Data Curation of a Findable, Accessible, Interoperable, Reusable Polymer Nanocomposites Data Resource - Materialsmine

by

Bingyin Hu

Department of Mechanical Engineering and Materials Science
Duke University

Date:_______________________

Approved:

___________________________
L. Catherine Brinson, Supervisor

___________________________
Wei Chen

___________________________
Samuel Wiseman

___________________________
Stefano Curtarolo

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mechanical Engineering and Materials Science in the Graduate School of Duke University

2022
ABSTRACT

Data Curation of a Findable, Accessible, Interoperable, Reusable Polymer Nanocomposites Data Resource - Materialsmine

by

Bingyin Hu

Department of Mechanical Engineering and Materials Science
Duke University

Date:_______________________

Approved:

___________________________
L. Catherine Brinson, Supervisor

___________________________
Wei Chen

___________________________
Samuel Wiseman

___________________________
Stefano Curtarolo

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mechanical Engineering and Materials Science in the Graduate School of Duke University

2022
Abstract

A polymer nanocomposite (PNC) is a composite material consisting of a polymer matrix and stiff fillers with at least one dimension smaller than 100 nm. With the addition of a small amount of filler to the polymer matrix, PNC demonstrates large reinforcement of mechanical, viscoelastic, dielectric, thermal, optical, and other physiochemical properties as compared to pure polymer or pure fillers acting alone. PNCs have thus attracted significant amounts of research interest over recent years.

To accelerate materials design, we need findable, accessible, interoperable, and reusable (FAIR) data resources to provide sufficient data for data-driven approaches to replace the traditional trial-and-error style of exploration in a lab. With the goal to build a FAIR data resource for the PNC community, we built NanoMine in 2016, which later evolves into MaterialsMine with the extension of MetaMine in the metamaterial domain. To be FAIR, we need a clear and extensible data representation to enable the interoperable knowledge exchange. We thus designed the NanoMine XML schema.

With the data framework and data representation in place, we still need tools and a user-friendly interface for data curation. This dissertation describes in detail the tools and data interfaces we developed to ensure a smooth data curation pathway for NanoMine/MaterialsMine. To reduce and prevent curation errors and thus improve data
quality, we need data validation mechanisms. To address the need, we discuss the validation mechanisms embedded both during and after the curation.

On many occasions, even without human-caused curation errors, the data resource cannot perform to its full capacity due to data inconsistencies. For example, the inconsistency of polymer indexing caused by the lack of uniformity in the expression of polymer names and the inconsistent use of mass fraction and volume fraction in specifying the composite composition. To address the need for data standardization, tools developed to bypass manual curation, the mass fraction – volume fraction conversion agent, and ChemProps, a RESTful API-enabled multi-algorithm-based polymer/filler name mapping methodology, are discussed in detail in this dissertation.

To create truly powerful and transformative materials design paradigms and towards a sustainable future for MaterialsMine, we need to harness the power of AI to efficiently extract a significant set of data from the published, archival literature. Natural Language Processing (NLP) offers an opportunity to make this data accessible and readily reusable by humans and machines. The first step is to generate a sample list where curators can easily find the number of samples, their compositions, and properties reported in the paper. The task is handled in a pre-training-finetuning fashion. Downstream tasks include Named Entity Recognition (NER) to detect sample code, sample composition, property, and group reference to samples in the captions, and Relation Extraction (RE) which predicts the relations between pairs of detected named
entities. In this dissertation, a detailed discussion of how the two corpora for pre-training and finetuning are constructed is provided. A T5-base model pre-trained on the caption-mention corpus and fine-tuned for the NER and RE tasks is proposed. We evaluated it along with an array of BERT-based models and seq2seq models for potential applications in the semi-automated curation pipeline for MaterialsMine.
Dedication

To my family.
Contents

Abstract ......................................................................................................................................... iv

List of Tables .................................................................................................................................. xiii

List of Figures .................................................................................................................................. xiv

Acknowledgements ......................................................................................................................... xvii

1 Introduction ...................................................................................................................................... 1

1.1 Needs for polymer nanocomposite materials ............................................................. 1

1.2 Needs for FAIR data resources .................................................................................. 1

1.3 Challenges in building a FAIR data resource for the materials science community .............................................................. 5

1.4 Challenges in data curation .......................................................................................... 6

1.5 Challenges in data validation and standardization ................................................... 7

1.6 Challenges in the sustainability of PNC data resource ........................................... 8

1.7 Research tasks .............................................................................................................. 9

1.8 Dissertation outline ................................................................................................... 11

2 Technical background ............................................................................................................ 12

2.1 Materials informatics ................................................................................................. 12

2.2 XML and XML schema ............................................................................................ 15

2.3 Curation pathway ....................................................................................................... 17

2.3.1 XML webform-based data curation ................................................................ 17

2.3.2 Excel-spreadsheet template-based data curation ............................................. 19

2.4 String matching and string comparison .................................................................... 21
4.2.3 XML data update agent ................................................................. 51
4.3 Conclusion ......................................................................................... 51

5 NanoMine data standardization ........................................................... 53
5.1 DOI crawler ....................................................................................... 53
5.2 ChemProps ....................................................................................... 54
  5.2.1 Introduction ................................................................................... 54
  5.2.2 Methods ....................................................................................... 57
    5.2.2.1 Structure of ChemProps API ............................................... 57
    5.2.2.2 Data ....................................................................................... 60
    5.2.2.3 Weighted voting algorithm for name mapping ...................... 63
  5.2.3 Results ......................................................................................... 70
    5.2.3.1 Optimization ....................................................................... 70
    5.2.3.2 Example of API access for ChemProps ............................... 72
    5.2.3.3 Example of GUI access for ChemProps.............................. 74
  5.2.4 Discussion .................................................................................... 76
  5.2.5 Conclusion ................................................................................... 79
5.3 Mass fraction - volume fraction conversion agent ............................ 81
5.4 Spectral data header parser ............................................................ 82
  5.4.1 Overview ..................................................................................... 82
  5.4.2 Data preparation .......................................................................... 84
  5.4.3 Name-unit separation ................................................................. 85
  5.4.4 Decision space pruning ............................................................... 86
List of Tables

Table 5-1: A list of input information of a GET request. ............................................................57
Table 5-2: A list of returned information of a GET request if successful (Code 200). ..........58
Table 5-3: Summary of the proposed mapping algorithms. ......................................................64
Table 5-4: Results of weight vector optimization and ten-fold validation...............................70
Table 5-5: Output name-unit pairs of inputting “CP (J/(g*K))” into name-unit separator. ............85
Table 6-1: Results on the NER task .........................................................................................106
Table 6-2: Results on the RE task. .........................................................................................107
List of Figures

Figure 2-1: Two XML’s containing identical materials data but with different structures. ................................................................................................................................................................................................................................................................. 15
Figure 2-2: A screenshot of the NanoMine webform-based curation system in 2017......18
Figure 2-3: Methods used to build hierarchy in Excel template.................................................. 20
Figure 2-4: A typical NLP pipeline. The shaded area is merged into one step for the deep learning model......................................................................................................................................................... 25
Figure 2-5: The architecture of the Transformer model. Modified from [75]. .........................29
Figure 3-1: Workflow of Excel-based data curation in NanoMine. ........................................ 32
Figure 3-2: Example of a sample list used for NanoMine curation........................................ 33
Figure 3-3: A screenshot of the toolbox design in the "3. Synthesis and Processing” worksheet in the master template......................................................................................................................................................... 35
Figure 3-4: A screenshot of the data uploader. .......................................................................... 36
Figure 3-5: A screenshot of the data uploader with the pop-up on successful job submission. ........................................................................................................................................................................ 38
Figure 3-6: An example email notification of a successful conversion................................. 39
Figure 3-7: An example email notification of an unsuccessful conversion......................... 39
Figure 3-8: The detailed workflow of XMLCONV. Yellow boxes indicate validation units that will terminate the process......................................................................................................................................................... 40
Figure 3-9: An example supplementary data file................................................................. 44
Figure 4-1: A dropdown menu in the master template for the unit of temperature......... 47
Figure 4-2: The easy csv plotter in MaterialsMine. ............................................................. 48
Figure 4-3: An example of a curation error detected by the XML schema validation. ..... 49
Figure 4-4: Viz validation GUI (left) and the original plot (right)................................. 50
Figure 5-1: The structure of the ChemProps API ................................................................. 59

Figure 5-2: Illustration of the data ingestion process, BOC correspondence are color coded ............................................................................................................................................ 62

Figure 5-3: Screenshots of synonyms of polystyrene on PubChem[119] and ChEBI[120]. ....................................................................................................................................................... 63

Figure 5-4: Structure of training data for weight optimization. ........................................ 67

Figure 5-5: Example of the score difference derivation. ....................................................... 69

Figure 5-6: The ChemProps Graphical User Interface[121]. .............................................. 75

Figure 5-7: An illustration of how the spectral data axis parser works. Input: <headers>. Output: <AxisLabel>.......................................................................................................................................................... 83

Figure 5-8: A screenshot of the dataset prepared for the spectral data header parser..... 84

Figure 5-9: Illustration of the inference by the merged graph with an example input.... 86

Figure 6-1: Our vision of the NLP-driven curation pathway for MaterialsMine.............. 94

Figure 6-2: Summary of the construction of the caption-mention corpus................... 95

Figure 6-3: A screenshot of the caption corpus annotation with doccano....................... 98

Figure 6-4: Pre-training history of T5 model with caption-mention corpus............... 105

Figure 6-5: Proposed pipelines for the application of NER and RE models in MaterialsMine data curation............................................................................................................................................. 109

Figure 7-1: A schematic of the proposed AI-assisted Data Ingestion interface............. 117

Figure 7-2: A proposed end-to-end pipeline for sample list generation in MaterialsMine. .................................................................................................................................................................................................. 120

Figure A-1: The landing page of the Duke OneLink account registration .................... 122

Figure A-2: Illustration of the request API token button on the ChemProps GUI page. 123

Figure A-3: Example token request page if users are not logged in ............................ 123
Figure A-4: Example token request page for first-time logged-in users ......................... 124

Figure A-5: Example of a successfully requested API token ........................................ 124
Acknowledgements

It has been a wonderful five years doing my Ph.D. at Duke University. I am grateful for all the help and support I received from many people along the way.

First and foremost, I would like to thank my advisor Dr. L. Catherine Brinson for her support, guidance, consideration, and trust. It is sincerely appreciated that I was granted the freedom to dig into the brand-new world of NLP. Your advice beyond research is lifelong beneficial to me. I would like to thank Dr. Samuel Wiseman for your guidance, inspiration, and help on NLP that transforms my interest into a well-defined research problem. I would also like to thank the other two of my committee members, Dr. Wei Chen from Northwestern University and Dr. Stefano Curtarolo for their suggestions on my research works and my dissertation.

I would like to thank all my colleagues in Brinson Advanced Materials Lab, Dr. Marc Palmeri, Dr. Pavan Kolluru, Dr. Xiaolin Li, Dr. Min Zhang, Dr. Yixing Wang, Dr. Anqi Hu, Dr. David Collinson, Dr. Richard Sheridan, Ms. Heer Majithia, Ms. Prajakta Prabhune, Dr. Boran Ma, Mr. Nicholas Finan, Dr. Mary Bastawrous, Dr. Shana McAlexander, Ms. Defne Cirici. Thanks to Mr. Erik Daughtrey and Mr. Tolulomo Fateye, for helping me and guiding me to bring many of my prototype research-level codes to production. It is always fun to chat with you Tolu.

I would like to thank my collaborators on the CSSI MaterialsMine project. Dr. Linda Schadler from University of Vermont, Dr. Wei Chen from Northwestern
University, Dr. Deborah McGuinness and Ms. Jamie McCusker from Rensselaer Polytechnic Institute, Dr. Chiara Daraio from California Institute of Technology, and Dr. Cynthia Rudin from Duke University for inspiring discussions and amazing endeavors to build the pioneering MaterialsMine data resource for the community. We are one heck of a team!

I would like to thank my friends from CMU, Dr. Xidong Pi, Dr. Susu Xu, Dr. Pinchao Zhang, Dr. Wei Ma, Dr. Shuguan Yang, Dr. Guannan He, and Dr. Zihang Dai, for supporting me in my decision to join Duke. I am fortunate to meet Dr. Haibei Zhu and Dr. Fan Wang in Durham, thank you for being a good friend and a helpful neighbor.

Last but not least, I would like to sincerely express my gratitude to my family. Without the unconditional support from my parents, Mr. Daming Hu and Mrs. Weining Li, I would never be where I am. Though I have not been home during the last four years due to various issues, I would like you to know that I miss you. You demonstrated to me by yourself to be kind, diligent, and up straight. Without your constant encouragement, I would have never reached one milestone after another in my academic life. Now, it is time for me to pay back. Anqi Lin, my love, you are a fighter and I know you have it in you. We have our ups and downs, but you are always there. Thank you for connecting me to Cate for this precious opportunity to join the amazing group and for connecting me to your friend for my job opportunity. You are my lucky star.
1 Introduction

1.1 Needs for polymer nanocomposite materials

A polymer nanocomposite (PNC) is a composite material consisting of a polymer matrix and stiff fillers with at least one dimension smaller than 100 nm[1]. With the addition of a small amount of filler to the polymer matrix, PNCs demonstrate large reinforcement of mechanical, viscoelastic, dielectric, thermal, optical, and other physiochemical properties as compared to pure polymer or pure fillers acting alone [2]–[8]. PNCs have thus attracted significant amounts of research interest over recent years[9].

1.2 Needs for FAIR data resources

The ultimate goal of PNC studies is to understand processing-structure-property (p-s-p) relationships, which thereafter can guide materials design toward applications. The presence of interphase, as a result of interactions between confining surfaces (e.g., nanoparticles) and polymer macromolecules, makes the p-s-p relationships of PNCs especially complicated[10]. In this sense, the traditional trial-and-error style of exploration becomes time-intensive, labor-intensive, and low in efficiency.

In the past decade, significant advances in computational power made possible many data-driven methods. With sufficient data, it has been demonstrated in biomedical, chemistry, and engineering domains, among others, that data-driven methods have the capability to make reliable predictions and inspiring explorations[11]–
The data-driven approach, or materials informatics some may call it, usually requires less time and labor investment as compared to the traditional trial-and-error style of work. With that in mind, the white house launched the Materials Genome Initiative (MGI), bringing a growing opportunity to integrate data science to forge new capabilities in the understanding and design of materials[18], [19].

However, as we mentioned above, sufficient data is a prerequisite for reliable data-driven methods. Unlike the inorganic and metallic alloy domain, polymers and PNCs do not have a well-established protocol for data deposit and sharing other than reporting data points in papers in textual format, because the data space of polymers and PNCs is high in complexity and dimensionality. For individual labs, the only option to apply data-driven methods for materials design is to collect data from their own labs and maybe ask their collaborators for data. First, the time and labor spent in collecting and cleaning data will be intensive. Second, the size of data generated from a single lab or a couple of labs is inevitably limited, especially when there is a huge reservoir of hundreds of thousands of published PNC papers, meaning millions of data points. If we could utilize even ten percent of the reservoir, we can train a decent model to help with the design of PNC. Unfortunately, most of the data points in the reservoir are not “digitally available”. For the portion that is “digitally available”, data points are very likely to be a compressed file uploaded to a lab website, sometimes with a dead link, or
an online repository like GitHub\textsuperscript{1}, Zenodo\textsuperscript{2}, or figshare\textsuperscript{3}, which cannot be queried. When you download the compressed file and decompress it, you might end up with a folder of random spreadsheets with arbitrary headers, or data files with an unknown file extension that must be opened with an unspecified commercial software, or maybe without any metadata to help you correctly integrate the data with other datasets. The so-called “digitally available” resources are not very helpful in terms of data-driven methods.

Recently, there is an increasing call for data resources to be findable, accessible, interoperable, and reusable (FAIR)\cite{fair1}, \cite{fair2}. To be findable, the data resource must have a globally unique identifier and be indexed in a searchable resource. To be accessible, a data resource must be not only open, meaning no paywalls blocking access, but also having metadata accessible even when data is not available. Interoperability calls for a shared language for knowledge representation. Basically, if an application programming interface (API) is configured and well-documented for a data resource, it becomes interoperable. Reusability asks for rich metadata to be associated with the scholarly data, enabling others to understand how the data was obtained in detail.

\textsuperscript{1} https://github.com/
\textsuperscript{2} https://zenodo.org/
\textsuperscript{3} https://figshare.com/
There exist a few online polymer data resources, namely the PoLyInfo database\(^4\) owned by the National Institute for Materials Science (NIMS) of Japan, the CROW Polymer Database\(^5\), the Polymer Property Predictor and Database\(^6\) (PPPDB) by the Center for Hierarchical Materials Design (CHiMaD), and the Polymer Genome\(^7\) by the Georgia Institute of Technology\([22]\), \([23]\). While the CROW Polymer Database and the Polymer Genome have a large collection of pure polymers with their properties, there is no data on PNCs. PPPDB takes one step further by cataloging polymer blends and polymer composites. But it only covers three properties: the Flory-Huggins Chi parameter, glass transition temperature, and binary polymer solution cloud point. The PolyInfo database hosts polymer blend and PNC data as well. Unfortunately, these databases do not provide an application programming interface (API), which prevents them from complying with the FAIR data principles due to limited interoperability. In addition, processing steps are left out of the scope of these databases. Though processing steps are challenging to represent in a model, failing to catalog those data makes it impossible to build the p-s-p relationship even when better numerical representations are developed.

\(^4\) https://polymer.nims.go.jp/PoLyInfo/
\(^5\) http://polymerdatabase.com/polymer%20index/home.html
\(^6\) https://pppdb.uchicago.edu/
\(^7\) https://www.polymergenome.org/
With the goal to build a FAIR data resource for the PNC community, we gave birth to NanoMine in 2016 along with a suite of user-friendly analytical and simulation tools[24]. Later, we extended our platform to the metamaterials community and built MetaMine, which together with NanoMine forms the MaterialsMine data resource[25].

1.3 Challenges in building a FAIR data resource for the materials science community

A FAIR data resource is more than a data repository. To be FAIR, we need a clear and extensible data representation to enable the interoperable knowledge exchange. At the time when NanoMine was first introduced, there did not exist an extensible data representation for the complicated PNC system. Thus, one major task back in the day was to develop a consistent, clear, and extensible data representation for PNCs and the materials science community. To promote interoperability, we collaborated with the National Institute of Standards and Technology (NIST) to utilize the XML framework they developed for the Materials Data Curation System (MDCS) as the backbone of the NanoMine schema, which enables other materials data resources registered in the NIST Materials Resource Registry (NMRR) to exchange data with NanoMine [26]–[28]. XML is not the optimal solution for data querying. We later transitioned to an ontology-based knowledge graph representation to be flexible, robust, and interoperable, as we integrated and collaborated with the data science and semantic web community.
1.4 Challenges in data curation

With the data framework and data representation in place, the final missing piece is the pathway to curate data into the system. As we mentioned above, the vision of the NanoMine data resource is to crowd-source PNC data from different labs for discoveries powered by data-driven methods and machine learning. We need tools and a user-friendly interface for data curation, which helps human curators to translate unstructured data in publications or unorganized in-house data into structured data representation in the data resource. Human curators are required to have domain knowledge, which means they are likely materials scientists who are usually more comfortable working with Excel spreadsheets than other data representations like XMLs. Thus, a couple of data interfaces are required to ensure a smooth data curation pathway from scattered in-house experimental data or publications to queryable data representation via human curators.

As we mentioned, the PNC data space is high in complexity and dimensionality. For example, the NanoMine schema has 845 fields and keeps evolving as needed with nested hierarchical structures. It is not easy to lay the fields out in a relatively flattened tabular data structure like Excel spreadsheets. We managed to use 11 worksheets and 954 fillable cells to capture all schema terms in a master Excel template. However, for users who are new to our curation system, the number is still huge and could be
intimidating. Methodology to significantly reduce the number of cells in our master template is necessary to make our data curation pathway user-friendly.

1.5 Challenges in data validation and standardization

The more user-friendly the data curation pathway, the less likely curation errors happen. However, as long as human curators are involved in the curation process, we should expect curation errors to exist because of human error, especially when curating complex PNC data that requires domain knowledge. To reduce and prevent curation errors and thus improve data quality, we need data validation mechanisms. To address the need for data validation, we discuss the validation mechanisms embedded both during the curation and after the curation.

On many occasions, even without human-caused curation errors, the data resource cannot perform to its full capacity due to data inconsistencies. The inconsistency of polymer indexing caused by the lack of uniformity in the expression of polymer names is a major challenge for the widespread use of polymer-related data resources and limits the broad application of materials informatics for innovation in broad classes of polymer science and polymeric-based materials. The current solution of using a variety of different chemical identifiers has proven insufficient to address the challenge and is not intuitive for researchers. Another example will be the use of mass fraction or volume fraction to report the composition of PNCs. The two types of fractions are interchangeable, but authors usually report one of them. Without
standardization, the data resource will only be able to return half of its data collections when queried with a common search term “show me all data with Property A vs. volume fraction” or “show me all data with Property A vs. mass fraction”. Therefore, data standardization tools are necessary to release the full capacity of the FAIR data resource. Tools developed to bypass manual curation, the mass fraction – volume fraction conversion agent, and ChemProps, a RESTful API-enabled multi-algorithm-based mapping methodology that is optimized to solve the polymer indexing issue with easy-to-update design both in-depth and in width, are discussed in detail in this dissertation[29].

1.6 Challenges in the sustainability of PNC data resource

Human curation is by far the most viable pathway for curating data into MaterialsMine. However, the efficiency of data curation by human curators is still limited. To create truly powerful and transformative materials design paradigms, we need to harness the power of AI to efficiently extract a significant set of data from the published, archival literature. Natural Language Processing (NLP), which seeks to automate the extraction of knowledge from human-written text, offers an opportunity to make this data accessible and readily reusable by humans and machines. The first step is to generate a sample list where curators can easily find the number of samples, their compositions and properties reported in the paper. We handle the task in a pre-training-finetuning fashion. Downstream tasks include Named Entity Recognition (NER) to
detect sample code, sample composition, property, and group reference to samples in
the captions, and Relation Extraction (RE) which predicts the relations between pairs of
detected named entities. It is reported that NLP models pre-trained on general corpora
do not work well on domain-specific tasks. However, domain-specific corpora are not
openly available. In this dissertation, we discuss in detail how the two corpora for pre-
training and finetuning are constructed (see Chapter 6). A T5-base model pre-trained on
the caption-mention corpus and fine-tuned for the NER and RE tasks is proposed. We
evaluated it along with an array of BERT-based models and seq2seq models for potential
applications in the semi-automated curation pipeline for MaterialsMine.

1.7 Research tasks

To address the needs and challenges described in the previous sections, and to
ensure the data size and quality of MaterialsMine data resource, we present the
following three research tasks:

**Task 1: To build a user-friendly human curation pathway for**

**NanoMine/MaterialsMine data resource.** Since the MDCS system is not intuitive to our
potential curators – materials scientists, and the initial master template has redundant
fields for different curators leading to a low level of user-friendliness, the objective of
this task is to 1) build a user-friendly front-end web app for Excel template and data
uploading, 2) build a back-end interface that reads Excel templates and converts data
into XML format, 3) design a methodology to reduce the number of fillable cells in the master template and update the back-end interface to support template customization.

**Task 2: Develop tools and mechanisms that facilitate the curation and enhance data quality.** As we discussed in Section 1.5, curation errors are inevitable as long as human curators are involved. Data standardization is necessary to push the utilization of data resources to their limit and reduce the workloads of human curators. The objective of this task is to develop tools and layers of data validation and standardization for MaterialsMine data resource to prevent curation errors and release the full data capacity of MaterialsMine in downstream querying.

**Task 3: Explore methods towards sustainable and (semi-)automated data curation of MaterialsMine data resource.** Human curation is limited in terms of efficiency as new PNC papers are published on a daily basis. Given that hundreds of thousands of PNC data are archived in publications in unstructured textual format, if we can utilize the NLP techniques to identify and organize some of the published data into MaterialsMine, or generate a sample list to guide human curators, it will significantly boost the curation efficiency. From the sustainability perspective, the future of MaterialsMine data curation lies in NLP-driven (semi-)automated approach. As a first demonstration of the concept, the objective of this task is to 1) collect and clean a pre-training corpus of figure/table captions and sentences that refers to a caption obtained
from PNC papers, 2) annotate finetuning corpus obtained from MaterialsMine figure captions, 3) train NLP models that perform NER and RE tasks on PNC captions.

1.8 Dissertation outline

Chapter 1 introduces the background information of PNCs and FAIR data along with the challenges to build a FAIR data resource for PNC data with a user-friendly data curation system, which motivates the research tasks. Chapter 2 provides technical background for the works described in Chapter 3-6. Chapter 3 discusses in detail how the customization of the master template is made possible from both the front-end and back-end. Chapter 4 lays out the data validation tools and mechanisms implemented during and after curation that prevent curation errors. In Chapter 5, we present data standardization tools that were developed to bypass manual curation and tools or services that help NanoMine to reach its full capacity in downstream querying like mass fraction-volume fraction conversion tools and ChemProps, which is broadly applicable to the whole polymer community. Chapter 6 presents a proposed prototype NLP-driven semi-automated curation pathway that serves as the first step towards a sustainable data curation future of NanoMine/MaterialsMine. Chapter 7 summarizes the contributions of works presented in this dissertation and discusses the potential directions of future works.
2 Technical background

In this chapter, we will review the technical backgrounds associated with the research tasks defined in Section 1.7. First, a review of the concept of materials informatics is presented, followed by a discussion of existing materials data resources. Second, background information about XML and XML schema is provided. In the next section, we provide a thorough walkthrough of the Excel-based curation workflow in NanoMine. After that, we present some fundamentals of NLP along with a survey on the pre-training then fine-tuning style of NLP model training. Transformers and several representative transformers-based models are discussed in detail. Lastly, we provide an overview of an emerging interdisciplinary domain where NLP techniques are applied to solve materials science problems.

2.1 Materials informatics

Long before the establishment of the MGI, the idea of materials informatics was first described in 2005 by Rajan as an approach that could potentially advance the discovery of structure-property relationships and materials design[18], [19], [30]. Essentially, materials informatics has two major building blocks. The first building block is the machine learning algorithms and their application scenarios on materials data. Over the past few years, applications including but not limited to steel fatigue prediction[31], intrinsic dielectric breakdown strength prediction for insulators[32], microstructure optimization for metal alloy[33], microstructure reconstruction via
transfer learning[12], guiding lab works to find polymers with high glass transition
temperatures faster with active learning[34], etc. have been reported. Several works did
a great job summarizing the machine learning algorithms that could be applied to
different types of materials data[35]–[39]. Besides the algorithms and applications, the
other building block is the data infrastructure and the data itself. The Materials Data
Research Alliance was founded in 2020 to push the community towards the goals of the
MGI in the data scene[1]. While the idea of materials informatics could be applied to all
kinds of materials, the progress of materials data generation and data infrastructure
construction differs by materials type. The inorganic and metal alloy fields have a longer
history of archiving structured materials data, partially due to the high-throughput
computation that is capable to generate huge datasets[40]. AFLOW[2], OQMD[3], and the
Materials Project[4] are three popular data resources in this realm, all of which are
providers of the Open Databases Integration for Materials Design (OPTIMADE)
consortium that provides a uniformed API for data exchange between the providers to
enhance interoperability[41]–[46]. On the other hand, the data resources for organic or
soft materials are less developed as compared to the inorganic world. Ten years after the
introduction of materials informatics, Rajan still did not mention organic and soft

---

[1] https://www.marda-alliance.org/
materials in a review paper, indicating the late start of the subfield [47]. The complexity of organic, and especially soft materials is probably one of the major reasons that stopped researchers to build a data resource in the early days. As we mentioned in Chapter 1, there are a few data resources for polymers, including PPPDB, PoLyInfo, CROW Polymer Database, and Polymer Genome, all of which are not FAIR. A new FAIR data resource named CRIPT\(^5\) was just released in June this year[48]. In addition to the specialized data resources that we have introduced, there are a few general-purpose data resources that do not limit materials systems. Reaxys\(^6\) by Elsevier is a powerful database that provides metadata, properties, and reaction steps of organic, inorganic, and organometallic substances[49]. Unfortunately, this subscription-based service is not accessible according to the FAIR principles. Citrination\(^7\) is less powerful than Reaxys but offers an academic license[50]. Unfortunately, the Open Citrination platform was decommissioned in September this year. Materials Commons\(^8\) is a FAIR data platform with a well-documented API and a built-in search function[51]. However, data points are not ingested and directly queryable. Instead, datasets and files are organized in individual projects with metadata. In terms of PNC, to our knowledge, we are the first and probably the only effort to build a FAIR data resource.

\(^{5}\) https://criptapp.org/
\(^{6}\) https://www.reaxys.com/
\(^{7}\) https://citrination.com
\(^{8}\) https://materialscommons.org/
It is worth mentioning that there is a unique research domain within materials informatics that replaces the emphasis on the data infrastructure with a fully automated experimental platform that generates data on the fly, namely the Autonomous Experimentation (AE) systems[52], [53]. The system is generally coupled with active learning which makes educated suggestions for experimental planning based on limited available data[54]. The level of complexity of processing and synthesis of PNCs drives the investment for such an AE system high and thus makes it a less viable materials informatics application in the PNC domain.

2.2 XML and XML schema

Extensible Markup Language (XML) is a markup language that stores data with arbitrary structures in a tree-like structure that is both human-readable and machine-readable. XML was recommended by the World Wide Web Consortium (W3C) in 1998[55]. The flexibility of XML enables it to represent arbitrary data structures, but it can also cause inconsistency issues. Figure 2-1 demonstrates such inconsistencies.

Figure 2-1: Two XML’s containing identical materials data but with different structures.
To represent a PNC with polycarbonate as the matrix, and BaTiO3 as the filler with 20 vol\%, both representations in Figure 2-1 are correct according to human intuition. However, the equivalence is not as intuitive to machines because machine reads first by the structure, then by the values. We will not be able to effectively extract data from XML’s if there is not a standard or schema to control the structure of XML. To regularize XML and resolve the inconsistencies, W3C recommended XML Schema Definition (XSD) in 1999[56]. An XSD defines the type, value, and neighboring structure of each element in XML. XML can be validated against XSD to ensure consistency. For example, if we define the “FillerComposition” element has a child element “volume” of type “xsd:double”, the XML on the left in Figure 2-1 will be considered valid while the XML on the right will not. Or the other way around, if the “volume” element is a complex type with child element “value” of type “xsd:double”, then the XML on the right becomes valid and the XML on the left becomes invalid.

Since its first release in 2017, 8 versions of NanoMine schema have been published and are openly available at the Duke-MatSci GitHub repository9.

2.3 Curation pathway

2.3.1 XML webform-based data curation

NIST developed the MDCS\textsuperscript{10} to provide a pathway for curating materials data into XML, which will then be logged into a NoSQL MongoDB database\cite{28}. Besides the curation system, the MDCS also includes an XSD template builder and a RESTful API for data exchange between different data resources that build with the MDCS. The graphical user interface (GUI) embedded in the MDCS utilized the human-readable characteristic of XML by displaying the XML tree directly as a data form for curators to fill in. The first version of the data curation system in NanoMine could be considered an instance of the MDCS. We developed the NanoMine XSD schema for the PNCs\cite{57}. Figure 2-2 shows a screenshot of the data curation GUI of NanoMine that utilized the MDCS in 2017.

\textsuperscript{10} https://mdcs.nist.gov/
As an early effort to provide a means for materials data curation, MDCS did a good job since it provides a working pathway. However, the drawbacks of the webform-based curation system are also obvious. The tree-structured XML is human-readable but not user-friendly. Materials scientists, who form the potential pool of data contributors of NanoMine, are not comfortable working with XML. On loading, the XML webform is by default collapsed to the top levels, otherwise, the web page would be so long that it could span tens of browser windows. Such a design decision brings several issues. First, it requires curators to memorize the XSD schema because every element other than the top-level ones are hidden from them. This requirement stops new users to contribute their data. Second, it could take hundreds of clicks for nodes to collapse and expand to
finish the curation for one sample, while a publication usually contains about dozens of
them. The third point is associated with the second point. MDCS loads the XML web
form dynamically based on the XSD schema, which means some nodes with many child
nodes will take a long time to load. Unfortunately, there was not a loading icon
displayed during the process, only a frozen browser. From the curators’ perspective,
they may believe that their clicks crashed the system and force-exit the browser, putting
all previous curation works unsaved and in vain. The first issue stopped new users from
trying the system, while the second and third issues kept frustrating experienced users.
Thus, the number of curations in NanoMine did not increase at a healthy pace when
webform-based data curation was the only option.

2.3.2 Excel-spreadsheet template-based data curation

To improve the user experience while maintaining interoperability with MDCS
and other data resources in NMRR, we decided to add another layer before the XML to
interface with materials scientists – our curators. After careful evaluation of various
options, we determined that an Excel spreadsheet-based template could be a good
interface. The decision was based on several reasons: 1) many materials scientists are
familiar with Excel and comfortable with it as a data form, 2) Excel spreadsheets are by
nature tabular and flattened as compared to the XML’s tree-like structure, which means
the number of clicks to search for the property or other terms are significantly reduced,
making it more friendly to the new users, who are not sure where to or if there exists a field for them to input their data.

As the saying goes, “Every coin has two sides.” The flattened structure of Excel makes it challenging to represent the highly hierarchical structure of PNC data. Figure 2-2 illustrates the methods we used to represent the hierarchical structure in a flattened Excel template.

<table>
<thead>
<tr>
<th>Worksheet - 1st level of hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Titles and spaces</strong></td>
</tr>
<tr>
<td>“#” — a repeatable row or section</td>
</tr>
<tr>
<td>font size and styles (e.g. bold)</td>
</tr>
<tr>
<td>types of cell color: light green: for free text; blue: a dropdown menu</td>
</tr>
</tbody>
</table>

**Figure 2-3: Methods used to build hierarchy in Excel template.**

Individual worksheets are designated for data origin (metadata), materials type, synthesis and processing, characterization methods, mechanical properties, viscoelastic properties, electrical properties, thermal properties, volumetric properties, rheological properties, and microstructures. Within each worksheet, different levels of fields will have different font sizes and styles. Blank rows are also utilized to separate fields with different levels. Boxes are used to group fields that belong to a shared parent in the schema. For fields that can be duplicated, i.e., properties tested under different testing conditions...
conditions like complex modulus as shown in the screenshot, we put a "#" mark following the name and bound the whole section with a box.

As stated in Section 1.4, the original Excel template, which is referred to as the "master template" in this dissertation, has 954 fillable cells, which can be intimidating to our curators at first glance. We need a methodology to significantly reduce the number of cells in our master template, which is necessary to make our data curation pathway user-friendly. The solution we propose is to leverage the fact that research groups usually have a few research focuses, which we can use to eliminate some irrelevant fillable cells beforehand by reading their publications to realize a certain level of customization. Of course, the back-end script to convert the Excel template into XML is required to support the duplication and removal of certain cells in the customized template. The details of the conversion script XMLCONV and the web app for template and data uploading will be discussed in detail in Chapter 3.

2.4 String matching and string comparison

String matching, as indicated by its name, is usually related to a searching act where a target string or pattern is used for searching in another string. In this dissertation, applications that involve string matching can be found in Section 5.2 and Section 5.4. It is also used for corpus generation and cleaning in Section 6. In this section, we will introduce the background of several string-matching tools, metrics, and algorithms used in developing various NanoMine data standardization tools.
2.4.1 Regular expressions

Regular expressions (regex) are widely used for string matching since it is a language designed for specifying text search patterns[58]. In python, a package called re is installed by default for regular expression matching operations[59].

Regex is very powerful. It can construct patterns with multiple containers delimiting a set of characters by the brackets “[]”. Within the brackets, regex allows ranges of characters like “[a-zA-Z0-9_]”, meaning it will only look for one alphanumeric character. If only the non-alphanumeric characters are wanted, you can specify the negation by “[^a-zA-Z0-9_]”. For non-alphanumeric characters, we can also use the shortcut “[^\w]” or “[\W]”, where “\w” is for alphanumeric characters and “\W” is for the opposite case. You can also specify the number of characters by the quantifiers like “?*+[n,m]”. For example, if we want to match a pattern of three, four, or five contiguous non-alphanumeric characters, we can do it by “[\W][3,5]”. If we want to force the search to start at the beginning of a string, we can start regex with “^”, note that it is different from the leading “^” in a bracket, which is for negation. Like other languages, “\” is used to escape special characters. You can form groups with “()” and apply look-ahead or look-behind assertions. Regex is widely used as a tool for string matching and keyword extraction in materials informatics [60]–[62].
2.4.2 Longest common substring

Regex is a tool that helps us find strings that match a given pattern. For each substring in a string, it is a binary decision – match or not. It is not designed for string comparison. If we want to check how similar two strings are, we need metrics like the longest common substring (LCS). LCS is the longest substring that exists in two or more strings that we are comparing. For example, “polymethyl methacrylate” and “methacrylic acid copolymer” have common substrings like “polyme”, “meth”, “methacryl”, etc. The LCS of the two strings is thus “methacryl” since it is the longest. LCS is an intuitive metric reflecting how similar two strings are. Many algorithms for string comparison were developed based on LCS.

2.4.3 Ratcliff-Obersharp algorithm

Ratcliff-Obersharp algorithm is one of the string-matching algorithms that include LCS in the equations. The similarity score is given by

\[ S_{RO} = \frac{2LCS_{rec}}{L_A + L_B} \]

where \( L_A \) and \( L_B \) are the lengths of the two input strings A and B, \( LCS_{rec} \) is the total length of all LCS obtained recursively on the remaining non-matching sections on both sides of the LCS. Using the example from the previous section again,

LCS(“polymethyl methacrylate”, “methacrylic acid copolymer”) = “methacryl”. The next step is to find LCS(“polymethyl “, “” = “” and LCS(“ate”, “ic acid copolymer”) = “a”.

For the left side, we hit an empty string and we stop. For the right side, we split again
and compute LCS(“”,”ic“) = “” and LCS(“te”,”cid copolymer”) = “e”, until we cannot find an LCS. In this case, the LCSrec is the length of “methacryl” + “a” + “e”, which equals to 11. Given that \( L_A \) is 23 and \( L_B \) is 26, the similarity score equals to \( \frac{2 \times 11}{23 + 26} = 0.449 \). The python difflib package implements the Ratcliff-Obershelp algorithm[63].

### 2.4.4 Levenshtein Distance

Another popular string-matching algorithm that includes LCS in its equation is the Levenshtein distance[64]. LCS and Levenshtein distance are both members of a string metric known as edit distance. Levenshtein distance tells you at least how many insertions, deletions, and substitutions are required to transform one string to another. The algorithm takes the form of

\[
lev(A[:i], B[:j]) = \min \begin{cases} 
lev(A[:i], B[:j-1]) + 1 \\
lev(A[:i-1], B[:j]) + 1 \\
lev(A[:i-1], B[:j-1]) + 1_{A[i] \neq B[j]} 
\end{cases}
\]

Edit distance is widely used in applications like spell-correction.

### 2.5 Natural Language Processing

Natural Language Processing is a subdomain of computer science and linguistics that seeks to teach computers to understand human language. Its applications include but are not limited to text classification, machine translation, part-of-speech (POS) tagging, named entity recognition, relation extraction, question answering, text summarization, speech detection, and so forth. Generally speaking, any machine
learning problems with human language involved in the dataset or target output can be considered an NLP problem.

### 2.5.1 Fundamentals of NLP

A typical NLP pipeline consists of the steps provided in Figure 2-4.

![Figure 2-4: A typical NLP pipeline. The shaded area is merged into one step for the deep learning model.](image)

The first step is always data collection. The typical data source includes webpages like Wikipedia, books, and news articles. The next step is text cleaning. The step is especially important if data is crawled from webpages in a markup language, like HTML and XML, where text is wrapped around by special tags that must be removed, along with some functional statements embedded in the file. Till this stage, we can argue that we have a corpus, which means a collection of cleaned text in NLP. Some corpora were generated after pre-processing steps like Unicode normalization, sentence segmentation, etc. Other pre-processing steps include tokenization, stop words removal, stemming, lemmatization, phrase-detection, and many more. Tokenization splits the sentence into tokens. A token is usually a word, while sometimes it is a sub-word
depending on the type of the tokenizer. Stop words are the words that commonly appear across all documents in the corpus but help little with the NLP task. For example, if we want to collect a corpus consisting of figure captions from PNC papers, the word “Figure” and “Fig” are considered stop words. Stemming and lemmatization are procedures to reduce the size of the vocabulary, which means a collection of all tokens that appeared in the corpus. Stemming reduces the word to its word stem by removing tailing characters. Lemmatization uses the context to determine a lemma for a group of words grouped by their intended meaning. Phrase-detection merges tokens that appear neighboring to each other frequently. Up until this point, our data is still in textual format. Feature engineering intends to craft features, usually vectors, from the text. Popular feature engineering methods for traditional statistical NLP include one-hot encoding, bag-of-words model, and term frequency – inversed document frequency (TF-IDF) encoding. Word embedding models like Word2Vec, Global Vectors (GloVe), fastText, and ELMo are more advanced methods to transform the text into vectors[65]–[68]. In fact, word embedding models by themselves are unsupervised NLP models. Then, we can apply a machine learning model, evaluate the results, and deploy the model. Sometimes, we might loop back to a previous step before we deploy the model. Use NER as an example, as a supervised learning task, we need to annotate the corpus by hand. However, we do not need to annotate a lot of documents at first pass. A small number of data can be annotated to train an ok model to help us pre-tag the sentences in
the second pass, which can save time and effort. The iteration goes on until we are satisfied with the result and deploy the model. Just like any other model development, we might start over from the first step when we discover something new that could help the model performance and release a new version of the model after the deployment.

The shaded area in Figure 2-4 is what differs the deep learning NLP models from its traditional statistical alternative. Unlike the traditional statistical approach, deep learning NLP models take care of feature engineering and modeling at the same time. These models learn the feature representation in the latent space.

2.5.2 Pretraining and finetuning

Deep learning models are powerful but usually contain a big number of parameters, which require a huge training set and a long training time. Deep learning models, like other machine learning models, are usually task-specific, meaning the model should be retrained for different tasks. However, as indicated by the name NLP, NLP tasks have a shared feature that they are tasks based on the understanding of natural language. Thus, if we train a deep learning model on some generic NLP tasks, the model might be able to learn the latent representations that could be helpful across all NLP tasks and lift the limitation posed by the size of training sets, which has been proven over recent years with models setting records on various tasks with new state-of-the-art performance[69]. The action of training a deep learning model on a large corpus for generic NLP tasks is called “pre-training”. “Fine-tuning” is to continue training the
pre-trained model, maybe with some extra layers or heads, on a small task-specific dataset.

A common self-supervised pre-training task is denoising. Denoising tasks come in various forms. A common denoising task is masked language modeling, where models are trained to predict tokens or spans that are randomly masked from sentences. The training objective is adopted by several encoder models like BERT, RoBERTa, DeBERTa, and encoder-decoder models like T5[70]–[73]. Note that some of these models have extra training objectives. Sentence permutation and text infilling used by BART are examples of some other denoising tasks[74].

There are several options one can pick on how to fine-tune the pre-trained model. The first option is to use the pre-trained model like a word embedding model. The hidden states are passed directly to a subsequent layer while freezing the pre-trained model parameters. The second option, which is quite popular, is to attach a head layer or two at the end of the pre-trained model for task-specific predictions. In this case, some, or all of the parameters in the pre-trained model will be updated along with the newly added parameters in the head layers at every optimization step.

### 2.5.3 Transformers

The original Transformer model developed for machine translation tasks was the first NLP model that introduces the attention mechanism[75]. The attention mechanism enables the model to pay special attention to some of the words in the context and
ignore some other words when generating the latent representation for each word.

Figure 2-5 shows the architecture of the Transformer model.

![Figure 2-5: The architecture of the Transformer model. Modified from [75].](image)

The model is constructed with an encoder that learns the latent representation of the input, and a decoder that takes the latent representation and the masked targeted output to generate a target sequence. The model architecture was then adopted in many creative ways, resulting in a family of Transformers models. There are three big categories of Transformers models depending on the use of the encoder and the decoder. As we briefly mentioned in the previous section, BERT and models derived
from BERT are examples of encoder models, where a series of encoders are stacked to form the model. Since encoders are used to understand the input, encoder models are good at tasks that seek to understand the input, like NER and POS tagging. Another class of models is decoder models, which are good for text generation. The third class of models is the encoder-decoder models or sequence-to-sequence models (seq2seq) since their inputs and outputs are both sequences. With an encoder to understand the inputs and a decoder for text generation, seq2seq models are good for generative tasks that require some understanding of inputs, like machine translation, text summarization, question and answering, etc. As you might have noticed, the original Transformer model is a seq2seq model. T5 is another powerful seq2seq model that is pre-trained for multi-tasks.

2.6 Applications of NLP in materials science

As we have discussed in Section 1.2, a vast amount of unstructured non-computational materials data is currently locked in the publications. To apply machine learning or deep learning models for materials discovery and design, we need to extract data from the publications efficiently. In 2016, a toolkit named ChemDataExtractor was introduced for chemical entity extraction with associated measurements and properties[62], [76]. The toolkit was showcased to generate a database of Curie and Neél magnetic phase-transition temperatures from a corpus of 68k chemistry and physics papers[77]. Many of the later works in the NLP + materials science domain benefited
from the tokenizer and chemical entity extraction provided in the ChemDataExtractor toolkit [78]–[82].

Works in the domain span across the subtasks in the pipeline. Some researchers focused on the annotation scheme of synthesis procedures [83], [84]. The annotated corpus was then used to extract the synthesis recipe and conduct a virtual screening of synthesis parameters [60], [78], [79], [85]–[89]. With the same corpus, several WordEmbedding models were trained and made publically available [79], [87]. Another Word2Vec embedding model trained on 3.3M scientific abstracts was reported [90]. Two BERT-based models, MatBERT and MatSciBERT, were released this year [82], [91]. MatBERT was pre-trained on a corpus consisting of 2M materials science journal articles, while MatSciBERT was pre-trained on ~150k full-text articles retrieved via Elsevier API. A few demonstrative works utilized NER models for so-called knowledge extraction are also reported in recent years [80], [81].

The language used in materials science publications is domain-specific, which differs a lot from the generic natural language used to pre-train Transformers models. Directly fine-tuning the pre-trained models is not recommended and the importance of domain-specific corpus is stressed [61]. Due to copyright issues, researchers have been working with their private domain-specific corpus, which remains a challenge in applying NLP to materials science.
3 NanoMine curation pipeline

In this chapter, a detailed end-to-end step-by-step walk through of the entire NanoMine curation pipeline will be presented.

3.1 Overview

Figure 3-1 illustrates the major steps involved in the curation pipeline.

Figure 3-1: Workflow of Excel-based data curation in NanoMine.

The steps marked with a human icon indicate that a human curator is in charge while the steps marked with a computer icon indicate that a script or several scripts are in charge. The first step is to collect the full-text paper and supporting information along with the metadata, which is usually retrievable all by a single DOI. The second step is for curators to read the paper, identify the samples or experimental units, identify and extract the data from figures, tables, and text, and link the data to the samples. The first two steps are required only if we are curating from publications since in-house data usually provides the data and their linkage to the samples. The third step is to generate a
sample list with information collected in the previous step. Figure 3-2 gives an example of a sample list.

![Figure 3-2: Example of a sample list used for NanoMine curation.](image)

As shown in the figure, the sample code, sample composition, and sample properties including microstructure images are listed in the sample list together with the source of the data. Curators can then use the sample list to locate data faster during curation and check their curation on a high-level after curation. The fourth step is to fill the data into the master template. Upon completion, curators can then open the data uploading web app on the MaterialsMine website, select or create a dataset for the DOI, and upload the curation. Then, the back-end scripts will take over. The XMLCONV conversion script will convert the Excel template to XML. During and after the conversion, multiple data standardization agents and data validation agents will be called to resolve potential data inconsistencies and detect any potential curation errors. The generated XML will be validated against the XSD schema and stored in MongoDB. When XMLCONV finishes running, an email will be sent to the curator with the status of the job. If the conversion failed for some reason, the email would include the error message to guide curators for correction. More often than not, the conversion error is
caused by misuse of the master template. To complete the full curation cycle, a final layer of conversion, developed by our collaborator, is called to transform XML into the RDF triples and publish the triples into the MaterialsMine knowledge graph. The final layer is beyond the scope of this dissertation.

3.2 Customized Excel template

Early curation trials showed that curators tend to frustrate and give up when their first exposure to the curation system is via the master template. In fact, many research labs will have a few research focuses instead of generating data spanning mechanical, viscoelastic, dielectric, thermal, volumetric, and rheological properties. With that in mind, together with my colleagues, we developed a customizable template and the associated back-end conversion script XMLCONV that supports the customization of the template.

A typical customization model works like this: 1) a few papers from a certain research group are selected to be curated into MaterialsMine, 2) an experienced curator from our side will skim the papers to determine property fields to be kept for curation, 3) worksheets deemed irrelevant will be hidden, 4) irrelevant properties within the remaining worksheets will be removed, 5) the customized template will be sent and a training session will be scheduled with the designated curator from the research group.
Another feature of the master template that contains some level of customization is the curation of processing steps. Figure 3-3 shows the worksheet designated for the processing steps.

![Figure 3-3: A screenshot of the toolbox design in the "3. Synthesis and Processing" worksheet in the master template.](image)

Unlike other worksheets, the order of processing steps adds another dimension to the curation. To keep track of the order, we used a toolbox design. Curators will copy and paste the processing steps from the toolbox one by one. Thus, XMLCONV cannot expect a static layout in this worksheet, which also calls for the support of customization.

An example of the customized template can be found at Duke-Matsci GitHub¹. After customization, the template has 538 fillable cells, meaning a significant 44% reduction as compared to the master template.

3.3 Data uploader – the webapp

Figure 3-4 is a screenshot of the data uploader web app in MaterialsMine.

![Data Uploader](image)

File Upload

**Description**

The simplest method to curate your sample into the database is by uploading an MS Excel spreadsheet. For each sample, select or create a dataset for your sample group, upload a completed Excel template file using the first uploading box and other supplementary images and raw data files using the second uploading box. The template Excel template contains all possible fields for nanocomposite sample data and therefore many fields will remain blank for your sample. Fill in only the parameters applicable to your sample. Customized templates are available upon request, please contact the Nanomine team if customization is required.

**Steps**

**NOTE:** Filesets for samples are grouped into datasets. The files for a sample (images, auxiliary spreadsheet data, completed Excel template, etc.) are uploaded as a set called a fileset. Uploading multiple samples requires multiple fileset uploads.

**Step 1:** Create a new dataset for the control sample and its related fileset, then when uploading each additional sample be sure to select the same dataset that was used for the control sample of the sample group.

**Step 2:** Click here to download the blank MS Excel template (157 KB). (Click here to see an example, 203 KB)

**Step 3:** Fill in the parameters for all applicable cells in the Excel template file. Prepare the supplementary images and raw data files.

**Step 4:** Select the completed Excel template file in the first uploading box.

**Step 5:** Select the supplementary images and other raw data files in the second uploading box (press “Ctrl” or “Command” when selecting multiple files), then click Submit to upload your data.

**Step 6:** Wait for the feedback message. Please read the message and follow the instructions if an error message is displayed.

**Note**

1. We recommend you to upload your control sample first and remember its sample ID.
2. Upload one sample data at a time (one template Excel file along with supplementary files).
3. Rows or sections followed by a “*” sign in the template Excel file can be duplicated. Copy them into additional rows if needed.
4. Acceptable image file format: JPG, PNG, TRIFF. Indicate the full image file name including the extensions in the corresponding cells in the Excel template file.

**Inputs**

**Choose or create a dataset**

Select a completed Excel Template File

Select Other Files (including raw data files and image files)

REFERENCES

![Footer](image)

Figure 3-4: A screenshot of the data uploader.
The description box gives general instructions on how to use the master template. It also contains a link to contact the team that should kick off the template customization process as described in the previous section.

Following the description box are the detailed step-by-step instructions. The first step is to create a dataset or select an existing dataset. This feature was enabled by the software engineers in our team to group curations and thus enable enhancement or difference calculation based on the control sample in a batch of experiments. In PNC studies, it is common to characterize the pure polymer, serving as the control sample, together with the PNCs, such that a change in properties caused by the addition of stiff nanoparticles can be investigated. With the dataset feature, we can group samples published in a single paper, or from the same batch of experiments in the in-house data. When the first template is uploaded and converted successfully, the dataset will be updated with the DOI, if provided, and populated with metadata retrieved by the DOI crawler, which we will discuss more in Section 5.1.1. The second step is to download the empty master template. An example of a filled template is also provided. The third step is to fill in the template and prepare supplementary data files like microstructure images and spectral data tabular files, which we utilize to capture spectral data since Excel cell does not accept non-scalar values. Users should put the names of supplementary data files in the “Datafile” column like the ones in Figure 2-3. The fourth step is to click the first “BROWSE” button to upload the template file. Next, users need to click the second
“BROWSE” button to upload the supplementary data files. The reason why we used two portals for data uploading is that we accept Excel files as supplementary data files. We thus cannot use the file extension to distinguish the template file. We also do not want to restrict the file name of the template since it will not be user-friendly. Using a separate uploading portal for the template is then the best option. The last step is to click the “SUBMIT” button, which will remain grayed out until a template file is uploaded. Successful submission will lead to a pop-up as shown in Figure 3-5.

![Figure 3-5: A screenshot of the data uploader with the pop-up on successful job submission.](image)

Usually, it takes less than a minute for the back-end scripts to finish running and notify users with an email. If the conversion of the job “xmlconv-xzTiWDT3jVNmfvGRtwohrk” in Figure 3-5 is successful, we should expect an email like the one shown in Figure 3-6.
Figure 3-6: An example email notification of a successful conversion.

Implicitly, there is a layer of data validation here. Figure 3-6 is an email notification of an unsuccessful conversion due to a missing file. Error messages are configured to be informative so that the curators could pinpoint and correct the errors quickly.

Figure 3-7: An example email notification of an unsuccessful conversion.

3.4 XMLCONV

Figure 3-8 layouts the full workflow of XMLCONV.
Figure 3-8: The detailed workflow of XMLCONV. Yellow boxes indicate validation units that will terminate the process.

When users successfully submit the job via the data uploader web app, XMLCONV will take over. Starting from the top left in Figure 3-8, XMLCONV will fetch the latest version of the NanoMine schema via the NanoMine REST service for later use. Then, a validation unit will check if the uploaded template file has a valid Excel “.xlsx” or “.xls” file extension. Note that all arrows coming out of yellow decision boxes in Figure 3-8 indicate “Yes” while all “No” situations will be short-routed to a termination.
of the process with an automatically generated email listing the reason for termination, like the one displayed in Figure 3-7. XMLCONV then reads the template with the xlrd package\(^2\) to extract the sample ID, which is a string to number samples within the same dataset or paper with the format “S1”, “S2”, “S27”, and so forth. A validation unit follows the extraction to make sure the sample ID is valid. Then, XMLCONV proceeds to extract DOI from the template. For in-house data, which are not published, no DOI is directly linked to them. However, we provide a field in the template to specify related DOI for cases when a sample is synthesized and characterized together with some other samples reported in a publication. It makes sense to group the unpublished portion of the samples with their published peers. Thus, XMLCONV will check if a related DOI is provided if no DOI is found. If no related DOI is reported, XMLCONV will do one final check to see if curators select “lab-generated” as the citation type. If so, it will generate a pseudo-DOI for the sample with the prefix “unpublished-” and extract metadata manually input by curators from the template if there is any. Otherwise, XMLCONV will terminate the process since we do not allow a sample to be published while associated with no DOI, which is most likely caused by a negligent curator. Taking one step back, if a DOI or a related DOI is found in any of the searches, XMLCONV will then examine the dataset to see if the DOI is seen before in the dataset, which is usually

\(^2\)https://xlrd.readthedocs.io/en/latest/ Note that we use xlrd 1.2.0 instead of the latest 2.0.1 version because 1.2.0 is the only version that supports python 3.7 and parsing .xlsx files at the same time.
positive except for the first curation into the dataset. During the first curation, the dataset will be initialized with no associated DOI. XMLCONV will call the DOI crawler to fetch the metadata. If metadata is returned, we will then save the metadata to the dataset with the DOI, which ensures the following curations to skip the DOI crawler call. With the metadata, via whatever pathway, XMLCONV will assemble the paper ID, sample ID, and other metadata to form the final ID like “L387_S14_ning_2019” or “E115_S5_bell_2016”. The letter “L” stands for literature while the letter “E” stands for experimental in-house data. The compiler script is called next to convert the Excel template into XML if the ID is successfully generated. More details about the compiler script will be discussed in the rest of this section. If the XML is generated successfully, we will validate it against the schema fetched in the very first step. If it passes the schema validation, XMLCONV will make a call to ChemProps, whose details can be found in Section 5.2, to insert unique SMILES, standardized names, and density into the XML. If the call returns with a greenlight, XMLCONV triggers the mass fraction – volume fraction conversion agent, which is discussed in Section 5.3, to convert between mass fraction and volume fraction if any of the two values are not provided. The next call is to the spectral data header parser that reads the axis headers of spectral data, splits properties with units, and tries to standardize properties and units on both axes so that the downstream visualization module can confidently plot spectral data from different studies in a single plot. Details of this parser can be found in Section 5.4. Next,
there is one more agent that interpolates special spectral data, which has a real part, an imaginary part, and a tan delta curve. We will discuss more about the agent in Section 5.5. Image files uploaded with the template are initially stored as a directory path in XML by the compiler script. To make it accessible, XMLCONV will try to upload images to the file system and assign it a Binary Large Objects (BLOB) identifier. A URL that contains the BLOB id will be generated and used to replace the directory path stored in the XML for the image. If that goes well, XMLCONV will attempt to insert the XML into the dataset and finish the curation by sending an email with the format shown in Figure 3-6.

The actual conversion from Excel to XML happens in the compiler. Within the compiler code, individual functions were developed for each worksheet. To support customization, the compiler code was not hard coded to memorize the position of each term. Instead, it memorizes only the non-leaf nodes in the XML schema. Note that top-level nodes in the schema are already represented by the individual worksheets. On the other hand, bottom-level nodes like description, value, unit, uncertainty, etc. are represented in a single row by different columns. Thus, we removed two levels already by the Excel template design, leaving at most three hierarchical levels for each worksheet. For each worksheet, the compiler iterates through the rows and keeps track of the current level and the name of the level. Whenever it reads a row with the same or higher level in the schema, it saves the data with the name of the level as a dictionary.
a parent dictionary and updates the current level and the name of the level. A nested
dictionary will be built and converted to XML by the dicttoxml package3.

As we mentioned before, the Excel template relies on separate tabular data for
spectral data curation. The tabular data file is required to have two columns of data with
the first row being axis labels including the property and the unit. Figure 3-9 is an
element of tabular data.

<table>
<thead>
<tr>
<th>Temperature (Celsius)</th>
<th>Tan delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.761435</td>
<td>0.0967781</td>
</tr>
<tr>
<td>32.026127</td>
<td>0.0922304</td>
</tr>
<tr>
<td>34.251652</td>
<td>0.0994788</td>
</tr>
<tr>
<td>36.238659</td>
<td>0.0917111</td>
</tr>
<tr>
<td>38.295902</td>
<td>0.0944467</td>
</tr>
</tbody>
</table>

**Figure 3-9: An example supplementary data file.**

A tabular data reader is embedded in the compiler code to extract data from the
uploaded tabular data and format it into a nested dictionary, which will later be
transformed into a predefined structure regulated by the schema. Here, the compiler
will check if the data file specified in the template is included in the upload, otherwise,
we shall expect an email that looks like the one in Figure 3-7 asking for missing files. It
will also check the extension of the data file, currently limited to ‘.xlsx’, ‘.csv’, and ‘.tsv’. If the data file is not correctly formatted with exactly two columns filled with values for
each row, the conversion fails. The name of the data file and the reason for failed

3 https://pypi.org/project/dicttoxml/
validation are provided in the email notification. The only exception is when a master curve file is provided, which should have three columns of data.

As you might have noticed, there are many validation units embedded in XMLCONV already, reflecting how complex and error-prone it could be for PNC data curation. In the next chapter, we will introduce some more tools and features we developed over the years for data validation.

3.5 Conclusion

In this chapter, a detailed walk through of the three key components of the NanoMine curation pipeline was provided. We developed the customized Excel template that could reduce nearly half of the fillable cells from the master template, a web app for data uploading, and a back-end script XMLCONV that supports customization of the template to improve the user-friendliness of the pipeline. The embedded validation units in XMLCONV for format checks were also introduced.
4 NanoMine data validation

The data validation units we touched on in the previous chapter exist in XMLCONV, which can be deemed as validation after the curation. In fact, if you look close enough, you can tell that they only conduct formatting checks. In other words, if a unit is curated incorrectly, or the spectral data of sample 9 is treated as that of sample 2, there is no way for the validation units in XMLCONV to find out. Thus, we will introduce features and tools that help validate the value that exists during and after the curation. Moreover, an XML data update agent is discussed as it makes sure that all existing XML will still be valid after schema updates.

4.1 Validation during the curation

In this section, two preventive features for data validation are introduced.

4.1.1 Dropdowns in the Excel template

For some of the fields in the template, we know beforehand the possible pool of values it could choose from such as publication type, rheometer mode, and units. The data validation feature in Excel is utilized to create a dropdown menu to limit the values for certain cells. Figure 4-1 shows an example of the dropdown menu for the temperature units “Kelvin” and “Celsius”.
We also limit the possible values in the schema for terms with the dropdown in the template by the `<xsd:restriction><xsd:enumeration>` combo as double insurance.

### 4.1.2 Easy csv plotter

When curators are curating from literature with no raw data provided by the author, they will have to digitize data points from curves plotted in figures with the help of the WebPlotDigitizer[92]. Unfortunately, the digitizing process is tedious and prone to error. For example, data points are not by default sorted so we observed many zigzagging data in the past. Curators will download digitized data as two-column csv files as supplementary data files. Being able to plot the digitized data and compare it with the original figure is a straightforward way to validate. However, a figure can
contain more than five curves. According to our curators, it is frustrating to repeatedly open each csv file, plot the data within Excel, and switch log scale if needed. A csv plotting tool that is easy to use and easy to toggle log scales on both axes is in desperate need. Figure 4-2 shows the GUI of the easy csv plotter in MaterialsMine.

![MaterialsMine GUI](image)

**Figure 4-2: The easy csv plotter in MaterialsMine.**

The web app is developed using Vue.js with a drag-and-drop area on the left, and the canvas on the right. Two switches for log scale toggling are placed at the top of the canvas. Curators can use up to three clicks and releases to plot the digitized data. No moves are required to reset the canvas as it automatically refreshes when a new csv file is dropped in the dropping zone.

### 4.2 Validation after the curation

In this section, we will go through the tools and agents we developed for value checking after the curation.
4.2.1 Validation by an XML schema

An XML schema can not only constrain the structure of XML, but also the values of certain elements. For example, we can control an element to be of type xsd:double, meaning it must be a number. We can even wrap around it with <xsd:restriction base="xsd:double"> and put a <xsd:minExclusive value="0"> in it to allow only positive values. In the NanoMine schema, we created such a simple type called “PositiveValueType”. It is used widely in fields like density, molecular weight, frequency, etc. Figure 4-3 shows the error message thrown by a failed value check by the schema.

Figure 4-3: An example of a curation error detected by the XML schema validation.
4.2.2 Validation by visualization

Before the easy csv plotter was introduced, a local python tool with GUI called viz-validation\(^1\) was developed to check local copies of XML files that belong to the same dataset or paper.

As you might observe from Figure 4-4, features like the side-by-side comparison and the log-scale toggle really inspire the easy csv plotter, the difference is that the easy csv plotter is moved to the early stage before the curation since there is no reason to wait until the spectral data become part of the XML for validation.

Though I did not develop the app, there is another validation mechanism via the visualization module that I would like to touch on\[^93\]. We have had a few value-related curation errors detected in the downstream visualization module. To date, we have found exceptionally high Young’s modulus due to a misused “GPa”, which should be

---
\(^1\) https://github.com/bingyinh/nanomine-xml-viz-validation
“Pa”, exceptionally high glass transition temperature due to a misused “Celsius”, which should be “Kelvin”, and so forth. Abnormal data points are obvious when plotted together in the visualization module.

4.2.3 XML data update agent

As we have mentioned, eight versions of the NanoMine schema have been released since it is evolving overtime as more feedbacks are available from the community. To maintain the validity of XML’s curated under an older schema version, an update agent was developed. Users need to prepare a csv file to indicate a sequence of actions to convert XML in the older schema version into the latest version. Supported actions include “change”, ”add”, ”remove”, ”move”, and ”rename”. For more details, please refer to the documentation in the GitHub repo2.

4.3 Conclusion

In this chapter, a suite of features and validation tools designed to perform value checks was introduced. During the curation, the dropdowns in the Excel template help unify some commonly used terms like unit and characterization mode. An easy csv plotter, inspired by another validation tool “viz-validation” designed years earlier for after-the-curation validation, was created as a web app in MaterialsMine with the drag-and-drop feature to plot digitized spectral data for fast comparison with the original

2 https://github.com/bingyinh/xml-info-update
data during the curation. How the XML schema could be used for value checks as an after-the-curation validation layer was discussed in detail. As the schema evolves, XML’s sometimes become no longer valid according to the new schema. An XML data update agent was developed to address the issue.
5 NanoMine data standardization

Data standardization seeks to remove data inconsistency caused by various reasons, mostly due to what we called habits or conventions. Different researchers call polymers and fillers differently because of their habits. Mass fraction and volume fraction are used sometimes based on habits. Equivalent spectral data headers like “Dielectric constant”, “Real permittivity”, and “Real part of dielectric permittivity” are used based on habits. Thus, we introduce in this section a suite of tools and agents for data standardization in MaterialsMine with an emphasis on ChemProps in Section 5.2 due to its broader impact.

5.1 DOI crawler

Metadata including publication volume, issue, authors, and so on can be retrieved by a single DOI. During the early days, NanoMine curators complained about the amount of work required for typing the metadata. Typos are frequently observed in the metadata section. To bypass the manual curation and ensure metadata consistency, a DOI crawler was developed\(^1\). The crawler has two modules, one web-scraping based, and one CrossRef Query Service API based. The web-scraping module supports 15 leading publishers in the PNC domain. Typically, the DOI crawler will activate the web-

\(^1\) https://github.com/bingyinh/doi-crawler
scraping module first, and then fill in the missing pieces with the results fetched by the API-based module.

5.2 **ChemProps**

In this section, we will present our published work entitled “ChemProps: A RESTful API enabled database for composite polymer name standardization”[29].

5.2.1 **Introduction**

Significant advances in computing power in the past decade have given birth to many data-driven approaches including Materials Informatics, which facilitates understanding of processing-structure-property relationships and is a cornerstone in the materials design process[47], [94]–[96]. However, as Ramprasad et al. point out, Materials Informatics requires data to be reliable, uniform, and stored in a controlled manner[97]. This seemingly simple requirement has posed many challenges for polymeric materials data due to the prevalent use of different naming conventions and abbreviations for polymers. While the various aliases are fluently acceptable for humans, they confound attempts to curate data in the robust and consistent manner essential for indexing into databases for Materials Informatics, which is considered a major impediment to the adoption of machine learning techniques[98], [99].

The correctness of data indexing can impact many aspects of a data resource. Because of the lack of uniformity in the expression of polymer names in publications and data sets, exploration of the data via search and visualization tools becomes
problematic, leading to difficulties in using a polymer data resource as a viable tool in data-driven discovery. For example, the results returned for searching for Poly(methyl methacrylate) (PMMA) on Reaxys[100], which is a powerful database with properties and reaction data available for a wide range of substances including polymers, varies by the expression of PMMA².

NanoMine is another example. NanoMine is an ontology-enabled open-source data resource for the polymer nanocomposite community. The visualization tool of NanoMine is hampered by the polymer nomenclature in terms of the existence of multiple different labels for the same polymer. As a result, users cannot perform simple operations such as filtering the plot by polymer matrix readily. A tool resolving the nomenclature discrepancy is useful to polymer data resources like NanoMine and the whole community.

Researchers have made efforts in polymer indexing through the use of chemical identifiers which fall into two categories. The first is source-based such as PubChem CIDs[101], ChEBI IDs[102], and CAS numbers[103]. These number-form identifiers are generated by common online chemical platforms. Regardless of the data quality issue, these chemical identifiers are usually not favorable for cross-database studies due to source dependency[104]. The second category of chemical identifiers is source

---

² Results for “polymethyl methacrylate” counts 3974, while results for Poly(methyl-methacrylate) counts 1 on Reaxys. Note that the quotation marks here are part of the searching term.
independent, such as IUPAC names, SMILES notations[105], and InChI strings[106]. However, Akhondi et al. found that the consistency of the source-independent chemical identifiers between data sources counterintuitively varies significantly[107]. A recent effort entitled BigSMILES[108] was made to build upon the SMILES notation for a better representation of the stochastic nature of polymers. BigSMILES could be a solution to the inconsistency use of chemical identifiers once it is canonicalized, which is still ongoing. Even when a polymer database is correctly indexed, it is not intuitive for users to search for chemicals with chemical identifiers. In many cases, researchers cannot read the chemical identifiers since these expressions are not typically used regularly in communications or record keeping. Consider “polystyrene” as an example, whose IUPAC name is “poly(1-phenylethene-1,2-diyl)”, SMILES notation can be “*CC(*)c1ccccc1” and InChI string is “1S/C8H8/c1-2-8-6-4-3-5-7-8/h2-7H,1H2”. Researchers and publications most typically use simply “Polystyrene” or “poly(styrene)” or “PS” or a handful of similar common shorthands to refer to this common chemical compound. A hub that can link the chemical identifiers with the common conventional names that researchers use is in urgent need. Here we present a multi-algorithm-based mapping methodology that is optimized to solve the indexing issue and promote easy data exchange and application of rapidly advancing machine learning methods to polymer systems.
To generate insights from data resources, in addition to consistent identifiers, computational methods and algorithms require large-scale, efficient data exchange. Application Programming Interface (API) is a concise solution. A lightweight REpresentational State Transfer (RESTful) API simplifies the user interaction with the hub to an HTTP request and a return. APIs are widely used in many data resources in the materials design domain[41], [109], [110]. In this work, we propose ChemProps, a RESTful API-enabled database that takes in common polymer names and returns chemical identifiers (unique SMILES) with a tolerance of expression differences as mentioned earlier. The rest of the paper is as follows. We describe the structure of ChemProps API, data storage/update protocols of ChemProps database, and how we optimize the polymer name mapping algorithms in detail in Section 5.3.2. The results of optimization and examples of how to use ChemProps either through HTTP requests or GUI are given in Section 5.3.3. We discuss the potential uses of ChemProps in Section 5.3.4 and summarize in Section 5.3.5.

5.2.2 Methods

5.2.2.1 Structure of ChemProps API

Table 5-1 shows the input names and descriptions of a GET request.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>polfil</td>
<td>string</td>
<td>Search type: Use “pol” for polymer or “fil” for filler</td>
<td>“polfil”: “pol”</td>
</tr>
</tbody>
</table>
To use the services, users only need to refer to Table 5-1 to send a GET request to the endpoint at the ChemProps API[111]. We would like to highlight that only the polfil and the ChemicalName parameters are required inputs. Users can combine these two parameters with any combinations or none of the other three optional inputs to form a valid payload to the ChemProps API. The results will be returned within seconds in the format given in Table 5-2.

**Table 5-2: A list of returned information of a GET request if successful (Code 200).**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>StandardName</td>
<td>string</td>
<td>Standard chemical name</td>
<td>“StandardName”: “Polystyrene”</td>
</tr>
<tr>
<td>density</td>
<td>string</td>
<td>Chemical density at 25°C in g/cm³</td>
<td>“density”: “1.04”</td>
</tr>
<tr>
<td>uSMILES</td>
<td>string</td>
<td>Polymer unique SMILES</td>
<td>“uSMILES”: “C(C1=CC=CC=C1)[<em>][</em>]”</td>
</tr>
</tbody>
</table>

Figure 5-1 shows the overall structure of the ChemProps API for polymer name standardization. Please note ChemProps is designed for standardizing both polymer
names and composite nano-filler names. The *polfil* parameter, which states whether users search for a polymer name or a filler name, is assumed to be “pol” throughout this paper.

![Figure 5-1: The structure of the ChemProps API.](image)

In Figure 5-1, only the yellow part is visible to users, showing the essence of a lightweight RESTful API, while the back-end scripts carry all the loads. In the back-end process, first, a script will check for the existence of the SMILES parameter in the input and send it to the SMILES translation module for standardization, if present. The SMILES translation module sends the input SMILES string to the NIH Online SMILES

---

3 The “fil” option for filler search will be described elsewhere.
Translator to generate the Kekule unique SMILES (uSMILES) representation[112]. This module can remove flavors from SMILES and unify them in most cases. The details of the uSMILES generation can be found in the work by Weininger et al.[113] After this step, the input package moves to the algorithms section, which is the heart of ChemProps. A set of internal evaluation queries are designed to represent a predefined set of mapping algorithms. Each query, which implements a single algorithm, may find multiple results or may not find any results within the ChemProps core database. To ensure the reliability of our service, we assign an optimized weight, which will be updated over time as new polymers are included in the system, to each query and let them vote for the result by summing up the weights of each returned chemical. We describe the optimization process in a later section in detail. The chemical with the highest sum of weights will be returned. There are two cases when we will log the input and notify the admin. First is when no results can be found by any queries, which means ChemProps might need to welcome new chemicals. Second is when two chemicals tie at the top, which we try to avoid through optimization.

5.2.2.2 Data

The core ChemProps database was built on MongoDB, a NoSQL database. As a starting point, we begin with a core subset of all possible polymers to create a set of robust algorithms that are continuously calibrated to optimal performance with the introduction of more polymers into ChemProps. Raw data is stored on the cloud in a
Google spreadsheet for stability. We pulled 515 polymer names, considered as standard names, from Polymer Genome[23] along with associated SMILES and density information as the skeleton of the raw data. We adopt the customization of marking the linking atoms of the polymer chain with [*] by Polymer Genome. We then augmented the raw data with 49 polymer names pulled from NanoMine for a total of 564 unique polymer names sitting in the Google spreadsheet. Of these 564 unique polymer names, 89 polymer names are fully indexed with their unique SMILES, abbreviations, synonyms, trade names, and density information, which are used in the rest of this current study. All SMILES notations were converted to unique SMILES by the translation module[114]. Additional information was collected from sources including MatWeb[115], Polymer Database[116], and Polymer Science Dictionary[117] semi-automatically. Overtime, more of the 564 polymer names will become fully indexed. 54 fillers (such as silica or carbon nanotubes) are also indexed in ChemProps with their synonyms and density information.

To begin the algorithm development, we borrow the idea of bag-of-words (BOW)[118] from natural language processing to build a bag-of-characters notation (BOC). BOC is a string of digits with its first 26 digits indicating the occurrence of 26 alphabetic characters from a to z and its last 10 digits representing the occurrence of 10 Arabic numerals from 0 to 9. We cap the occurrence of a character at 9 in its BOC string since it is not common for a character or numeral to occur more than 9 times in a
polymer name. All strings are converted to lowercase before transforming into BOC strings. The merits of BOC include disregarding the character order and ignoring non-alphabetic and non-digit characters like whitespaces, dashes, and parentheses. These merits are especially useful for polymer name translation as minor changes in spacing and hyphenation are common in polymer representation. We also introduce a BOC-alph representation which is an alphabet-only version of the BOC string. Upon ingesting data from the cloud into the database, a script will convert standard names, abbreviations, synonyms, and trade names into BOC strings for storage. Figure 5-2 shows an example of the ingestion and conversion process.

![Figure 5-2: Illustration of the data ingestion process, BOC correspondence are color coded.](image)

On the cloud side, since the skeleton already covers chemical names, SMILES, and density, an administrator only needs to fill in the common abbreviations, synonyms, and trade names to “activate” a new polymer. A python script is called automatically
every week to back up the spreadsheet from the cloud, scan inside the 564 polymers for all the “activated” polymers that have either abbreviations, synonyms, or trade names filled in, which currently numbers 89, format the data, and push into the MongoDB for an update. This design makes it easy to update the ChemProps database both in-depth and in width, which is critical for a growing database, by only interacting with the Google spreadsheet. In-depth, administrators can add more polymers by simply adding a row in the Google spreadsheet. In width, a new property such as glass transition temperature can be added to the ChemProps database by adding a column in the Google spreadsheet and a few more lines in the digesting scripts.

5.2.2.3 Weighted voting algorithm for name mapping

The conventional way of dealing with synonyms is the brute force approach, which simply tabulates all known synonyms as shown in Figure 5-3.

![Figure 5-3: Screenshots of synonyms of polystyrene on PubChem[119] and ChEBI[120].]

This method, however, is unstable and still subject to errors as new differences in expression are encountered. Based on our experience with the 1500+ samples in
NanoMine, we propose 12 mapping algorithms to vote for the final match given the user input. Table 5-3 summarizes the proposed algorithms.

Table 5-3: Summary of the proposed mapping algorithms.

<table>
<thead>
<tr>
<th>Algorithm ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search within unique SMILES with translated SMILES</td>
</tr>
<tr>
<td>2</td>
<td>Search within standard names with ChemicalName</td>
</tr>
<tr>
<td>3</td>
<td>Search within abbreviations with ChemicalName</td>
</tr>
<tr>
<td>4</td>
<td>Search within synonyms with ChemicalName</td>
</tr>
<tr>
<td>5</td>
<td>Search within abbreviations with Abbreviation</td>
</tr>
<tr>
<td>6</td>
<td>Search within BOC with BOced Abbreviation</td>
</tr>
<tr>
<td>7</td>
<td>Search within tradenames with TradeName using relaxed BOW comparison</td>
</tr>
<tr>
<td>8</td>
<td>Search within BOC with BOC-alphed TradeName by comparing the first 26 digits</td>
</tr>
<tr>
<td>9</td>
<td>Search within BOC with BOced ChemicalName</td>
</tr>
<tr>
<td>10</td>
<td>Search within BOC with BOC-alphed ChemicalName by comparing the first 26 digits</td>
</tr>
<tr>
<td>11</td>
<td>Search within standard names with ChemicalName using relaxed BOW comparison</td>
</tr>
<tr>
<td>12</td>
<td>Search within synonyms with ChemicalName using relaxed BOW comparison</td>
</tr>
</tbody>
</table>

We create a variety of BOC and BOW algorithms for high-precision polymer name identification that is robust to expression differences. For example, in Table 5-3 algorithm #6 considers abbreviations like PA 6-6, PA 66, and PA 6/6 as the same.

However, the BOC method cannot identify that “Diglycidyl ether of bisphenol-A epoxy resin” and “diglycidyl ether of bisphenol-A” are equivalent. To address this issue, we introduce the relaxed BOW comparison, which is a variation from BOW. We first split the input by non-alphabetic characters to form a bag of words. Then we search the
database for entries that contain all the words in the bag. Though the two DGEBA epoxy expressions have different BOC strings, they can be identified by algorithm #12 since the BOW of the second expression is a subset of the first expression. Due to the difference in reliability, we apply a weight to each algorithm and optimize the weights for the best mapping performance. The collection of the weights is denoted as a vector \( w \).

In order to optimize the weights of algorithms, we use the data from NanoMine divided chronologically into training and validation sets. We extract the reported polymer data following the format given in Table 5-1. We hereby define a searching package (\( SP \)) as a combination of ChemicalName, Abbreviation, TradeName, and SMILES. The training set contains 166 unique sets of \( SPs \), while the test set contains 54 unique sets of \( SPs \). To start the training process, the initial weight of all 12 algorithms is assigned to one. When an \( SP \) is passed to ChemProps, all 12 algorithms will be used for evaluation one by one. For each algorithm, ChemProps will look at all of the 129 indexed polymers to see if there is a match evaluated through the algorithm using the terms in the \( SP \). Intuitively, we create a vector \( R \) of length 12 for each indexed polymer for initialization. We then fill in the nth spot with either 0 or 1 depending on the evaluation using the nth algorithm with 0 indicating not a match and 1 otherwise. When the \( SP \) is evaluated by all 12 algorithms, we will have 129 \( R \) vectors in theory. However, many of the 129 \( R \) vectors are filled with 12 zeros, which are trivial and discarded. For that specific \( SP \), ChemProps only records the indexed polymers with non-trivial \( R \) vectors.
and computes their scores by taking the dot product of $w$ and $R$. Only the non-trivial $R$ vectors are generated for better computational efficiency, and we denote the names of indexed polymers associated with those non-trivial $R$ vectors as candidate chemical names, one of which must be the ground truth chemical name based on our training data.

In the first pass through ChemProps of the 166 training SPs, 102 SPs get paired with more than one $R$ vector and 64 SPs get paired with only one $R$ vector, which we will discuss later. For each of the 102 SPs, we select from the recorded non-trivial $R$ vectors the ones with score ranked top two. Since there could be a tie in the score for $R$ vectors, we end up collecting 262 $R$ vectors from the 102 SPs. For this subset of data, our goal is to maximize the score difference between the ground truth chemical name and other top-ranked candidate chemical names via an optimal weight vector $w$. To accomplish this goal, for each of the 102 SPs, we define a vector $x$ as the difference between the $R$ vectors of wrongly mapped candidate chemical names and the $R$ vector of the ground truth chemical name. $x$ is a vector of length 12 with values of -1, 0, or 1: -1 results if the algorithm finds an incorrectly mapped chemical name, 0 results if the algorithm either finds both or finds neither of the correct and the wrongly mapped chemicals, and 1 results if the algorithm only finds the ground truth chemical name. Then the score difference can be represented by the dot product of $x$ and $w$. It is worth mentioning that the $R$ vector of the ground truth chemical name is not necessarily 1 in
all spots. For the rest 64 of the 166 SPs, each of them is paired with only one R vector.
Since the associated candidate chemical names are indeed the ground truth chemical name, we denote these 64 R vectors as Rc with the letter “c” standing for “correct”.

An illustration of the training data structure is shown in Figure 5-4.

![Figure 5-4: Structure of training data for weight optimization.](image)

From the bottom up it shows the breakdown of the training data set. The grey blocks give examples of R vectors and Rc vectors. y being 1 stands for the ground truth chemical name and being 0 stands for the wrong mapping. From SP1 to SP102, ChemProps returns multiple candidate chemical names meaning multiple R vectors mapped to one SP. Consider SP1 as an example, there are three candidate chemical
names returned with $R_1$ being the $R$ vector of the ground truth. Our next step is to obtain the difference between the ground truth $R_1$ and the wrongly mapped $R_2$ and $R_3$, leading to the formation of $x_1$ and $x_2$. $SP_{103}$ to $SP_{166}$ all return the only ground truth forming the 64 $R_c$ vectors.

We seek to maximize the score difference between the ground truth chemical name and the wrongly mapped candidate chemical names, while maintaining the weights to be non-negative and the score of 64 $R_c$ vectors to be positive, ensuring that the ground truth chemical name will still be returned when $w$ is converged. Therefore, we propose the null negative form of the optimization problem as follows:

$$\text{minimize} \quad f(w) = -\min (x_i \cdot w^T), \quad i = 1, \ldots, 160$$

subject to

$$-w_j \leq 0, \quad j = 1, \ldots, 12$$

$$-Rc_k \cdot w^T < 0, \quad k = 1, \ldots, 64$$

The dot product of $x$ and the weight vector $w$ is the score difference that we would like to maximize, where a positive score difference indicates that the ground truth chemical name has a higher score than the other candidates and thus will be returned by ChemProps. Instead of maximizing the score difference in the entire training set, we can reduce the objective function to only maximizing the minimum value of all the score differences, as stated in the function above. A detailed example of how we obtain the score difference is given in Figure 5-5.
The example SP in Figure 5-5 provides “polyvinyl alcohol” as ChemicalName and “PVA” as Abbreviation. There are two candidates returned, the ground truth Poly(vinyl alcohol) and the wrongly mapped Poly(vinyl acetate). Since both candidates can be abbreviated as “PVA”, algorithms #5 and #6 return both candidates. However, algorithms #9 and #10 can only return the ground truth because Poly(vinyl alcohol) has the same BOCed and BOC-alphed chemical name as the input ChemicalName. Therefore, the score difference is the sum of the weights assigned to algorithms #9 and #10, i.e. \( w_9 \) and \( w_{10} \). It is worth noting that algorithm #2 does not find the ground truth due to the
existence of the parentheses in the standard name highlighting the need for algorithms
#9 and #10 using BOC.

We select the widely used Generalized Reduced Gradient (GRG) nonlinear
method for optimization. We also apply a ten-fold validation to the optimization
problem in which the 160 x’s are randomly divided into ten groups. For each iteration,
we omit a group of 16 x vectors and conduct optimization on the w vector using the
remaining 144 x vectors. We then use the omitted x vectors for validation each time and
enforce that the minimum score of the omitted x vectors must be positive to pass. This
process is repeated ten times.

5.2.3 Results

5.2.3.1 Optimization

The results of the optimization and ten-fold validation are summarized in Table
5-4.

<table>
<thead>
<tr>
<th>Fold</th>
<th>w_1</th>
<th>w_2</th>
<th>w_3</th>
<th>w_4</th>
<th>w_5</th>
<th>w_6</th>
<th>w_7</th>
<th>w_8</th>
<th>w_9</th>
<th>w_{10}</th>
<th>w_{11}</th>
<th>w_{12}</th>
<th>Min training score</th>
<th>Ten-fold test score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>1.2</td>
<td>1</td>
<td>2</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>1.2</td>
<td>1</td>
<td>2</td>
<td>0.8</td>
<td>2.4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>1.2</td>
<td>1</td>
<td>2</td>
<td>0.8</td>
<td>2.4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>1.2</td>
<td>1</td>
<td>2</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>1.2</td>
<td>1.2</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>-1</td>
</tr>
</tbody>
</table>
The \( w \) values in Table 5-4 are the converged weight factors. For each iteration, the minimum training score is the minimum score difference in the 144 \( x \) vectors used for optimization, while the ten-fold test score is the minimum score difference in the omitted 16 \( x \) vectors used for validation. The results show that all but iterations 5 and 6 converge to a consistent set of weight factors \( w^* = \{1, 1, 1, 0.8, 0.8, 0, 0, 1.2, 1.2, 1, 2\} \) with a positive ten-fold test score. In contrast, the ten-fold test score of iterations 5 and 6 are non-positive, which means that within the 16 omitted cases, at least one wrongly mapped chemical name has a score that is higher or equal to the ground truth. If we adopt the converged weight factors in iterations 5 and 6, we know that ChemProps will return wrong chemical names in these cases. Our next step is to check the validity of \( w^* \) over the training data in iterations 5 and 6 by recomputing the minimum training score and the ten-fold test score of the two iterations when the original converged weight factors are replaced with \( w^* \). Positive values for both updated scores indicate that we can consider \( w^* \) a feasible solution to our optimization problem on the training set. \( w^* \) works for iteration 5, whose min training score decreases from 1 to 0.8 and the ten-fold test score increases from 0 to 0.8. \( w^* \) also works for iteration 6, whose min training score
remains the same at 0.8 and the ten-fold score increases from -1 to 1.6. Therefore, we have shown that $w^*$ is a feasible solution for the training set.

The final step is to verify that $w^*$ also works for the 54 SPs from the test set. We configure the ChemProps algorithm section to use $w^*$ and feed the 54 SPs to the API. All SPs have the correct mapping returned, indicating $w^*$ is a feasible solution to our optimization problem. Note that $w^*$ assigns zero weight to algorithms #7 and #8, which are TradeName related. It shows trade names are, at this point, not an important mapping feature. However, we retain them in the overall framework as the weight factors will continue to evolve through the presented optimization pipeline as ChemProps grows. It is also noted that due to the source of NanoMine data being mostly published literature, none of the current SPs, training or test, contain SMILES. That is why the weight of algorithm 1 is the same as the initial value in any iteration. We assume SMILES as a chemical identifier is reliable if provided. We keep the weight for algorithm #1 as one and as more data enters the system with SMILES fields, the coefficients will evolve.

5.2.3.2 Example of API access for ChemProps

In this example, we introduce how to use python to access ChemProps API. More examples for accessing by Javascript, Java, Go and PHP are provided [see Appendix A]. The package requirement includes requests, which can be installed through “pip install requests” in the command line. Users must login into the NanoMine platform by InCommon or OneLink accounts to request tokens needed for
API calls. Procedures to apply for the account and to request tokens are provided elsewhere [see Appendix A]. Once the token is obtained, users can use the following scripts to use ChemProps API and search, for example, polystyrene:

```python
import requests
TOKEN = '$your_token'
CP_URL = 'https://materialsmine.org/nmr/api/chemprops'
payload = {
    'dataformat':'json',
    'polfil':'pol',
    'chemicalname': 'polystyrene',
    'abbreviation': 'PS',
    'tradename': 'Styrofoam',
    'smiles': '*CC(*)c1ccccc1'
}
headers = {'Authorization': 'Bearer {}'.format(TOKEN)}
r = requests.get(CP_URL, params=payload, headers=headers)
data = r.json()

The returned data variable is a dict object that takes the form of:

{'StandardName': 'Polystyrene', 'density': '1.04',
'uSMILES': 'C(C(C1=CC=CC=C1)[*])[*]'}

As guided in Table 5-1, users can replace the corresponding payload parameters with their values. Optional parameters like Abbreviation, TradeName, and SMILES can be removed from the payload in the previous example.
5.2.3.3 Example of GUI access for ChemProps

In addition to the ChemProps API, we also developed a Graphical User Interface (GUI) for users that are not familiar with API calls. We make the GUI open for users without an InCommon or OneLink account. Figure 5-6 shows the ChemProps GUI.
Figure 5-6: The ChemProps Graphical User Interface[121].
Users can select search for either polymer or filler and put the keywords into the corresponding text boxes. For polymer search, a convenient quick search box is enabled, where users can input any one of the chemical name, abbreviation, SMILES, or trade name to do the search. An extra feature as compared to the API call is that the chemical structure and formula[122] will be displayed on the page when a result is found in the ChemProps polymer database. If no results are found in ChemProps, the page will display a warning message “No results found. Admin alerted to update the database. Please try again in a week.” with an error code 404.

**5.2.4 Discussion**

ChemProps is a hub that acknowledges the large number of variations of names for polymers, merges those aliases in an intelligent and systematic way, and speaks the languages that both computer and materials scientists understand. Apart from its potential in Materials Informatics in the future, it can be useful to the materials science community immediately. Similar to the example of Reaxys, widely used chemical databases like ChEBI use text search for chemicals by default, which brings inconsistency in the results⁴.

With ChemProps, those chemical databases can comb their polymer data and remove duplicates as needed.

---

⁴ 5 entries were found by searching for Poly(methyl-methacrylate), while 52 entries were found by searching for polymethyl methacrylate on ChEBI.
Another immediate usage of ChemProps is to convert common polymer names to uSMILES. Since the introduction of SMILES, major polymer-related chemical databases gradually support the more accurate “search by SMILES” function. However, many researchers do not read or speak SMILES. This language barrier stops users from utilizing the more advanced searching functionalities. ChemProps can act as a bridge to enable general chemical name querying in those databases instead of using the complicated SMILES. A general workflow could be 1) users input chemical names in the search box of those databases as usual 2) the search function makes an API call to ChemProps 3) ChemProps returns the result including the uSMILES 4) the search function uses the returned uSMILES for accurate searching. We must point out that since SMILES are software-generated, it has different flavors. To reduce the impact of those software-generated flavors, ChemProps uses uSMILES throughout the system. So ChemProps returns uSMILES only. To use ChemProps as a bridge for querying by SMILES, it is recommended that our potential collaborators also use uSMILES generated by the SMILES translation module of ChemProps to be consistent with ChemProps such that the returned value of an API call to ChemProps can be used in the query directly.

Beyond the original purpose of ChemProps, we have discovered that a robust solution to the polymer indexing problem in NanoMine has led to the solutions of related problems. For example, the amount of filler in a composite system can be expressed by a mass fraction or volume fraction, which are convertible if the densities of
the polymer matrix and the filler are available. However, such data are not typically provided in publications. By standardizing the polymer and filler names, we have been able to use ChemProps to automatically populate the density data to initiate the mass fraction – volume fraction conversion process. Now that the naming issue is solved, there are yet other automation agents that can be enabled beyond the mass fraction – volume fraction conversion agent. For example, an autonomic inference agent that populates storage modulus and loss modulus for polymers can be written readily to deploy. Single-value properties that do not vary significantly within the same polymer class like dielectric constants in the polymer database prepared by Huan et al. [123] can also be readily integrated into ChemProps thanks to its ability to resolve polymer names in different forms and flexible data ingestion design. Beyond properties, we can couple with another effort to produce better polymer indexing entitled BigSMILES[108] by ingesting the BigSMILES representations that were specifically designed for polymer systems into the ChemProps database to provide another machine-readable representation in the output of the ChemProps API.

An emerging machine-readable format of chemical mixtures is valuable to Materials Informatics, especially polymer composite data resources like NanoMine since mixtures have a significant presence throughout the processing steps[124]. However, currently, polymers need to be defined by names or database identifiers due to the size of macromolecules as stated by the authors. ChemProps could help with the issue since
it standardizes polymer names and provides the associated uSMILES, which is machine-readable. In line with the effort to describe the processing steps, a recent work named PolyName2Structure[125] seeks to convert polymer name representations to monomer structures, predict the polymerization pathway and identify the reaction groups. In addition to IUPAC names and source-based names, PolyName2Structure also accepts common names by passing them to an in-house dictionary to resolve the structures. PolyName2Structure can expand the range of feasible common names by making an API call to ChemProps.

5.2.5 Conclusion

In this work, we propose a twelve-algorithm-based mapping methodology named ChemProps that is optimized to solve a polymer indexing issue that routinely impedes the progress of Materials Informatics for polymeric-based systems. ChemProps acknowledges the large number of variations of names for polymers, merges those aliases in an intelligent and systematic way, and speaks the languages that both computer and materials scientists understand. A cloud-based design makes ChemProps not only reliable but also easy for admins to add data and expand properties. With the RESTful API, users can access the powerful service with a few simple lines of an HTTP request or through the user-friendly graphical user interface powered by ChemProps API with a quick search function enabled.
To ensure the accuracy of ChemProps, we assign a weight factor to each algorithm to generate scores for candidate chemical names and optimize them using the data from NanoMine. We favor the cases when ground truth chemical names have a higher score than other candidate chemical names as the correct mapping will be returned. Therefore, our goal is to maximize the score difference between the ground truth and other candidate chemical names. To further reduce the computational problem, we maximize the minimum value of the score difference between the ground truth chemical name and the other candidate chemical names. Ten-fold validation is utilized on the training data points to prevent overfitting issues. We configured ChemProps with the converged set of weight factors using training data and tested it on the test-set searching packages. A 100% test accuracy is achieved. The weight factors of two algorithms related to trade names currently converge to zero, showing that trade names might not be an important mapping feature at present. However, we retain the two algorithms in the overall framework since the weight factors will evolve through the same optimization pipeline as ChemProps grows. In the current set of data, we use 166 searching packages (each corresponding to a unique polymer) as training data, retaining 54 searching packages for testing. It is important to note that ChemProps is continuously growing and the living algorithm presented here will be continuously run to update the weighting functions as new polymers are added to the system.
We believe ChemProps can contribute to the community in several ways. First, other polymer databases can use ChemProps to comb their data and remove duplicate entries. Second, other polymer databases that enable the more accurate “search by SMILES” function can allow users to input common human-readable names while using ChemProps as a translator through API calls to conduct SMILES search in the back-end. Third, the easy-to-update design makes ChemProps a good tool to auto-populate polymer properties, which is useful to enable a growing array of automated agents for materials data resources.

### 5.3 Mass fraction - volume fraction conversion agent

The idea of a mass fraction - volume fraction conversion agent is briefly introduced in Section 5.2.4 as it directly benefits from ChemProps. The code base is available on GitHub\(^5\). The agent was developed with no assumptions about the number of matrix components or filler components. Since the volume fraction or mass fraction is provided, we can assume the total volume to be one or the total mass to be one with not units. We can then compute the dimensionless volume or mass of each component with the density given either by curators or by ChemProps. The final step is to compute the volume fraction or mass fraction by calculating the ratio of each component and the composite with dimensionless volume or mass. A final check of the calculated fraction

\(^5\) [https://github.com/bingyinh/mf-vf_conversion](https://github.com/bingyinh/mf-vf_conversion)
with the reported fraction is deemed valid if their difference is less than 1E-5. The mass fraction – volume fraction conversion agent has a potential weakness because it does not support one corner case. If a sample with multiple filler components reports only mass fraction for a filler component and only volume fraction for another, the agent will not be able to compute correctly. Fortunately, it is a safe assumption that the corner case is rare.

5.4 Spectral data header parser

This work was completed based on the preliminary work done by the summer Research Experiences for Undergraduates student Joshua Facello, who was my mentee.

5.4.1 Overview

When curators upload supplementary data files for spectral data, we require the first row to be axis labels. More specifically, in a scientific plot, the axis label usually contains the name or symbol of a property followed by a unit. Take Figure 3-9 as an example, the axis label for the x-axis is “Temperature (Celsius)” and the axis label for the y-axis is “Tan delta”. Though the use of axis labels might be consistent within the dataset, it is usually inconsistent from paper to paper. Authors use different representations for property names, units, and the separation between names and units. The inconsistency could also stem from the typos of curators. The ability to standardize property names and units is necessary for the downstream querying and visualization module to group and plot spectral data correctly. Without standardization, the
visualization module would not be able to plot “E’ (MPa) vs. T (°C)”, “Storage Modulus [\*10^6 Pa] vs. Temperature [Celsius]”, and “storage modulus/MPa vs. temperature/deg C” reported in different works together for comparison. Essentially, there are four tasks for the standardization tool: 1) split property names and units, 2) map property names to a standardized expression, 3) map units to a standardized expression, and 4) pick a unit from the pool of standardized units like (Pa, kPa, MPa, GPa) or (Hz, rad/s) and convert data to use that unit. The last task is handled by a unit conversion agent developed by our collaborator on the knowledge graph side of NanoMine. We will focus on the parser we developed to handle the first three tasks in this section.

Figure 5-7 is an illustration of the input and output of the spectral data header parser.

![Figure 5-7: An illustration of how the spectral data axis parser works. Input: <headers>. Output: <AxisLabel>.

The parser will read the input XML and find all <headers> elements. For each of the <headers> elements, the two axis labels along with the XML path (XPath) will be
used to generate the standardized property names and units for the x-axis and y-axis, which will be inserted as a sibling element of the <headers> element back into the XML. We included the XPath because we can infer what property the header is about from the XPath and thus narrow down the pool for name and unit standardization.

5.4.2 Data preparation

All headers curated into NanoMine were collected along with their XPath are collected. A simple heuristic rule-based name-unit separator was used to coarsely separate property names and units, followed by a manual correction of the separation results. From the collected data, we defined a few unit types like temperature, pressure, time, etc. A set of standardized names and units are also selected. Then, we assign all instances in our dataset with the standardized names, denoted as stdName, and units, denoted as stdUnit, along with the unit type, denoted as unitType. Figure 5-8 is a screenshot of the dataset. Note that stdName is a merge of stdXName and stdYName. Similarly, stdUnit is a merge of stdXUnit and stdYUnit.

Figure 5-8: A screenshot of the dataset prepared for the spectral data header parser.

Thus, we have the following mappings: stdName-name, stdUnit-unit, unitType-unit, stdName-unitType, and XPath-stdName. Finally, we merge the pairs to form the
graph that serves as the reference for the mapping algorithms. We will explain with examples of how the graph is utilized for inference in Section 5.4.4.

5.4.3 Name-unit separation

Before we can use the graph prepared in the previous section, we must first separate the name and unit. Most authors use “()”, “[ ]”, or “/” to separate units from property names. However, there exist some corner cases where units contain nested parenthesis or mixed use of common separators. For example, the unit of specific heat capacity $C_p$ is $(J/(g*K))$. Our solution is to split the name and unit in every possible way. Table 5-5 demonstrates the results.

<table>
<thead>
<tr>
<th>Name</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_p$</td>
<td>$J/(g*K)$</td>
</tr>
<tr>
<td>$C_p(J/)$</td>
<td>$g*K$</td>
</tr>
<tr>
<td>$C_p(J$</td>
<td>$(g*K))$</td>
</tr>
</tbody>
</table>

Table 5-5: Output name-unit pairs of inputting "CP (J/(g*K))" into name-unit separator.

We kept the cases with open parentheses because similar typos were seen in the NanoMine curation. All possible splits will be fed into the downstream modules for mapping. The mapping algorithm will be the judge to return the mapping with the highest level of confidence.
5.4.4 Decision space pruning

We could start mapping the name and unit to their standardized form now. However, XPath can help us significantly prune the decision space or the graph since XPath infers the property, which infers the possible stdName and stdUnit. An illustration of how the graph is pruned and utilized for inference is provided in Figure 5-9 with an example (XPath, name, unit) input triple.

![Diagram](image)

**Figure 5-9: Illustration of the inference by the merged graph with an example input.**

The XPath is used first to narrow down the stdName selections. For each candidate stdName, we know all previously curated names associated with it. We can also infer the unitType associated with it, which usually has less than two candidate unit types. From the unitType, we can find the associated stdUnit. Then, the final step is to determine the corresponding stdName and stdUnit for the name and unit in the input.
5.4.5 Algorithms

Three layers of mapping algorithms are implemented. Layer 1: direct search in the dictionary of standard names and units. Layer 2: pattern matching using the python difflib package with Ratcliff-Obershemp algorithm with a cutoff score set at 0.6. Layer 3: pattern matching with Levenshtein distance with a cutoff distance set at 2. Details of the algorithms can be found in Section 2.4. If a layer finds a match, the result will be returned with a confidence score without executing subsequent layers. Any match found in layer 1 will be assigned 0.95, which is an arbitrary value that could be optimized. The metric returned by the Ratcliff-Obershemp algorithm is by nature a similarity score. Any match found in layer 2 will thus be assigned the similarity score. In layer 3, the Levenshtein distance is not a value that ranges from 0 to 1. If a match is found in layer 3, we divide the Levenshtein distance by the length of the input string.

Of all the matches returned, we will pick the one with the highest score as the final match. This step is necessary to handle the name-unit splitting challenge stated in Section 5.4.3. One final step is to build the <AxisLabel> subtree and insert it back into the XML.

5.4.6 Discussion

The spectral data header parser is a useful tool for data standardization, that resolves many data inconsistencies in the spectral data headers for the downstream visualization module. There is plenty of room for improvement. First, the cutoff
parameters were selected either using the default value or by rules of thumb. More data are needed to optimize the parameters. Second, the parser cannot generalize. It is built with heuristics from curated NanoMine data. There exist a few properties in the NanoMine schema that could be reported as spectral data but are never used. For these unseen data, our parser will not be able to find a match from the assembled graph. Third, the parser currently conducts string-match individually for name and unit, which should be correlated with each other.

5.5 Spectral data interpolation agent

As we mentioned, if authors do not provide the raw data, curators have to digitize data from figures. Digitizing works well on most occasions but there are exceptions. There is a special group of spectral data that has a real part, an imaginary part, and a tan delta curve, which is interchangeable. In the NanoMine schema, there are three of them: the dynamic modulus under viscoelastic properties, the rheological complex modulus, and the dielectric permittivity. The three spectral data are among the most frequently curated spectra in NanoMine. In publications, authors usually pick two from real, imaginary or loss, and tan delta to report because tan delta is the imaginary part divided by the real part. To be able to obtain the third curve, we must ensure that the two reported curves are digitized with the exact same layout on the x-axis, which is
too hard to accomplish. Instead, we solved the issue with an interpolation agent. A typical run works like this: 1) the agent takes an XML as input, 2) it searches for the three possible properties, 3) the agent checks if there is a missing piece that could be computed from the reported spectral data, 4) if so, the agent interpolates/extrapolates the two reported spectra so that they share an x-axis, 5) the agent calculates the missing piece with the interpolated/extrapolated data, 6) the agent inserts the new data and updates the reported spectral data with interpolation/extrapolation.

### 5.6 Conclusion

In this chapter, a thorough introduction was given to an array of data standardization tools utilized in the NanoMine curation workflow, including the DOI crawler that fetches metadata for curators with a simple DOI, the ChemProps API service that recognizes and standardizes common polymer and filler names, the mass fraction – volume fraction conversion agent that calculates and inserts the missing fraction to the XML, the spectral data header parser that detects and resolves property names and units from curated axis labels for spectral data, and the spectral data interpolation agent that interpolates or extrapolates multiple spectral data to achieve a shared x-axis. These tools and agents help the downstream querying and visualization module in MaterialsMine fetch and display more relevant data.

---

*https://github.com/bingyinh/spectra_data_interpolation*
6 NLP-driven semi-automated curation

6.1 Introduction

With the advent of the Materials Genome Initiative (MGI) [18], there is a growing vision to integrate data science and machine learning to forge new capabilities in the understanding and design of materials, with applications from health care to advanced structures to renewable energy[38]. The work to date has focused on the development of machine learning algorithms to leverage data from computational models and make new discoveries[95]. The lack of FAIR (Findable, Accessible, Interoperable, and Reusable) materials data resources means that it is difficult to utilize data generated by peers in the field, pushing researchers to rely on computational models that could generate a fair amount of data, which is necessary for training a decent machine learning model, within their own lab. To address this issue, we have been developing MaterialsMine\(^1\), an ontology-driven open-source FAIR data resource for polymer nanocomposites (PNC) and metamaterials[24], [25], [57], [93]. Besides the data generated in our own labs, we have been curating data from the literature. Data curation is challenging in that it requires domain knowledge and it is highly repetitive for humans, prone to errors. In a subdomain like PNC study, there are dozens of new publications every week, which is too many for experienced human curators to catch up with unless

\(^1\) https://materialsmine.org/nm#/
they are full-time curators. Unfortunately, a good funding model is not yet in place for curators to work full-time. Thus, the limited curated data significantly drags down the data-driven design of materials. As an alternative, an additional urgent need is to harness the power of AI to extract data from the vast, published, archival literature, and to make that data FAIR by incorporating it into a robust materials data framework.

Access to this enormous array of published data, both experimental and computational, would transform our ability to use existing knowledge in understanding and developing design paradigms for new materials. Natural language processing (NLP), which seeks to automate the extraction of knowledge from human-written text, offers an opportunity to make this data accessible and readily reusable by humans and machines.

In the MaterialsMine curation workflow, the first step is to create a sample list, which summarizes the experimental units reported in the paper and their composition and properties. Human curators use sample lists as guidance for curation. Five major steps are involved in sample list generation: 1) identify experimental units or sample code, 2) identify sample composition, 3) identify properties characterized for each sample, 4) associate sample code with composition, and 5) associate sample code with properties. The five steps can be translated smoothly into two structured prediction tasks in NLP. The task of named entity recognition (NER), which involves identifying and categorizing word-spans in a document, is suitable for the “identify” steps. The
relation extraction (RE) task that seeks to predict the relationship between entities can be applied to the “associate” steps.

To date, a few materials researchers have begun to apply NLP techniques to the NER task, focusing on inorganic materials like metal oxides[126], zeolites[89], and nanomaterials[127]. A similar recent effort utilizes rule-based heuristics and an unsupervised Snowball algorithm for relation extraction (RE) to generate ontologies for a class of crystallographic materials[62]. However, NER and RE for inorganic, crystalline materials are relatively simple because the compact chemical formula of inorganic materials acts as unique identifiers. In contrast, organic materials, especially polymers, cannot be uniformly represented. PNCs are even more complex, with the introduction of nanofillers of complex geometry and chemistry. Accordingly, authors of PNC papers refer to experimental samples with fluid language; a single sample in a paper may be referred to as “1 wt.% silica in epoxy,” “epoxy/1 wt.% SiO2,” “epoxy-SiO2-0.01,” or “Ep-SiO2-01” interchangeably, which makes annotating a corpus for training an NER system difficult[128].

The large pre-trained transformer-based NLP models have achieved state-of-the-art performance in various downstream tasks, including NER and RE, in recent years[69]. Most of the NLP + materials science works leverage the “pre-training then fine-tuning” paradigm to train their models. It has been concluded in multiple works that the transformer-based models pre-trained on domain-specific corpus outperforms
the ones pre-trained on generic natural languages since sentences in materials science publications are extremely specialized\cite{61}, \cite{129}. Due to the uniqueness of the PNC language and the fact that materials science corpora used in existing works are kept private, we need to create our unannotated pre-train corpus and annotated fine-tune corpus from PNC publications.

Of all available models in the broad transformers model family, a group of BERT-based encoder models is typically selected for structured prediction tasks like BERT, RoBERTa, and DeBERTa\cite{70}–\cite{72}. Examples in the materials science domain include SciBERT, MatBERT, and MatSciBERT, all of which selected to pre-train BERT-based transformers with domain-specific corpora\cite{82}, \cite{91}, \cite{130}. The “matching the blanks” (MTB) architecture as an extension to the BERT model is reported to perform decently on RE tasks\cite{131}. Recently, formulating structured prediction tasks as text generation tasks becomes increasingly popular, as the structure of BERT-based models is typically adapted to the specific task, causing its incapability of multitasking. Seq2seq models like T5 have shown their versatility on an array of NLP tasks, structured or unstructured, with one single model\cite{73}. The TANL framework was developed for an array of structured prediction tasks to be formed as a translation task between the target sequence and the input augmented natural language, building on top of T5\cite{132}.
As illustrated in Figure 6-1, our vision is to create a semi-automated curation pathway by generating sample lists from PNC journal articles, that could gradually evolve into an automated curation pathway that could populate the MaterialsMine knowledge graph directly from the articles in the future. This work serves as a first step towards the sustainable future of MaterialsMine driven by the automated-curation pipeline. In this work, we will focus on the following topics: 1) a detailed description of our PNC corpus generation, 2) a summary of pre-training T5 with PNC corpus, 3) exploration and assessment of viable options for NER and RE tasks for sample list generation, 4) discussion of the results.

6.2 Methods

6.2.1 Dataset collection and preparation

During sample list generation, curators are suggested to pay special attention to figures and tables since sample codes, composition, and properties are most likely reported in those components. Information can also be extracted from paragraphs where
a discussion of figures or tables is presented. Learning from this practice, we focus on
the figure caption, table caption, and sentences that refer to a figure or a table when
building our PNC corpus. All datasets use an 80:20 split for the training set and
validation set.

6.2.1.1 Caption-mention corpus – the pre-train corpus

The caption-mention corpus discussed in this work consists of 1M (1,002,904)
sentences sourced from figure captions, table captions, and sentences that mention a
figure or a table in the body text of 23,090 PNC papers. Figure 6-2 demonstrates the
construction process of the caption-mention corpus.

![Diagram of the construction process of the caption-mention corpus]

Figure 6-2: Summary of the construction of the caption-mention corpus.
A Scopus API query was utilized to obtain 99,985 doi’s with keyword filtering of “polymer + composite”. The obtained doi’s are further filtered by keywords (“poly” or “rubber”) and “composite” in the abstract. Doi’s of book chapters are removed from the collection. The list of doi’s is then grouped by the publishers, resulting in 18,210 doi’s from Elsevier, 1,922 doi’s from RSC, and 2,958 doi’s from ACS. The Elsevier corpus is obtained via the Elsevier API, which returns XML’s. The RSC and ACS are obtained via an HTML scraper developed in-house. The markup language files are then parsed with a modified HtmlReader of the ChemDataExtractor package. For each doi, we store the abstract, the full text structured with top-level headers and content, figure captions, and table captions, all of which are normalized with the python unicodedata package. We then extract sentences that mention a figure or a table from the full-text content. Finally, we use ChemDataExtractor to perform sentence segmentation on all the figure captions, table captions, and sentences that mention a figure or a table, to build our caption-mention corpus. Sentences with a length between 10 and 256 after tokenization are kept in the pre-train corpus.

---

2 https://dev.elsevier.com/documentation/ScopusSearchAPI.wadl
3 https://dev.elsevier.com/documentation/FullTextRetrievalAPI.wadl
4 https://docs.python.org/3/library/unicodedata.html
6.2.1.2 Annotated caption corpus – the fine-tune corpus

The annotated caption corpus discussed in this work consists of 1896 captions collected from 214 PNC papers curated into the MaterialsMine data resource. The doccano annotation platform is used for NER tagging and RE tagging. One graduate student leads the annotation while another graduate student is in charge of double-checking the annotation.

For NER tagging, we propose four classes of named entities for sample list generation purposes: Sample code (S), composition (C), group reference (G), and property (P). For a span to be labeled with S, it must either be able to point any materials scientists to a unique experimental unit without reading through the full paper or used explicitly as a sample code in the paper. Spans that indicate nanofiller loadings like a mass fraction or volume fraction will be labeled with C. Similarly, spans that describe the property of interest in the figure or table are labeled with P. In PNC papers, it is common to compare properties within a group of PNCs with different nanofiller loadings in a figure or a table. Instead of listing all sample codes in the captions, authors usually use spans like “epoxy nanocomposites” or “silane-modified samples” for group reference. Such spans are labeled with G. It is worth mentioning that C spans sometimes overlap with S spans. For example, “epoxy/1 wt.% SiO2” is a S span while “1 wt%” is a C span. Since most of the BERT-based NER models do not support overlapping named entities, we will remove overlapped C span in this case for simplicity in downstream tasks.
For RE tagging, we propose three relation classes: `isPropertyOf`, `isCompositionOf`, and `isMemberOf`. `isPropertyOf` can be applied to the (P, S) pair, (P, C) pair, and (P, G) pair, indicating a P span is reported for the other entity in the pair. `isCompositionOf` is straightforward as it can only be applied to the (C, S) pair. It is common that a caption contains multiple C tags and S tags, making the `isCompositionOf` class necessary. `isMemberOf` can be applied to the (S, G) pair and (C, G) pair, bridging the group reference to a sample or a smaller group of samples with identical nanofiller loadings.

For detailed annotation guidelines, please refer to Appendix B.

![Figure 6-3: A screenshot of the caption corpus annotation with doccano.](image)

Figure 6-3 is an example of a figure caption annotated for NER and RE in the doccano platform[133]. The resulting corpus has 2028 entities with the S label, 491 entities with the C label, 1606 entities with the G label, 2465 entities with the P label, 4262 entity pairs labeled with `isPropertyOf`, 633 entity pairs labeled with `isCompositionOf`, and 872 entity pairs labeled with `isMemberOf`. 
6.2.1.3 Datasets

For the NER task, an 80:20 split was adopted, resulting in 3045 sentences for training and 762 sentences for testing.

For the RE task, similarly, an 80:20 split resulted in 9328 sentences for training and 2332 sentences for testing. An additional “Other” label was added for 5893 entity pairs with no relation. Since no entity pairs with the same NE labels were annotated with a relation, we did not include those in generating the sentences for the RE task. For each entity pair with different NE labels, we generate a sentence with either entity markers (EM) or the augmented natural language pre-processing for TANL. For the EM style pre-processing, each NE of the entity pair was wrapped around with entity markers “<e1>“, ”<e1>“, ”<e2>“, and “</e2>“. For the TANL style pre-processing, please refer to the description of the relation classification task in their original paper.

6.2.2 Pre-training of T5

The T5-base model is pre-trained with multiple optimizers and training arguments on the caption-mention corpus. We used the same denoising pre-training objective as reported in the T5 paper that replaces dropped-out spans with sentinel tokens with a 15% corruption rate and an averaged 3 tokens per corrupted span. Our models are implemented with HuggingFace[134].

We evaluated the following training settings with a batch size of 16 and 8 gradient accumulation steps resulting in 128 total train batch size: 1) AdamW optimizer
with a peak learning rate of 5e-4, 2) AdamW optimizer with a peak learning rate of 5e-5, 3) AdaFactor optimizer with the AdaFactor scheduler that adjusts learning rate internally, 4) AdaFactor optimizer with an external peak learning rate of 1e-3, and 5) AdaFactor optimizer with an external constant learning rate of 1e-3. AdamW cases used a weight decay of 1e-3. Each model was scheduled to warmup for 5000 steps. A linear scheduler was utilized unless otherwise specified. Models were evaluated every 2500 steps. The maximum length of the input sequence is limited to 256. The best pre-trained model was trained on an NVIDIA Quadro P5000 GPU with 16GB GPU RAM for 6 days. Pre-training codes are available at our GitHub repository\(^5\).

**6.2.3 Downstream tasks**

**6.2.3.1 NER**

For the NER task, the BILOU tagging scheme was adopted for pre-processing the labels. The input and label encodings generated by the tokenizers are truncated or padded to a fixed length of 200. For baselines, we assessed encoder models like DeBERTa-base, MatBERT, and MatSciBERT, and seq2seq models like TANL for NER with the T5-base model as the starting point, and two other formulations of the target sequence for T5-base model to treat NER as a text generation task. In the first formulation, the T5 model predicts a sequence of label tokens. The second option is to

\(^5\) [https://github.com/bingyinh/NLP_PNC_sample_list](https://github.com/bingyinh/NLP_PNC_sample_list)
predict an interleave style of word token and label token. An example of the two formulations is as follows.

**Input:** Fig 3. Tg of PMMA-silica-0.1.

**Output (label sequence):** “<O><O><O><U-P><O><B-S><I-S><I-S><I-S><L-S><O>”

**Output (interleave):** Fig<O>3<O>.<O>Tg<U-P>of<O>PMMA<B-S>-<I-S>silica<I-S>-<I-S>0.1<L-S>.<O>

For simplicity, we add label tokens as additional special tokens to the T5 tokenizer, so that each label token is guaranteed not to be tokenized into multiple sub-tokens. To avoid disturbing the embedding of the existing vocabulary, like the “O” tag, we wrapped every label token with “<” and “>” before we added them to the tokenizer. 17 label tokens were added to the T5 tokenizer.

Apart from the baselines, we also assessed three seq2seq formulations, namely TANL, label sequence, and interleave token label sequence, with our pre-trained T5 model for the NER task.

Micro-averaged precision, recall, and F1 score are used as the metric for the NER task. Each model is fine-tuned until the F1 score evaluated on the test set stops increasing.

### 6.2.3.2 RE

For the RE task, baselines include the “matching the blank” (MTB) architecture on top of the BERT, MatBERT, and MatSciBERT model with entity marker (EM) as a
state-of-the-art architecture for RE task among the encoder models, and the TANL model built on top of the T5-base model with the augmented natural language for the relation classification task. We assessed our pre-trained T5-base model on the RE task with two proposed approaches: 1) use the TANL framework but with our pre-trained T5-base model, and 2) an EM-style input sequence and relation triple style output sequence. For the EM-style fine-tuning, like what we did in the NER task, we added entity markers and relation labels wrapped in “<” and “>” as additional special tokens to the T5 tokenizer. An example target sequence will be “<isPropertyOf><e2><e1>". Input sequences were truncated or padded to a fixed length of 200.

Micro-averaged F1 score is used as the metric for model evaluation. Each model is fine-tuned until the F1 score stops increasing.

Since our goal is to create a pipeline for sample list generation in MaterialsMine, using individual single-task models for NER and RE separately might bring deployment concerns. The charm of the seq2seq model lies in its versatility. There are two options for us to use one seq2seq model for both tasks, namely a multi-task seq2seq model and a joint-entity-relation extraction model.

### 6.2.3.3 Multi-task NER + RE

A multi-task TANL on top of our pre-trained T5 model was trained in a multi-task setting with a separated NER dataset and RE dataset. A task prefix, like “NM_NER :” and “NM_REL :” was added to each sentence as a prefix. The micro-
averaged F1 score of this model on the NER task and the RE task will be compared with single-task models as well.

6.2.3.4 Joint-entity-relation extraction

A TANL model starting with our pre-trained T5 model was trained for a joint-entity-relation extraction task as well. After being translated into the augmented natural language, an example input sentence will become:

**Input:** Fig 3. Tg of PMMA-silica-0.1.

**Output:** Fig 3. [Tg \(\mid P\mid isPropertyOf = PMMA\text{-silica}-0.1\) \(\mid S\)].

6.2.3.5 Hyperparameters

For the NER task, our hyperparameter selections are described as follows. For our T5-base model, we tested a wide range of learning rates from \([5e-5, 1e-4, 2e-4, 3e-4, 4e-4, 5e-4, 1e-3]\), weight decay from \([1e-4, 1e-3, 1e-2, 1e-1]\), batch size from \([8, 16]\), number of beams from \([5, 10]\). We did not set a cap on the training epochs for any of the models being assessed. The stopping criteria are purely based on the validation F1 score.

For the experiments with our pre-trained T5-base model. The pre-trained T5-base with label sequence style output generation for the NER task was trained with a learning rate of \(1e-3\) and a weight decay of \(1e-2\) for 300 epochs with a warmup step of 1000. The peak F1 score was reached at epoch 203. The other option with the interleave style output for the NER task shared the same training configurations as the previous model,
with a quick converge to the peak F1 score at epoch 23. The training configuration for
the TANL + pre-trained T5-base setup is identical to the baseline TANL model for NER
as described in Section B.2.1.

For the RE task, a fixed weight decay of 1e-2, a batch size of 32, and a linear
scheduler with AdamW optimizer were adopted for fine-tuning the pre-trained T5-base
model. The model was trained with a learning rate of 5e-5 for 240 epochs with a
warmup step of 200, followed by 60 epochs with a learning rate of 1e-4, 30 epochs with a
learning rate of 2e-4, and 30 epochs with a learning rate of 3e-4. The training
configuration for the TANL + pre-trained T5-base setup is again identical to the baseline
TANL model for RE as described in Section B.2.1.

For the multi-task NER + RE TANL model, default configurations were used
other than a max input sequence length of 256, a max output sequence length of 512, a
batch size of 8, and a gradient accumulation step of 2. The model was trained for 60
epochs with the same datasets used for the single-task NER and RE cases.

For the joint-entity-relation extraction TANL model, a set of training
configurations that is similar to the multi-task model was used. The input and output
sequence lengths were set to 256 and 512, respectively. The batch size was set to 8. The
model was trained for 200 epochs on a joint-entity-relation dataset that has the same format as the CoNLL04 dataset⁶ used in SpERT[135].

For detailed hyperparameters of each baseline experiment, please refer to Appendix B. The number of beams for beam search is set at 5 for all T5 implementations.

6.3 Results and discussion

6.3.1 Pretraining of T5

The pre-training history is provided in Figure 6-4.

![Pre-training history of T5 model with caption-mention corpus.](image)

**Figure 6-4: Pre-training history of T5 model with caption-mention corpus.**

The five cases were experimented on one after another. Cases 2) and 3) perform significantly worse than case 1) at around 10,000 steps. Thus, they were terminated early. Case 4) showed a decreasing trend at 30,000 steps, so we kept it training for another 30,000 steps with the scheduler started afresh until the validation loss

⁶ http://lavis.cs.hs-rm.de/storage/spert/public/datasets/conll04/conll04_train.json
converged. Case 5), which performs the best on the validation set, was also kept training
till 60,000+ steps. Case 5), AdaFactor with constant external learning rate at 1e-3, as
reported in the T5 paper for fine-tuning, obtained the best validation loss at 0.457.

6.3.2 Downstream tasks

6.3.2.1 NER

Table 6-1 summarizes the micro-averaged precision, recall, and F1 scores of the
assessed models on the NER task. Note that the multi-task TANL model and the joint-
entity-relation extraction TANL model are included in the table along with other single-
task NER models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeBERTa base</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MatBERT</td>
<td>85.6</td>
<td>83.9</td>
<td>84.7</td>
</tr>
<tr>
<td>MatSciBERT</td>
<td>78.1</td>
<td>82.7</td>
<td>80.4</td>
</tr>
<tr>
<td>TANL</td>
<td>79.7</td>
<td>80.6</td>
<td>80.2</td>
</tr>
<tr>
<td>T5-base label seq</td>
<td>79.6</td>
<td>79.4</td>
<td>79.5</td>
</tr>
<tr>
<td>T5-base interleave</td>
<td>73.0</td>
<td>77.2</td>
<td>75.0</td>
</tr>
<tr>
<td>Pre-trained T5-base label seq</td>
<td>80.9</td>
<td>82.3</td>
<td>81.6</td>
</tr>
<tr>
<td>Pre-trained T5-base interleave</td>
<td>57.0</td>
<td>80.2</td>
<td>66.6</td>
</tr>
<tr>
<td>TANL + Pre-trained T5-base</td>
<td>80.7</td>
<td>79.3</td>
<td>80.0</td>
</tr>
<tr>
<td>TANL multi-task + Pre-trained T5-base</td>
<td>81.6</td>
<td>81.0</td>
<td>81.3</td>
</tr>
<tr>
<td>TANL joint-entity-relation + Pre-trained T5-base</td>
<td>85.1</td>
<td>81.2</td>
<td>83.1</td>
</tr>
</tbody>
</table>

As expected, encoder models perform well on the NER task. We can tell that
MatBERT, which was pre-trained on a corpus consisting of 2M full-text materials science
journal articles, performs the best in the NER task. The DeBERTa-base model, as an
advanced BERT model with disentangled attention, also achieves a good F1 score of 82.4 even though it was not pre-trained on a domain-specific corpus. Interestingly, the TANL multi-task model and TANL joint-entity-relation model obtains better F1 scores than the single-task TANL model with our pre-trained T5-base model as the starting point. It suggests that learning for the RE task can be beneficial to the NER task. Our pre-trained T5-base model helps the label sequence formulation increase its F1 score from 79.5 to 81.6, which is still impressive given that our caption-mention corpus is more than 100 times smaller than the MatBERT pre-train corpus since we only used caption-related sentences excerpted from a total of 23k papers, and that it outperforms the MatSciBERT model, which was pre-trained on ~150k full-text materials science journal articles. Surprisingly, the pre-trained T5-base model does not help the interleave formulation outperform its baseline alternative.

6.3.2.2 RE

The micro-averaged F1 scores after evaluating multiple RE models on our annotated caption corpus can be found in Table 6-2. Again, the multi-task TANL model and the joint-entity-relation extraction TANL model are included in the table as well.

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTB + BERT&lt;sub&gt;EM&lt;/sub&gt;</td>
<td>91.6</td>
</tr>
<tr>
<td>MTB + MatBERT&lt;sub&gt;EM&lt;/sub&gt;</td>
<td>94.9</td>
</tr>
<tr>
<td>MTB + MatSciBERT&lt;sub&gt;EM&lt;/sub&gt;</td>
<td>93.0</td>
</tr>
<tr>
<td>TANL</td>
<td>95.5</td>
</tr>
<tr>
<td>Model Configuration</td>
<td>Score</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Pre-trained T5-base\textsubscript{EM}</td>
<td>95.8</td>
</tr>
<tr>
<td>TANL + Pre-trained T5-base</td>
<td>95.8</td>
</tr>
<tr>
<td>TANL\textsubscript{multi-task} + Pre-trained T5-base</td>
<td>96.2</td>
</tr>
<tr>
<td>TANL\textsubscript{joint-entity-relation} + Pre-trained T5-base</td>
<td>71.2</td>
</tr>
</tbody>
</table>

Since only 3 relation classes were annotated in our fine-tune corpus, the performance of all models listed in Table 6-2 is decent. While the MTB + MatBERT\textsubscript{EM} model performs the best among the encoder models with a score of 94.9, the best micro-averaged F1 score of 96.2 was reached by the TANL\textsubscript{multi-task} model. In fact, the multitask model could reach a higher F1 score of 97.0 with a lower NER F1 score of 80.5. A decent score of 95.8 was obtained by our pre-trained T5-base model with entity markers, and the TANL for relation classification formulation with our pre-trained T5-base model as the starting point. The TANL\textsubscript{joint-entity-relation} model, on the other hand, did not do well in the RE task, probably due to around 30% of the generated sequences cannot match the input tokens exactly, which is called a “wrong construction” in the TANL framework.

As shown in the example given in section 6.2.3.4, the augmented natural language allocates a longer span to RE expressions than NER expressions. A failed reconstruction thus impacts the RE task more than the NER task, leading to a high NER score but a low RE score for the TANL\textsubscript{joint-entity-relation} model.

### 6.3.3 Discussion

Figure 6-5 summarizes the three potential approaches to implement the encoder or seq2seq models for NER and RE tasks.
Figure 6-5: Proposed pipelines for the application of NER and RE models in MaterialsMine data curation.

From the performance perspective, the top three single-task NER models are MatBERT (84.7), DeBERTa-base (82.4), and Pre-trained T5-base label seq (81.6). The top three single-task RE models are Pre-trained T5-base EM (95.8), TANL + Pre-trained T5-base (95.8), and baseline TANL (95.5). Thus, the best performance of a NER-then-RE task for sample list generation could be achieved with a MatBERT model for NER and a Pre-trained T5-base EM model or a TANL + Pre-trained T5-base model for RE. For the NER-then-RE task, we can also use a multitasking model. In our case, the TANL multi-task + Pre-trained T5-base model achieves a score of 81.3 on the NER task and a score of 96.2 on the RE task. Apart from the NER-then-RE task, a joint-entity-relation extraction task can also serve the purpose of sample list generation. The TANL joint-entity-relation + Pre-trained T5-base model achieves a score of 83.1 on the NER task and a score of 71.2 on the RE task.
The model performs well on NER with a score of 83.1 but poorly on RE with a score of 71.2. The reason for the significantly low RE score is probably that the relation extraction part of a joint-entity-relation extraction task in TANL does not use input sentences with correctly labeled named entities as other single-task RE or multi-task RE models do. Thus, the RE score of 71.2 is a reflection of the combined performance of the NER part and the RE part of the joint-entity-relation extraction task. As for the other two approaches, if the performance is our pure focus, we would prefer the single-task-model approach as it outperforms the multi-task-model approach on the NER task by an F1 score of 3.4, though its performance on the RE task is lower by 0.4. However, if we include production deployments into our scope, a pipeline with a single multi-task model has many advantages over one with two single-task models in terms of resource and code base simplicity.

In addition, we believe that the multi-task model has the potential to do better on the NER task. The reason why the single-task MatBERT model performs the best on the NER task is because of the size of its pre-train corpus. As we have mentioned, the MatBERT model was pre-trained from the BERT-uncased-base model with 2M full-text materials science publications. On the contrary, our pre-train corpus consists of captions and caption-related sentences extracted from 23k polymer nanocomposite papers. It is safe to say that the MatBERT model was pre-trained on a corpus that is more than 100k times bigger than ours. Limiting our pre-train corpus to only caption-related sentences
might also limit the semantic understanding of T5. Thus, our next step is to extend our
pre-train corpus to include full-text PNC papers and re-run the pre-training of T5.

NER and RE can generate entity-relation-entity triples, which is still one step
away from the sample list generation. With the detected named entities and relations,
we need an extra layer of entity resolution to merge the detected entity-relation-entity
triples into a graph, which can then compose the sample list, which will be future work.

6.4 Conclusions

In this work, we presented the methods we used to collect a domain-specific
unannotated corpus for pre-training and a domain-specific annotated corpus for fine-
tuning an array of BERT-based models and seq2seq models for NER and RE tasks on
captions excerpted from PNC publications. A T5-base model was pre-trained using 1M
caption-related sentences collected from 23k PNC articles. A fine-tune corpus containing
1896 figure captions from PNC papers was annotated with named entities from 4 classes
and relations from 3 classes. Three approaches were proposed and evaluated for NER
and RE tasks. For the NER task, our baseline includes three BERT-based models, two of
which were pre-trained on domain-specific corpora, and three T5 models with different
formulations of the target sequence, with one being TANL, one being a sequence of label
tokens, and the other being an interleaved sequence of input token and label token. The
three T5 models were evaluated with our pre-trained T5-base model as the starting point
for comparison. The result shows that the MatBERT model outperformed other models
with an F1 score of 84.7. For the RE task, our baselines include the three BERT-based models used in the NER baselines with the MTB framework and entity markers inserted into the input sequence for relation extraction, and a T5 model with TANL formulation of the target sequence for relation extraction. The TANL formulation was also evaluated with our pre-trained T5-base model along with a formulation of input sequence with entity markers and output sequence being relation statement triples, both of which achieved 95.8 in F1. In addition to single-task models, a multi-tasking TANL formulation of our pre-trained T5-base model was also fine-tuned and evaluated on the NER task with a score of 81.3, and the RE task with a score of 96.2. A third approach of fine-tuning our pre-trained T5-base model with the TANL formulation for the joint-entity-relation extraction task was tested to perform well on NER with a score of 83.1 but poorly on RE with a score of 71.2.

Purely from the performance perspective, using MatBERT for NER and pre-trained T5-base with entity marker or TANL formulation for RE should outperform other options. However, if we include the difficulty of production deployment in our scope, the TANL_{multi-task} + pre-trained T5-base model is preferred, whose performance on NER has the potential for improvement if we extend our pre-train corpus to include full-text PNC papers instead of caption-related-only sentences. We hope our work could inspire data curation for a variety of scientific fields beyond the materials science community.
7 Conclusions

7.1 Summary of contributions

MaterialsMine, an open-source FAIR data resource for the polymer nanocomposites community and metamaterials community, was established to facilitate data-driven materials design that is faster than the traditional trial-and-error style of exploration. To be FAIR, we need a clear and extensible data representation to enable the interoperable knowledge exchange. We thus designed the NanoMine XML schema. With the data framework and data representation in place, we still need tools and a user-friendly interface for data curation. In Chapter 3, we explained in detail the customized Excel template we designed as a more comfortable user interface for our potential curators – materials scientists, the construction of the web app for data uploading, and the realization of ingesting the Excel template along with the uploaded supplementary data files by the back-end XMLCONV, and the small validation units for formatting checks across the workflow. To reduce and prevent curation errors and thus improve data quality, we need data validation mechanisms for value checks. To address the need, we discussed the validation mechanisms embedded both during and after the curation in Chapter 4. We also introduced a tool that we developed to update existing XML’s curated in an older version of the schema to comply with the latest schema. On many occasions, even without human-caused curation errors, the data resource cannot perform to its full capacity due to data inconsistencies. In Chapter 5, we introduced a
suite of tools developed for data standardization in MaterialsMine. We introduced ChemProps, a RESTful API-enabled multi-algorithm-based polymer/filler name mapping methodology, with a special emphasis due to its general applicability to the whole polymer informatics community. Maximum efforts have been spent to improve the manual data curation pathway up until this point. To create truly powerful and transformative materials design paradigms and towards a sustainable future for MaterialsMine, we need to harness the power of AI to efficiently extract a significant set of data from the published, archival literature. Thus, we introduced our first step towards a semi-automated or fully automated curation future in MaterialsMine with NLP in Chapter 6. We argued that the sample list generation step in the manual curation pathway can be translated into NER and RE tasks in NLP. We discussed in detail how the two corpora for pre-training and finetuning are constructed from PNC publications. Three approaches for NER and RE pipeline construction were proposed and an array of viable BERT-based models and seq2seq models were evaluated for potential applications in the semi-automated curation pipeline for MaterialsMine. Following is a detailed bullet of contributions corresponding to each research task stated in Section 1.7:

**Task 1: To build a user-friendly human curation pathway for NanoMine/MaterialsMine data resource.**

- Together with my colleague, a customized Excel template was created with the potential of more than a 40% reduction in fillable cells as compared to the master template.
• A user-friendly webapp for template and supplementary data uploading with carefully designed instructions was developed.
• A back-end pipeline named XMLCONV was developed to perform end-to-end conversion from Excel to XML with the support of a customized template.
• Many formatting validation units are embedded in XMLCONV. An email will be sent to the user once the conversion job is complete or aborted due to failed validation. Informative error messages are included to help pinpoint errors fast.

Task 2: Develop tools and mechanisms that facilitate the curation and enhance data quality.

• Implemented value constraints in template and schema, in addition to the structural constraints of XML schema, for data validation.
• Developed a python application named Viz-Validation with a GUI that re-plots digitized spectral data from XML for comparison with the original figures.
• Inspired by Viz-Validation, developed an easy csv plotter webapp in MaterialsMine that supports drag-and-drop with convenient log-scale toggling on both axes.
• A DOI crawler that automatically collects metadata with a crawler module and a query module.
• Developed ChemProps, a RESTful API-enabled multi-algorithm-based polymer/filler name mapping methodology to solve the polymer/filler indexing issue.
• A mass fraction – volume fraction conversion agent was developed to automatically convert the author-reported mass fraction or volume fraction into its counterpart with the density retrieved from the ChemProps API.
• A spectral data header parser was developed to standardize the property names and units reported as axis labels for spectral data.

Task 3: Explore methods towards sustainable and (semi-)automated data curation of MaterialsMine data resource.

• A caption-mention corpus was generated with 1M caption-related sentences sourced from 23k PNC publications.
• An annotated caption corpus was generated with 1896 captions from 214 NanoMine/MaterialsMine papers annotated with 4 NER labels and 3 RE labels.
• A T5-base model was pre-trained with the caption-mention corpus with the denoising objective.
• BERT-based baselines DeBERTa, MatBERT, and MatSciBERT, and seq2seq baselines TANL, T5-base\textsuperscript{label seq}, and T5-base\textsuperscript{interleave} were fine-tuned for the NER task. The three seq2seq baselines were also evaluated using our pre-trained T5-base model as the starting point for the NER task.

• BERT-based baselines BERT, MatBERT, and MatSciBERT were evaluated under the “matching the blank” framework with entity markers inserted for the RE task. TANL, as the seq2seq baseline, was evaluated along with two seq2seq models fine-tuned with our pre-trained T5-base model, one with TANL formulation for RE and the other with entity-marker style input and relation statement triple style output.

• Our pre-trained T5-base model was fine-tuned with a TANL multi-task formulation.

• Our pre-trained T5-base model was fine-tuned with a TANL formulation for the joint-entity-relation extraction task.

• Potential applications of NLP in semi-automated data curation in scientific data resources were discussed and demonstrated.

7.2 Future works

The following research directions are recommended as an extension of the works presented in this dissertation.

An AI-assisted data ingestion interface based on a stepper curation system for MaterialsMine

Materials data curation bears a resemblance to tax preparation, with high
complexity, quantitiveness, and high co-occurrence of certain data point pairs.

Wizard-based curation tools like common commercial tax preparation software provide
a good user experience and have a quick learning curve. Therefore, as an alternative to
the Excel template approach, an AI-assisted “stepper” curation tool could be developed
for a better curation experience. During curation, the system will ask for the DOI if the
work is published, ask for basic information about the materials system, ask for some
key processing steps, and let users select from a list of common characterization methods ranked by the co-occurrence frequency of those methods being applied to the materials system reported by users earlier, and finally guide users to input properties obtained via the specified characterization methods.

Figure 7-1: A schematic of the proposed AI-assisted Data Ingestion interface.

As shown in Figure 7-1, the process will be AI-assisted in that it prunes irrelevant data forms based on 1) the co-occurrence probability of materials systems-characterization methods and characterization methods-properties we obtained via the corpus mentioned in Chapter 6 (red path), and 2) the frequently used characterization methods reflected by the curated samples with similar groups of authors queried via DOI (blue path).

PNC papers usually report multiple samples with shared matrix materials, processing steps, characterization methods, and properties. Thus, similar to when tax
software preloads tax forms from previous years, when users indicate a DOI and curate one of the samples, they will not have to go through the process again when they curate other samples from the same work. Instead, the stepper curation will be further pruned to guide users to modify the values they reported for a previous sample, as displayed in the green path in Figure 7-1.

**NLP-enhanced ChemProps 2.0**

In Section 5.2, we introduced ChemProps with a multi-algorithm-based mapping methodology. Now, we can utilize the full-text corpus as described in Section 6.2.1 to enhance ChemProps in two aspects. First, we can extract polymer names and their abbreviations from the corpus with either rule-based heuristics or ChemDataExtractor to potentially increase the data collection in ChemProps[76]. Second, the traditional statistical pipeline utilized for name mapping in ChemProps can be upgraded in three different ways: 1) use a WordEmbedding model, such as Mat2Vec[90], to generate embeddings from input polymer names, abbreviations, tradenames, etc., and compute the cosine similarity of the embeddings against the embeddings of all reported names in the ChemProps collection, similar to what is done in this paper[81]. 2) instead of a WordEmbedding model, use a pre-trained encoder model, such as MatBERT and MatSciBERT, to pull the latent representation of input names from the hidden layer and compute cosine similarity. 3) translate the named entity normalization task into a
seq2seq task, pre-train a T5 model with our full-text corpus, and use it for the end-to-end mapping.

**NLP-driven spectral data header parser for MaterialsMine**

As introduced in Section 5.4, the spectra header parser has plenty of room for improvement. First, the name-unit separator is primitive. This task can be translated into a token classification task, by assigning property name tokens “<NAME>” label and unit tokens “<UNIT>” label. Second, the mapping of names and standard names, units and standard units, can be upgraded with NLP models, similar to what was recommended in the previous paragraph for ChemProps 2.0.

**A complete pipeline for sample list generation in MaterialsMine**

The work we presented in Chapter 6 is the first step to generate a sample list. It stops at the stage where entity-relation-entity triples are detected, which is one step away from generating the sample list as shown in Figure 7-2.
Figure 7-2: A proposed end-to-end pipeline for sample list generation in MaterialsMine.

An extra entity resolution layer is required to merge the entity-relation-entity triples into a graph, which will be equivalent to a row in a sample list. We also left the table content out-of-scope for the time being. It is also an important piece for sample list generation as it could suggest the mapping of sample codes (in its header row or column) and group reference (in its caption). Once that is accomplished, we could move one step forward to try to extract actual values from text and tables and fill them into the sample list. Note that extra labels and thus extra annotations will be required, though values are usually numerical characters that could be extracted with regex in a reliable way.
Appendix A

This document provides text and figures pertaining to (1) procedures for Duke OneLink account applications; (2) procedures to request tokens needed for ChemProps API; (3) examples for accessing ChemProps API by Javascript; (4) examples for accessing ChemProps API by Java; (5) example for accessing ChemProps API by GO; (6) example for accessing ChemProps API by PHP.

A.1 Duke OneLink account application

Users can go to https://accounts.oit.duke.edu/onelink/register to apply for a Duke OneLink account. Without a Duke OneLink account or InCommon account (under development), users may not be able to apply for the API token and thus cannot use the ChemProps service through API calls. Figure A-1 shows the landing page of the Duke OneLink account registration.
Figure A-1: The landing page of the Duke OneLink account registration

A.2 ChemProps API token request

To use the ChemProps API, users must go to the ChemProps GUI at https://materialsmine.org/nm#/ChemProps first to request a token. This can be done by clicking on the “REQUEST API TOKEN” button located at the top of the ChemProps GUI page as shown in Figure A-2 or directly by the URL https://materialsmine.org/nm#/ChemPropsAPIToken.
Figure A-2: Illustration of the request API token button on the ChemProps GUI page.

If users are not logged in on NanoMine through OneLink or InCommon account, an alert message will display for redirection to login or OneLink account registration as shown in Figure A-3.

Figure A-3: Example token request page if users are not logged in

If users have already logged in and they have not requested a ChemProps API token before, they will be asked to provide a domain secret as shown in Figure A-4.
Then the “REQUEST TOKEN” button will be activated. Clicking on the button and the token will display as shown in Figure A-5.

Users should keep their tokens safe. In case users forgot their token, the same message containing the token information will display if users log back into the system and navigate to the token request page.

**A.3 Example Javascript code for using ChemProps API**

Assume the token for all examples is **Bearer**

OiJIUzI1NiIsInR5cCI6IkpXVCGxlLmNvbSIsImlzQW5vbnltb3VzIjpmYWxzZSwi
A.4 Example Java code for using ChemProps API

OkHttpClient client = new OkHttpClient().newBuilder().build();
Request request = new Request.Builder()
   .url("https://qa.materialsmine.org/nmr/api/chemprops?chemicalname=polystyrene&abbreviation=PS&tradename=Styrofoam&smiles=*CC(*)c1cccccl&polfil=pol")
   .method("GET", null)
   .addHeader("Authorization", "Bearer OiJIUzI1NiIsInR5cCI6IkpXVCGxlLmNvbSIsImlzQW5vbnltb3VzIjpmYXxZSwidXNlc0cyI6dHJ1ZSwia")
   .build();
Response response = client.newCall(request).execute();
A.5 Example GO code for using ChemProps API

```go
package main

import (
    "fmt"
    "strings"
    "net/http"
    "io/ioutil"
)

func main() {
    url := "https://qa.materialsmine.org/nmr/api/chemprops?chemicalname=poly styrene&abbreviation=PS&tradename=Styrofoam&smiles=*CC(*)c1ccccc1&polfil=pol"
    method := "GET"
    payload := strings.NewReader(""")
    client := &http.Client {
    }
    req, err := http.NewRequest(method, url, payload)
    if err != nil {
        fmt.Println(err)
    }
    req.Header.Add("Authorization", "Bearer OiJIUzI1NiIsInR5cCI6IkpXVCJ9")
    res, err := client.Do(req)
    defer res.Body.Close()
    body, err := ioutil.ReadAll(res.Body)
    fmt.Println(string(body))
}
```
A.6 Example PHP code for using ChemProps API

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://qa.materialsMine.org/nmr/api/chemprops?chemicalname=polystyrene&abbreviation=PS&tradename=Styrofoam&smiles=*CC(*)c1cccccl&polfil=pol');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array('follow_redirects' => TRUE));
$request->setHeader(array('Authorization' => 'Bearer OiJIUzI1NiIsInR5cCI6IkpXVCGx1LmNvbSIsImlzQW5vbnpmYWxzZSwidXNlc0N0cyI6dHJ1ZSwia
kV4aXN0cyI6dHJ1ZSwia
));
$request->setBody('');
try {
    $response = $request->send();
    if ($response->getStatus() == 200) {
        echo $response->getBody();
    } else {
        echo 'Unexpected HTTP status: ' . $response->getStatus() . ' .
        $response->getReasonPhrase();
    }
} catch(HTTP_Request2_Exception $e) {
    echo 'Error: ' . $e->getMessage();
}
```
Appendix B

This document provides a detailed description of (1) the annotation guidelines for NER and RE tasks, and (2) the baseline models evaluated on NER and RE tasks.

B.1 Annotation Guidelines for NER and RE tasks

B.1.1 Named entity classes

Sample code (S): A unique name given to an experimental unit by the author.

Composition (C): The mass fraction or volume fraction that indicates the nanofiller loadings of the nanocomposites, including non-numerical characters like “wt%” or “phr”. C class is often overlapping with S and G. Label them as we need them for the relation annotation.

Group reference (G): A name used by the author to refer to a group of unique experimental units. The common feature of the group of experimental units is usually adopted as the group reference. For example, “PMMA composites”, “composites with 5wt% loadings”, or “samples with “silane-modified samples”.

Property (P): A measurable that is characterized in materials science research. Sometimes, a short discussion will be included in the caption with some derived property names. Label all of them for the NER task. Do not label those for the RE task.

B.1.2 Relation classes

isPropertyOf: The relation that indicates a property P is reported in the caption for a sample code S, a composition C, or a group reference G. More than often, you will
find two or more (P, S), (P, C), (P, G) pairs that can be labeled. In such a case, the annotation level of priority is (P, S) > (P, C) > (P, G). There is one exception, when G refers to a group of samples with the same volume fraction or mass fraction like “<P>Tg</P> of <G>composites with <C>5 wt%<C> loadings</G>.” Then we should label (isPropertyOf, <P>, <G>) and (isMemberOf, <C>, <G>).

**isCompositionOf**: The relation that links a nanofiller loading to a unique sample. In other words, only the (C, S) pair is allowed for this relation.

**isMemberOf**: The relation that indicates a parent-child relation between (S, G) pairs or (C, G) pairs. The (S, G) pairs indicate the unique sample labeled as S, is a member of the group of samples in G. The (C, G) pairs is used for cases that a top-level group reference G is given in the caption, and then multiple subgroups, usually with shared nanoloadings, is discussed separately. In such case, link (P, C) pairs with **isPropertyOf**, and link (C, G) pairs with **isMemberOf**.

**B.2 Baselines**

**B.2.1 NER**

**DeBERTa-base**: The DeBERTa base model was downloaded from the Hugging Face “microsoft/deberta-base” repository. The max length of the input and output sequence was set to 200. A batch size of 8, a learning rate of 2e-5, a weight decay of 1e-2, and an AdamW optimizer wrapped in a linear scheduler were adopted. The model was fine-tuned for 100 epochs with the maximum F1 score evaluated at the 51st epoch.
**MatBERT:** The MatBERT model was downloaded from the author’s figshare repository\(^1\). We adopted default values for training arguments\(^2\), including a batch size of 10, an embedding learning rate of 1e-4, a transformer learning rate of 2e-3, a classifier learning rate of 1e-2, a weight decay of 0, a LAMB optimizer with an exponential scheduler that exponentially decreases learning rate to 10% of its initial value at the final epoch, except that the model was fine-tuned for 20 epochs. A default value of 512 was selected as the max length.

**MatSciBERT:** The MatSciBERT was downloaded from the Hugging Face “m3rg-iitd/matscibert” repository. The max length of the input and output sequence was set to 200. A batch size of 16, a learning rate of 1e-5, a weight decay of 1e-2, and an AdamW optimizer wrapped in a linear scheduler were adopted. The model was fine-tuned for 50 epochs with the maximum F1 score evaluated at the 42\(^{nd}\) epoch.

**TANL:** The code of all TANL models is available on our fork\(^3\) of the TANL GitHub repository by amazon-science\(^4\). For the named entity recognition task, a max input sequence length of 256, a max output sequence length of 512, a 5-beam search, and a batch size of 8 with a gradient accumulation step of 2 resulting in a total batch size of 16 were adopted. Other configurations were kept at default values set by TANL with a

---

\(^1\) https://figshare.com/articles/software/MatBERT-NER_models/15087276  
\(^2\) https://github.com/CederGroupHub/MatBERT_NER/blob/main/matbert_ner/train.py  
\(^3\) https://github.com/bingyinh/tanl  
\(^4\) https://github.com/amazon-science/tanl
learning rate of 5e-4, a weight decay of 1e-2, and an AdamW optimizer wrapped in a linear scheduler, which is the default setting of Hugging Face transformers. The model was fine-tuned for 60 epochs.

**T5-base**

The T5-base model was downloaded from the Hugging Face “microsoft/t5-base” repository. The max length of the input and output sequence was set to 200. A batch size of 16 was used. The baseline model was trained with a learning rate of 1e-3 and a weight decay of 1e-2 for 400 epochs, followed by a learning rate of 3e-4 and a weight decay of 1e-1 for 100 epochs. The best F1 score was obtained at the 47th epoch. A linear scheduler with AdamW optimizer and no warmups was used.

**T5-base-interleave**

The T5-base model was downloaded from the Hugging Face “microsoft/t5-base” repository. The max length of the input and output sequence was set to 200. A batch size of 16 was used. The baseline model was trained with a learning rate of 1e-3 and a weight decay of 1e-2 for 200 epochs and reached its best F1 score at the 100th epoch. A linear scheduler with AdamW optimizer and no warmups was used.

**B.2.2 RE**

**MTB**: The code of all MTB models is available on our fork of the MTB PyTorch implementation by plkmo. A batch size of 32 with a gradient accumulation step of 2 resulting in a total batch size of 64 was used. A learning rate of 7e-5 was used in an

---

5 https://github.com/bingyinh/BERT-Relation-Extraction
6 https://github.com/plkmo/BERT-Relation-Extraction
Adam optimizer wrapped in a multistep learning rate scheduler with a gamma of 0.8. The BERT-base uncased model, courtesy of Hugging Face, was used for the MTB + BERT\textsuperscript{EM}. The MatBERT model\textsuperscript{7} was used for the MTB + MatBERT\textsuperscript{EM}. The MatSciBERT model, courtesy of Hugging Face, was used for the MTB + MatSciBERT\textsuperscript{EM}.

**TANL:** Our setting of TANL for the RE task is similar to the setting we reported in section B.2.1. For the relation extraction task, a max input and output sequence length of 200, a 5-beam search, and a batch size of 16 were adopted. Other configurations were kept at default values set by TANL with a learning rate of 5e-4 and an AdamW optimizer wrapped in a linear scheduler. The model was fine-tuned for 30 epochs.

\textsuperscript{7} Available at https://figshare.com/articles/software/MatBERT-NER_models/15087276
References


133


[120] “polystyrene polymer (CHEBI:61642).” https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI:61642


Biography

Bingyin Hu received his B.Eng. degree in Environmental Engineering (2014) at Tsinghua University, Beijing, China. He received his M.S. degree in Civil and Environmental Engineering (2015) at Carnegie Mellon University, PA, USA. He is graduating with his Ph.D. in Mechanical Engineering and Materials Science and M.S. in Electrical and Computer Engineering at Duke University, NC, USA. His research interests include materials informatics, natural language processing, and data curation.