low-level tasks and debugging, but also on design and implementation. I probably could have achieved the same results by spending more time trying different approaches and searching for hints and help on the internet. But that is not an efficient or good use of my time. After all, I am not a computer scientist or a professional software engineer. My project is about the application of existing AI/ML methods in a novel way to solve, or at least ameliorate, a problem common to DH research and scholarship: attending to tedious, repetitive tasks on a tight or non-existent budget.

That is also why I chose to use Hugging Face's API instead of writing custom code. Having experimented with Keras, Tensorflow, and PyTorch—all widely used for AI/ML development—I realized that I did not have the time to learn how to write custom code to

accomplish my objectives. Fortunately, Hugging Face's API includes code for standard AI/ML operations, along with many more specialized methods and functions. Indeed, one call to the API can activate dozens or even hundreds of lines of code written by experts in AI/ML, and the API's main classes have names that clearly explain their purpose, such as 'Logging', 'Tokenizer', 'Trainer', etc. Moreover, the API is well documented, and the code is verbosely and lucidly commented, formatted according to recognized standards, and available for inspection in version-controlled repositories, so I am confident that another researcher could reproduce my work.[36]

In short, I make no claim to originality for the code I have used in this project. Indeed, my use of AI chatbots (OpenAI's ChatGPT, Google's Gemini, and Anthropic's Claude) and frameworks like the Hugging Face API are essential parts of my argument for using AI/ML to accelerate DH research. Nevertheless, I have made the code, the models, and the datasets available for anyone who might want to use them or to investigate my results.

11. Financial and Environmental Costs

I tried to keep the financial and environmental costs of this work as low as possible. I used Free and Open-Source Software (FOSS) whenever possible, including all the Python packages required for the code. I worked almost exclusively in Microsoft's freely available VS Code. I used Git for version control, and I stored the data, code, and models in publicly available repositories (e.g., on GitHub and Hugging Face). I spent less than $100 of my own funds on subscriptions and other resources for this project.

I used the 'codecarbon' Python library to estimate the environmental impact of training the two models for this project.[37] The full output of 'codecarbon' is available in the model cards in the Hugging Face repositories, but I present the most important information here:

Table 1. Environmental impact of the models

Metric	Author Reconciliation	Greek-Latin Identification
Duration (seconds)	2197	1348
Emissions (kilograms of carbon)	0.015	0.009
Energy consumed (total watts)	0.031	0.019

For the sake of comparison, driving the average gas-powered car for one mile produces around 400 grams of CO_2, according to the United States Environmental Protection Agency.[38] By contrast, one fine-tuning run for the Author Identification model produces around 15 grams of CO_2; the Greek-Latin model produces around 9 grams. The energy consumed in fine-tuning runs for both models combined is less than the amount consumed by a smoke detector or carbon monoxide detector designed for home use, according to the Lawrence Berkeley National Laboratory.[39]

Of course, these numbers do not take into account the amount of energy consumed or carbon generated in training the original BERT model or its 'distilled' versions. Considering that each pretraining of the $BERT_{LARGE}$ model on 16 Cloud TPUs (64 TPU chips total) took 4 days to complete, the total environmental impact of this project could be said to be much greater.[40] To mitigate that as much as possible, I used the 'distilled' version as the basis for my models, since they take less time to fine-tune and require less storage than the full version would have.

12. Conclusion

My assumptions were optimistic, but the results are promising. I was able to fine-tune models to identify authors more accurately than standard deterministic and probabilistic matching algorithms can and more rapidly than humans can, and I did it at a very low cost and without access to special resources or paid developers. Moreover, I discovered a way to identify

candidates for addition to the DLL Catalog's authority records. Although titles remain an unsolved problem, at least I understand the issues better, and I have identified some techniques and approaches to try in the next stages of development. Even so, I can look forward to saving a considerable amount of time adding records to the DLL Catalog. I hasten to add, of course, that final review and approval of all metadata for this (and any) project should remain the exclusive domain of humans with subject-matter expertise.[41]

Regardless of this specific use case, I wish to make a larger point, namely that scholars with low or no funding and modest technical skills can use AI/ML technology to accelerate some of the tedious but necessary tasks in DH research.

Endnotes

A version of this article was presented at the 2025 annual meeting of the Society for Classical Studies during the panel 'Opening Up Classics with AI', organized by the Digital Classics Association. I wish to thank Rebecca Huskey, Ron Kneusel, Patrick Burns, Tyler Pearson, Mark Laufersweiler, Varun Sayapaneni, Caleb Carr, Alex Ward, and Wolfgang Jentner for their suggestions and guidance. I also wish to thank the two anonymous readers of this paper for their thoughtful and constructive comments.

[1] The Digital Latin Library, *The Digital Latin Library Project*, https://digitallatin.org/, last accessed 25 June 2025.

[2] For an overview of work on the library component of the DLL, see J. Abbas, et al., 'How I learned to love classical studies: information representation design of the Digital Latin Library', *Proceedings of the Association for Information Science and Technology*, 53 (2016), 1–10; and S. J. Huskey, 'The Digital Latin Library: cataloging and publishing critical editions of Latin texts', in Monica Berti, ed., *Digital classical philology: Ancient Greek and Latin in the digital revolution* (Boston, 2019), 19–34.

[3] The Digital Latin Library, *The Digital Latin Library Catalog*, https://catalog.digitallatin.org/, last accessed 25 June 2025.

[4] The Andrew W. Mellon Foundation supported the project with a planning grant (11200693) and two implementation grants (21400643 and 21500706).

[5] The Digital Latin Library, *The Library of Digital Latin Texts*, https://ldlt.digitallatin.org/, last accessed 25 June 2025.

[6] A list of standards and ontologies for bibliographical data would be long, but good examples are the The Library of Congress' *Metadata Object Description Schema* (MODS, https://www.loc.gov/standards/mods/, last accessed 25 June 2025) and *Metadata Authority Description Schema* (MADS, https://www.loc.gov/standards/mads/, last accessed 25 June 2025), and the International Federation of Library Associations and Institutions' *Functional Requirements for Bibliographical Records* (FRBR, https://www.ifla.org/resources/?oPubId=591, last accessed 25 June 2025) are good examples.

[7] OCLC, Inc. *Virtual International Authority File 8194433*, http://viaf.org/viaf/8194433, accessed on 19 January 2025.

[8] Strictly speaking, *Bucolica* ('Pastoral Songs') is the title of the collection of ten poems, each of which is an *Ecloga* ('Selection'). See R. Coleman, *Vergil: Eclogues* (Cambridge, 1977), 14–15.

[9] For example, Virgil's author page at https://catalog.digitallatin.org/dll-author/a4830, last accessed 25 June 2025.

[10] We continue to refine our implementation of Apache Solr (Apache Software Foundation, *Solr* https://solr.apache.org/, last accessed 25 June 2025) for faceted search to accommodate these variations.

[11] On the phenomenon of records 'disappearing' from library catalogs because of typographical errors, see J. Beall and K. Kafadar, 'The effectiveness of copy cataloging at eliminating typographical errors in shared bibliographic records', *Library Resources & Technical Services*, 48 (2004), 92–101.

[12] With a nod to A. Sweigart, *Automate the boring stuff with Python: practical programming for total beginners* (San Francisco, 2020).

[13] The information for these records was gleaned from a variety of print (e.g., P. G. W. Glare, ed. *Oxford Latin Dictionary*, Oxford, 1996) and digital (e.g., VIAF) sources. The records are available on the DLL Catalog site (https://catalog.digitallatin.org/) and in a variety of formats in a code repository

(DLL, *linked-open-data,* https://github.com/DigitalLatin/linked-open-data, last accessed 25 June 2025).

[14] F. Chollet, *Deep learning with Python*, 2nd edition (Shelter Island, 2021), recommends 20,000 words as the 'right vocabulary size for text classification' (310).

[15] Specifically, we include variant name forms from the following authorities: International Standard Name Identifier (ISNI), Biblioteca Nacional de España, Bibliothèque Nationale de France, Deutsche National Bibliothek, Istituto Centrale per il Catalogo Unico delle Biblioteche Italiane, WorldCat, and Wikidata.

[16] I selected the number eight because that is the average number of variant name forms per author in the DLL Catalog. The script for this process is available in my dataset repository *sjhuskey/latin_author_dll_id*, https://huggingface.co/datasets/sjhuskey/latin_author_dll_id, last accessed 25 June 2025.

[17] For example, an edition of Muḥammad ibn 'Umar, al-Wākidī's history of Mesopotamia in Arabic bears the title *de Mesopotamiae expugnatae historia*, but the text of the work is, in fact, in Arabic, though the editor's introduction is in Latin.

[18] Specifically, the TLG makes lists of authors publicly available at *TLG Authors*, https://stephanus.tlg.uci.edu/tlgauthors/post_tlg_e.php and *Authors included in TLG disks D and E*, https://stephanus.tlg.uci.edu/tlgauthors/cd.authors.php (both last accessed 25 June 2025).

[19] The Perseus Digital Library, *The Perseus Catalog*, https://catalog.perseus.org/, last accessed 25 June 2025. Many thanks to Alison Babeu, Digital Librarian and research coordinator for the Perseus Project, for access.

[20] The Greek-Latin dataset is available in my data repository *sjhuskey/greek_latin_authors*, https://huggingface.co/datasets/sjhuskey/greek_latin_authors, last accessed 25 June 2025.

[21] I followed the progression of examples in Chollet, *Deep learning with Python,* 309–363.

[22] J. Devlin, et al., 'BERT: pre-training of deep bidirectional transformers for language understanding', arXiv:1810.04805 (2018).

[23] Hugging Face (https://huggingface.co/) is a company that develops and maintains software for AI/ML applications and provides a space for sharing models and datasets. The full list of BERT models is at https://huggingface.co/models?sort=trending&search=bert, last accessed 25 June 2025.

[24] DistilBERT Community, *distilbert/distilbert-base-multilingual-cased*, https://huggingface.co/distilbert/distilbert-base-multilingual-cased, last accessed 25 June 2025.

[25] On the multilingual nature of the model, see T. Pires, E. Schlinger, and D. Garrette, 'How multilingual is multilingual BERT?', arXiv:1906.01502 (2019). The full list of languages is available at Google Research, *bert/multilingual.md,* https://github.com/google-research/bert/blob/master/multilingual.md#list-of-languages, last accessed 25 June 2025.

[26] D. Bamman and P. Burns, 'Latin BERT: A contextual language model for classical philology', arXiv:2009.10053 (2020).

[27] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter', arXiv:1910.01108 (2020).

[28] Hugging Face, *Transformers: BERT*, https://huggingface.co/docs/transformers/en/model_doc/bert, last accessed 25 June 2025.

[29] F. Pedregosa, et al., 'Scikit-Learn: Machine Learning in Python', *Journal of Machine Learning Research*, 12 (2011), 2825–30. I used 'f1 score', a standard metric in binary and multi-class classification tasks, as the primary metric, as well as the more generic 'accuracy' score. For documentation on both metrics, see Scikit-Learn, *sklean.metrics*, https://scikit-learn.org/stable/api/sklearn.metrics.html (last accessed 25 June 2025).

[30] For a summary and discussion of these issues, see H. Semmelrock, et al, 'Reproducibility in machine learning-based research: overview, barriers and drivers', arXiv:2406.14325 (2024).

[31] The TSV of the data downloaded from HathiTrust and the code I used to clean and format the data are available in the DLL's GitHub repository *dll_catalog_reconciliation*, https://github.com/DigitalLatin/dll_catalog_reconciliation, last accessed 25 June 2025.

[32] Patrick J. Burns, 'LatinCy: synthetic trained pipelines for Latin NLP', arXiv:2305.04365 (2023).

[33] N. Cooke and R. Litvack-Katzman, 'Open times: the future of critique in the age of (un)replicability', *International Journal of Digital Humanities*, 6 (2024), 71–85; B. Joyeux-Prunel, 'Digital humanities in

the era of digital reproducibility: towards a fairest and post-computational framework', *International Journal of Digital Humanities*, 6 (2024), 23–43; T. Ries, K. van Dalen-Oskam, and F. Offert, 'Reproducibility and explainability in digital humanities', *International Journal of Digital Humanities*, 6 (2024), 1–7; see also N. Z. Da, 'The computational case against computational literary studies', *Critical Inquiry*, 45 (2019), 601–39.

[34] The concept of 'good enough practices' originated with G. Wilson, et al., 'Good enough practices in scientific computing', *PLOS Computational Biology*, 13 (2017): e1005510, as noted in F. Karsdorp, M. Kestemont, and A. Riddell, *Humanities data analysis: case studies with Python*, Princeton, 2021, 323. On guidelines specifically developed for machine learning research see Semmelrock et al., 'Reproducibility in machine learning-based research' and the editorial 'Moving towards Reproducible Machine Learning', *Nature Computational Science*, 1 (2021), 629–30.

[35] R. T. Kneusel's books *Practical deep learning: a Python-based introduction* (San Francisco, 2021) and *How AI works: from sorcery to science* (San Francisco, 2024) were essential reading for this project. I am grateful to the author for discussing my work with me during a visit to my home institution.

[36] Documentation for the Hugging Face API, including links to the full code and the repositories where it is stored, is at Hugging Face, *Documentations*, https://huggingface.co/docs, last accessed 25 June 2025.

[37] B. Courty, et al, 'Mlco2/codecarbon: V2.4.1' doi.org/10.5281/zenodo.11171501, last accessed 25 June 2025.

[38] United States Environmental Protection Agency, *Greenhouse Gas Emissions from a Typical Passenger Vehicle*, https://www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle#driving, last accessed 25 June 2025.

[39] Lawrence Berkeley National Laboratory, *Products That Use Standby Power*, https://standby.lbl.gov/standby-power-data-metering, last accessed 25 June 2025.

[40] Devlin et al., 'BERT', 12. On the costs of each training of BERT models, see E. Strubell et al., 'Energy and policy considerations for deep learning in NLP', arXiv:1906.02243 (2019) and X. Wang et al., 'Energy and carbon considerations of fine-tuning BERT', in Houda Bouamor, Juan Pino, Kalika

Bali, eds., *Findings of the Association for Computational Linguistics EMNLP 2023* (Singapore, 2023), 9058–69.

[41] See J. Grimmer, M. E. Roberts, and B. M. Stewart, *Text as data: A new framework for machine learning and the social sciences* (Princeton, 2022) on amplifying, not replacing, 'the human efforts of scholars' (31).