

JLSC

ISSN 2162-3309 | JLSC is published by the Iowa State University Digital Press | <http://jpsc-pub.org>

Volume 13, 1 (2025)

Leveraging LLMs in Library Publishing: JATS XML Encoding with ChatGPT

Matthew Vaughn & Richard Higgins

Vaughn, M. & Higgins, R. (2025). Leveraging LLMs in Library Publishing: JATS XML Encoding with ChatGPT. *Journal of Librarianship and Scholarly Communication*, 13(1), eP18048. <https://doi.org/10.31274/jpsc.18048>

This article underwent semi-anonymous peer review in accordance with JLSC's peer review policy.



© 2025 The Author(s). This is an open access article distributed under the CC BY license (<https://creativecommons.org/licenses/by/4.0/>)

RESEARCH ARTICLE

Leveraging LLMs in Library Publishing: JATS XML Encoding with ChatGPT

Matthew Vaughn

Indiana University Bloomington

Richard Higgins

Indiana University Bloomington

ABSTRACT

Introduction: Reliable and lightweight conversions of Microsoft Word documents to HTML have long eluded library publishers. We demonstrate how off-the-shelf large language models (LLMs) like ChatGPT offer a lean pathway forward for generating JATS XML, which current platforms are equipped to render into user-friendly HTML publications.

Methods: With careful prompting, ChatGPT can turn a plain text typescript into valid JATS. Leveraging a one- and few-shot approach for the <front> part of an XML file ensures that boilerplate data included in example(s) prompts the LLM to populate the correct data in its output. In <body> and <back/references> parts, zero-shot prompts with only the name and version of our JATS specification produce valid XML in ChatGPT 4.0.

Results: One- and few-shot prompting proved effective in directing ChatGPT 3.5 to consistently encode discrete, sequential sections of article typescripts. In retesting with ChatGPT 4.0, zero-shot approaches demonstrated that <body> and <back/references> parts need only the JATS specification name and version to convert typescript into valid XML. The <front> parts still benefit from a one- and few-shot approach.

Discussion: The primary bottleneck is token or source size limitations. Content must be broken up into separate sections for input and the output manually “stitched” together to form a complete XML file.

Conclusion: LLMs may offer a solution for publishers without the resources to encode JATS files by other means. As LLMs increase in scale, we expect workflows for encoding research articles in JATS to become even more accurate, with fewer restrictions on capacity.

Keywords: library publishing, JATS XML, ChatGPT, prompt engineering

Received: 05/04/2024 Accepted: 09/05/2024



© 2025 The Author(s). This is an open access article distributed under the CC BY license (<https://creativecommons.org/licenses/by/4.0/>)

IMPLICATIONS FOR PRACTICE

1. HTML full-text publishing, or HTML + PDF article formats, should become the standard over PDF-only publications in order to provide flexible browser-based reading experiences and user accessibility.
2. The standard source for full-text academic publishing is encoded JATS XML files. JATS is a vital international markup standard for academic journal publishing. Scholarly communication practice would benefit from its wider adoption.
3. Our work demonstrates that, with effective prompting, LLMs like ChatGPT are capable of radically reducing the resources necessary to encode plain-text journal article content into JATS.
4. As under-resourced library publishers and other open access practitioners struggle to move away from PDF-only publishing, we believe our results represent a promising opportunity to enable the wider adoption of efficient Word-to-HTML conversion workflows.
5. We demonstrate and document a practical, step-by-step ChatGPT prompting approach that will allow library publishers to begin encoding their article content in JATS—even if they have little or no experience with XML, LLMs, or programming.

INTRODUCTION

The proliferation of library-published academic journals speaks to the ways in which scholarly publishing aligns with librarians' strengths in data management, digital scholarship, and information preservation and dissemination (Lippincott, 2017, p. 3). Library publishers, with their emphasis on education, open access (OA), innovative publications, and information equity, continue to claim a unique position for themselves as vital contributors to the production and dissemination of original academic research (Lippincott, 2017). In contrast to well-financed commercial publishers, however, library publishers face significant challenges in terms of staffing, capacity, and technological support (Perry et al., 2011; Meetz & Boczar, 2022). Consequently, library publishers favor "lightweight, nofrills workflows" that help to reduce costs while also giving them "their characteristic agility" (Lippincott, 2017, p. 5–6). One particularly troublesome and persistent barrier to this lightweight approach has been the difficulty in converting manuscripts authored in Microsoft Word into more flexible formats like the Journal Article Tag Suite (JATS) Extensible Markup Language (XML). Large language models (LLMs) like OpenAI's ChatGPT, however, offer a potential solution for library-based publishers who wish to publish full-text content and to gain experience with the JATS specification. This article describes our

efforts to develop an LLM-assisted JATS publishing workflow for the journal *Studies in Digital Heritage* (SDH).

The OA journal publishing program at [Indiana University Libraries](#) supports over 50 OA journals on a self-hosted instance of the [Open Journal Systems](#) (OJS) platform, which is developed and maintained by the Public Knowledge Project (PKP). Most of our program's journals are published only in PDF. And while we do now publish three journals in full-text HTML derived from JATS XML source files, we have struggled to adopt JATS more widely because of the added time and difficulty involved in converting author submissions into publication-ready JATS XML files. Nevertheless, we are eager to expand our adoption of JATS both because of the readability of full-text HTML and because of the sustainability of machine-readable XML files. JATS, specifically the *Journal Archiving and Interchange Tag Set v.1.3* in our implementation, is an XML tag set format that has become an international standard for scholarly journal publishing, indexing, sharing, and archiving ([McGlone, 2013](#); [Bazargan, 2015](#); [Lapeyre, 2019](#); [Odu & Ekanger, 2020](#)). Lapeyre (2019) explains that JATS is a set of “elements and attributes that describe the content and/or metadata of journal articles,” with the goal of providing a common format for archives and publishers to “produce, exchange, and store journal article content,” (para. 2). Originally developed by the National Library of Medicine (NLM) to preserve STEM articles, JATS is now used worldwide by journals in every discipline ([McGlone, 2013](#); [Lapeyre, 2019](#); [NISO, 2021](#)). The JATS standard is used by both small and mid-sized publishers in at least 25 countries, and all the largest academic publishing houses use JATS for exchange and preservation. In addition, many public archives and repositories accept or require JATS for deposit, and conversion vendors, web-hosting services, and software developers increasingly offer products and support for JATS workflows ([Lapeyre, 2019](#)).

The ubiquity of this tag suite is attributable to the numerous advantages it provides. In a 2020 article on JATS workflow implementation, [Odu and Ekanger](#) explain that XML allows publishers to “encode content in a machine-readable and layout-independent way, which is more flexible and reusable for a variety of formats (PDF, HTML, etc.). XML makes documents more searchable and accessible, and better-suited for preservation; it allows text mining and opens for content enrichment through multimedia and semantic tagging” ([Odu & Ekanger, 2020](#), p. 157). As a descriptive tag suite, JATS is flexible by design, enabling many ways to mark up the same article and giving editors latitude in customizing their approach to tagging content ([Lapeyre, 2019](#)). XML files are also more resilient as software systems and browsers evolve, making JATS archiving somewhat “future proof” ([Bazargan, 2015](#)). In addition, JATS XML is an inherently accessible format that provides markup for accessibility features like alt-text that make article images recognizable to pronouncing screen readers ([Kasdorf, 2018](#)).

Nevertheless, converting research articles into JATS XML poses considerable challenges to publishers with limited staff and resources. Research articles often total over 1000 lines of XML and usually include complex elements like graphs, abstracts, footnotes, references, and section headings that are more challenging to transform from Word-based manuscripts into JATS. Due to these complexities, our current XML workflow for research articles involves sending completed typescripts to a typesetting vendor for conversion from Microsoft Word documents. The XML produced by the vendor is then reviewed and corrected by in-house staff, who must often spend considerable time performing manual corrections before the XML can be uploaded for publication.

However, we have found that the LLM-based tool ChatGPT can produce promising results in tagging plain-text versions of research articles in accurate, publishable JATS. Trained on enormous sets of texts, LLMs like ChatGPT can handle a myriad of user-provided writing tasks, complex problems, and the automation of routine tasks (Sänger et al., 2023). Users do this by guiding the LLM with prompts entered by them directly into the ChatGPT interface. In an attempt to reduce the overhead of our investment in JATS-to-HTML publishing workflows, we developed a series of prompts in ChatGPT that directed the LLM to encode each section of a research article using a specific JATS specification. Once these sections are manually stitched together, this ChatGPT-based workflow produces a valid, publishable JATS galley file, demonstrating a promising step forward for library publishing.

LITERATURE REVIEW

Despite the widespread adoption of JATS XML and the considerable advantages it provides, publishers are still struggling with the challenges involved in converting article submissions from common Word documents to JATS and generating JATS galleys. One of the primary obstacles to the wider adoption of this standard is the dominance of Microsoft Word as the preferred manuscript submission format for most academic authors (Bazargan, 2015; Gebhard & Rosenblum, 2016; Garnett et al., 2017; Kasdorf, 2018). Garnett et al. (2017) note that the ubiquity of Word as an authoring tool “creates substantial inefficiencies in the scholarly authoring ecosystem” (para. 2). Many journal publishing workflows are organized around multiple rounds of “uploading and downloading incrementally updated drafts of Word manuscripts, creating a difficult-to-manage ecosystem of individual change-tracked files” (Garnett et al., 2017, para. 2). And while most authors have settled on Word for composition, most publishers have settled on JATS XML for production. However, as Bazargan (2015) points out, Word’s page-based, visual composition system does not map easily onto JATS’s logically structured tag set. An ambiguous graphical formatting choice in Word could potentially signal any number of different JATS XML tags (Bazargan, 2015). Authors have chosen a tool (Word) for its ease of use rather than for its advantages in publishing, while

publishers have chosen a powerful format (JATS) that remains difficult to produce efficiently: “This disconnect between the tools and formats used for authoring and the formats required for publishing has meant that, for several decades now, manuscripts received from authors will need to be entirely XML-typeset by publishers at considerable expense” (Garnett et al., 2017, para. 2). The solution for some publishers has been to outsource JATS production and its complex encoding apparatus to experienced vendors, while many smaller publishers simply forgo JATS publication altogether and stick with PDF-only publications.

Another solution publishers have been experimenting with are various proprietary or open-source options for automatic or semi-automatic Word-to-JATS conversions. Garnett et al. (2015) document some of the early efforts by PKP to develop an automatic stack for Word-to-JATS XML conversion. Using the *meTypeset* conversion engine written by Martin Paul Eve as their foundation, Garnett et al. (2015) developed the Open Typesetting Stack (OTS). By bundling *meTypeset* with several other open-source tools, the OTS project achieved some impressive results in converting Word documents to JATS. However, the output from OTS required more manual intervention than most journal editors found feasible. The complexity and maintenance needs of the OTS technical stack was judged to be out of the reach of most independently run journals. Garnett et al. (2017) note, for instance, that neither *meTypeset*- nor OTS-produced XML files were accurate enough to relieve vendors and editors from needing to have significant in-house XML expertise for revising and correcting the encoded files. In response to this, Alex Garnett and others went on to develop Texture, an easy-to-use visual interface that allows users with minimal XML expertise to edit imperfect Word-to-JATS conversions into structured, production-ready JATS galley (Garnett et al., 2017). While the Texture interface is more user-friendly than manual XML tagging, the process of turning an imperfect JATS conversion into a structured, semantically tagged, and metadata-enhanced galley remains challenging. Development and funding for Texture has since devolved after the complexities of competing use cases and stakeholder needs became too difficult to manage in one project (Aufreiter, 2020).

Bazargan (2015) successfully developed a complete end-to-end publishing system based on JATS (the RVPublisher), but this approach requires authors to forgo Word for a web-based interface that handles each article component separately and commits publishers to high, ongoing service costs. Dartmouth Journal Services, likewise, created an automated JATS-based production workflow, which requires trained editors to structure Word manuscripts in *eXtyles* in preparation for XML creation and copyediting. Proofing and editing are then completed in a web-based XML editing environment that authors must learn to use in order to review and correct their proofs (O’Connor et al., 2015). Clearly, mastering these types of semi-automated workflows would require significant buy-in and investment from editors and authors who are probably already proficient in workflows that rely mainly on Word and

InDesign. Furthermore, this JATS-based approach still necessitates the use of additional proprietary tools like eXtyles. While styling and structuring Word manuscripts can be an effective way to prepare for conversion to JATS XML, someone still must be trained to format Word correctly, and journals must have the capacity to take on the manuscript styling work (Kasdorf, 2018). For less well-resourced journals with established Word-to-PDF workflows, switching to a JATS-based production workflow looks out of reach and impractical.

In order to make JATS realistic for under-resourced journals, something closer to a completely automated Word-to-JATS conversion represents a more workable solution. Maier & Stutzmann (2019) developed an automated, single-source JATS workflow for the journal *Judaica*, primarily using a combination of Markdown, the open source Pandoc converter, and the OJS XML Lens Viewer. They created a Makefile that automates the build workflow, allowing them to quickly create publication-ready files in JATS, PDF, and HTML. Nevertheless, the implementation of this workflow requires considerable technical expertise, and the JATS produced is in some respects sparsely tagged. Odu & Ekanger (2020) attempted to develop a JATS-based publishing workflow using the Texture JATS editor and the OJS-developed plugin *docxToJats* converter, a PHP-based library that leverages Word OOXML standards to output JATS XML. Both OJS and *docxToJats* are written in the PHP programming language, and OOXML is the XML specification underlying docx files. While this process produced promising results, the project ultimately failed due to *docxToJats*'s lack of support for some non-textual elements and failures following system upgrades.

While significant progress has been made in automating Word-to-JATS conversion and in JATS publication workflows, a conversion process that is both efficient and accessible to under-resourced academic journal editors remains elusive. A number of free, open-licensed online tools are able to automatically convert Word documents to JATS, including

- ALLDOCS (<https://alldocs.app/convert-word-docx-to-jats-xml>),
- CERMINÉ (<http://cermine.ceon.pl/index.html>),
- Pandoc (<https://pandoc.org/>), and
- Vertopal (<https://www.vertopal.com/en/convert/docx-to-xml-jats>).

However, we tested each of these tools using Word versions of some of our published *SDH* research articles, and unfortunately, none of the conversions were successful enough with our research articles to indicate that any of these tools are a viable alternative to vendor conversions to JATS.

METHODS

Since 2017, the Indiana University Open Access Journal Publishing program has published *SDH*, a peer-reviewed online journal that publishes innovative research articles applying new digital technologies to the various fields of cultural heritage. Starting in 2018, we have published *SDH* in both PDF and in HTML, which is derived from JATS XML galleys, on the OJS publishing platform. Our current workflow involves sending the journal's DOCX manuscript files and PDF galleys to a third-party vendor for conversion into JATS. Once the vendor has converted these articles into JATS, we take the JATS files through a final correction stage using Oxygen and other text editors. Once finalized, the publication-ready JATS galleys are uploaded and rendered on-the-fly into HTML by the eLife Lens Article Viewer plugin. While this workflow is effective in producing accurate JATS-to-HTML publication, it would still be much too costly and labor-intensive to adopt for all the journals we publish. Nevertheless, the advantages of XML publishing have encouraged us to explore any available options to expand our use of JATS. Indeed, in 2023 we began experimenting with the generative AI tool ChatGPT to determine whether it could be used to automate some or all of the conversions to JATS in our publishing workflow.

The growing public availability of LLM-based tools like those released by [OpenAI](#) and [Microsoft](#), for instance, has given practitioners in a number of disciplines the opportunity to explore new approaches to automating common tasks through generative AI. LLMs are trained on “vast sets of texts from the web and other repositories, enabling them to encode syntactic and semantic relationships in human language” ([Sänger et al., 2023](#), p. 2). In response to user input, generative language models like ChatGPT are often capable of producing convincing approximations of human writing, of providing possible solutions to complex problems, and of automating various tasks ([Sänger et al., 2023](#)). Working with an LLM involves providing it with “prompts,” which are “natural language instructions used to provide context to the LLM and guide its generation of textual responses” ([White et al., 2023](#), p. 1). In ChatGPT, these take the form of chat messages that are typed or pasted directly into browser- and app-based user interfaces ([White et al., 2023](#)). While LLMs “hold immense potential for helping to automate common tasks,” care must be taken in structuring prompts correctly and in validating the output these prompts produce ([White et al., 2023](#), p. 14). Even with careful use, LLMs have been demonstrated to produce faulty results and false or unexpected information, and the “design of prompts entered into [an] LLM has a decisive influence on the output quality” ([Sänger et al., 2023](#), p. 5). Prompt engineering is the practice of “utilizing carefully selected and composed NL [natural language] instructions to achieve the desired output from a GenAI model such as a pre-trained LLM” ([Ronanki et al., 2023](#), p. 2). Natural language instructions are inherently ambiguous

and susceptible to misinterpretation by the LLM, making prompt engineering more of an art than a science (Ronanki et al., 2023).

Our LLM assisted approach to automating the process of tagging *SDH* research articles in JATS involves prompting ChatGPT with a series of zero-shot, one-shot, and few-shot instructions. In this type of LLM prompting, “shots” refer to the examples the user provides to ChatGPT to guide it in the completion of a task (Reynolds & McDonell, 2021; White et al., 2023).

- Zero-shot prompting involves directing the LLM to complete a task without any examples (Dang et al., 2022): “Tag the following reference in JATS XML.”
- In one-shot prompting, the user provides a single example: “Tag in JATS XML any references I provide you, using this tagged reference as a guide: [provide a relevant reference tagged in JATS XML].”
- Few-shot prompting uses two or more examples (Dang et al., 2022).

Depending on the task, zero-shot prompting with LLMs is often at least as effective if not more so than one- or few-shot prompting in completing tasks (Kojima et al., 2022; Sohail et al., 2023). And while we initially experimented with one- and few-shot prompting strategies, our results support the argument that LLMs are zero-shot reasoners (Kojima et al., 2022).

Due to the token limitations of both ChatGPT v.3.5 and v.4.0, it is necessary to break up the automatic tagging process into several steps for each research article. We first tag the header/front section of the article—usually in one pass—and then the body sections in chunks of six to fifteen paragraphs, depending on the version of the LLM; finally, we tag the references/back section, which requires the most resources for the LLM and must be split up into five to eight references per input.

In order to tag the header and <front> metadata-heavy section of an article, we use a one-shot prompt that provides ChatGPT with the context of the task (tagging research articles in JATS) and a mostly blank template of the entire tag structure we use for the <front> section of our articles (as an example for the processor).

Prompt: “I will give you plain text research article content to tag in JATS XML. There may be multiple authors (<contrib contrib-type=“author”>). All the research article content I provide you should be tagged in JATS XML with a header, doctype declaration, and front section that are structured like this example template: [followed by our template].”

After we supply this prompt, ChatGPT will respond with a formulaic expression of willingness to assist and request the plain-text content to begin tagging it in JATS: “Sure, please provide the research article content that you would like to be tagged in JATS XML format.” We then cut and paste the header material (title, author, abstract, keywords, volume, issue, etc.) from a plain-text version of the article directly into the interface. ChatGPT will then tag the header content as requested (Figure 1).

Partial output:

```
xml
Copy code

<!DOCTYPE article PUBLIC "-//NLM/DTD JATS (Z39.96) Journal Archiving and Interco
<article dtd-version="1.0" article-type="research-article"
  xmlns:xlink="http://www.w3.org/1999/xlink">
  <front>
    <journal-meta>
      <journal-id>SDH</journal-id>
      <journal-title-group>
        <journal-title>Studies in Digital Heritage</journal-title>
      </journal-title-group>
      <issn pub-type="epub">2574-1748</issn>
      <publisher>
        <publisher-name>Indiana University</publisher-name>
      </publisher>
    </journal-meta>
    <article-meta>
      <article-id pub-id-type="doi">10.14434/sdh.v7i1.35741</article-id>
      <article-categories>
        <subj-group subj-group-type="heading">
          <subject>Articles</subject>
        </subj-group>
      </article-categories>
      <title-group>
        <article-title>Ethically Digital: Contested Cultural Heritage in
      </title-group>
```

Figure 1. Partial ChatGPT output after using a one-shot prompt to encode plain-text article header content.

In the case of the <front> section, we learned how important it is to be precise in our prompts. When we ask for the same front section without an example, when we use a zero-shot approach instead of a one- or few-shot prompt, in other words, ChatGPT creates its own

data instead of flagging it as unknown. The LLM will, in many cases, output incorrect data without notifying the user of the potential error.

Output with the wrong ISSN:

```

xml
Copy code
<!DOCTYPE article PUBLIC "-//NLM//DTD JATS (Z39.96) Journal Archiving and Interco
<article xmlns:xlink="http://www.w3.org/1999/xlink" article-type="research-artic
<front>
  <journal-meta>
    <journal-id journal-id-type="publisher-id">SDH</journal-id>
    <journal-title-group>
      <journal-title>Journal of Digital Heritage</journal-title>
    </journal-title-group>
    <issn pub-type="ppub">1234-5678</issn>
    <publisher>
      <publisher-name>Society for Digital Heritage</publisher-name>
    </publisher>
  </journal-meta>
  <article-meta>
    <title-group>
      <article-title>Ethically Digital: Contested Cultural Heritage in Digital
    </title-group>
    <contrib-group>
      <contrib contrib-type="author">
        <name>
          <surname>Rouhani</surname>
          <given-names>Bijan</given-names>
        </name>

```

Figure 2. Partial ChatGPT output after prompting without an example. The incorrect ISSN is emphasized.

By providing a one-shot prompt with an example of a properly tagged <front> section, we also provide the AI processor with the boilerplate data for our journal.

```

<journal-title-group>
<journal-title>Studies in Digital Heritage</journal-title>
</journal-title-group>

```

```
<issn pub-type="epub">2574-1748</issn>

<publisher>

<publisher-name>Indiana University</publisher-name>

</publisher>
```

This is particularly true in the case of a journal’s ISSN. In the image above, for example, ChatGPT has inserted “1234-5678” as the ISSN (Figure 2); in other instances, it outputs less obvious sets of random numbers formatted like an ISSN. A fully formatted example with the correct general metadata of the journal ensures that ChatGPT outputs a more accurate document. For instance, the LLM will correctly “consume” the journal’s ISSN from the example and include it in the output.

The <body> sections of an article are generally less complex than the front matter, and this gives us an opportunity to pare down our prompts to the JATS specification on its own. This saves us time we would spend tinkering with examples and helps us to avoid introducing inadvertent errors in our prompts. In ChatGPT 4.0, the processor only needs the correct JATS set.¹ In this case, we are using the “Journal Archiving and Interchange Tag Set v. 1.3” specification.

Prompt: “I will provide you with individual sections of research article content to tag in JATS standard. Use the Journal Archiving and Interchange Tag Set v. 1.3 specification and 'B0' as a placeholder for in-text citations.”

In this case, we do add one additional instruction to make “B0” the placeholder reference ID. But this is for our convenience to avoid confusion in the manual follow-up work. Without the instruction, the processor will helpfully add sequential IDs as placeholders that do not correspond to the correct IDs in the references list. (With further prompt work, we should be able to add connected references as a feature in the workflow.) After the prompt, ChatGPT, again, responds by requesting the content and offering to tag it, with output like the image below after the content has been input (Figure 3).

¹ All references are to the paid “Personal Plus” version of ChatGPT 4.0.

Partial output:

```

<sec id="sec1">
  <title>Introduction</title>
  <p>The FAIR guiding principles have been developed by data stakeholders, including academia, industry, funding agencies, and scholarly publishers, to ensure the reusability of data as well as findability, accessibility, and interoperability [<xref ref-type="bibr" rid="B0">Wilkinson et al. 2016</xref>]. Although FAIR principles advocate for open and reusable data and facilitate data sharing, the rights of indigenous communities to control, access, and benefit from data about their people, land, and resources and the related ethical challenges are not addressed. To complement FAIR Principles, the Global Indigenous Data Alliance developed the CARE Principles for Indigenous Data Governance. The CARE Principles, standing for Collective benefit, Authority to control, Responsibility, and Ethics, acknowledge the concerns about people and the purpose of data, power relationships, historical, social, and structural conditions and inequalities, and data sensitivity for each community [<xref ref-type="bibr" rid="B0">Carroll et al. 2021</xref>, <xref ref-type="bibr" rid="B0">Kansa 2022</xref>]. The scope of CARE principles is expanding beyond indigenous peoples to include the data rights of the broader community. These principles can include any community whose cultural heritage has become digital data.</p>
  <p>In 2021, UNESCO's Member States adopted a set of recommendations on the ethics of artificial intelligence (AI). The recommendations are based on universal values, including human rights, environment, diversity and inclusiveness, and peaceful, just, and interconnected societies. These fundamental values are not the same as the UNESCO concept of outstanding universal value, an increasingly contentious issue [<xref ref-type="bibr" rid="B0">Jokilehto 2008</xref>]. The document provides a framework of principles to help states in developing their national legislation and policies on AI. These principles are: proportionality and do no harm; safety and security; fairness and non discrimination; sustainability, right to privacy and data protection; human oversight and determination; transparency and explainability; responsibility and accountability; awareness and literacy; and multi-stakeholder and adaptive governance and collaboration [<xref ref-type="bibr" rid="B0">UNESCO 2022</xref>, 18–23].</p>
</sec>

```

Figure 3. Partial ChatGPT output for the body of an article after prompting with the appropriate JATS specification.

The evolution of ChatGPT is notable in its ability to properly render a specific specification like JATS XML. When we began this project using ChatGPT 3.5, we often had to remind the LLM to use correct formatting and syntax. Our original prompts included prompting for properly tagging sections and inline references: “Please tag in JATS XML any section (<sec>) content that I provide you. Use ‘B0’ as a placeholder for in-text citations like <xref ref-type=“bibr” rid=“B0”>-Surname 2018</xref>.” We are able to pare away the complexity of a prompt as the LLM versions evolve. With ChatGPT 4.0, it is the AI engine that reminds us of how it has formatted the correct JATS. From the output for a recent body section prompt, the interface writes

In this structure, each paragraph is enclosed within <p> tags, and in-text citations are referenced using <xref> tags with a ref-type of “bibr” to indicate bibliographic references. Each reference is assigned an ID which needs to be defined in the bibliography section with corresponding <bibl> tags. This XML assumes placeholders for references, which should be replaced with actual IDs as per the bibliography provided.

The token limits, or capacity of the processor for a given input, has also increased with newer versions. (According to ChatGPT itself, for standard releases, 3.5 is set at 4,096 tokens, 4.0 at 8,192 tokens.) This improves the overall efficiency and organizational demands of the workflow, even if we are still entering each “section” of an article manuscript one by one.

Once all the body sections have been tagged, we can move on to the references list. In order to tag the references, we give ChatGPT a zero-shot prompt which provides the context of the task (tagging references in JATS XML):

Prompt: “I will provide you with plain text research article content to tag in JATS XML. Use the Journal Archiving and Interchange Tag Set v. 1.3 specification. Please tag in JATS XML all references and references lists that I provide you.”

Partial output:

```
xml Copy code
<ref-list>
  <title>References</title>
  <ref id="B1">
    <mixed-citation publication-type="report">
      <person-group person-group-type="author">
        <name><surname>AL-Azm</surname><given-names>Amr</given-names></name>
        <name><surname>Paul</surname><given-names>Katie A.</given-names></name>
      </person-group>
      <year>2019</year>
      <article-title>Facebook's Black Market In Antiquities Trafficking, T
      <ext-link ext-link-type="uri" xlink:href="http://atharproject.org/wp
    </mixed-citation>
  </ref>
  <ref id="B2">
    <mixed-citation publication-type="web">
      <person-group person-group-type="author">
        <name><surname>Ameneyro</surname><given-names>Van</given-names></name>
      </person-group>
      <year>2022</year>
      <article-title>Nfts As Cultural Heritage</article-title>
      <ext-link ext-link-type="uri" xlink:href="https://undrgrnd.io/undrgr
    </mixed-citation>
  </ref>
  <ref id="B3">
    <mixed-citation publication-type="web">
```

Figure 4. Partial ChatGPT output for a references list after prompting with the appropriate JATS specification.

ChatGPT responds by requesting the content and offering to tag it. We then paste in a plain-text version of the list of references, and ChatGPT begins tagging them (Figure 4). Longer reference lists will need to be broken into sections to avoid hitting the token limit. References, as many will know from using other conversion tools, are especially difficult to process. ChatGPT 4.0 can generally process eight to ten references at a time.

With the references tagged, they can be combined with the previously created front and body sections in a text editor to create a well-formed JATS version of the plain-text article content we started with. However, the resulting document is still not publication-ready, and some additional editing both manually and with ChatGPT will be required.

Our overall prompting strategy has produced consistent results over the past year. Our first successful use of this strategy occurred in July of 2023 when one author was able to create a JATS version of a research article that was almost completely generated by one- and few-shot prompts in ChatGPT 3.5 alone. In April of 2024, the co-author was then able to independently replicate much of the first author's results for JATS headers, body sections, and references, using a different article and a combination of the original prompts and new zero-shot prompts in ChatGPT 4.0. Finally, in August of 2024, the authors were again able to replicate the results for JATS headers, body sections, and references using the same ChatGPT 4.0 prompting strategy. Nevertheless, as these models evolve, users can expect variability in results across different versions and with different "shot" prompts. Much like search, prompting requires users to develop skill with testing strategies, allowing real-time results to guide necessary adjustments. No single prompt or strategy should be relied on rigidly.

RESULTS

Our experiment with automating JATS XML encoding through ChatGPT demonstrates that LLMs are capable of completing this kind of work with a high level of accuracy. With zero- and one-shot prompting using relevant examples, ChatGPT is able to effectively tag plain-text article content. At the section level, the JATS produced by ChatGPT is comparable in accuracy to that produced by our vendor (Figure 5). By using this method and several additional steps, including separate steps for completing tables and in-text citations, we are able to produce a publishable JATS file that can be rendered as HTML on the *SDH* website. However, the piecemeal approach necessitated by the ChatGPT token limit makes this approach time-consuming and impractical to adopt on a large scale. We have certainly not reached a point where we can quickly and efficiently tag complete research articles in publishable JATS using ChatGPT alone. However, it should be noted that mid- or small-scale individual journals with the capacity and desire to

publish in full text could produce their own JATS galleys in the manner outlined above, depending on the publication tempo.

Our current production workflow using a commercial vendor to produce XML is still much faster and easier than the tagging we can currently do with ChatGPT 3.5 and 4.0. Nevertheless, if the token limit were increased, and if we could provide ChatGPT with all our prompts at once, fully automated JATS tagging seems to be well within reach. Ideally, we would only need to provide a single, longer prompt with the appropriate JATS specification, followed by the plain text of an entire issue's worth of articles, and ChatGPT would tag everything in one pass. As LLMs increase in power and sophistication, we anticipate that this kind of large-scale automated text processing is an inevitability.

Finally, however, we note that our ChatGPT prompts have already helped us to streamline our current XML workflow. Our vendor-produced JATS galleys typically require some level of correction on our part before they can be published, and ChatGPT has allowed us to automate some of this correction process. Reference lists, for example, have often proved difficult for our vendors to tag accurately and consistently. But our ChatGPT prompt for tagging reference lists now allows us to quickly produce an accurate list and add it to our JATS galleys, saving us considerable time in the correction process (Figure 5).

Original plain text reference: De Luca, L., Veron, P., and Florenzano, M. 2006. Reverse engineering of architectural buildings based on a hybrid modeling approach. *Computers & Graphics* 30, 2, 160–176.

Reference tagged in JATS by ChatGPT.	Reference tagged in JATS by vendor.
<pre> <ref id="B13"> <mixed-citation publication-type="journal"> <person-group person-group-type="author"> <name> <surname>De Luca</surname> <given-names>L.</given-names> </name> <name> <surname>Veron</surname> <given-names>P.</given-names> </name> <name> <surname>Florenzano</surname> <given-names>M.</given-names> </name> </person-group> <year>2006</year> <article-title>Reverse engineering of architectural buildings based on a hybrid modeling approach</article-title> <source>Computers and Graphics</source> <volume>30</volume> <issue>2</issue> <fpage>160</fpage>-<lpage>176</lpage> </mixed-citation> </ref> </pre>	<pre> <ref id="B13"> <mixed-citation publication-type="journal"> <person-group person-group-type="author"> <string-name> <surname>De Luca</surname>, <given-names>L.</given-names> </string-name>, <string-name> <surname>Veron</surname>, <given-names>P.</given-names> </string-name>, and <string-name> <surname>Florenzano</surname>, <given-names>M.</given-names> </string-name> </person-group> <year>2006</year>. <article-title>Reverse engineering of architectural buildings based on a hybrid modeling approach</article-title>. <source>Computers & Graphics</source> <volume>30</volume>, <issue>2</issue>, <fpage>160</fpage>-<lpage>176</lpage>. </mixed-citation> </ref> </pre>

Figure 5. An example reference tagged in JATS by ChatGPT next to the same reference tagged by a JATS vendor.

DISCUSSION

Our results indicate that, with effective prompting, LLMs like ChatGPT are capable of automatically tagging plain-text research article content in JATS XML. While the resulting tagged output is not error-free, the mistakes that do occur are easy to correct and comparable to those that we find in vendor-produced JATS. After a JATS galley is rendered into HTML by the eLife Lens Article Viewer plugin, it is easy to spot errors in the HTML and to compare it side by side with a publication-ready PDF for further validation.

Publishers have explored and developed a number of approaches to JATS conversion and workflows, but JATS adoption still presents considerable challenges for smaller publishers. Our work suggests that LLMs may eventually offer a workable solution for publishers with limited resources or who lack expertise to adopt most of the existing workflows. In addition, researchers in a number of fields are investigating the efficacy of various LLM prompting strategies, and our results suggest that zero-, one-, and few-shot prompting with detailed examples are effective in text processing of this kind. With higher token limits, greater sophistication, and additional JATS training input, LLMs will likely be able to accurately tag entire issues or more with a single prompt.

LLMs have the potential to put JATS publishing in the reach of small publishers and under-resourced journals, and wider JATS adoption is beneficial for scholarly communication. JATS facilitates the encoding and exchange of journal article content, promoting archiving, flexibility, and interoperability. In addition, XML tagging ensures machine-readable content, fostering research by enabling searchability, accessibility, and text mining.

While LLMs hold enormous potential for automating digital publishing tasks, no single existing model can or should be relied on uncritically or indefinitely. ChatGPT, while currently “free-to-use,” is a proprietary system with no particular responsibility to its users or to the scholarly publishing community. Proprietary systems frequently offer convenience, efficiency, and ease of use at the price of user data, dependency, and a lack of reliability (Liesenfeld et al., 2023). In addition, proprietary systems often lack “transparent ethical oversight” and are motivated by profit instead of a commitment to principled research and open knowledge production (Liesenfeld et al., 2023, p. 2). Furthermore, the creation and development of proprietary systems often involves “exploitative labour practices, privacy implications, and murky copyright situations,” and concerns have been raised about OpenAI/ChatGPT in each of these areas (Liesenfeld et al., 2023, p. 2). Finally, LLMs are prone to produce errors, faulty results, and false information. They are known to generate hallucinations, which are “outputs that, while statistically plausible and seemingly believable, are factually incorrect” (Sänger et al., 2023). Recent research has also suggested that ChatGPT’s performance and behavior can

vary over time, and its ability to follow instructions has decreased (Chen et al., 2023). As a result, prompts and prompting strategies may not be reliable over time with ChatGPT, and publishers should consider this when integrating LLM assistance into their work. Given these issues, librarians and library publishers should seek opportunities to create and to support transparent, ethical, and open-source AI tools that are academy owned, trained, and hosted.

Both the Association of Research Libraries and Creative Commons support the broad use of “modern in-copyright materials” to train AI as essential to innovation and in the public interest. Although using copyrighted content to train an AI tool or using an LLM to process a copyrighted text are both, at present, considered to be fair use, authors and editors may still be uncomfortable sharing journal content with LLMs like ChatGPT (Vézina & Pearson, 2021; Klosek & Blumenthal, 2024).² When users share content, their input may be used by ChatGPT to further refine its model. However, ChatGPT users do have the option to opt out of allowing their content to be used as training data by the model, and editors can make the realities of LLM-assisted workflows clear to their authors (OpenAI, 2024b). We note, for instance, that LLMs do not actually store user input or other training data. These models, instead, “read” content to enhance the LLM’s ability to predict relationships between words. That is to say, training data influences and enhances a model but is not permanently stored by that model (OpenAI, 2024a). In addition, while author agreements routinely grant journals a non-exclusive license to use a published work in any form or medium and to create derivative work from it, editorial teams using an LLM-assisted publishing workflow may, in the interest of transparency, wish to make this detail explicit in the text of their agreements. Finally, we see this approach as an ethical use of research content, in part, because we are using publicly available OA research articles, and our primary aim is to leverage ChatGPT’s processing power to provide greater access to this research by publishing it in an HTML version as a supplement to less accessible PDF versions.

Future research should explore similar prompting strategies with other LLMs, particularly those with higher token limits. If ChatGPT or other LLMs could manage more comprehensive JATS prompts and ingest and tag entire articles or issues at once, that would represent an important step forward in automating JATS publishing workflows. In addition, the efficiency and accuracy of JATS tagging would likely be substantially increased if an LLM could be augmented or customized by additional training on a large corpus of research articles tagged in well-formed JATS. While our investigations relied on zero-, one-, and few-shot prompting, future research should also investigate other prompting patterns for their efficacy in automating JATS tagging.

² In the U.S., the ultimate legal status of training AI with copyrighted works remains unresolved, and concepts of fair use and fair dealing vary considerably internationally (Ahmad & Gross, 2024, para. 3).

CONCLUSION

While many smaller academic publishers could benefit from the utility of the JATS format, the difficulty in converting article submissions files into JATS galleys represents a significant barrier. Publishers and editorial teams have investigated a number of approaches to streamline JATS XML production. However, to our knowledge, no one has yet explored LLM prompting as a method for automatically tagging research articles. We find that with suitable prompting, ChatGPT has the ability to automatically convert plain-text research article content into well-formed JATS. Zero-, one-, and few-shot prompting proved effective in directing ChatGPT to consistently and accurately tag discrete sections of content and data from article typescripts. As LLMs increase in power, flexibility, and sophistication, we expect they will be able to encode research articles in JATS quickly, accurately, and at large scale.

REFERENCES

- Ahmad, A., & Gross, A. (2024). *Generative AI systems tee up fair use fight*. The National Law Review. <https://naltlawreview.com/article/generative-ai-systems-tee-fair-use-fight>
- Aufreiter, M. (2020, September 10). *Is this project still active?* [Github issue comment]. Github. <https://github.com/substance/texture/issues/1358>
- Bazargan, K. (2015, April 22). A complete end-to-end publishing system based on JATS. In *Journal Article Tag Suite Conference (JATS-Con) Proceedings 2015*. National Center for Biotechnology Information, Bethesda, MD. <https://www.ncbi.nlm.nih.gov/books/NBK279828/>
- Chen, L., Zaharia, M., & Zou, J. (2023). *How is ChatGPT's behavior changing over time?* arXiv. <https://doi.org/10.48550/arXiv.2307.09009>
- Dang, H., Mecke, L., Lehmann, F., Goller, S., & Buschek, D. (2022). *How to prompt? Opportunities and challenges of zero- and few-shot learning for human-AI interaction in creative applications of generative models*. arXiv. <https://doi.org/10.48550/arXiv.2209.01390>
- Garnett, A., Aufreiter, M., Buchtala, O., & Alperin, J. P. (2017). Introducing Texture: An open source WYSIWYG Javascript editor for JATS. In *Journal Article Tag Suite Conference (JATS-Con) Proceedings 2017*. National Center for Biotechnology Information, Bethesda, MD. <https://www.ncbi.nlm.nih.gov/books/NBK425544/>
- Garnett, A., Alperin, J. P., & Willinsky, J. (2015, April 21). The Public Knowledge Project XML publishing service and meTypeset: Don't call it "Yet another Word-to-JATS conversion kit." In *Journal Article Tag Suite Conference (JATS-Con) Proceedings 2015*. National Center for Biotechnology Information, Bethesda, MD. <https://www.ncbi.nlm.nih.gov/books/NBK279666/>
- Gebhard, C., & Rosenblum, B. (2016). Wrangling math from Microsoft Word into JATS XML workflows. *Learned Publishing*, 29(4), 271–279. <https://doi.org/10.1002/leap.1058>

- OpenAI. (2024a). *How ChatGPT and our language models are developed*. <https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed>
- OpenAI. (2024b). *How your data is used to improve model performance*. <https://help.openai.com/en/articles/5722486-how-your-data-is-used-to-improve-model-performance>
- Kasdorf, B. (2018, October 18). *Getting from Word to JATS XML*. Association of Learned and Professional Society Publishers blog. <https://blog.alpsp.org/2018/10/getting-from-word-to-jats-xml.html>
- Klosek, K., & Blumenthal, M. (2024, January 23). *Training generative AI models on copyrighted works is fair use*. Association of Research Libraries' ARL Views blog. <https://www.arl.org/blog/training-generative-ai-models-on-copyrighted-works-is-fair-use>
- Kojima, T., Matsuo, Y., Iwasawa, Y., Gu, S. S., & Reid, M. (2022). *Large language models are zero-shot reasoners*. arXiv. <https://doi.org/10.48550/arXiv.2205.11916>
- Lapeyre, D. A. (2019, July 29). *Balisage Paper: Customizing JATS (Journal Article Tag Suite) [Presentation]*. Balisage Series on Markup Technologies 24. Presented at the Symposium on Markup Vocabulary Customization, Washington, DC. <https://doi.org/10.4242/BalisageVol24.Lapeyre01>
- Liesenfeld, A., Lopez, A., & Dingemans, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *Proceedings of the 5th International Conference on Conversational User Interfaces, 1–6*. Association for Computing Machinery Digital Library. <https://doi.org/10.1145/3571884.3604316>
- Lippincott, S. K. (2017). *Library as publisher: New models of scholarly communication for a new era (Charleston Briefings)*. Against the Grain (Media), LLC. <https://doi.org/10.3998/mpub.9944345>
- Maier, D. C., & Stutzmann, J. (2019). JATS for Judaica. Designing an XML-first publication workflow for a Humanities journal [Conference presentation]. PKP 2019 Conference, Barcelona, Spain. <https://boris.unibe.ch/id/eprint/135828>
- McGlone, J. (2013). Preserving and publishing digital content using XML workflows. In A. P. Brown (Ed.), *The library publishing toolkit* (pp. 97–108). IDS Project Press. <http://hdl.handle.net/2027.42/99563>
- Meetz, J., & Boczar, J. (2022). Library publishing programs at capacity: Addressing issues of sustainability and scalability. *Journal of Librarianship and Scholarly Communication, 10*(1). <https://doi.org/10.31274/jlsc.12909>
- NISO (National Information Standards Organization). (2021). *Standardized markup for journal articles: Journal Article Tag Suite (JATS)*. <https://www.niso.org/standards-committees/jats>
- O'Connor, C., Haenel, S., Gnanapiragasam, A., Hepp, M., & Fleischer, T. (2015, April 22). Building an automated XML-based journal production workflow. In *Journal Article Tag Suite Conference (JATS-Con) Proceedings 2015*. National Center for Biotechnology Information, Bethesda, MD. <https://www.ncbi.nlm.nih.gov/books/NBK279927/>

- Odu, O., & Ekanger, A. (2020). How we tried to JATS XML. *Ravmetrykk*, (39), 156–162. <https://doi.org/10.7557/15.5517>
- Perry, A. M., Borchert, C. A., Deliyannides, T. S., Kosavic, A., Kennison, R., & Dias-Correia, S. (2011). Libraries as journal publishers. *Serials Review*, 37(3), 196–204. <https://doi.org/10.1080/00987913.2011.10765382>
- Reynolds, L., & McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–7. Association for Computing Machinery Digital Library. <https://doi.org/10.1145/3411763.3451760>
- Ronanki, K., Cabrero-Daniel, B., Horkoff, J., & Berger, C. (2023). *Requirements engineering using generative AI: Prompts and prompting patterns*. arXiv. <https://doi.org/10.48550/arXiv.2311.03832>
- Sänger, M., De Mecquenem, N., Lewińska, K. E., Bountris, V., Lehmann, F., Leser, U., & Kosch, T. (2023). *Large language models to the rescue: Reducing the complexity in scientific workflow development using ChatGPT*. arXiv. <https://doi.org/10.48550/arXiv.2311.01825>
- Sohail, S. S., Farhat, F., Himeur, Y., Nadeem, M., Madsen, D. Ø., Singh, Y., Atalla, S., & Mansoor, W. (2023). Decoding ChatGPT: A taxonomy of existing research, current challenges, and possible future directions. *Journal of King Saud University - Computer and Information Sciences*, 35(8), 101675. <https://doi.org/10.1016/j.jksuci.2023.101675>
- Vézina, B., & Pearson, S. (2021, 4 March). *Should CC-licensed content be used to train AI? It depends*. Creative Commons blog. <https://creativecommons.org/2021/03/04/should-cc-licensed-content-be-used-to-train-ai-it-depends>
- White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2023). *Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design*. arXiv. <https://doi.org/10.48550/arXiv.2303.07839>