

# Collage: Decomposable Rapid Prototyping for Co-Designed Information Extraction on Scientific PDFs

Sireesh Gururaja<sup>1\*</sup> Yueheng Zhang<sup>2\*</sup>

Guannan Tang<sup>2</sup> Tianhao Zhang<sup>2</sup> Kevin Murphy<sup>2</sup> Yu-Tsen Yi<sup>2</sup> Junwon Seo<sup>2</sup>

Anthony Rollett<sup>2</sup> Emma Strubell<sup>1,2</sup>

<sup>1</sup>Language Technologies Institute, School of Computer Science

<sup>2</sup>Department of Materials Science and Engineering  
Carnegie Mellon University

sgururaj@cs.cmu.edu, yuehengz@andrew.cmu.edu

## Abstract

Recent years in NLP have seen the continued development of domain-specific information extraction tools for scientific documents, alongside the release of increasingly multimodal pretrained language models. While applying and evaluating these new, general-purpose language model systems in specialized domains has never been easier, it remains difficult to compare them with models developed specifically for those domains, which tend to accept a narrower range of input formats, and are difficult to evaluate in the context of the original documents. Meanwhile, the general-purpose systems are often black-box and give little insight into preprocessing (like conversion to plain text or markdown) that can have significant downstream impact on their results.

In this work, we present Collage, a tool intended to facilitate the co-design of information extraction systems on scientific PDFs between NLP developers and scientists by facilitating the rapid prototyping, visualization, and comparison of different information extraction models on the content of scientific PDFs. For scientists, Collage provides side-by-side visualization and comparison of multiple models of different input modalities in the context of the PDF content they are applied to; for developers, Collage allows the rapid deployment of new models by abstracting away PDF preprocessing and visualization into easily extensible software interfaces. We also enable both developers and scientists to inspect, debug, and better understand modeling pipelines by providing granular views of intermediate states of processing. We demonstrate our system in the context of information extraction to assist with literature review in materials science.

## 1 Introduction

In recent years, systems based on large language models (LLMs) have broadened the public visibility

\* Equal contribution.

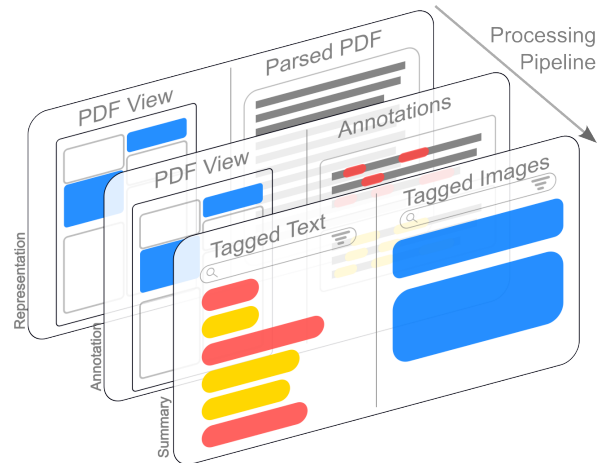


Figure 1: Collage allows users to inspect multiple models in different modalities by presenting a stage-by-stage, decomposed view of the PDF modeling pipeline. Here, we see a PDF composed of text and tables, with entities from different models shown in red and yellow. The summary view shows extracted content, while annotations and inspection views allow the user to step back in the modeling pipeline

ity of developments in NLP. With the advent of tools that have publicly accessible, user-friendly interfaces, experts in specialized domains outside NLP are empowered to use and evaluate these models inside their domains, for example to automatically mine insights from scientific literature. Further, an increasing number of these tools are multimodal, handling not only text, but frequently images, or even PDFs directly. However, despite the accessibility of these tools, the processing pipelines they employ remain as end-to-end black boxes and provide little interpretability or debuggability in case of failure. Further, these systems usually rely only on large, deployed models, potentially leaving other user priorities, such as interpretability, efficiency, or domain specialization, unaddressed.

Domain specific research in domains like clinical (Naumann et al., 2023), legal (Preotiuc-Pietro et al., 2023), and scientific (Knoth et al., 2020; Co-

han et al., 2022) NLP have long histories. Models in these areas remain less accessible; in order to run and evaluate these models on your own data, custom code is often needed. Further, because many of these models are text-only, evaluating their results in the context of their eventual use — for example, directly on a PDF — poses a challenge.

This paper presents Collage, a tool that facilitates the rapid prototyping, visualization, and comparison, of multiple models across modalities on the contents of scientific PDF documents. Collage was designed to address the interface between developers of NLP-based tools for scientific documents and the scientists who are the intended users of those tools. To address scientists’ needs, we ground our design in a series of interviews with domain experts in multiple fields, with a particular focus on materials science. Further, in cases where model results may not meet scientists’ or developers’ expectations, we visualize the intermediate representation at each step, giving the user a granular view of the modeling pipeline, allowing shared debugging processes between developers and users. Collage is domain-agnostic, and can visualize any model that conforms to one of its three interfaces - for token classification models, text generation models, and image/text multimodal models. We provide implementations of these interfaces that allow the use of any HuggingFace token classifier, multiple LLMs, and several additional models without requiring users to write any code. All of the interfaces are easily implemented, and we provide instructions and reference implementations in our repository <sup>1</sup>.

## 2 Motivation

Collage is based on collected themes from interviews with 15 professionals across materials science, law, and policy, in which the authors ask about their practices for working with large collections of documents. For a reasonable scope, we focus on the 9 materials scientists in our sample, whose responses concern their process of literature review. We focus on three themes that emerged consistently from these interviews to inform our design of Collage:

**Varied focuses.** One of the most prominent themes to emerge in our interviews is the variety of focuses that scientists, even in very closely related subfields, can have when reading a paper and

<sup>1</sup>[github.com/gshireesh/ht-max](https://github.com/gshireesh/ht-max)

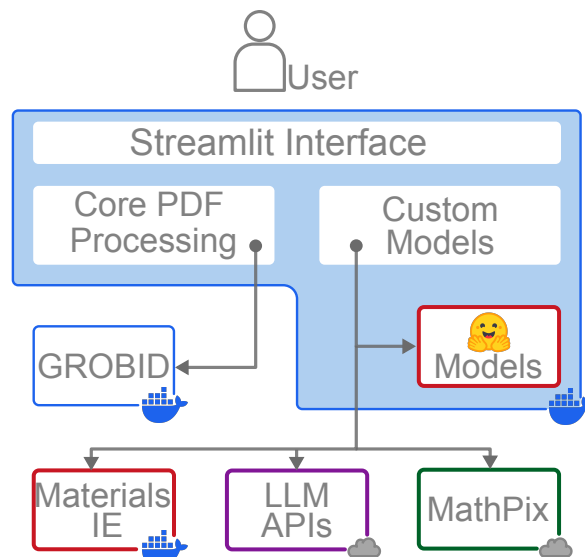




Figure 2: System architecture with currently implemented models. All custom models implement our interfaces, outline color indicates which: **Token Classification**, **Text Generation**, or **Image Processing**.  indicates components running in the same Docker container, and  indicates models running in the cloud. "Materials IE" refers to materials-specific models, like ChemDataExtractor.

evaluating it for relevance to their purpose. While many participants focused on paper metadata, such as the reputation of the publication venue or citation count, others focused on cues from within the content of the paper. For the design of Collage, we focus on accelerating co-design of models that address specific information extraction needs on paper content, by reducing the burden of deploying new models on PDF content, and providing a shared, user-friendly view of the results upon which scientists and developers can base subsequent efforts.

**Information in tables.** As pointed out above, many of our participants relied heavily on information provided in tables, rather than solely in the document text. As such, an important concern in the design of Collage would be to allow multimodality in the models that it interfaces with and visualizes.

**Older documents.** Our participants noted that they regularly work with documents across a wide time range. Several participants noted that the work that they relied on most frequently were technical reports from the 1950s to the 1970s. These reports are now digitized, but are otherwise highly variable in their accessibility to modern processing tools:

The OCR used when digitizing them can be inaccurate, they often contain noise in the scanned images, and layouts are less standardized. This can lead to confusion on whether issues with performance are the fault of models themselves, or preprocessing choices that cause that degraded performance. We therefore aim to provide an interface that allows users to inspect intermediate stages of processing, to better understand where a model may have failed, and what subsequent development should target next: whether better performing models, or better preprocessing.

### 3 Design and Implementation

We conceptualize our system in three parts: PDF representation, which parses and makes the content of PDFs easily available to downstream usage; modeling, i.e. applying multiple models to that PDF representation, backed by common software interfaces, which facilitate the rapid extension of the set of available models; and a frontend graphical interface that allows users to visualize and compare the results of those models on uploaded PDFs. We discuss the design choices and implementation details of each stage in the following subsections, and show an architectural overview in Figure 2.

#### 3.1 PDF Representation

To produce a PDF representation amenable to our later processing, we build a pipeline on top of the PaperMage library (Lo et al., 2023), which provides a convenient set of abstractions for handling multimodal PDF content. PaperMage allows the definition of Recipes, i.e. combinations of processing steps that can be reused. We base our pipeline off of its CoreRecipe pipeline, which identifies visual and textual elements of a paper, such as tables and paragraphs.

We then introduce several new components to the CoreRecipe, to make the paper representation more suitable to our use case. First, we introduce a parser based on Grobid (GRO, 2008–2023), which provides a semantic grouping of paragraphs into structural units, allowing us to segment processing and results by paper section. Second, to address issues with text segmentation in scientific documents, we replace PaperMage’s default segmenter (based on PySBD) with a SciBERT (Beltagy et al., 2019)-based SciSpaCy (Neumann et al., 2019) pipeline.

At the end of this stage of processing, we have the PaperMage representation of a document, in

```
class LiteLlmCompletionPredictor(TextGenerationPredictorABC):
    def __init__(
        self,
        model_name: str,
        api_key: str,
        prompt_generator_function: Callable[[str], List[LLMMessage]],
        entity_to_process="reading_order_sections",
    ):
        super().__init__(entity_to_process)
        self.model_name = model_name
        self.api_key = api_key
        self.generate_prompt = prompt_generator_function

    def generate_from_entity_text(self, entity: Entity) -> str:
        messages = [asdict(m) for m in self.generate_prompt(entity.text)]
        llm_response = completion(
            model=self.model_name, api_key=self.api_key, messages=
            messages, max_tokens=2500
        )
        response_text = llm_response.choices[0].message.content
        return response_text
```

Figure 3: Partial implementation of the TextGenerationPredictor to allow LLM predictions given an Entity extracted from the PDF. LLMMessage is a data class wrapper around the system and user messages for LLMs in the OpenAI format. Not shown are the property declarations; full listing can be found in our code repository.

the form of Entity objects, organized in Layers. Entity objects can be e.g. individual paragraphs by section or index, images of tables, and individual sentences.

#### 3.2 Modeling and Software Interfaces

To facilitate the easy implementation of new information extraction tools, we define common interfaces that simplify the process of adding additional processing to a document’s content. These interfaces standardize three kinds of annotation on PDF content, allowing users convenient access to the PDF’s content as images or strings (though they can access the PaperMage representation) and automatically handling visualization in several supported formats. This requires users to implement only a few simple functions in the modalities their models already use. All models currently in Collage are implementations of these interfaces. We describe the interfaces, the requirements for implementation, and current implementations below. All interfaces are defined in the papermage\_components/interfaces package of our repository. In order to add a new custom processor, users must define a class that extends one of the interfaces specified below, and then register their predictor in the local\_model\_config.py module.

Figure 4: LLM Selector, as it appears in the File Upload view. Users specify an LLM to query, enter their API key, customize the prompt for an LLM, and repeat for any number of LLMs and prompts.

**Token Classification Interface:** This interface is intended for any model that produces annotations of spans in text, i.e. most “classical” NER or event extraction models. Users are required to extend the `TokenClassificationPredictorABC` class and override the `tag_entities_in_batch` method, which takes a list of strings to tag, and produces a list of lists of tagged entities per-sentence. Tagged entities are expected to have the start and end character offsets, and the interface’s code automatically handles mapping indices from the sentence level to the document level, and visualizing annotated text using the `displacy` visualizer<sup>2</sup>.

To demonstrate this interface, we provide two implementations: one with a common materials information extraction system, `ChemDataExtractor2` (Swain and Cole, 2016; Mavracic et al., 2021), which we wrap in a simple REST API and Dockerize to streamline environment and setup, as well as a predictor that can apply any HuggingFace model that conforms to the `TokenClassification` task on the HuggingFace Hub<sup>3</sup>.

**Text Generation Interface:** Given the prominence of large language model-based approaches, this interface is designed to allow for text-to-text prediction. Users are required to extend the `TextGenerationPredictorABC` class, and to

implement the `generate_from_entity_text()` method, which takes and returns a string. This basic setup allows users to e.g. prompt an LLM and display the raw response. A popular prompting method, however, is to request structured data e.g. in the form of JSON. To accommodate this, and to allow for aggregating LLM predictions into a table, users can also implement the `postprocess_text_to_dict()` method. The default implementation of this method attempts to deserialize the entirety of the LLM response into a dictionary, but users can implement custom logic.

Our implementation of this interface uses `LiteLLM`<sup>4</sup>, a package that allows accessing multiple commercial LLM services behind the same API. We allow users to specify the endpoint/model, their own API key, and a prompt, and display predictions from that model. We show a partial implementation of this predictor in Figure 3, and a sample of its results in Figure 5.

**Image Prediction Interface:** Given the focus on tables and charts that many of our interview participants discussed, and the fact that table parsing is an active research area, we additionally provide an interface for models that parse images, the `ImagePredictorABC` in order to handle multimodal processing, including tables. This interface allows users two options of method to override: In cases where only image inputs are needed (e.g. if a table extractor performs its own OCR), the `process_image()` method; in cases where the method is inherently multimodal, implementors can instead override the `process_entity()` method, which allows them full access to PaperMage’s multimodal Entity representation. This interface requires implementors to return at least one of three types of data: a raw string representation, which we view as useful for e.g. image captioning tasks; a tabular dictionary representation, for the case of table parsing; or a list of bounding boxes, in the case of models that segment images. Implementations of this interface are free to return more than one type of output; all of them will be visualized in the frontend.

We demonstrate implementations of both types. For raw image outputs, we implement a predictor that calls the `MathPix` API<sup>5</sup>, a commercial service for PDF understanding. For the multimodal approach, we implement a predictor that builds on

<sup>2</sup><https://demos.explosion.ai/displacy-ent>

<sup>3</sup>Model list available [here](#).

<sup>4</sup><https://docs.litellm.ai/>

<sup>5</sup><https://mathpix.com/>



the Microsoft Table Transformer model (Smock et al., 2023). This model predicts bounding boxes around table cells, which we then cross-reference with extracted PDF text in the PaperMage representation to provide parsed table output. An example of parsed table output from this predictor can be seen in figure 5.

### 3.3 Visualization Frontend

We present the results of the PDF processing in an interactive tool built using Streamlit<sup>6</sup> that allows the user – whether scientist or developer – to upload a PDF, define a processing pipeline, and inspect the results of that processing pipeline at each stage. More concretely, after the paper is uploaded and processed, we present the results of the pipeline in three views, in decreasing order of abstraction from the paper. The intention of this is to first show the user the potential output of their chosen pipeline for a given paper, then allow them to inspect each step of the pipeline that led to that final output. Each view is described in more detail below, and has a screenshot in Appendix A.

**File Upload and Processing.** The first view we present to a user allows them to upload a file, and to define the processing pipeline applied to that file. Basic PDF processing is always performed, and users can then toggle which custom models will be run. Users can additionally specify any number of HuggingFace token classification models or LLMs with the provided widget, which allows users to search the HuggingFace Hub, select LLMs, and customize the prompts for them. We show a view of the LLM model selector in Figure 4.

**File Overview.** This view presents the high-level extracted information from the paper, as candidates for what could be shown to the user as part of their search process. In particular, we show a two-column view, with tables of tagged entities from both token-level predictors and LLMs on the left, and the processed content of images on the right. Users can filter based on sections, to e.g. find materials mentioned in the methods section of a paper. If the user finds the content extracted with the pipeline useful, the model and processing pipeline could be further developed into a more integrated prototype. If not, the user can proceed to the succeeding views, to see where models may have failed.

<sup>6</sup><https://streamlit.io>

**Annotations.** This view allows the user to compare the results of models in the context of the PDF. We present another two-column view, in which the PDF is visualized on the left, and allows the user to select a paragraph or table at a time, and visualize the results of each model on it. In the case of text annotation, we visualize the entities identified by token prediction models as well as predictions from LLMs. In the case of images, all of the available output types from the image processing interface are visualized. We show a composite screenshot of this interface in Figure 5.

**Representation Inspection.** This view presents visualization of the PDF representation available to any downstream processing that the user might select. In the sidebar, users can choose to visualize any PaperMage Layer, i.e. set of Entity objects, tagged by the basic processing steps. Then, in a view similar to the raw annotations view, they can see all of those entities highlighted on the PDF in the left-side column. Once the user selects an object, they see the raw content extracted from that object in the right-side column, in the form of its image representation and the text extracted from it, along with the option to view how the text is segmented into sentences. This view allows users to inspect how the PDF processing choices may have affected the text they send to models, which often have significant effects on their downstream performance (Camacho-Collados and Pilehvar, 2018).

## 4 Addressing Needs from Interviews

Our system is specifically designed to respond to the concerns raised in our interviews. First, to accommodate the varied processes of materials scientists, we design interfaces that allow for easy implementation of new models into our framework; our existing implementations of those interfaces also allow for the application of multiple LLMs and HuggingFace models directly in the context of the PDFs under review. This allows users to search for and evaluate models that suit their existing workflows. For tables, we both provide an interface and implementations that allow the comparison of proprietary and open-source table parsing systems. Extending this work to new table models and evaluating them is simplified by our software and visualization interfaces. Our inspection view is designed to address concerns about older PDFs: in being able to inspect the results of processing, users and engineers of this system can identify fail-

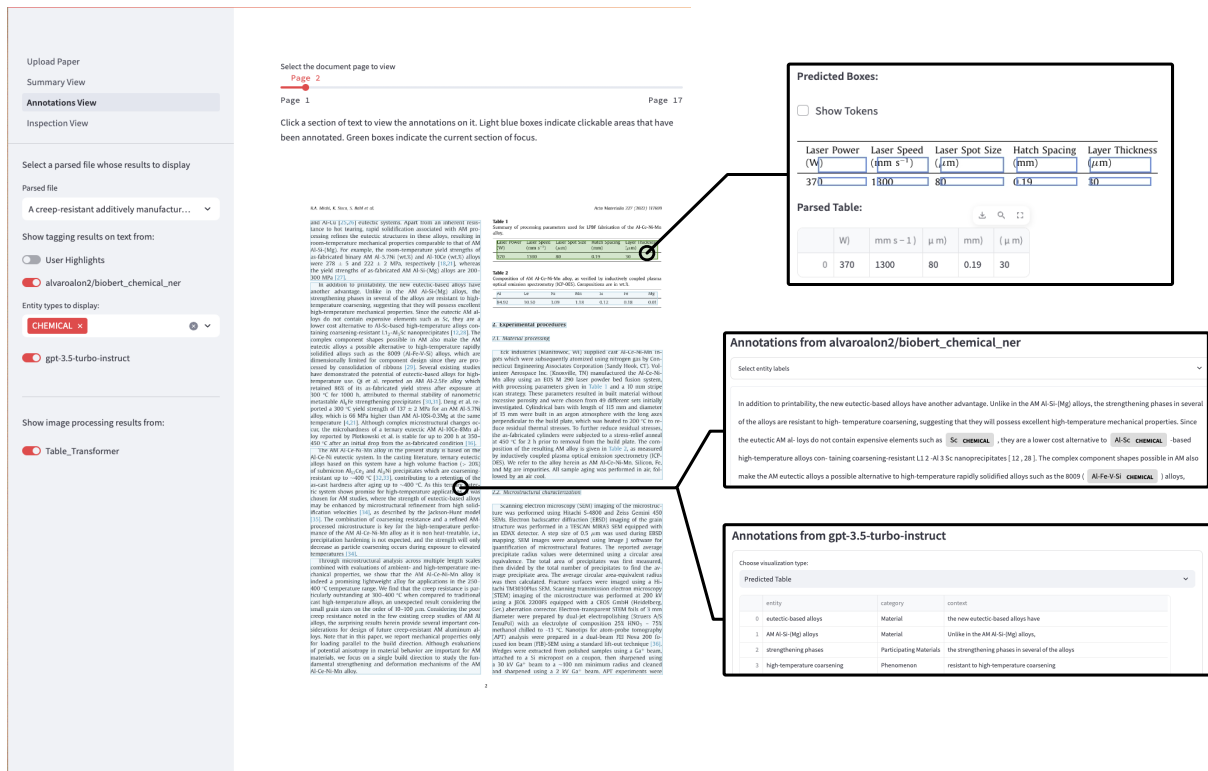


Figure 5: The annotations view. On the left, a screenshot showing the sidebar, allowing file and model selection, and the left pane, a visualization of the PDF with clickable regions highlighted. On the right, screenshots showing visualizations from the Table Transformer model with bounding boxes and parsed table (top), a HuggingFace transformer model with token-level tags (middle), and GPT-3.5 Turbo, with JSON output parsed into a table (bottom).

ure modes in both the upstream and downstream processing.

## 5 Co-design with Collage

In this section, we walk through an example of how Collage might facilitate the development of an information extraction pipeline for a materials scientist. In this scenario, Bob, a materials scientist, wishes to extract the synthesis parameters of a class of materials called zeolites from a dataset of PDFs from the 1980s to the 2010s. Papers discussing Zeolite synthesis often report parameters both in the text of the paper as well as in tables, so multimodal extraction is crucial. He has worked with Alice, an NLP developer, before but they have not yet collaborated on this project.

**Evaluating off-the-shelf models.** Bob begins in Collage by trying to see if there is an existing model that already works for his case. Using the Hugging-Face model selector in the upload paper view, he searches for tagging models, but only finds models trained on general scientific or biomedical text, not materials. He is, however, able to write a prompt

for an LLM model to extract this information, and he adds predictors that call out to two popular commercial models to extract the information that he’s interested in. He uploads a recent paper that he’s been reading, and waits for Collage to process it.

**Finding modeling opportunities.** Once Collage has processed the paper, Bob heads to the **summary view**, and compares the results from the two commercial models. He’s able to view the parameters that they extracted, filtered by section, to develop an understanding of what heuristics might get him the information he wants: parameters identified in the related works section, for example, are frequently irrelevant to his search. In the summary view, he’s also able to see the tables that Collage has identified and parsed with the Table-Transformer and MathPix models, along with their labels and captions, and the tagged bounding boxes for the table cells.

To make sure those annotations are reasonable, he heads to the **annotations view**, where he can visualize the extracted information side-by-side with the original PDF content, and compare the annota-

tions from his two LLMs. He's also able to check whether the table detection model has predicted sensible bounding boxes that both don't exclude content like table footnotes, but also don't include irrelevant, non-table content. He notes that while the table parsing from both models is reasonable, the paper he's reading reports values in ratios that may not be comparable across papers. To have a single pipeline that produces normalized results, he'd like to use a multimodal LLM, but in Collage currently, LLMs can only be applied to text. He decides to get in touch with Alice, to see if she can develop an LLM-based table information extraction model.

**Prototype model development.** Alice begins work on a table information extraction tool, but there are a lot of possible options to evaluate: should she use a multimodal model and process the table in image format? Should she linearize the table into text, and have a text-only LLM work with it? In Collage, both options involve little more than implementing the LLM call, so it's easy to do both and then compare. For the multimodal case, Alice extends the **image predictor interface**, which allows her to receive as input the cropped image of any element on the page and pass that to an LLM; for the text-only case, Alice can easily access the underlying document representation use the already identified and parsed tables (which are in a DataFrame-compatible format) and convert them into markdown for her linearization. She is able to return a dictionary in the same schema for both predictors, which will automatically be visualized in the frontend as a Pandas dataframe. She commits her code, registers the predictors, and asks Bob to take a look in the Collage interface.

**In-context evaluation.** Bob then re-processes his paper through Collage, making sure to check the boxes for Alice's new table parsing predictors. In the summary view, he's able to compare the predicted, normalized tables to the original PDF, to verify that the models are performing the normalization correctly. He then picks the better performing model, and asks Alice to create a pipeline that can process his entire dataset. Alice is able to take the predictor, add it to the PaperMage recipe that underlies Collage, and run it over Bob's set of PDF documents, adding a step to export the parsed tables that Bob saw in the Collage interface.

**Diagnosing errors.** Bob looks through the parsed tables from processing all of the PDFs, and notes that for the older PDFs, the parsed content doesn't look right. He'd like to diagnose the problem. Because the processing that Alice and Bob run on these documents is the same as that underlying Collage, the results can be visualized in the tool, even if they were not directly processed through it. Bob loads the representation of the parsed older document, and is able to view the results from the model that didn't look right. While the bounding boxes for the table look correct in the annotations view, he's also able to see in the **inspection view** that the text detected within the table has not been correctly OCR'd. He can now contact Alice to see if there's a fix for that problem, but in the meantime, he can examine the visualizations for his PDFs to understand how the publication year might affect whether the deployed suite of models can correctly extract and normalize information, and what the cutoff year might be for the results to be trustworthy.

In this case, Collage enables Bob to self-serve cutting-edge NLP for his own use case, requiring that he involve Alice only when Collage's functionality needs extension. When that happens, Bob and Alice can both see results in the same interface, and can discuss errors and how to prioritize new work. When Alice develops new predictors to address Bob's needs, she is required to do no PDF processing or visualization, which are built into the tool, and Bob can evaluate and compare the results of these new predictors in the same interface he's been using the whole time. For debugging, both Bob and Alice have access to the same representation and visualization as a shared source of truth, and collaborate to involve both NLP and subject matter expertise in how to fix the problem. Collage can accelerate the process of collaboration between NLP developers and scientists, allowing for co-design and rapid prototyping with a shared representation.

## 6 Related Work

Collage situates itself at the intersection of tools that offer reading assistance for scientific PDFs and tools that partially automate the process of literature review by means of information extraction. Tools for scientific PDFs often focuses on interfaces that augment the existing PDF with new information, such as citation contexts ([Rachata-](#)

sumrit et al., 2022; Nicholson et al., 2021), or highlights that aid skimming (Fok et al., 2023). However, most of these works are designed around and purpose-built for specific models. By contrast, Collage draws from projects like PaperMage (Lo et al., 2023), by attempting to be model-agnostic, while at the same time providing a visual interface to prototype and evaluate those models.

Scientific information extraction and literature review automation also have long histories. Collage’s focus on materials science was driven by the field’s existing investment into data-driven design (Himanan et al., 2019; Olivetti et al., 2020), which focuses on using information extraction tools to build up knowledge graphs to inform future materials research. This adds to the existing body of work in chemical and material information extraction, including works like ChemDataExtractor (Swain and Cole, 2016; Mavracic et al., 2021) and MatSciBERT (Gupta et al., 2022). Works like Dagdelen et al. (2024) showcase the growing interest in LLM-based extraction; as LLMs increasingly become multimodal, this capability is likely to be used for tasks like scientific document understanding. While all of these tools are intended to be applied to documents from the materials science domain, they do not share an interface: most tools expect plain text, some, like ChemDataExtractor allow HTML and XML documents, and some work with images. Collage aims to be a platform on which multiple competing approaches can be evaluated, regardless of the input and output formats they require.

## 7 Conclusion

In this work, we present Collage, a system designed to facilitate co-design and rapid prototyping of mixed modality information extraction on PDF content between scientists and NLP developers. We focus on a case study in the materials science domain, that allows materials scientists to evaluate models for their ability to assist in literature review. We intend for this work to be a platform on which to evaluate further modeling work in this area.

## Ethics and Broader Impacts

Our interview protocol was evaluated and approved by the Carnegie Mellon University Institutional Review Board as STUDY2023\_00000431.

In developing a tool to facilitate the automated processing of scientific PDFs, we feel that it is im-

portant to acknowledge that that automation may propagate the biases of the underlying models. Particularly in the case of English that does not reflect the training corpora that models were built on top of, models can perform poorly, leading to fewer results from those papers, and the potential to inadvertently exclude them. However, we hope that in providing a tool to inspect model outputs before such automation tools are deployed, that we can encourage critical evaluation and uses of these tools.

## Acknowledgements

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-22-2-0121. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

## References

- 2008–2023. Grobid. <https://github.com/kermitt2/grobid>.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.
- Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 40–46, Brussels, Belgium. Association for Computational Linguistics.
- Arman Cohan, Guy Feigenblat, Dayne Freitag, Tirthankar Ghosal, Drahomira Herrmannova, Petr Knoth, Kyle Lo, Philipp Mayr, Michal Shmueli-Scheuer, Anita de Waard, and Lucy Lu Wang, editors. 2022. *Proceedings of the Third Workshop on Scholarly Document Processing*. Association for Computational Linguistics, Gyeongju, Republic of Korea.
- John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418. Publisher: Nature Publishing Group.



- Raymond Fok, Hita Kambhamettu, Luca Soldaini, Jonathan Bragg, Kyle Lo, Andrew Head, Marti A. Hearst, and Daniel S. Weld. 2023. *Scim: Intelligent Skimming Support for Scientific Papers*. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pages 476–490. ArXiv:2205.04561 [cs].
- Tanishq Gupta, Mohd Zaki, NM Anoop Krishnan, and Mausam. 2022. Matscibert: A materials domain language model for text mining and information extraction. *nlp Computational Materials*, 8(1):102.
- Lauri Himanen, Amber Geurts, Adam Stuart Foster, and Patrick Rinke. 2019. Data-driven materials science: status, challenges, and perspectives. *Advanced Science*, 6(21):1900808.
- Petr Knoth, Christopher Stahl, Bikash Gyawali, David Pride, Suchetha N. Kunnath, and Drahomira Hermannova, editors. 2020. *Proceedings of the 8th International Workshop on Mining Scientific Publications*. Association for Computational Linguistics, Wuhan, China.
- Kyle Lo, Zejiang Shen, Benjamin Newman, Joseph Z Chang, Russell Authur, Erin Bransom, Stefan Candra, Yoganand Chandrasekhar, Regan Huff, Bailey Kuehl, et al. 2023. Papermage: A unified toolkit for processing, representing, and manipulating visually-rich scientific documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 495–507.
- Juraj Mavracic, Callum J Court, Taketomo Isazawa, Stephen R Elliott, and Jacqueline M Cole. 2021. Chemdataextractor 2.0: Autopopulated ontologies for materials science. *Journal of Chemical Information and Modeling*, 61(9):4280–4289.
- Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Anna Rumshisky, editors. 2023. *Proceedings of the 5th Clinical Natural Language Processing Workshop*. Association for Computational Linguistics, Toronto, Canada.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. *ScispaCy: Fast and robust models for biomedical natural language processing*. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.
- Josh M. Nicholson, Milo Mordaunt, Patrice Lopez, Ashish Uppala, Domenic Rosati, Neves P. Rodrigues, Peter Grabitz, and Sean C. Rife. 2021. *scite: A smart citation index that displays the context of citations and classifies their intent using deep learning*. *Quantitative Science Studies*, 2(3):882–898.
- Elsa A Olivetti, Jacqueline M Cole, Edward Kim, Olga Kononova, Gerbrand Ceder, Thomas Yong-Jin Han, and Anna M Hiszpanski. 2020. Data-driven materials research enabled by natural language processing and information extraction. *Applied Physics Reviews*, 7(4).
- Daniel Preotiuc-Pietro, Catalina Goanta, Ilias Chalkidis, Leslie Barrett, Gerasimos (Jerry) Spanakis, and Nikolaos Aletras, editors. 2023. *Proceedings of the Natural Legal Language Processing Workshop 2023*. Association for Computational Linguistics, Singapore.
- Napol Rachatasumrit, Jonathan Bragg, Amy X. Zhang, and Daniel S Weld. 2022. *CiteRead: Integrating Localized Citation Contexts into Scientific Paper Reading*. In *27th International Conference on Intelligent User Interfaces, IUI '22*, pages 707–719, New York, NY, USA. Association for Computing Machinery.
- Brandon Smock, Rohith Pesala, and Robin Abraham. 2023. Aligning benchmark datasets for table structure recognition. In *International Conference on Document Analysis and Recognition*, pages 371–386. Springer.
- Matthew C. Swain and Jacqueline M. Cole. 2016. *ChemDataExtractor: A Toolkit for Automated Extraction of Chemical Information from the Scientific Literature*. *Journal of Chemical Information and Modeling*, 56(10):1894–1904. Publisher: American Chemical Society.

## A Appendix: Screenshots of Interface Views

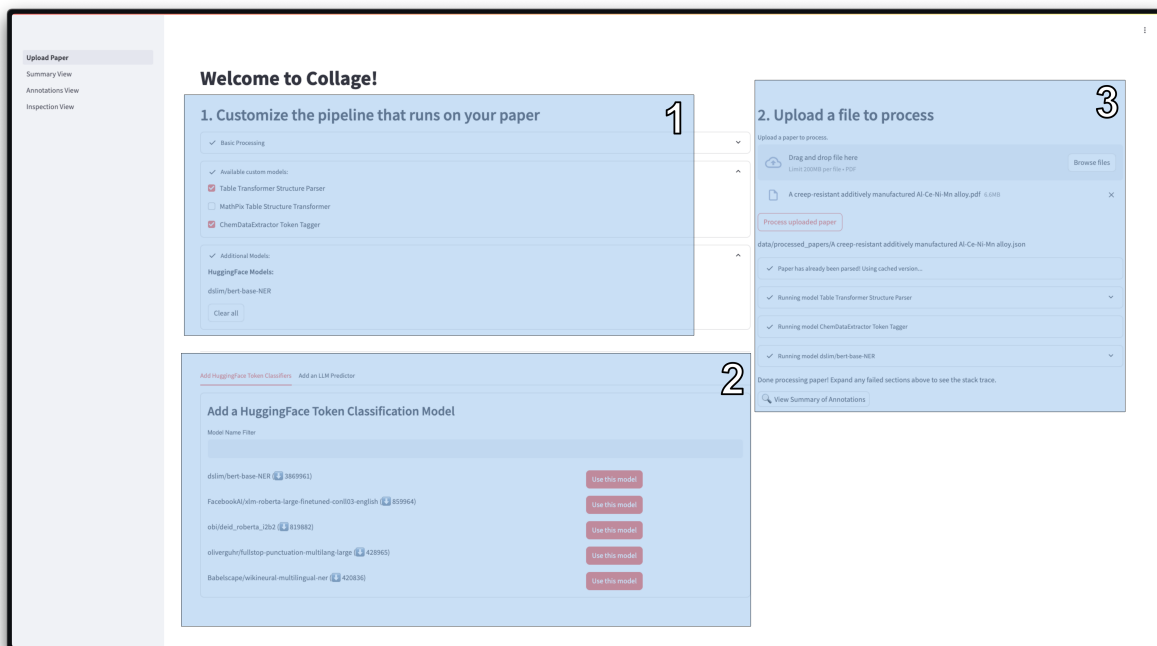


Figure 6: The Upload Paper view, showing (1) The currently selected models, (2) widget for selecting HuggingFace and LLM Classifiers, (3) File upload and progress visualization.

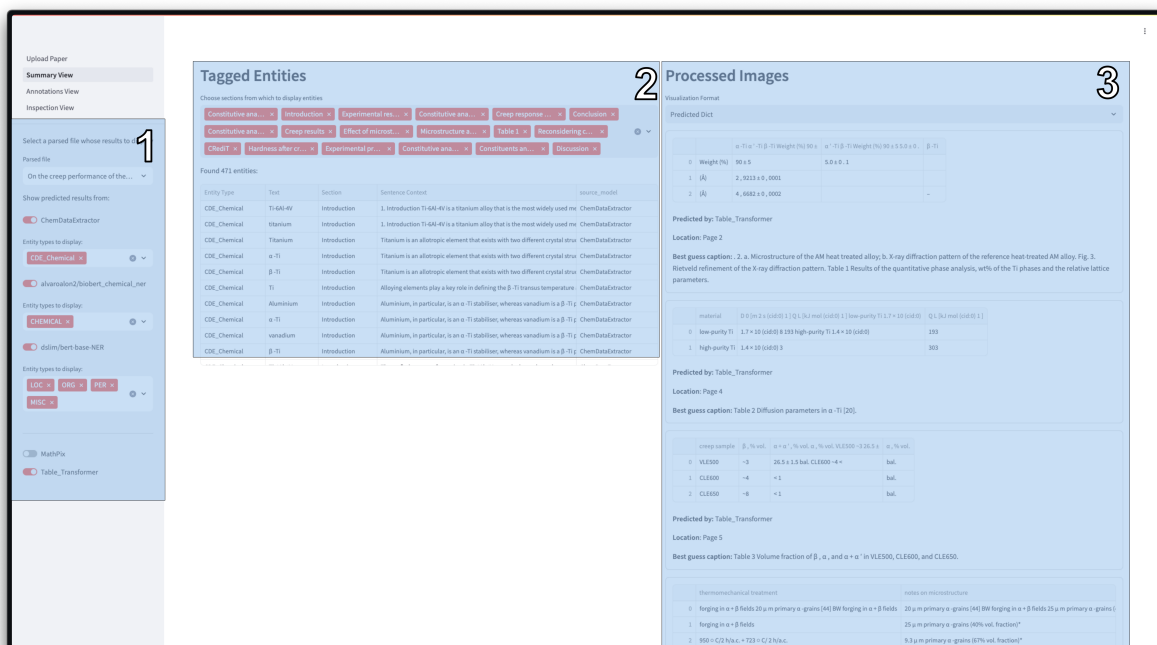


Figure 7: The Summary view, showing (1) the sidebar allowing model and entity type selection, (2) visualized tagged entities from the selected tagging models, (3) visualized image processing results.



Figure 8: The Annotations view, showing (1) the sidebar allowing model and entity type selection, (2) the visualized PDF, showing clickable regions (3) visualized annotations on the clicked region.

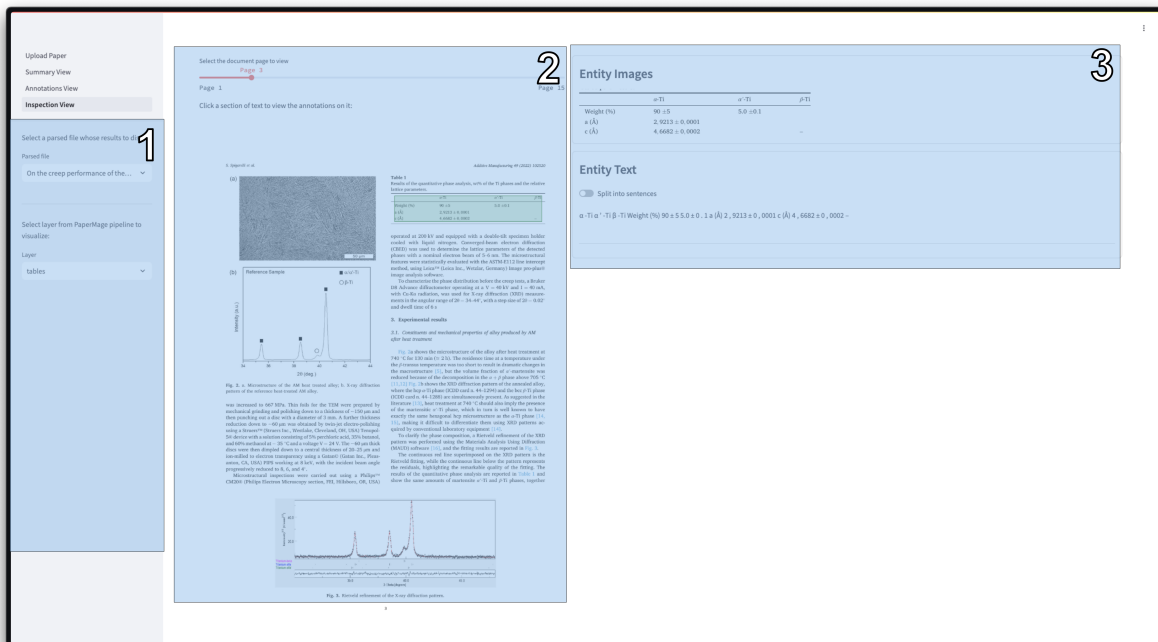


Figure 9: The Inspection view, showing (1) the sidebar allowing PaperMage layer selection, (2) the visualized PDF, showing clickable regions (3) the image and the text of the selected Entity