THE UNIVERSITY OF CHICAGO

ENABLING SCIENTIFIC INFORMATION EXTRACTION WITH NATURAL LANGUAGE PROCESSING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
ZHI HONG

CHICAGO, ILLINOIS
MARCH 2024

TABLE OF CONTENTS

LIST OF FIGURES iv

LIST OF TABLES vi

ACKNOWLEDGMENTS ix

ABSTRACT x

1 INTRODUCTION 1
  1.1 Research Question 1
  1.2 Thesis Statement 1
  1.3 Introduction and Motivation 2
  1.4 Formalization 4

2 BACKGROUND AND RELATED WORK 5
  2.1 Overview of the Scientific Information Extraction Workflow 5
  2.2 Challenges of Scientific Information Extraction 9
    2.2.1 The Computer-(un)friendly Formats of Scientific Texts 9
    2.2.2 The Need for (and Lack of) Training Data 10
    2.2.3 The Sparsity of Information of Interest in Literature 11
    2.2.4 The Difficulties of Applying Models Trained on General Corpora to Scientific Text 13
  2.3 Related Work 15
    2.3.1 Named Entity Recognition 15
    2.3.2 Generic Information Extraction in NLP 17
    2.3.3 Scientific Masked Language Models 18
    2.3.4 Causal Language Models 20

3 RAPID TRAINING DATA COLLECTION 22
  3.1 Model-in-the-loop Training Data Collection 22
  3.2 Data-Performance Tradeoffs of the Model-in-the-loop 27
    3.2.1 Model Size 27
    3.2.2 Amount of Training Data 28
    3.2.3 Training Epochs 30
  3.3 Data Collection Results 31
    3.3.1 Performance of Human Annotators 31
    3.3.2 Effects of Training Data Quality on Model Performance 31
  3.4 Analysis of Human and Model Errors 32
  3.5 Data Availability and Formats 36

4 SCIENTIFIC INFORMATION EXTRACTION WITH DEEP NEURAL NETWORKS 39
  4.1 Named Entity Recognition (NER) 39
    4.1.1 Problem Definition 39
    4.1.2 A Lexicon-infused Bidirectional LSTM with CRF Model 40
    4.1.3 Model Features 43
    4.1.4 Experimental Analysis 44
    4.1.5 Experimental Results 45
  4.2 Relation Extraction with Syntax Tree Parsing 46

5 LARGE LANGUAGE MODELS FOR SCIENCE 50
  5.1 Data 50
    5.1.1 The Public Resource Dataset 50
    5.1.2 The PILE dataset 51
    5.1.3 Deduplication of the Training Corpora 51
    5.1.4 ScholarBERT Pretraining 56
  5.2 Vector space analysis of models prior to fine tuning 60
  5.3 ScholarBERT Performance on Downstream Scientific Tasks 66
    5.3.1 Sensitivity Analysis 66
    5.3.2 Finetuning 67
    5.3.3 Discussion 73
  5.4 Combining ScholarBERT and Graph Models for Semi-structured Extraction 75
    5.4.1 Problem Definition 77
    5.4.2 Methodology 77
    5.4.3 Experimental Evaluation 80

6 REAL-WORLD SCIENTIFIC INFORMATION EXTRACTION APPLICATIONS 86
  6.1 Drug Molecule Screening 86
    6.1.1 Automated Drug Extraction 86
    6.1.2 Validating the Utility of Identified Molecules 89
  6.2 Mining Material Properties from Literature 89
  6.3 Context-based Molecule Discovery 91
    6.3.1 Qualitative Method 93
    6.3.2 Quantitative Method 96

7 CONCLUSION 108

REFERENCES 109

LIST OF FIGURES

2.1 The three steps in a Scientific Information Extraction (SciIE) pipeline: Pre-processing (blue), data curation (orange), and learning (green). Components marked with a red alarm symbol are particularly challenging. 6
2.2 Example tasks in SciIE (visualized with Prodi.gy [94]): (a) Classifying sentences based on whether they mention polymers, (b) recognizing named entities such as polymers, and (c) identifying relations between named entities (e.g., polymers, properties, property values). 7
2.3 PDF (left) files are designed to capture page layout, while HTML (right) files are designed to save logical structure. It is thus much easier to identify which parts of a file are the desired body texts with the help of HTML tags. 10
2.4 Comparison of data collection and annotation solutions: Harvesting existing data takes the least effort if such data is available; crowdsourcing is the most straightforward, but requires careful task decomposition; computer-aided methods reduce manual workload but quality may vary; distant-supervised and unsupervised methods do not require manual labeling but may produce ambiguous or incoherent labels. 12
2.5 Comparison of text filtering techniques. Heuristic methods are simple to implement, and often achieve higher precision but lower recall. Statistical models are more complex, and commonly have higher recall but lower precision. 14
2.6 Challenges with applying generic NLP methods to SciIE: Datasets compiled from non-scientific sources cannot be applied to train SciIE models; word embeddings pre-trained on generic texts need to be enriched with embeddings trained on scientific texts to include terminologies; pre-trained language models requires fine-tuning on scientific texts to achieve better understanding of domain-specific language. 16

3.1 Overview of the training data collection workflow, showing the three phases described in the text and with the parameter values used in the study reported here. Each phase pulls paragraphs from the CORD-19 dataset (blue dashed line) according to the **Select** criterion listed in its shaded box. Phases B and C repeatedly update the weights for the NER model (green arrows) that they use to identify and label uncertain paragraphs; human review (yellow and gold arrows) corrects those silver paragraphs to yield gold paragraphs. Total human review work is ∼278+600+960=1838 paragraphs. 23
3.2 Precision, recall, and F-1 scores of medium and large SpaCy models trained on 278 examples. 28
3.3 Training curves for the SpaCy and Keras models for different number of examples collected. 29
3.4 Loss, precision, recall, and F-1 of SpaCy model during training for 100 epochs on 278 paragraphs. 30

4.1 The model with external lexicon knowledge. 42

4.2 Comparison of Positional Embedding and Dependency Path: In the example, the relative position of the words (Polystyrene, temperature) is 5, while the lengths of shortest dependency path between the two words is 2. 47
4.3 Model Architecture 48
4.4 Precision, recall, and F1 score of the the proposed model and the traditional RE model on each type of relation in the dataset. 49

5.1 Pretraining loss plots for the ScholarBERT models listed in Table 5.5. The vertical dashed lines indicate the approximate locations of the iteration checkpoints selected for evaluation. 56
5.2 Difference vector = avg(domainA entities) – avg(chemistry entities) 64
5.3 As Fig. 5.2, except that the similarity for each entity e is with the difference vector $\overrightarrow{diff_{X,\ Chemistry,\ e}}$ as computed with Eq. 5.2, i.e., one in which e is not considered. 65
5.4 NER F1 scores with standard deviations represented by error bars. 72
5.5 Sentence classification F1 scores from Table 5.8, with standard deviations represented by error bars. 74
5.6 The GraphScholarBERT model architecture 78
5.7 Graph representation of a webpage. DOM node attributes propagate bi-directionally through the black edges but not the dashed orange edges (“virtual edges”), which are only used for classification. 79

6.1 Percentage of detected entities that are also found in DrugBank, when running either on all words found by our model or on just the balanced subset, and with “found” defined as either a full or partial match. 87
6.2 The Web interface for the Solid State Material Database, which allows users to search by material name, descriptor type, and/or descriptor value. In this example, the search term is “thermoelectric”. Relevant materials and references are returned. 92
6.3 t-SNE plot with 10% of contexts in the `Known` set 95
6.4 Each column shows a `Known` molecule on the bottom and its top candidate molecule from the `Random` set on the top 95
6.5 Each column shows a `Known` molecule on the bottom and its top candidate molecule from the `Relevant` set on the top 96

LIST OF TABLES

3.1 K-fold (K=5) validation of the SpaCy model on 96 paragraphs with original vs. corrected labels. The first five rows are the results of each fold; the last row is the average F-1 score of the five folds. 32
3.2 K-Fold (K=5) validation of the Keras-LSTM model on 96 paragraphs with original vs. corrected labels. The first five rows are the results of each fold; the last row is the average F-1 score of the five folds. 33
3.3 The 10 most frequent tokens, excluding stopwords and punctuation marks, within various window sizes around entities incorrectly labeled by human reviewers. . . 34
3.4 The 10 most frequent tokens, excluding stop words and punctuation marks, within various window sizes around entities correctly labeled by human reviewers. . . . 35
3.5 The 20 most frequent tokens, including stop words and punctuation marks, within various window sizes around entities incorrectly labeled by human reviewers. Words that are neither stop words nor punctuation words are in boldface. 35
3.6 The 20 most frequent tokens, including stop words and punctuation marks, within various window sizes around entities incorrectly labeled by the Keras model. Words that are neither stop words nor punctuation words are in boldface. 36

4.1 An example of encoding external knowledge from DBpedia 42
4.2 SciNER on polymer dataset 45
4.3 SciNER on drug molecule dataset 45
4.4 SciNER on sociology dataset 45
4.5 The number of sentences in each type of relations (`COMPARE, CONJUNCTION, EVALUATE-FOR, FEATURE-OF, PART-OF, USED-FOR`) in the training, development, and test set 49

5.1 A minhash characteristic matrix for 3 documents T_1, T_2, T_3. a – d are the tokens in the documents. The rows are randomly permutated. A ‘1’ in the matrix means the token exists in that document, whereas a ‘0’ indicates the opposite. 53
5.2 Pretraining corpora used by models in this study. The domains are Bio=biomedicine, CS=computer science, Gen=general, Materials=materials science and engineering and Sci=broad scientific. 57
5.3 Pretraining hyperparameters. All ScholarBERT variants use the same pretraining hyperparameters. 59
5.4 NER F1 scores for each model. Models are finetuned five times for each dataset and the average result is presented. Underlined results represent the F1-scores of models trained on in-distribution data for the given task, and bolded results indicate the best performing model on that task. `SB` = ScholarBERT. Task domain: Biomedical = {BC5CDR, JNLPBA, NCBI-Disease, ChemDNER}, CS = {SciERC}, Materials = {MatSciNER}, Sociology = {Coleridge}, Multi-Domain = {ScienceExam}. 67

5.5 Characteristics of the 14 BERT models considered in this study. The BERT-Base architecture has 12 layers, hidden size of 768, and 12 heads; BERT-Large has 24 layers, hidden size of 1024, and 16 heads [25]; the BERT-XL architecture has 36 layers, hidden size of 1280, and 20 heads. Details of the pretraining corpora are in Table 5.2. The domains are Bio=biomedicine, CS=computer science, Gen=general, Mat=materials science and engineering, and Sci=broad scientific. 68
5.6 F1 scores for each model on Relation Extraction (SciERC, ChemProt) and Sentence Classification (PaperField, Battery) tasks. Models are finetuned five times for each dataset and the average result is presented. Underlined results represent the F1-scores of models trained on in-distribution data for the given task, and bolded results indicate the best performing model on that task. `SB` = SCHOLARBERT. Task domain: Biomedical = {ChemProt}, CS = {SciERC}, Materials = {Battery}, Multi-Domain = {Paperfield}. 71
5.7 NER F1 scores for each of 14 models (rows), when the model is finetuned on eight different domain datasets and the resulting finetuned model applied to that dataset's associated NER task (columns). In each case, we give the average value and its standard deviation over five runs. 73
5.8 Sentence classification F1 scores for each of 14 models (rows), when the model is finetuned on one of four different domain datasets and the finetuned model is applied to that dataset's associated sentence classification task (columns). In each case, we give the average value and its standard deviation over five runs. . 74
5.9 Statistics of the SWDE, Retail, and Expanded SWDE datasets used in the experiments. 81
5.10 Examples of search keywords and descriptions used as inputs. The "Ground Truth Relations" column lists the expressions used by different websites to represent the same relation. The search keywords and descriptions are manually provided. The search keywords are usually synonyms or hypernyms of the ground truth relations. 81
5.11 Intra- and inter-vertical OpenIE extraction performances on the SWDE dataset. 82
5.12 Intra- and inter vertical OpenIE extraction performances on the expanded SWDE dataset. 82
5.13 Intra- and inter-vertical OpenIE extraction performances on the PPPDB dataset. 82
5.14 The top (green) and bottom (orange) 3 relations in SWDE and Expanded SWDE datasets based on inter-vertical extraction performance of the GraphScholarBERT model. 84
5.15 Ablation results on the SWDE dataset. 84

6.1 Extracted materials and descriptors 91
6.2 Metadata on the three molecule datasets 94
6.3 Spearman's correlation between the ground truth rankings and the ScholarBERT-predicted rankings with "hydrogen" as the center word. Detailed rankings are provided in Table 6.5 and 6.6. 98

6.4 Spearman's Correlation between the LLM-predicted rankings and the ground truth rankings by capacity and cost. The numbers in **bold** are the strongest correlation with the correct sign. The scholarBERT-predicted rankings on both the Relevant and the Random set have the strongest correlation with the ground truth rankings by cost. The ground truth rankings are listed in Table 6.5 and 6.6, while the LLM-predicted rankings are listed in Table 6.7 and 6.8. 101
6.5 Rankings for 32 molecules in the Relevant set. The embedding rank is based on the cosine similarity between the ScholarBERT embedding of the molecule name and the embedding of the word "hydrogen". 104
6.6 Rankings for 32 molecules in the Random set. The embedding rank is based on the cosine similarity between the ScholarBERT embedding of the molecule name and the embedding of the word "hydrogen". 105
6.7 Rankings for 32 molecules in the Relevant set by LLMs. The rank of a molecule is based on the maximum cosine similarity between its embedding by the LLM and the embeddings of the Known molecules by the same LLM. 106
6.8 Rankings for 32 molecules in the Random set by LLMs. The rank of a molecule is based on the maximum cosine similarity between its embedding by the LLM and the embeddings of the Known molecules by the same LLM. 107

ACKNOWLEDGMENTS

It is with immense gratitude that I acknowledge the support and help of my advisor Dr. Ian Foster and co-advisor Dr. Kyle Chard for providing their support and guidance at all times. They have given me all the freedom to pursue my research, while giving inspirations and suggestions to ensure I stay on course. Without their guidance this thesis would not have been possible.

I would like to acknowledge the help I received from my groupmates, Dr. Roselyne Tchoua, J. Gregory Pauloski, and Aswathy Ajith. With their expertise on this topic, they have given me many constructive suggestions and helped me avoid many roadblocks.

My sincere thanks also goes to Dr. Logan Ward, Marcus Schwarting, and Dr. Debra Audus for contributing their time and domain expertise to compiling and reviewing ground truth datasets. Without them the development and evaluation of the proposed models would not be complete.

Last but not least, I would like to thank my fellow labmates in Globus Labs for the stimulating discussions, the helpful suggestions, and of course, their friendship.

ABSTRACT

Scientific discoveries have traditionally been communicated through written papers, but these documents pose challenges for computer understanding due to the inherent ambiguity and variability of natural languages. Consequently, valuable knowledge and groundbreaking insights often remain buried within an overwhelming volume of publications, rendering them undiscoverable, inaccessible, and unusable for both humans and machines. While efforts have been made to construct scientific databases and repositories from these papers, these initiatives typically rely on laborious and error-prone manual extraction processes, which are not scalable to keep up with the millions of papers published annually. The inability to efficiently extract experimental data from existing literature poses a significant obstacle that hinders the adoption of cost-effective, safe, and data-driven simulations to inform traditional experiments across multiple disciplines.

To address this challenge and enable disciplines to leverage data-based simulations for cheaper, safer, and easier insights and guidance, Natural Language Processing (NLP) methods have emerged as a promising solution. Leveraging advanced machine learning techniques, NLP empowers computers to analyze, comprehend, and derive meaningful information from the ever-expanding scientific literature.

This work explores various methodologies for automatically extracting precise and structured information from unstructured scientific text. Major contributions include a model-in-the-loop pipeline for rapid training data annotation without relying on domain experts, special features designed to adapt classical named entity recognition and relation extraction models to scientific texts, a foundational large language model for science with up to 770M parameters, a joint graph-language model for processing semi-structured data, as well as several real-world applications. By investigating these techniques, we seek to demonstrate their potential in facilitating real-world scientific research and enabling researchers to efficiently leverage the vast knowledge accumulated in literature for accelerated scientific breakthroughs.

CHAPTER 1

INTRODUCTION

1.1 Research Question

This thesis aims to answer the following research question:

Can Natural Language Processing models make scientific facts in publications more discoverable, accessible, and usable?

1.2 Thesis Statement

The scientific process has long relied on publication as a means of documenting and disseminating results. The data accumulated in scientific publications may be of great value to future research as references and building blocks. Discovering and extracting the data from past publications is nevertheless a tedious task. Traditional keyword-based article searches (such as Google Scholar) are limited in granularity. A piece of data that a scientist needs may be buried under thousands of other papers on the same topic. The scientists would still need to sift through large volumes of papers to locate just the data they need. However, the variability and ambiguity inherent in natural language and the unstructured nature of such data make them undiscoverable, inaccessible, and unusable.

In this thesis, I investigate Natural Language Processing (NLP) techniques to understand if they can automatically extract scientific facts from scholarly publications. I develop models targeting individual tasks in the IE pipeline as well as explore the use of more generalizable large language models (LLMs). I apply the developed models on downstream scientific applications and evaluate the use of the extracted data on researchers' workflows.

1.3 Introduction and Motivation

The continuous growth of scientific knowledge and the rapid advancements in various fields has led to an unprecedented increase in the volume of scientific publications—estimated to be over 5M per year [23]. These publications serve as the primary medium for disseminating new research findings and theories, which enables researchers to build upon existing knowledge and advance their respective fields. However, the sheer volume of published work has led to an information overload [32], making it increasingly difficult for researchers to stay up-to-date with the latest developments and to apply computational methods that require large amounts of data (e.g., deep neural networks) on their research. This has motivated the need for new approaches to manage and extract value from the ever-growing body of scientific literature.

In response to this challenge, Natural Language Processing (NLP) methods have emerged as a promising solution. NLP is a subfield of artificial intelligence that deals with the development of algorithms and methods to analyze, understand, and generate human language. Through the use of advanced machine learning techniques, NLP enables computers to process and extract meaningful information from large volumes of text. In the context of scientific publications, NLP techniques can be applied to a wide range of tasks, such as information extraction [87, 130, 83], text summarization [3, 16], and classification [49]. These applications have the potential to greatly enhance the efficiency and effectiveness of knowledge discovery in the scientific community, allowing researchers to focus on their core research while benefiting from the distilled insights provided by NLP tools.

This thesis aims to investigate NLP techniques to help the extraction of data from scientific publications, with the goal of making historical research data discoverable, accessible, and usable for future researchers.

Research questions:

1. How can we automatically extract accurate structured information from unstructured

text in scientific publications?

   1.1. How to identify which scientific entities (e.g., materials, molecules) are discussed in a paper?
   1.2. How to extract the attributes relevant to an entity in a paper?
   1.3. How can LLMs be used to improve scientific information extraction?

2. Can a scientific LLM help in real-world scientific research?
   2.1. How to apply LLMs on scientific Information Extraction?
   2.2. How can scientists make use of the extracted data?

The major contributions of this dissertation include:

1. Data
   1.1. Data collection: a rapid model-in-the loop data annotation workflow
   1.2. Data cleaning: an efficient text deduplication pipeline
2. Models
   2.1. Classical Deep Learning models that target each individual component of the scientific information extraction workflow and work in tandem to extract scientific entities and values from literature
   2.2. Transformer-based LLMs that are specifically designed to process scientific contexts. The tradeoffs of various parameters and the model prediction performance are also investigated.
3. Applications
   3.1. Drug-like molecule discovery from coronavirus research

3.2. Material Descriptor Database (MDDB), which contains approximately 3M descriptors for 50k inorganic materials

3.3. Hydrogen-capable molecule discovery based on LLM embeddings

1.4 Formalization

Given a scientific article $\mathcal{A}$, which comprises texts $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$, tables $\mathcal{T} = \{t_1, t_2, \ldots, t_p\}$, and figures $\mathcal{F} = \{f_1, f_2, \ldots, f_q\}$, extract all scientific entities $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ and their properties $\mathcal{P} = \{p_{11}, p_{12}, \ldots, p_{1k}, p_{21}, \ldots, p_{nk}\}$ from $\mathcal{S}$.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Overview of the Scientific Information Extraction Workflow

Three common steps are involved in a Scientific Information Extraction (SciIE) pipeline (Fig. 2.1): data preprocessing, curation and annotation, and learning.

The first step, preprocessing, is to break down scientific articles into chunks of clean text for later steps. In addition to text in the body of an article, scientific documents may also have figures, tables, and publisher embellishments (e.g., logos, running titles, page numbers). The first step in preprocessing is thus to *parse the document* and extract the body text. This is particularly complicated in science due to the complex formats of scientific articles. While tables and figures may also contain valuable facts, extracting information from them relies on a completely different set of technologies, such as computer vision, which are beyond the scope of this thesis. Interested readers may refer to [18, 69, 37, 56] for relevant research. After document parsing, texts are then split into smaller units. This step is called *tokenization*. Sentence tokenization splits passages into sentences, which is the expected input format for many later steps. Word tokenization further splits sentences into tokens to reduce the entropy of the vocabulary. Once text are extracted and cleaned, the next step is to create the tools that allow for so-called intelligent processing of the language. In principle, it should be possible to manually define rules to perform each of the tasks just listed. For example, units can be identified by matching words that optionally begin with k, m, or M and end with a character from a known list (e.g., m, s, Ω). The complexity of human language, however, is too high to allow for the enumeration of a complete set of rules except in trivial cases (e.g., units). Rather, the modern approach is to use ML techniques to *learn* such rules automatically from many examples.

The second major step in performing IE is *curating and annotating* enough data to train ML models for each SciIE subtask. Tokenized texts are first selected and annotated to form

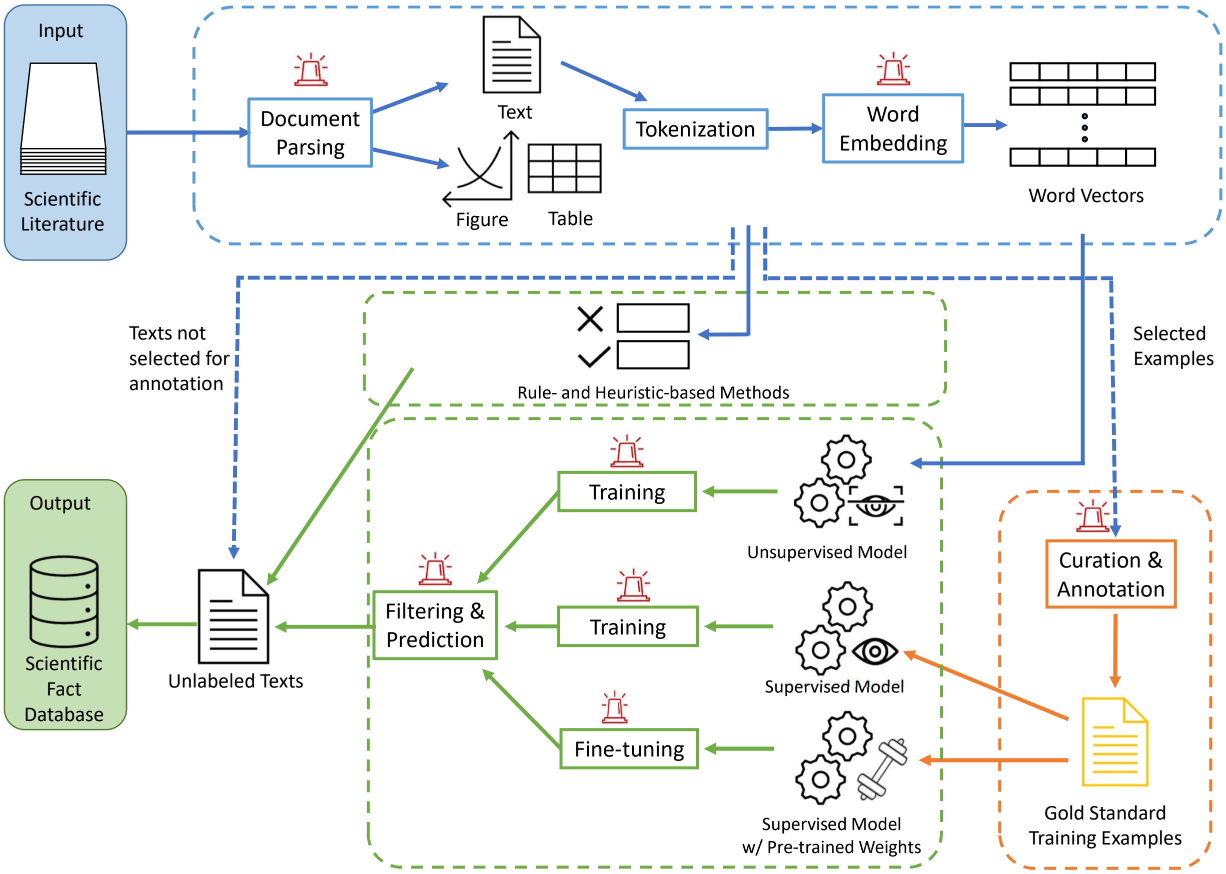
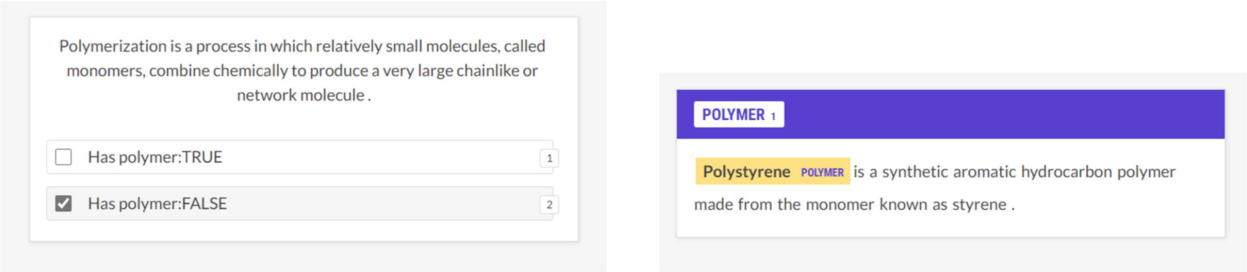


Figure 2.1: The three steps in a Scientific Information Extraction (SciIE) pipeline: Preprocessing (blue), data curation (orange), and learning (green). Components marked with a red alarm symbol are particularly challenging.



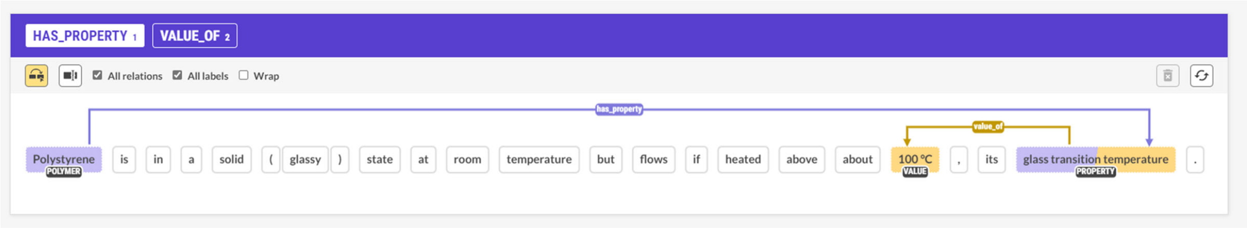
Polymerization is a process in which relatively small molecules, called monomers, combine chemically to produce a very large chainlike or network molecule .

☐ Has polymer:TRUE
☑ Has polymer:FALSE

Polystyrene is a synthetic aromatic hydrocarbon polymer made from the monomer known as styrene .

(a) Text Classification

(b) Named Entity Recognition



☑ All relations ☑ All labels ☐ Wrap

Polystyrene is in a solid (glassy) state at room temperature but flows if heated above about 100 °C , its glass transition temperature .

(c) Relation Extraction

Figure 2.2: Example tasks in SciIE (visualized with Prodi.gy [94]): (a) Classifying sentences based on whether they mention polymers, (b) recognizing named entities such as polymers, and (c) identifying relations between named entities (e.g., polymers, properties, property values).

a so-called *gold standard* training set for supervised models. Labels could be per example (sentence) or per token, depending on the task. A sentence could be given the label "True" or "False" if we simply want to classify whether it mentions any polymer. If we want to find out which polymers are mentioned in a sentence, then each word will need a separate label indicating whether it is a polymer or not. Annotation is where human intelligence is injected and it sets the performance ceiling for the models trained on the annotated data. The annotation process is usually costly, and time-consuming, and is especially difficult for scientific data due to the expertise required, the limited bandwidth of the people who do have the expertise, and the rarity of desired data across the scientific literature.

The final step in the pipeline is *model learning*. Traditional rule- and heuristic-based methods do not require learning. However, statistical models are the most widely used nowadays because of performance and adaptability advantages. Machine learning models for NLP are classified into two categories: supervised and unsupervised models. Most models used in IE are supervised, meaning labels are required to accompany the training data.

State-of-the-art techniques use neural networks that automatically account for the context of a word (e.g., recurrent, convolutional networks) and are flexible enough to express the exquisitely complicated model forms required for expressing language [82, 68].

The entire process of extracting information from text is often accomplished by a pipeline of complementary tools rather than a single program that produces data directly from tokenized text. The steps in these pipelines often include (Fig. 2.2):

1. *Text Classification* assigns a label or score to an entire block of text. For example, they could determine whether a block of text is an abstract or whether it contains the desired data.

2. *Named Entity Recognition* (NER) classifies whether a word or phrase belongs to a specific category. Categories could be broad (e.g., noun) or specific (e.g., place name, polymer).

3. *Relationship Extraction* produces pairs of named entities that are connected by a certain relationship. A common example of IE in materials science and engineering is to associate words that are materials with those that are property names. Complex relationships can then be built from multiple pair relationships, such as the material "iron" *has property* "density" *with value* "7.9 g/cc."

In the following sections, we will carefully examine the major challenges faced by SciIE, including the scarcity of labeled data, the sparsity of interested information in publications, and the difficulties of applying ML or DL models trained on general corpora to scientific texts.

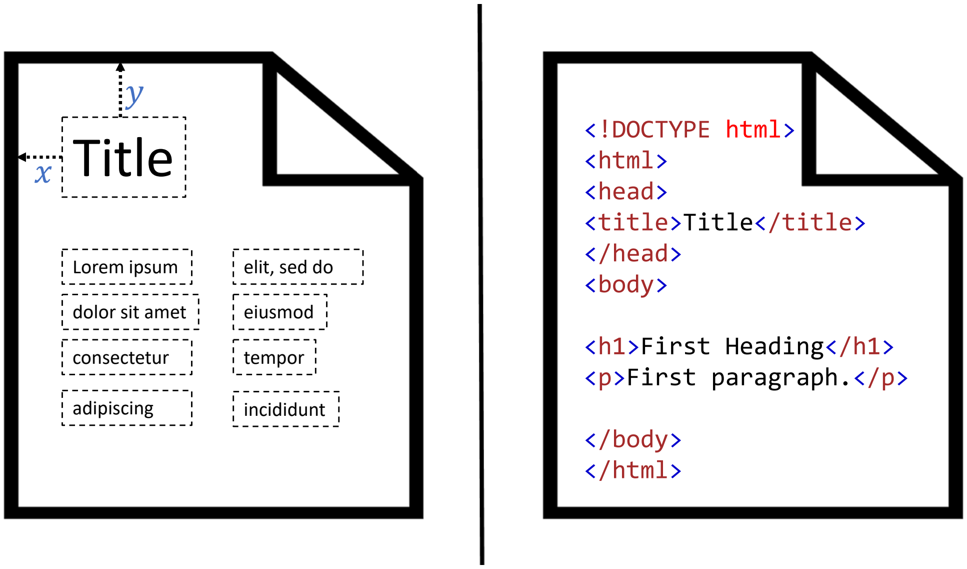
2.2 Challenges of Scientific Information Extraction

2.2.1 The Computer-(un)friendly Formats of Scientific Texts

The data needed to train or use NLP models are texts, ideally clean, well-formed texts that can be easily ingested by computers (and humans). Many generic NLP datasets exist, such as the CoNLL-2003 dataset for training Named Entity Recognition (NER) models [123] and the TACRED dataset for relation extraction models [139]. One common feature of these NLP datasets is that they are distributed as plain text and have well-defined homogeneous internal formats. For example, in the CoNLL dataset, each word is placed on a separate line with an empty line at the end of each sentence, and on each line, the word is followed by three tags: a part-of-speech tag, a syntactic tag, and a named entity tag. Such structured datasets can be fed into a model with only a few lines of code and without any extra pre-processing. Feeding scientific literature to models, unfortunately, is quite a different story.

Useful documents in material engineering are stored in a few different kinds of documents each with different processing challenges. Specification sheets and handbooks often express data in tabular formats instead of natural language and are best treated with special-purpose software tailored to their specific formats given the predictable form in which data are expressed. Information from journal articles, conference proceedings and technical report are held in text in a variety of document formats. Older articles are often only available as scanned images whereas more modern articles are expressed in a variety of digital formats. This multitude of formats for engineering text presents a major barrier to extracting knowledge.

Portable Document Format files (PDFs), commonly used for sharing papers, maintain a fixed layout, but lack structural information, making it difficult to programmatically discern different elements like body text and page numbers. Older PDFs often require Optical Character Recognition (OCR) due to being scanned images. The extraction of text from scientific papers in PDF format is complicated by their complex layouts, such as double-



```
<!DOCTYPE html>
<html>
<head>
<title>Title</title>
</head>
<body>

<h1>First Heading</h1>
<p>First paragraph.</p>

</body>
</html>
```

Figure 2.3: PDF (left) files are designed to capture page layout, while HTML (right) files are designed to save logical structure. It is thus much easier to identify which parts of a file are the desired body texts with the help of HTML tags.

column formats, and the presence of non-textual elements like publisher names. Tools like LA-PDFText [96] and GROBID [39] claim to be designed specifically for scientific literature in PDF formats. However, as they employ machine-learning-based approaches to classify and stitch together text blocks into coherent sections, they are not universally effective due to varying typesetting styles. On the other hand, HTML, increasingly popular in web-based technology, offers easier text extraction due to clear tagging of page elements, making it a preferred format for many studies. HTML's adaptability and straightforward tagging system simplify the identification of paper sections, thus eliminating the guesswork required with PDFs. This has led to its use in various research projects for extracting and analyzing data [80, 120, 121, 117].

2.2.2 The Need for (and Lack of) Training Data

Modern artificial intelligence (AI) methods derives their "intelligence" from the data on which they are trained. No model architecture, regardless of its sophistication, can do better than random guessing without training data. Moreover, having the data itself is often not sufficient. Many models are created for predictive purposes, i.e., to respond to a query: “Given $\mathbf{x}$, predict $\mathbf{y}$.” Having the data ($\mathbf{x}$) alone is of limited utility without the corresponding labels ($\mathbf{y}$). Such models are called *supervised* models and modern machine learning tools can require thousands of examples of (x, y) pairs.

The dearth of training data in materials engineering can be simply attributed to IE in materials being in its beginnings, though there are ways to rapidly accelerate developing the training sets. In this section, we first examine methods for curation of labeled training data, discussing the difficulties involved and presenting progresses towards solving the difficulties. Then, we discuss models that can train on data without labels (*unsupervised* models) and analyze their strengths and limitations. Fig. 2.4 shows a summary of the pros and cons of different solutions to the data collection and annotation problem.

2.2.3 The Sparsity of Information of Interest in Literature

SciIE must address a challenging range of scales. At one extreme, according to a bibliographical study on global publications from 2000 to 2018 by the National Science Foundation (NSF) [131], the scientific literature published in that time span encompasses some 35.5M articles, which ultimately we would like to analyze in their entirety. At the other extreme, the literature associated with individual disciplines and subdisciplines, each characterized by distinct vocabularies and conventions for communicating information, are much smaller. For example, the same study finds that the materials science literature encompasses 1.1M articles during that period: 3.1% of the total. A particular subdiscipline, such as polymer science, accounts for just a portion of those 1.1M articles, of which a yet smaller subset contain information relevant to any specific question.

Thus, even if we can identify the relevant publications accurately and efficiently, the sparsity of interesting information in text can be yet another obstacle to efficacious IE from scientific literature [97, 112]. One study on extracting glass transition temperature (Tg) shows that only 64 (0.67%) of 9518 sentences in 31 papers from *Macromolecules* contain

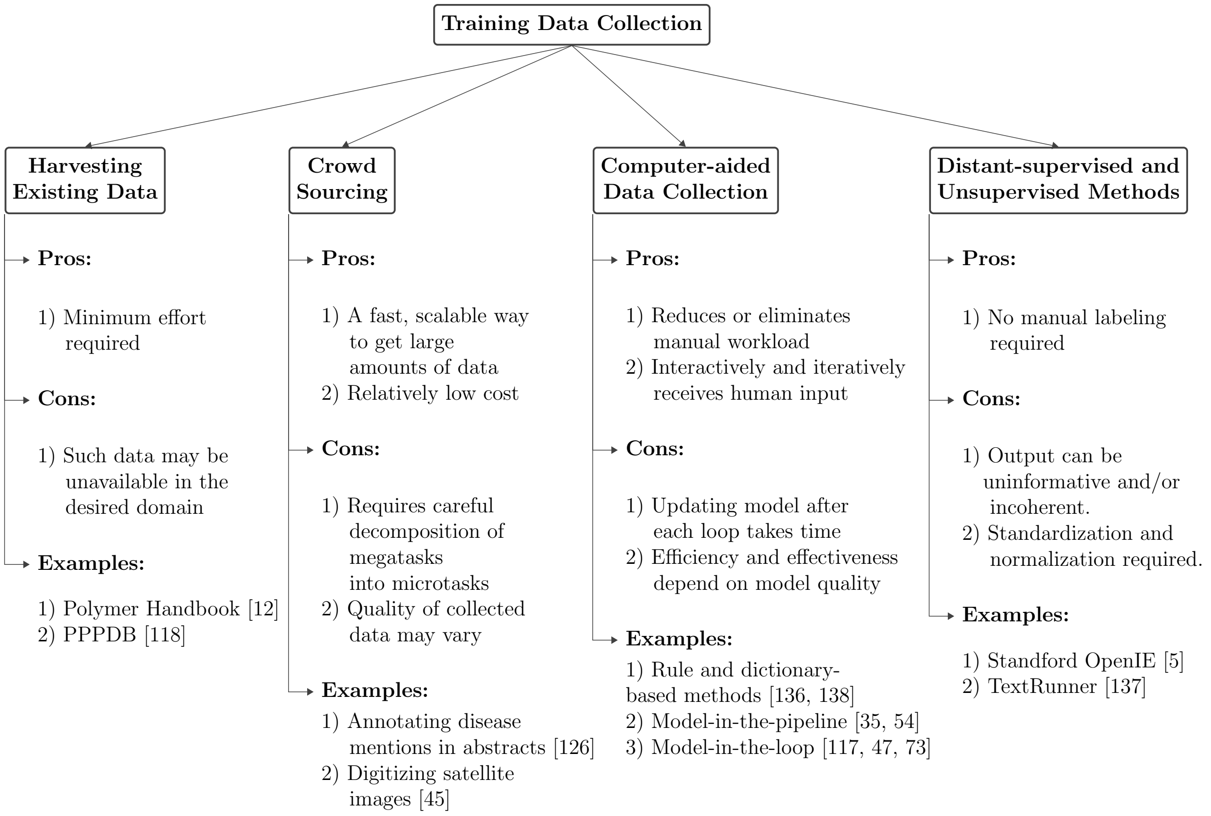

Training Data Collection
Harvesting Existing Data
Pros:
1) Minimum effort required
Cons:
1) Such data may be unavailable in the desired domain
Examples:
1) Polymer Handbook [12]
2) PPPDB [118]
Crowd Sourcing
Pros:
1) A fast, scalable way to get large amounts of data
2) Relatively low cost
Cons:
1) Requires careful decomposition of megatasks into microtasks
2) Quality of collected data may vary
Examples:
1) Annotating disease mentions in abstracts [126]
2) Digitizing satellite images [45]
Computer-aided Data Collection
Pros:
1) Reduces or eliminates manual workload
2) Interactively and iteratively receives human input
Cons:
1) Updating model after each loop takes time
2) Efficiency and effectiveness depend on model quality
Examples:
1) Rule and dictionary-based methods [136, 138]
2) Model-in-the-pipeline [35, 54]
3) Model-in-the-loop [117, 47, 73]
Distant-supervised and Unsupervised Methods
Pros:
1) No manual labeling required
Cons:
1) Output can be uninformative and/or incoherent.
2) Standardization and normalization required.
Examples:
1) Standford OpenIE [5]
2) TextRunner [137]

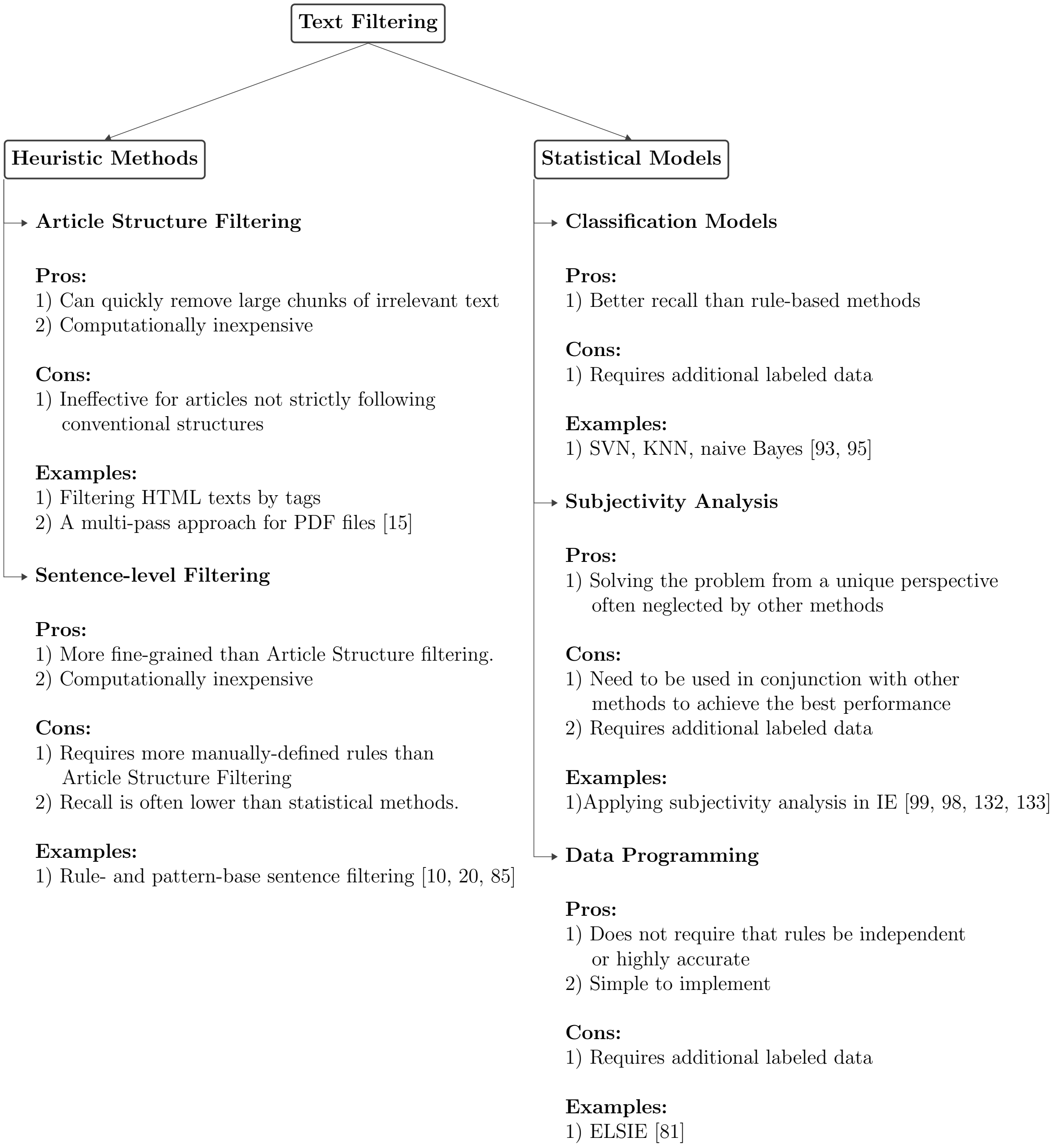
Figure 2.4: Comparison of data collection and annotation solutions: Harvesting existing data takes the least effort if such data is available; crowdsourcing is the most straightforward, but requires careful task decomposition; computer-aided methods reduce manual workload but quality may vary; distant-supervised and unsupervised methods do not require manual labeling but may produce ambiguous or incoherent labels.

both a polymer and its Tg value [81]. The remaining 99.33% of sentences are just noise for this task. Such imbalance is rare in standard NLP datasets that most state-of-the-art models are designed for and trained on, but will be common in extracting data from other materials engineering literature For comparison, the widely used SemEval 2010 Task 8 relation extraction dataset only has 17.63% and 16.71% of "Other" sentences (i.e., not belonging to known relations in the dataset) in the training and test set, respectively [44]. Extracting information of such sparsity will be difficult for ML/DL models. The high percentage of noise in texts will lead to more false positives and thus diluting the extraction results. To avoid this problem, it is advisable to apply a filtering step during preprocessing to remove as much noise as possible before feeding the texts to an IE model.

Fig. 2.5 provides a review of the techniques developed to filter out noisy texts from traditional intuitive heuristic-based methods to modern statistical model-based methods.

2.2.4 The Difficulties of Applying Models Trained on General Corpora to Scientific Text

It is standard practice in most applications of NLP to re-use ML models trained on plentifully available text (e.g., websites, newspapers), with only *retraining* performed to adapt the models to new domains. For general NLP, there are many datasets (e.g., Penn Treebank [75], CoNLL [107], OntoNotes [129]), pre-trained word vectors (e.g., GloVe [92], FastText [78]), and pre-trained models (e.g., BERT [26], Turing-NLG [101]) available online. However, their training corpora are usually taken from general sources such as newspapers [75], Twitter posts [105], online reviews [74], and Wikipedia pages [31] rather than scientific publications. In addition to common obstacles like long-distance dependencies [114] and polysemant disambiguation [124], the unique terminologies and language styles used in scientific publications pose unique challenges to the IE problem and make model re-use problematic. Elsevier has conducted an evaluation of OpenIE systems on two datasets: one consisting of 200 sentences randomly selected from Wikipedia, and the second made up of 220 sentences from the sci-



Text Filtering

Heuristic Methods

Article Structure Filtering

Pros:
1) Can quickly remove large chunks of irrelevant text
2) Computationally inexpensive

Cons:
1) Ineffective for articles not strictly following conventional structures

Examples:
1) Filtering HTML texts by tags
2) A multi-pass approach for PDF files [15]

Sentence-level Filtering

Pros:
1) More fine-grained than Article Structure filtering.
2) Computationally inexpensive

Cons:
1) Requires more manually-defined rules than Article Structure Filtering
2) Recall is often lower than statistical methods.

Examples:
1) Rule- and pattern-base sentence filtering [10, 20, 85]

Statistical Models

Classification Models

Pros:
1) Better recall than rule-based methods

Cons:
1) Requires additional labeled data

Examples:
1) SVN, KNN, naive Bayes [93, 95]

Subjectivity Analysis

Pros:
1) Solving the problem from a unique perspective often neglected by other methods

Cons:
1) Need to be used in conjunction with other methods to achieve the best performance
2) Requires additional labeled data

Examples:
1)Applying subjectivity analysis in IE [99, 98, 132, 133]

Data Programming

Pros:
1) Does not require that rules be independent or highly accurate
2) Simple to implement

Cons:
1) Requires additional labeled data

Examples:
1) ELSIE [81]

Figure 2.5: Comparison of text filtering techniques. Heuristic methods are simple to implement, and often achieve higher precision but lower recall. Statistical models are more complex, and commonly have higher recall but lower precision.

entific literature for the 10 most published disciplines. Extractions were checked manually by five humans to guarantee evaluation accuracy, and the results showed that the extractors perform better on encyclopedic sentences (54% precision) than on scientific sentences (34% precision) [40].

Many IE challenges observed in other fields of science will certainly also be challenges in materials engineering. There is plenty of terminology that is not specific to materials engineering and yet is critical to understanding the content of materials text. In previous work, we have also encountered many challenges in that the same materials can be referred to by different names (e.g., trade names, specification number, composition), which complicates both learning language models and organizing extracted data. Long-distance dependencies are also common in materials article, such as when the processing path for a material is described in a separate section from where its properties are discussed. Journal articles require significant training for humans to understand well, and it is similar for machines.

Fig. 2.6 lists the many gaps that prevent state-of-the-art NLP models from reaching their full potential on scientific literature, as well as techniques proposed to bridge the gaps.

2.3 Related Work

2.3.1 Named Entity Recognition

Named entity recognition has been a widely studied since the 1990s. It has attracted much attention because it is an essential step of extracting structured information from free text. Earlier works are mostly either rule-based or requires an extensive ontology, since those methods are less computationally intensive. With the emergence of hardware-accelerated computing, such as GPU and TPU, deep learning NER models have gained tremendous traction.

Early NER approaches often rely on crowdsourcing [134, 113] or rule-based systems [108, 62]. AQL, used in IBM's SystemT, allows users to define rules for query optimization. How-

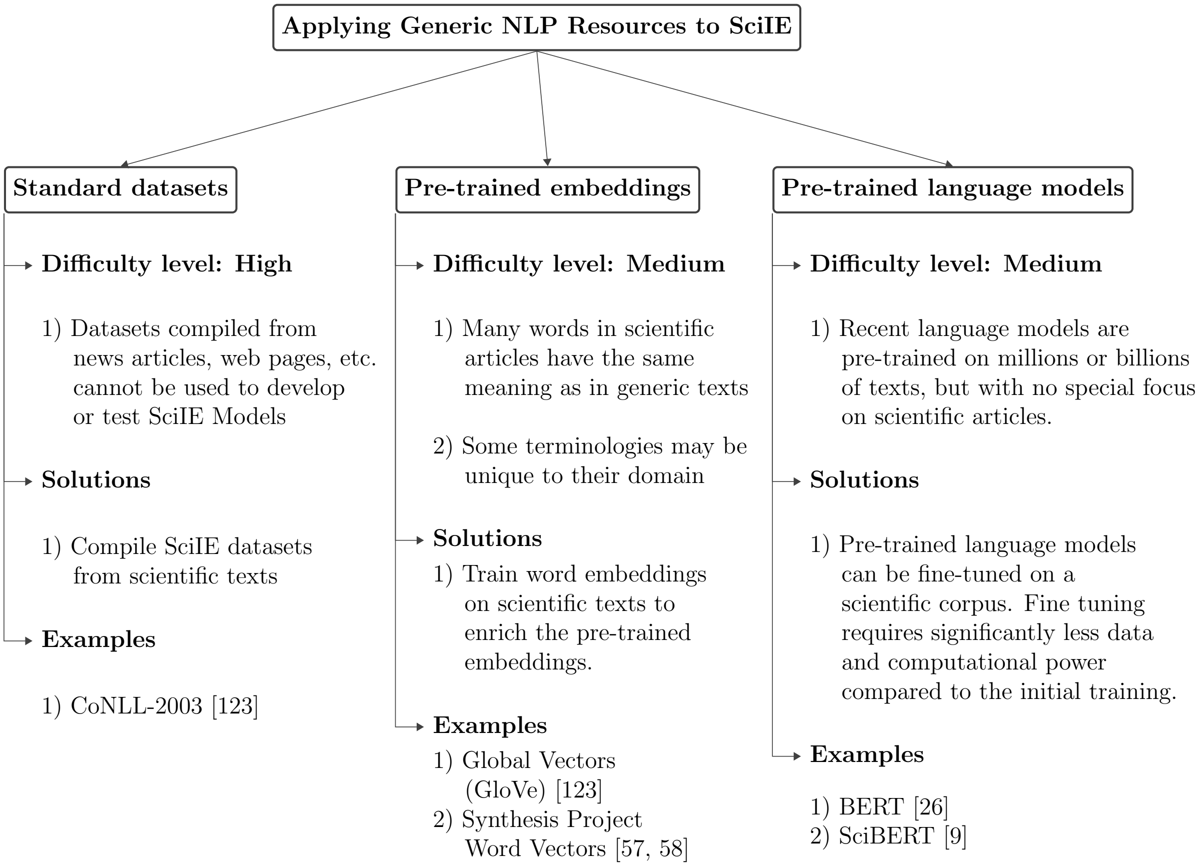

Applying Generic NLP Resources to SciIE
Standard datasets
Difficulty level: High
1) Datasets compiled from news articles, web pages, etc. cannot be used to develop or test SciIE Models
Solutions
1) Compile SciIE datasets from scientific texts
Examples
1) CoNLL-2003 [123]
Pre-trained embeddings
Difficulty level: Medium
1) Many words in scientific articles have the same meaning as in generic texts
2) Some terminologies may be unique to their domain
Solutions
1) Train word embeddings on scientific texts to enrich the pre-trained embeddings.
Examples
1) Global Vectors (GloVe) [123]
2) Synthesis Project Word Vectors [57, 58]
Pre-trained language models
Difficulty level: Medium
1) Recent language models are pre-trained on millions or billions of texts, but with no special focus on scientific articles.
Solutions
1) Pre-trained language models can be fine-tuned on a scientific corpus. Fine tuning requires significantly less data and computational power compared to the initial training.
Examples
1) BERT [26]
2) SciBERT [9]

Figure 2.6: Challenges with applying generic NLP methods to SciIE: Datasets compiled from non-scientific sources cannot be applied to train SciIE models; word embeddings pre-trained on generic texts need to be enriched with embeddings trained on scientific texts to include terminologies; pre-trained language models requires fine-tuning on scientific texts to achieve better understanding of domain-specific language.

ever, rule-based systems require manual effort and depend on rule quality. Hybrid human-machine systems improve results by applying heuristics to filter results and manually verifying rule-based extraction results, but they have limitations in recovering false negatives[76]. Domain-specific ontologies can be another way to match named entities [88, 103, 33]. The feasibility, however, depends on the availability of comprehensive ontologies, which is increasingly challenging with the rapid growth of publications. High NER accuracy is achieved in biomedicine due to structured databases and unique identifiers, allowing researchers to explore advanced models like machine learning classifiers [30, 13], but few other scientific communities have achieved such a high level of standardization.

Classical machine learning-based approaches, such as decision tree and KNN classifiers, show promise for NER. Recurrent neural networks (RNNs) are widely used for sequence-based tasks but suffer from catastrophic forgetting for NLP tasks [53, 119]. LSTM networks, with gates controlling information flow, offer improvements. Bidirectional LSTM can read text in both directions, enhancing context understanding, but LSTM only uses the information at the last step in the prediction of the following step. Attention-based models retain information at every step, preventing information loss, and automatically assign higher weights to the outputs that contribute more to the correct predictions.

2.3.2 Generic Information Extraction in NLP

It is standard practice in most applications of NLP to re-use ML models trained on plentifully available text (e.g., websites, newspapers), with only *retraining* performed to adapt the models to new domains. For general NLP, there are many datasets (e.g., Penn Treebank [75], CoNLL [107], OntoNotes [129]), pre-trained word vectors (e.g., GloVe [92], FastText [78]), and pre-trained models (e.g., BERT [26], Turing-NLG [101]) available online. However, their training corpora are usually taken from general sources such as newspapers, Twitter posts, online reviews, and Wikipedia pages rather than scientific publications. In addition to common obstacles like long-distance dependencies [114] and polysemant disambiguation [124], the unique terminologies and language styles used in scientific publications pose unique challenges to the IE problem and make model re-use problematic. Elsevier has conducted an evaluation of OpenIE systems on two datasets: one consisting of 200 sentences randomly selected from Wikipedia, and the second made up of 220 sentences from the scientific literature for the 10 most published disciplines. Extractions were checked manually by five humans to guarantee evaluation accuracy, and the results showed that the extractors perform better on encyclopedic sentences (54% precision) than on scientific sentences (34% precision) [40].

The performance gap between general and scientific IE can be attributed to the plethora of terminologies that are critical to understanding the content of the scientific text. For instance, the same materials can be referred to by different names (e.g., trade names, specification number, composition), which complicates both learning language models and organizing extracted data. Long-distance dependencies are also common in scientific articles, such as when the processing path for a material is described in a separate section from where its properties are discussed. Journal articles require significant training for humans to understand well, and it is similar for machines.

2.3.3 Scientific Masked Language Models

Masked Language Models (MLMs), such as BERT and RoBERTa, have been highly successful in a variety of IE tasks, including text classification, named entity recognition, and relation extraction. BERT's phenomenal success in understanding natural language texts has attracted attention from the science community, where valuable knowledge is being created at a rate faster than scientists can reasonably process. Thus, the use of BERT-like transformer models for automated knowledge extraction from literature is of great interest to the community. The original BERT model, however, was pretrained (and evaluated) on the general corpus, the context, terminology, and writing styles of which varies vastly from that of scientific literature. A number of efforts have been made to adapt BERT to various science domains, primarily by pretraining new BERT models on domain-specific corpora.

SciBERT [9] focuses on two domains: biomedical and computer science. Based on the BERT-Base architecture, the four SciBERT models are pretrained on 82% biomedical and 18% computer science texts, for a total of 3.17B words, sourced from SemanticScholar [4]. The four models are cased and uncased versions of SciBERT-BaseVocab and SciBERT-SciVocab. SciBERT-BaseVocab, similar to BioBERT, uses the original BERT vocabulary and continual pretrains from BERT-Base. SciBERT-SciVocab, trains BERT-Base from scratch using a new vocabulary built from the scientific pretraining corpus. The SciBERT variants outperform BERT-Base by an average of 1.66% on biomedical tasks and an average of 3.55% on computer science tasks.

BioBERT [64] is a BERT model for biomedical text. BioBERT added bio-focused texts, namely 4.5B words of abstracts from PubMed and another 13.5B words of full-text articles from PubMedCentral, to the original BERT pretraining corpus. At the time of writing, BioBERT's latest version is v1.2. BioBERT shares the same architecture with BERT-Base and is trained on a cased corpus. Compared to the original BERT model, BioBERT achieves improvements of 0.62% on biomedical named entity recognition, 2.80% on biomedical relation extraction, and 12.24% on biomedical question answering.

PubMedBERT [41], another BERT-Base model targeting the biomedical domain, is also pretrained on text from PubMed and PubMedCentral. Whereas BioBERT starts with a pretrained BERT-Base model that is pretrained further with biomedical specific text, PubMedBERT trains a new BERT-Base model from scratch on only text from PubMed and PubMedCentral. As a result, the vocabulary used in PubMedBERT varies significantly from that used in BERT and BioBERT. PubMedBERT is an uncased model. The pretraining corpus contains 3.1B words from PubMed abstracts and 13.7B words from PubMedCentral articles. PubMedBERT achieves state-of-the-art performance on the Biomedical Language Understanding and Reasoning Benchmark, outperforming BERT by 1.16%.

MatBERT [127] is a materials science-specific model pretrained on 2M journal articles

(8.8B tokens). Experimental results show that it consistently outperforms BERT-Base and SciBERT models on recognizing materials science entities related to solid states, doped materials, and gold nanoparticles, with ∼10% increase in F1 score compared to BERT-Base, and a 1% to 2% improvement compared to SciBERT.

These prior works demonstrate how pretraining BERT models on a domain-specific corpus can improve performance on downstream tasks in that domain. However, as scientific literature corpora grow in size and access to more powerful computing platforms enables larger model architectures, pretraining new language models for each scientific domain will become prohibitively expensive. Thus, there remains a need for larger, multi-disciplinary BERT models that harness the increased availability of pretraining text, available for domain scientist to adapt (i.e., fine tune) to their unique needs.

2.3.4 Causal Language Models

Although MLMs have shown great results in many scientific NLP tasks as discussed above, the recent emergence of Causal Language Models (CLMs), such as GPT-3 [14], GPT-4 [89], and their chatbot application, ChatGPT, may make it seem that the utility of MLMs for science has diminished.

CLMs are designed to generate coherent and fluent text by predicting the next word in a sequence of words. CLMs use an autoregressive architecture, which means that the model generates one word at a time, conditioning on the previously generated words. The goal of CLMs is to learn the probability distribution over all possible next words in a sequence, given the previously generated words. CLMs are particularly useful for language generation tasks such as text completion, machine translation, and summarization.

However, the performance improvements of CLMs are primarily a function of the model size and compute budget, as they tend to be two to three magnitudes larger than popular MLMs such as BERT and require compute budgets not available to the average researcher. Furthermore, the generative nature of CLMs makes them ill-suited for scientific IE tasks.

CLMs may “hallucinate” information rather than extracting them from the input because their architecture allows the output to comprise any token from their vocabulary, not just the ones from the input sequence. CLMs’ strength in generating fluent text makes model hallucinating especially troublesome for scientific IE tasks. While the false positive extraction results from MLMs are wrong spans of tokens from the input and are often incoherent, CLMs can make up seemingly coherent texts that are factually untrue, and spotting such errors requires careful manual validation by domain experts, which defeats a major purpose of automatic IE for science.

CHAPTER 3

RAPID TRAINING DATA COLLECTION

As described in Section 2.2.2, the effectiveness of modern AI techniques relies heavily on the data used for training. Because most SciIE tasks are supervised in nature, both the input data ($\mathbf{x}$) and their corresponding labels ($\mathbf{y}$) are essential. Training such models in modern machine learning typically requires a large number of ($\mathbf{x}$, $\mathbf{y}$) pairs.

In many scientific domains, IE applications are in their early stages, and as a result, there are few annotated training datasets available. In this chapter, we will examine methods for curation of labeled training data, discuss the difficulties involved and present progress towards addressing these challenges.

3.1 Model-in-the-loop Training Data Collection

We demonstrate in Figure 3.1 how to address the lack of labeled training data with an example of assembling a set of human- and machine-labeled data for druglike molecules from the CORD-19 corpus [128]. In describing this process, we refer to paragraphs labeled automatically via a heuristic or model as *silver* and to silver paragraphs for which labels have been corrected by human reviewers as *gold*. We use the Prodigy machine learning annotation tool [94] to manage the review process: reviewers are presented with a silver paragraph, with putative drug entities highlighted; they click on false negative and false positive words to add or remove the highlights and thus produce a gold paragraph. Prodigy saves the corrected labels in standard NER training data format.

Our algorithm involves three main phases, as follows. In the first bootstrap phase, we assemble an initial test set of gold paragraphs for use in subsequent data acquisition. We create a first set of silver paragraphs by using a simple heuristic: we select N_0 paragraphs from CORD-19 that contain one or more words in DrugBank with an Anatomical Therapeutic Chemical Classification System (ATC) code,label those words as drugs, and ask human

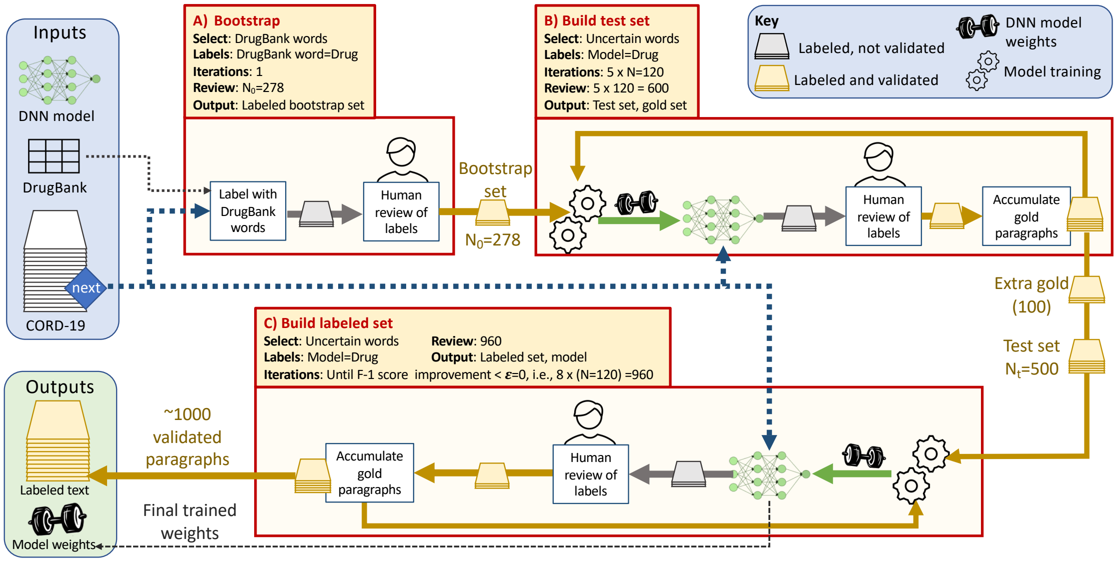


Figure 3.1: Overview of the training data collection workflow, showing the three phases described in the text and with the parameter values used in the study reported here. Each phase pulls paragraphs from the CORD-19 dataset (blue dashed line) according to the **Select** criterion listed in its shaded box. Phases B and C repeatedly update the weights for the NER model (green arrows) that they use to identify and label uncertain paragraphs; human review (yellow and gold arrows) corrects those silver paragraphs to yield gold paragraphs. Total human review work is ∼278+600+960=1838 paragraphs.

reviewers to correct both false positives and false negatives in our silver paragraphs, creating gold paragraphs.

In the subsequent build test set phase, we repeatedly use all gold paragraphs obtained so far to train an NER model; use that model to identify and label additional silver paragraphs, and engage human reviewers to correct false positives and false negatives, creating additional gold paragraphs. We repeat this process until we have N_t initial gold paragraphs.

In the third build labeled set phase, we repeatedly use an NER model trained on all human-validated labels obtained to date, with the N_t gold paragraphs from the bootstrap phase used as a test set, to identify and label promising paragraphs in CORD-19 for additional human review. To maximize the utility of this human effort, we present the reviewers only with paragraphs that contain one or more uncertain words, i.e., words that the NER model identifies as drug/non-drug with a confidence in the range [`min`, `max`]). We continue this process of model retraining, paragraph selection and labeling, and human review until the F-1 score improves by less than ϵ.

The behavior of this algorithm is influenced by six parameters: N_0, N, N_t, ϵ, `min`, and `max`. N_0 and N are the number of paragraphs that are assigned to human reviewers in the first and subsequent steps, respectively. N_t is the number of examples in the test set. ϵ is a threshold that determines when to stop collecting data. The `min` and `max` determine the confidence range from which words are selected for human review. In the experimental studies described below, we used N_0=278, N=120, N_t=500, ϵ=0, `min`=0.45, and `max`=0.55.

The NER model used in the model-in-the-loop annotation workflow to score words might also be viewed as a parameter. In the work reported here, we use SpaCy exclusively for that purpose, as it integrates natively with the Prodigy annotation tool and trains more rapidly We first need to decide which SpaCy model to use for model-in-the-loop annotation. Model size is a primary factor that affects training time and prediction performance. In general, larger models tend to perform better, but require both more data and more time to train effectively. As our model-in-the-loop annotation strategy requires frequent model retraining, and furthermore will (initially at least) have little data, we hypothesize that a smaller model may be adequate for our purposes.

To explore this hypothesis, we study the performance achieved by the SpaCy medium and large models [48] on our initial training set of 278 labeled paragraphs. We show in Figure 3.2 the performance achieved by the two models as a function of number of training epochs. Focusing on the harmonic mean of precision and recall, the F-1 score (a good measure a model's ability to recognize both true positives and true negatives), we see that the two models achieve similar prediction performance, with the largest difference in F-1 score being around 2%. As the large model takes over eight times longer to train per epoch, we select the medium model for model-in-the-loop data collection.

This semi-automated method saves time and effort for human reviewers because they are only asked to verify labels that have already been identified by our model to be uncertain, and thus worth processing. Furthermore, we find that we do not need to engage biomedical

Algorithm 1: Model-in-the-loop Annotation Workflow

```
$\mathcal{C}$ := randomized_paragraphs(CORD) ;    // Prepare CORD-19 paragraphs
/* A) Bootstrap: Identify first silver paragraphs */
$\mathcal{D} := \{s : s \in \text{DrugBank} \,\&\, \text{ATC}(s)\}$ ;    // Extract names from DrugBank
$\mathcal{P}_0 := \{p : p = \textbf{next}(\mathcal{C} \,\&\, p \cap \mathcal{D} \neq \varnothing\} \,\&\, |\mathcal{P}_0| = N_0$ ;    // Assemble first silver set
$\mathcal{B} = \textbf{verify}(\mathcal{P}_0)$ ;    // Verify labels; initialize bootstrap set
/* B) Build test set: Expand on bootstrap set from (A) */
while $|\mathcal{B}| < N_t$ do
    $\mathcal{M} := \text{NER}(\text{train} = 0.6\mathcal{B}, \text{test} = 0.4\mathcal{B}$ ;    // Train:  60-40 train-test split
    $\mathcal{P} := \{p : p = \textbf{next}(\mathcal{C} \,\&\, \exists w \in p : \mathcal{M}(w) \in [\texttt{min}, \texttt{max}])\} \,\&\, |\mathcal{P}| = N$ ;    // Next silver
    $\mathcal{V} := \textbf{verify}(\mathcal{P})$ ;    // Engage crowd to verify labels
    $\mathcal{B} := \mathcal{B} \cup \mathcal{V}$ ;    // Add corrected paragraphs to bootstrap set
end
$\mathcal{T} = \mathcal{B}[: N_t]$ ;    // Initialize test set to first $N_t$ examples
$\mathcal{G} = \mathcal{B}[N_t :]$ ;    // Initialize gold set to any remaining examples
/* C) Build labeled set: Use test set from (B) to evolve model */
while F-1 score improvement $> \epsilon$ do
    $\mathcal{M} := \text{NER}(\text{train} = \mathcal{G}, \text{test} = \mathcal{T}$ ;    // Train on $\mathcal{G}$, test on $\mathcal{T}$
    $\mathcal{P} := \{p : p = \textbf{next}(\mathcal{C} \,\&\, \exists w \in p : \mathcal{M}(w) \in [\texttt{min}, \texttt{max}])\} \,\&\, |\mathcal{P}| = N$ ;    // Next silver
    $\mathcal{V} := \textbf{verify}(\mathcal{P})$ ;    // Engage crowd to verify labels
    $\mathcal{G} := \mathcal{G} \cup \mathcal{V}$ ;    // Add corrected paragraphs to gold set
end
```

professionals to label drugs in text: untrained people, armed with contextual information (and online search engines), can spot drug names in text with accuracy comparable to that of experts.

We provide further details on the three phases of the algorithm in the following, with numbers in the list referring to line numbers in Algorithm 1.

A) Bootstrap

1 We start with the 2020-03-20 release version of the CORD-19 corpus, which contains 44 220 papers [128]. We create $\mathcal{C}$, a random permutation of its paragraphs from which we will repeatedly fetch paragraphs via `next`$(\mathcal{C})$.

2 We bootstrap the labeling process by identifying as $\mathcal{D}$ the 2675 items in the DrugBank ontology with a Anatomical Therapeutic Chemical Classification System (ATC) code attached (eliminating many, but not all, drug-like molecule entities).

3 We create an initial set of silver paragraphs, $\mathcal{P}_0$, by selecting N_0 paragraphs from $\mathcal{C}$ that include a word from $\mathcal{D}$.

4 We engage human reviewers to remove false positives and label false negatives in $\mathcal{P}_0$, yielding an initial set of gold paragraphs, $\mathcal{B}$.

B) Build test set

5 We expand the test set that we will use to evaluate the model created in the next phase, until we have N_t validated examples.

6 We train the NER model on 60% of the data collected to date and evaluate it on the remaining 40%, to create a new trained model, $\mathcal{M}$, with improved knowledge of the types of entities that we seek.

7 We use the probabilities over entities returned by the model to select, as our N new silver paragraphs, $\mathcal{P}$, paragraphs that contain at least one uncertain word (see above).

8 We engage human reviewers to convert these new silver paragraphs, $\mathcal{P}$, to gold, $\mathcal{V}$.

9 We add the new gold paragraphs, $\mathcal{V}$, to the bootstrap set $\mathcal{B}$.

11–12 Having assembled at least N_t validated examples, we select the first N_t as the test set, $\mathcal{T}$, and use any remaining examples to initialize the new gold set, $\mathcal{G}$.

C) Build labeled set

13 We assemble a training set $\mathcal{G}$, using the test set $\mathcal{T}$ assembled in the previous phases for testing. This process continues until the F-1 score stops improving.

14–17 Same as Steps 6–9, except that we train on $\mathcal{G}$ and test on $\mathcal{T}$. Human reviewers are engaged to review new silver paragraphs and produce new gold paragraphs, which are then added to $\mathcal{G}$ instead of $\mathcal{T}$.

3.2 Data-Performance Tradeoffs of the Model-in-the-loop

As noted in Section 3.1, our model-in-the-loop annotation workflow requires repeated retraining of a SpaCy model. Thus we conducted experiments to understand how SpaCy prediction performance is influenced by model size, quantity of training data, and amount of training performed.

3.2.1 Model Size

We first need to decide which SpaCy model to use for model-in-the-loop annotation. Model size is a primary factor that affects training time and prediction performance. In general, larger models tend to perform better, but require both more data and more time to train effectively. As our model-in-the-loop annotation strategy requires frequent model retraining, and furthermore will (initially at least) have little data, we hypothesize that a smaller model may be adequate for our purposes.

To explore this hypothesis, we study the performance achieved by the SpaCy medium and large models [48] on our initial training set of 278 labeled paragraphs. We show in Figure 3.2 the performance achieved by the two models as a function of number of training epochs. Focusing on the harmonic mean of precision and recall, the F-1 score (a good measure a model's ability to recognize both true positives and true negatives), we see that the two models achieve similar prediction performance, with the largest difference in F-1 score being around 2%. As the large model takes over eight times longer to train per epoch, we select the medium model for model-in-the-loop data collection.

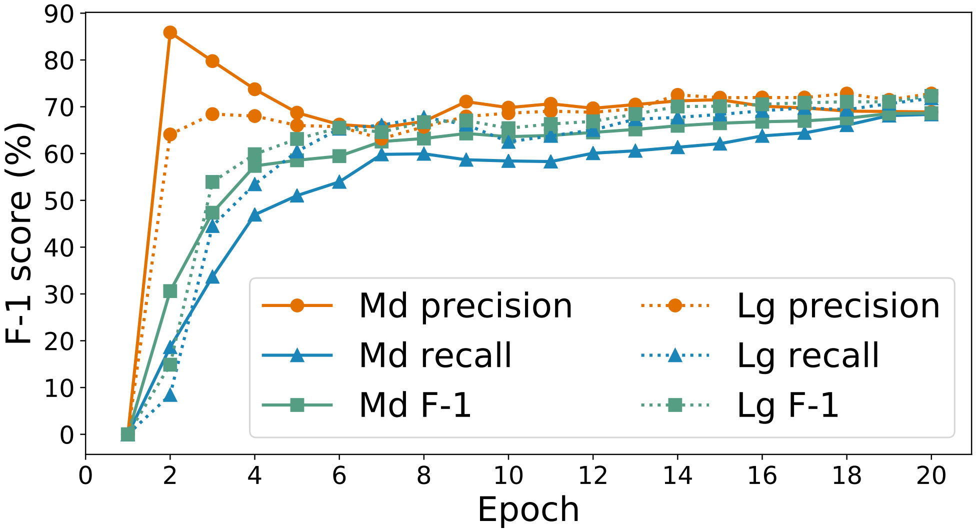


Figure 3.2: Precision, recall, and F-1 scores of medium and large SpaCy models trained on 278 examples.

3.2.2 Amount of Training Data

As data labeling is expensive in both human time and model training time, it is valuable to explore the tradeoff between time spent collecting data and prediction performance. To this end, we manually labeled a set of 500 paragraphs selected at random from CORD-19 [128] as a test set. Then, we used that test set to evaluate the results of training models on increasing numbers of the paragraphs produced by our human-in-the-loop annotation process. In addition to the SpaCy model that is used as the model-in-the-loop, another

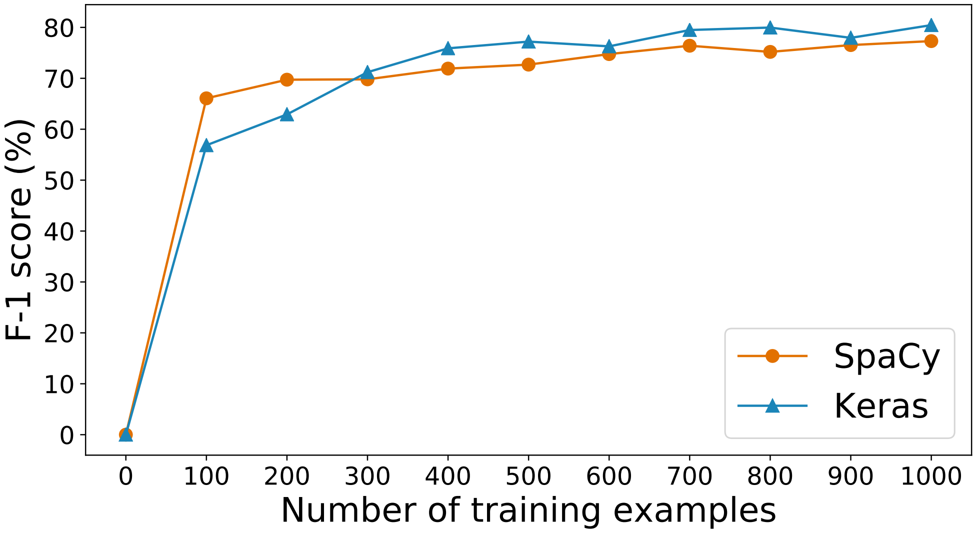


Figure 3.3: Training curves for the SpaCy and Keras models for different number of examples collected.

NER model built with Keras (detailed in Section 4.1) was also trained and evaluated on the same data to act as a control group and to corroborate that the SpaCy model's performance was within expectations. Figure 3.3 shows their F-1 score curves as we scale from 0 to 1000 training samples. With only 100 training examples, SpaCy and Keras-LSTM achieve F-1 scores of 57% and 66%, respectively. SpaCy performs better than Keras-LSTM with fewer training examples (i.e., less than 300), after which Keras-LSTM overtakes it and maintains a small but steady 2–3% advantage as the number of examples increases.

We stopped collecting training data after 1000 examples. We see in Figure 3.3 that the performance of the SpaCy and Keras-LSTM models is essentially the same with 1000 training examples as with 700 examples, with the F-1 score even declining when the number of available examples increases to 800 or 900. At 1000 examples the F-1 score is greatest for both models. We conclude that the 1000 training examples, along with the other 500 withheld as the test set, are best-suited to train our models. There are 4244 and 1861 entities in the training and test set, respectively.

3.2.3 Training Epochs

Prediction performance is also influenced by the number of epochs spent in training. The cost of training is particularly important in a model-in-the-loop setup, as human reviewers cannot work while an model is offline for training.

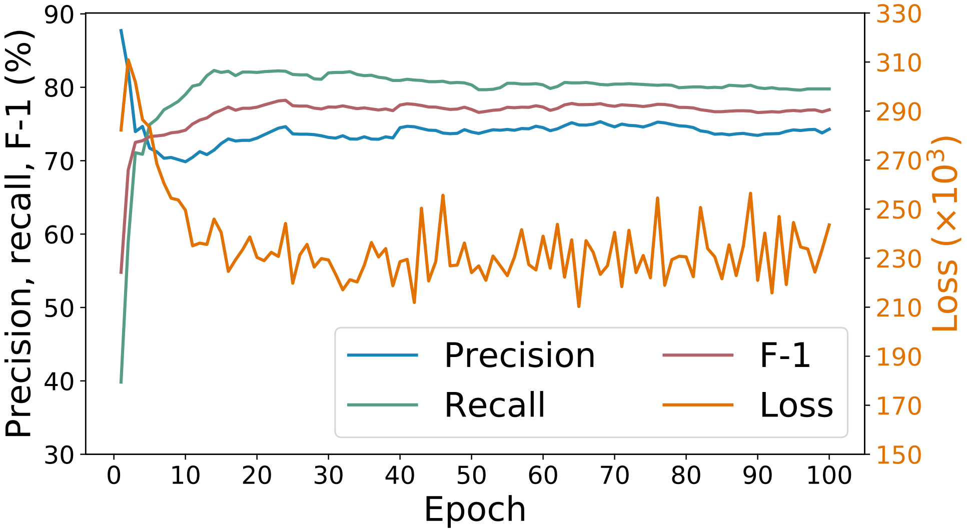


Figure 3.4: Loss, precision, recall, and F-1 of SpaCy model during training for 100 epochs on 278 paragraphs.

Figure 3.4 shows the progression of the loss, precision, recall, and F-1 values of the SpaCy model during 100 epochs of training with the initial 278 examples. We can see that the best F-1 score is achieved within 10 to 20 epochs. Increasing the number of epochs does not result in any further improvement. Indeed, F-1 score does not tell us all about the model's performance. Sometimes training for more epochs could lead to lower loss values while other metrics (such as precision, recall, or F-1) no longer improve. That would still be desirable because it means the model is now more "confident," in a sense, about its predictions. However, that is not the case here. As shown in Figure 3.4, after around 40 epochs the loss begins to oscillate instead of continuing downwards, suggesting that in this case training for 100 epochs does not result in a better model than only training for 20 epochs.

3.3 Data Collection Results

3.3.1 Performance of Human Annotators

Recognizing drug-like molecules is a difficult task even for humans, especially non-medical professionals (such as our non-expert annotators). To assess the accuracy of the annotators, we asked three people to examine 96 paragraphs, with their associated labels, selected at random from the labeled examples. Two of these reviewers had been involved in creating the labeled dataset; the third had not. For each paragraph, each reviewer decided independently whether each drug molecule entity was labeled correctly (a true positive), was labeled as a drug when it was not (a false positive), or was not labeled (a false negative). If all three reviewers agreed in their opinions on a paragraph (the case for 88 of the 96 paragraphs), we accepted their opinions; if they disagreed (the case for eight paragraphs), we engaged an expert.

This process revealed a total of 257 drug molecule entities in the 96 paragraphs, of which the annotators labeled 201 correctly (true positives), labeled 49 incorrectly (false positives), and missed 34 (false negatives). The numbers of true positives and false negatives do not sum up to the total number of drug molecules because in some cases an annotator labeled not to a drug entity but the entity plus extra preceding or succeeding word or punctuation mark (e.g. “sofosbuvir,” instead of “sofosbuvir”) and we count such occurrences as false positives rather than false negatives. In this evaluation, the non-expert annotators achieved an F-1 score of 82.9%.

3.3.2 Effects of Training Data Quality on Model Performance

We described in the previous section how the review of 96 paragraphs labeled by the non-expert annotators revealed an error rate of about 20%. This raises the question of whether model performance could be improved with better training data. To examine this question, we compare the performance of our models when trained on original vs. corrected data. As we only have 96 corrected paragraphs, we restrict our training sets to those 96 paragraphs in each case.

We sorted the 96 paragraphs in both datasets so that they are considered in the same order. Then, we split each dataset into five subsets for K-fold cross validation (K=5), with the first four subsets having 19 paragraphs each and the last subset having 20. Since K is set to five, the SpaCy and Keras models are trained five times. In the i-th round, each model is trained on four subsets (excluding the i-th) of each dataset. The i-th subset of the corrected dataset is used as the test set. The i-th subset of the original dataset is not used in the i-th round.

We present the K-fold cross validation results in Tables 3.1 and 3.2. The models performed reasonably well when trained on the original dataset, with an average F-1 score only 2% less than that achieved with the corrected labels. Given that the expert input required for validation is hard to come by, we believe that using non-expert reviewers is an acceptable tradeoff and probably the only practical way to gather large amounts of training data.

Table 3.1: K-fold (K=5) validation of the SpaCy model on 96 paragraphs with original vs. corrected labels. The first five rows are the results of each fold; the last row is the average F-1 score of the five folds.

Original labels			**Corrected labels**		
Precision	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
100.0	25.6	40.7	100.0	20.9	34.6
93.8	50.6	65.7	88.9	53.9	67.1
93.0	63.5	75.5	92.0	73.0	81.4
76.5	45.2	57.1	68.6	61.4	64.8
98.9	92.3	95.5	99.2	92.1	95.5
Average		66.9	Average		68.7

3.4 Analysis of Human and Model Errors

In this section, we explore the contexts in which human annotators and models make mistakes. Specifically, we study the tokens that appear most frequently near to incorrectly

Table 3.2: K-Fold (K=5) validation of the Keras-LSTM model on 96 paragraphs with original vs. corrected labels. The first five rows are the results of each fold; the last row is the average F-1 score of the five folds.

Original Labels			**Corrected Labels**		
Precision	**Recall**	**F-1**	**Precision**	**Recall**	**F-1**
88.5	53.5	66.7	80.6	69.1	74.4
70.6	57.8	63.6	90.2	59.1	71.4
70.0	67.7	68.9	82.9	47.5	60.4
80.8	55.3	65.6	80.8	51.2	62.7
78.1	56.8	65.8	75.0	67.9	71.3
Average		66.1	Average		68.0

labeled entities. To investigate the effects of immediate and long-distance context, we control, as *window size*, the maximum distance between a token and a entity for that token to be considered as "context" for that entity.

One difficulty with this analysis is that the most frequent tokens identified in this way were mostly stop words or punctuation marks. For instance, when the window size is set to three, the 10 most frequent tokens around mislabeled words are, in descending order, "comma(,)," "and," "mg," "period(.)," "right parenthesis())," "with," "of," "left parenthesis(()," "is," and "or." Only "mg" is neither a stop word nor punctuation mark.

Those tokens provide little insight as to why human reviewers might have made mistakes, and furthermore are unlikely to have influenced reviewer decisions. Thus we exclude stopwords and punctuation marks when providing, in Table 3.3, lists of the 10 most frequent tokens within varying window sizes of words that were *incorrectly* identified as molecules by human reviewers.

We see that there are indeed several deceptive contextual words. With a window size of one, the 10 most frequent tokens include "oral," "dose," and "intravenous." It is understandable that an untrained reviewer might label as drugs words that immediately precede or follow such context words. Similar patterns can be seen for window sizes of three and five. Without background knowledge to draw from, non-experts are more likely to rely on their experience gained from labeling previous paragraphs. One may hypothesize that after the reviewers have seen a few dozen to a few hundred paragraphs, those deceptive contextual words must have left a deep impression, so that when those words re-appear they are likely to label the strange unknown word close to them as a drug.

Table 3.3: The 10 most frequent tokens, excluding stopwords and punctuation marks, within various window sizes around entities incorrectly labeled by human reviewers.

	Window size = 1		**Window size = 3**		**Window size = 5**	
#	**Token**	**Count**	**Token**	**Count**	**Token**	**Count**
1	300	4	mg	19	mg	23
2	oral	3	once/day	7	daily	8
3	dose	3	treatment	6	treatment	8
4	intravenous	2	300	5	once/day	7
5	500	2	treated	4	300	7
6	intravenously	1	oral	4	oral	6
7	include	1	once	4	recipients	5
8	Both	1	dose	4	treated	4
9	resistance	1	cidofovir	3	twice	4
10	treatment	1	resistance	3	include	4

To investigate this hypothesis, we also explored the most frequent words around drug entities that are *correctly* labeled by human reviewers: see Table 3.4. Interestingly, we found overlaps between the lists in Tables 3.3 and 3.4: in all, three, four, and two overlaps for window sizes of one, three, and five, respectively, when treating all numerical values as identical. This finding supports our hypothesis that those frequent words around real drug entities may confuse human reviewers when they appear around non-drug entities.

We repeat this comparison of context words around human and model errors while considering stopwords and punctuation marks. Tables 3.5 and 3.6 show the 20 most frequent tokens in each case. We see that 20–25% of the tokens in Table 3.5, but only 5–10% of those in Table 3.6, are *not* stop words or punctuation marks. As the model only learns its word embeddings from the input text, if a token often co-occurs with drug entities in the training corpus the model will treat it as an indication of drug entities near its presence, regardless of whether or not it is a stopword. This apparently leads the model to make incorrect inferences. Humans, on the other hand, are unlikely to think that stopword such as “the” is

Table 3.4: The 10 most frequent tokens, excluding stop words and punctuation marks, within various window sizes around entities correctly labeled by human reviewers.

	Window size = 1		**Window size = 3**		**Window size = 5**	
#	**Token**	**Count**	**Token**	**Count**	**Token**	**Count**
1	resistance	176	Tetracycline	230	Tetracycline	230
2	treatment	9	resistance	177	resistance	178
3	mM	4	Trimethoprim	118	Trimethoprim	118
4	oral	3	treatment	11	treatment	14
5	after	3	20∼	7	20∼	8
6	analogue	3	Figure	5	placebo	7
7	responses	3	concentration	5	effects	6
8	antibiotics	2	compared	4	Figure	6
9	exposure	2	100	4	KLK5	6
10	pharmacokinetics	2	mM	4	matriptase	6

Table 3.5: The 20 most frequent tokens, including stop words and punctuation marks, within various window sizes around entities incorrectly labeled by human reviewers. Words that are neither stop words nor punctuation words are in boldface.

	Window size = 1		**Window size = 3**		**Window size = 5**	
#	**Token**	**Count**	**Token**	**Count**	**Token**	**Count**
1	,	27	,	49	,	74
2	.	14	and	21	.	28
3	and	14	**mg**	19	and	28
4	with	8	.	18	**mg**	23
5	(	7	)	13	)	21
6	is	7	with	10	of	18
7	of	6	of	10	(	17
8	was	6	(	9	with	13
9	or	4	is	9	to	12
10	**300**	4	or	7	the	11
11	**oral**	3	**once/day**	7	a	11
12	has	3	a	7	in	10
13	to	3	the	6	is	9
14	[	3	was	6	or	9
15	**dose**	3	to	6	**daily**	8
16	**intravenous**	2	**treatment**	6	was	8
17	in	2	in	5	**treatment**	8
18	may	2	**300**	5	]	8
19	**500**	2	be	4	**once/day**	8
20	a	2	**treated**	4	were	7

Table 3.6: The 20 most frequent tokens, including stop words and punctuation marks, within various window sizes around entities incorrectly labeled by the Keras model. Words that are neither stop words nor punctuation words are in boldface.

	Window size = 1		**Window size = 3**		**Window size = 5**	
#	**Token**	**Count**	**Token**	**Count**	**Token**	**Count**
1	,	166	,	347	,	468
2	(	86	and	126	and	176
3	and	81	(	117	(	162
4	of	58	)	89	)	143
5	)	30	of	85	of	130
6	to	28	the	73	the	121
7	or	24	to	60	to	85
8	.	22	.	48	in	75
9	**mM**	18	with	44	with	72
10	with	17	a	41	.	68
11	in	15	in	37	a	62
12	for	15	or	35	was	52
13	is	14	was	33	or	47
14	as	14	**mM**	32	is	43
15	the	13	is	31	for	42
16	was	12	as	29	that	37
17	that	11	that	25	**mM**	36
18	[	11	by	22	by	35
19	**treatment**	9	for	22	as	32
20	a	9	[	22	were	31

indicative of drug entities, no matter how frequently they appear together.

3.5 Data Availability and Formats

We have made our annotated training data, trained models, and the results of applying the models to the CORD-19 corpus publicly available online. [8].

In order to facilitate training of various models, we published the training data in two formats—an unsegmented version in line-delimited JSON (JSONL) format, and a segmented version in Comma Separated Value (CSV) format. The JSONL format contains the most comprehensive information that we have collected on the paragraphs in the dataset. We choose JSONL format rather than a JSON list because it allows for the retrieval of objects without having to parse the entire file. A JSON object in the JSONL file has the following structure:

- `text`: The original paragraph stored as a string without any modification.
- `tokens`: The list of tokens from `text` after tokenization.
    - `text`: The text of the token as a string.
    - `start`: The index of the first character of the token in `text`.
    - `end`: The index of the first character after the token in `text`.
    - `id`: Zero-based numbering of the token.
- `spans`: The list of spans (sequences of tokens) that are labeled as named entities (drugs)
    - `start`: The index of the first character of the span in `text`.
    - `end`: The index of the first character after the span in `text`.
    - `token_start`: The index of the first token of the span in `text`.
    - `token_end`: The index of the last token of the span in `text`.
    - `label`: The label of the span ("drug")

Another commonly adopted labeling scheme for NER datasets is the "IOB" labeling scheme, in which the original text is first tokenized and each token is assigned a label "I," "O," or "B." The label "B(eginning)" means the corresponding token is the first in a named entity. A label "I(nside)" is given to every token in a named entity except for the first token. All other tokens gets the label "O(utside)" which means they are not part of any named entity. The aforementioned JSONL data are converted according to the IOB scheme and stored in Comma Separated Value (CSV) files with one training example per line. Each line consists of two columns: a first of tokens that made up of the original texts, and a second of the corresponding IOB labels for those tokens. In addition to a different labeling scheme, the samples in the CSV files are segmented, meaning that each sentence is treated as a training sample instead of an entire paragraph. This structure aligns with that used in standard NER training sets such as CoNLL03 [107].

CHAPTER 4

SCIENTIFIC INFORMATION EXTRACTION WITH DEEP NEURAL NETWORKS

In this chapter, we tackle the SciIE problem step-by-step. First, we identify all the entities of interest with an NER model, and then determine the relations between entities with a Relation Extraction model. We present experimental results on a variety of domains to demonstrate the generalizability of the proposed SciIE pipeline.

4.1 Named Entity Recognition (NER)

We first describe the specific extraction problems and then present the Bidirectional LSTM and Lexicon-infused LSTM models used in this work.

4.1.1 Problem Definition

Given a paper, comprised of sections containing natural language text, our task is to identify scientific named entities of interest.

For example, in materials science publications researchers publish facts about polymers, such as their melting point or glass transition temperature. A crucial first step in extracting these facts is the need to identify polymer names. When given the following text from a materials science paper, the goal is to extract the polymer *"polystyrene"*.

> "...For instance, O'Shea et al. prepared bromo-terminated polystyrene (PS-Br) by ATRP, which was then treated with potassium phthalimide followed by the hydrazinolysis in order to obtain amino-terminated polystyrene (PS-NH2)..."

In social science publications our task is slightly different. Here researchers explore hypotheses and make assertions based on analysis of known datasets. Unfortunately, the datasets used are often embedded in the natural language text of the paper rather than cited like other artifacts. Thus, when given the following paragraph from a social science paper, we aim to extract the title of the dataset, i.e. *"Longitudinal Study of American Youth."*

> "...With this deficit looming, efforts to understand the factors that contribute to degree attainment have redoubled. This article is a piece of that effort. By analyzing data from 3279 individuals who participated in the Longitudinal Study of American Youth, this study examines the relative importance of personal, peer, and parent educational expectations in the degree attainment of students 15 years after high school graduation..."

For biomedical research, our task is to identify potential drug-like molecules for the SARS-CoV-2 virus from past publications. In this task, the goal is not simply finding any drug-like molecule, but the ones that have been discussed in coronavirus contexts. For example, when given the following paragraph from a biomedical paper, we aim to extract the name of the drug "Ribavirin"

> "This is a retrospective cohort study of critically ill patients with laboratory-confirmed MERS from 14 hospitals in Saudi Arabia diagnosed between September 2012 and January 2018. We evaluated the association of Ribavirin with 90-day mortality and MERS coronavirus (MERS-CoV) RNA clearance using marginal structural modeling to account for baseline and time-varying confounders."

4.1.2 A Lexicon-infused Bidirectional LSTM with CRF Model

The Long-Short Term Memory (LSTM) network has been applied to a variety of natural language processing tasks including language modeling [115], speech recognition [38], as well as named entity recognition [17]. One major advantage that LSTM has over other traditional methods is that it does not require any specific (and often manually selected) features. A label, "B," "I," or "O" is assigned to every word in the training corpus. "B" is assigned to the first word in a named entity as well as single-word named entities, while “I” marks the rest of a multi-word entity. Non-named entity words are given the label “O.”

It is a simple sequential neural network model with one input (embedded text) and one output(predicted label). The Bi-LSTM network evaluates every word based on the words around it in the input text, and outputs the label that it considers the most likely. This, however, means that the network has no awareness of the validity of the label sequence that it generates so it may output sequences such as “OIO”, which makes no sense under the ”BIO” labeling scheme.

To penalize invalid label sequences, a Conditional Random Field (CRF) layer is added on top of the Bi-LSTM network, as shown in Fig. 4.1. Rather than considering the words in the input text one by one, the CRF also takes the context into account, which makes it suitable for tasks that involves sequences of input/output samples. The CRF layer tries to find patterns in the labels from the training data, and assigns penalties to the predictions that mismatch.

To further improve the accuracy of our model, an external source of knowledge, DBpedia [6], is introduced. Simply put, DBpedia is a structured version of Wikipedia, which consists of categorized entries on people, organizations, locations, etc. By encoding these information and feeding it to the LSTM, the network can achieve a whole other level of understanding of the internal structure of named entities. With the addition of external lexicons, the neural network is no longer a strictly sequential model, as there are two inputs, one for the embedded text, one for the one-hot encoded lexicon features (which will be discussed in detail in the following section). A Concatenation layer is added to the network to combine these two inputs together and feed the concatenated vector to LSTM, as illustrated in Fig. 4.1.

For example, without the external lexicon, the LSTM can only learn about the order of the words, and seeing ”*poly(vinyl methyl ether)*” in the training data would not give it a lot of hint that ”*poly(ethylene glycol)*” is also a polymer. However, with the external

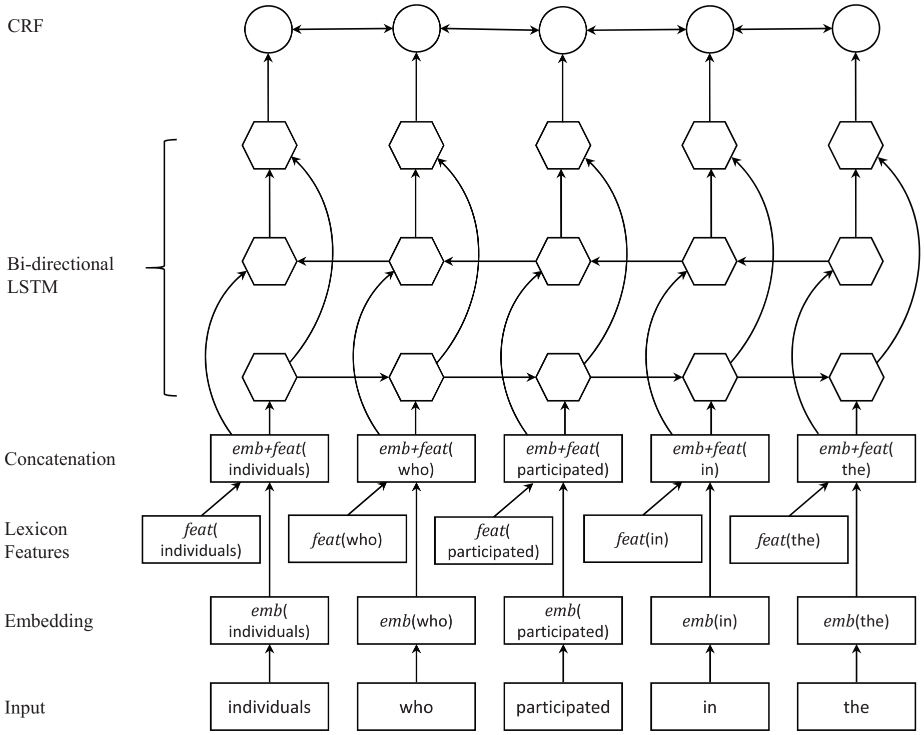


Figure 4.1: The model with external lexicon knowledge.

Table 4.1: An example of encoding external knowledge from DBpedia

	The	U.S.	Bureau	of	Labor	Statistics	Industry	Injury	and	Illness	Data	reveals	that	···
LOC	O	B	O	O	O	O	O	O	O	O	O	O	O	···
ORG	O	B	I	I	I	I	O	O	O	O	O	O	O	···

knowledge from DBpedia, the LSTM will learn that *vinyl methyl ether* and *ethylene glycol* are both chemical compounds and they both follow the pattern ”*poly([chemical compounds])*”. Therefore the network will be more likely to recognize ”*poly(ethylene glycol)*” as a polymer even though it has never seen it before. To encode the knowledge from the external lexicon, the same ”BIO” annotation as described above is used, as shown in Table 4.1. It can be seen from the table that there can be overlapping between multiple classes. Within each class overlapping is also possible. For example, “US Bureau of Labor Statistics” and “Bureau of Labor Statistics” are both present in the Organization lexicon. In such cases the longest match is used.

4.1.3 Model Features

The input we have on hand is a series of words. The neural network, however, requires numerical data (i.e. vectors) to compute. Therefore, it is necessary to find proper numerical representations for words in the corpus.

FastText Word Embedding Model

The FastText model from Facebook [11] provides a flexible architecture for encoding word embeddings compared to its predecessors such as CBOW [77] and Skipgram [79]. Aside from representing a word based on its contexts, it also makes use of character n-grams. Words are mapped to character n-grams, which are then embedded in vectors. The n-gram vectors will make up a part of the embeddings for the words that do not appear frequently enough and thus do not have enough context in the training corpus. The addition of n-gram embeddings also greatly helps when the target word is not in the vocabulary of the pre-trained word embeddings. There is little that classic methods such as CBOW or Skip-gram can do when faced with an unknown word. They may either give it a random vector or the average of all the other vectors in the vocabulary, but, unsurprisingly, such a vector does not reflect the actual meaning of the out-of-vocabulary word. FastText, meanwhile, can capture the meaning of an unknown word better by making a word vector out of its character n-grams.

Lexicon Features

The current ontology of DBpedia has over 4.2 million entries in 774 classes. Matching all of them to the training and testing corpus is a rather computationally intensive task. Appending the one-hot encodings of the “BIO” labelling for all the classes to a word vector will result in an extra $774 \times 3 = 2322$ dimensions, but for each word vector most of these dimensions will be zero. In other words, the concatenated vector will be very sparse and inefficient to compute. This is undesirable because it defeats the purpose of word embedding.

To avoid diluting the dense word vector too much, only the few DBpedia classes that are relevant to the NER task at hand should be used. For example, to identify polymers, "Chemical Elements", "Chemical Compounds", and "Chemical Substance" are potentially helpful classes so we should look for entries only from these classes in the training and testing corpus.

4.1.4 Experimental Analysis

Data

In the sociology domain, The Inter-university Consortium for Political and Social Research (ICPSR) provides a large number of manually curated relationships between datasets and papers in the field of social science, from which we obtained a "gold standard" dataset of publication-dataset relationships. ICPSR has indexed over 72,000 papers, out of which 6,368 are used in our evaluation. These papers are chosen because they are hosted by Elsevier, which provides an easy-to-use API to obtain the full text of publications in JSON format. The 6,368 papers are 8-2 split. 80% of the papers are used as training data, while the rest 20% are reserved for testing. Further, we crawled the full list of datasets that are indexed by ICSPR, which includes 60,566 dataset names.

For the materials science domain, a corpus of 100 materials science publications is gathered and manually labeled. Each paper was reviewed by 2 people to tag polymers, and when disagreement arose a third more senior domain expert made the final judgement. This materials science corpus includes 7,092 sentences that have at least one polymer.

For the biomedical domain, 1500 paragraphs from the CORD-19 dataset [128] were annotated using the Rapid Training Data Collection method described in Chapter 3.

Table 4.2: SciNER on polymer dataset

#	NER Task	Precision	Recall	F1 Score
1	SciNER with FastText Word Vectors	89.55%	92.26%	**0.9088**
2	CDE (NLP module only) on material science dataset	54.26%	58.28%	0.5620
3	CDE (NLP+regex+dictionary) on material science dataset	65.08%	58.66%	0.6170

Table 4.3: SciNER on drug molecule dataset

#	NER Task	Precision	Recall	F1 Score
4	SciNER with FastText Word Vectors	86.65%	75.08%	**0.8045**
5	SpaCy (web_md) NER on the drug molecule dataset	86.64%	69.88%	0.7736

Table 4.4: SciNER on sociology dataset

#	NER Task	Precision	Recall	F1 Score
6	SciNER with FastText Word Vectors	0.8253%	0.8704%	0.8473

4.1.5 Experimental Results

In this section, the proposed model is applied to three NER tasks in different domains to demonstrate its effectiveness and generalizability. Table 4.2, 4.3, and 4.4 shows the precision, recall, as well as the F1 score for each task.

The lexicon-infused Bi-LSTM architecture is applied to three NER tasks in different domains to demonstrate its effectiveness and generalizability.

The first task is to identify polymers in materials science publications (Table 4.2). The lexicon-infused Bi-LSTM achieved a F1 score of 0.9088, better than that of the comparison group, ChemDataExtractor (CDE), which is the state-of-the-art model for recognizing chemical entities [116]. CDE combines machine learning with classic rule-based approaches such as regular expression, dictionary matching, etc. When tested only with its NLP module, it gets an F1 score of 0.5620. Using its entire pipeline the score increases to 0.6170, In either case, SciNER manages to beat CDE by about 50%.

The second task is extracting druglike molecules from COVID-19 research articles. We used the data collection methods described in Section 3 to collect a training set, and trained both SpaCy and lexicon-infused Bi-LSTM NER models to recognize and extract drug-like molecules in text. We find that the trained `en_core_web_md` SpaCy model achieved a F-1 score of 77.3%, while our model achieved a F-1 score of 80.5%, outperforming SpaCy.

Finally, we applied the model to sociology, a domain outside of natural sciences. The task is to identify datasets mentioned in Social Science articles. SciNER with FastTest word embeddings achieved an F1 score of 0.8473. It shows that our proposed model adapts well to different domains.

4.2 Relation Extraction with Syntax Tree Parsing

After identifying the entities being discussed in text, the next step in the IE pipeline is to determine relations between entities: the Relation Extraction (RE) step,

Traditional RE methods are based on manually defined features and rules [34]. More recently, attention-based statistical models have shown promise [109]. However, RE on scientific texts poses unique challenges: the vocabulary includes domain-specific terminologies, acronyms, and abbreviations, and the sentences are longer and more complex with many modifiers and clauses. We aim to address these challenges in our model.

Position embeddings capture the relative positions of words in a sentence. They are commonly used in RE models due to the intuition that words that are closer together are more likely to be related [109]. This assumption, however, is not always the case, especially in complex sentences with many modifiers or clauses between entities. Dependency parsing, on the other hand, identifies the relation between modifier words and "head words" in a sentence by means of syntactic analysis. After dependency parsing, a sentence can be represented as a tree with the words as nodes and the dependency paths as edges. The tree is a connected, undirected graph, so Dijkstra's Algorithm can find the shortest path between any two nodes, which captures the semantic relations better than relative positions (Fig. 4.2) [27]. We encode the dependency parsing information as a feature for each word by concatenating the index of its head word and the lengths of its shortest paths to the two entities.

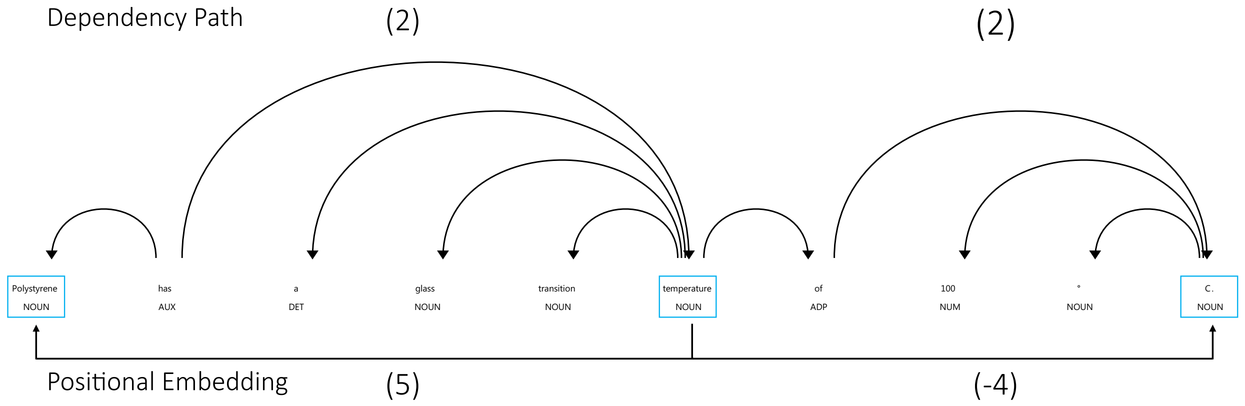


Figure 4.2: Comparison of Positional Embedding and Dependency Path: In the example, the relative position of the words (Polystyrene, temperature) is 5, while the lengths of shortest dependency path between the two words is 2.

Fig. 4.2 illustrates the advantages of dependency paths over simple relative positional embeddings: In the sentence *"Polystyrene has a glass transition temperature of 100°C"*, the relative position between the words *Polystyrene* and *temperature* is 5, but the shortest path between the same words in the dependency tree has a length of 2, so it captures the close semantic relations between the two words better than positional embeddings. In our model, we encode the information from dependency parsing as a feature for each word by concatenating the index of its head word and the lengths of the shortest path to the two entities.

Neural networks assume all words are of equal importance. *Attention* mechanisms are used as a way of highlighting important input data. In traditional RE models, the attention layer computes attention weights for each word in a sentence based on the word's embedding and the embeddings of the two entity words in the sentence. This is based on the assumption that there is a salient connection between the entity words and the relation terms. In our model we also take the relation term into consideration. The relation terms are stripped of stop words (e.g., "FEATURE OF" becomes "FEATURE"), and for each word, a new attention weight is calculated based on its embeddings and embeddings of the relation terms. Concatenating the new and the traditional attention weights produces the final attention-based feature of a word.

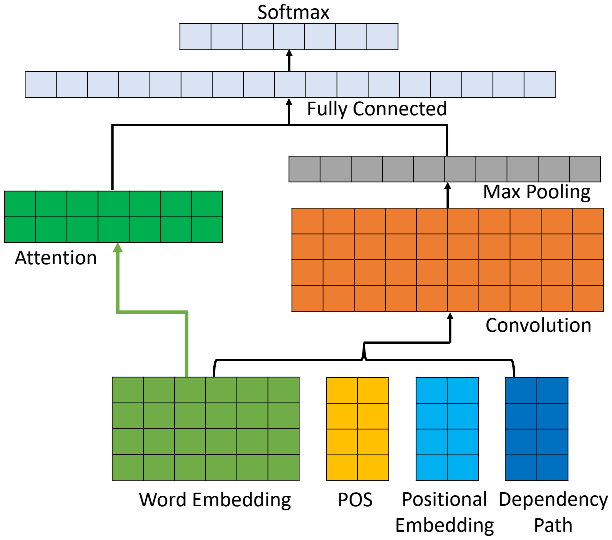


Figure 4.3: Model Architecture

Fig. 4.3 shows the overall model architecture. The inputs are the concatenation of four features: word embeddings, Part-of-Speech tags, positional embeddings, and dependency path features. The inputs are fed into a convolutional network to extract sentence features. Neural networks by default assume all words are of equal importance, so the attention mechanism is needed to highlight words that are more indicative of the relations present in a sentence. The attention layer takes word embeddings as inputs. For each word in an input sentence, it calculates attention weights between it and every entity in the sentence. The outputs from the attention layer and the convolutional network are concatenated and taken as inputs to a dense layer, after which a softmax is applied to generate the final probability distribution of the input sentences over the potential relation types.

We evaluate our model on the Scientific Information Extraction Relation Extraction dataset [1]. This is the same dataset that is used by SciBERT [9]. The dataset includes gold standard annotations for 4648 sentences from scientific publications, including 3219 in the training set, 455 in the development set, and 974 in the test set. Table. 4.5 shows the number of sentences that consists of each type of relations in the training, development, and test set.

Table 4.5: The number of sentences in each type of relations (`COMPARE, CONJUNCTION, EVALUATE-FOR, FEATURE-OF, PART-OF, USED-FOR`) in the training, development, and test set

	COMP	CONJ	EVAL	FEAT	HYPO	PART	USED
TRAIN	166	400	313	173	298	179	1690
DEV	29	59	50	32	44	27	214
TEST	38	123	91	59	67	63	533

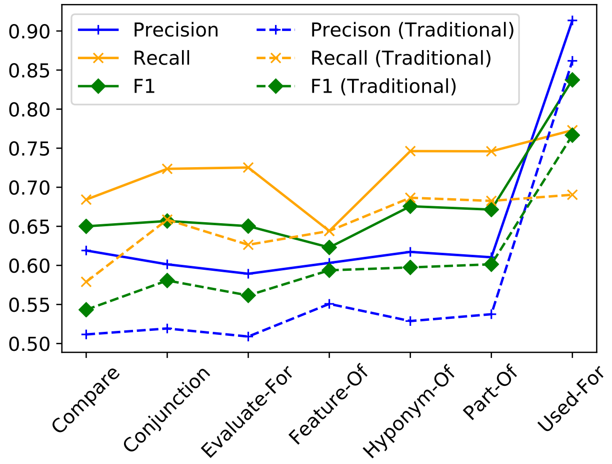


Figure 4.4: Precision, recall, and F1 score of the the proposed model and the traditional RE model on each type of relation in the dataset.

Our model consistently outperformed the traditional approach (Fig. 4.4). The macro-average F1 of our model and the traditional model across all relation types is 0.681 and 0.606, respectively. The weighted-average F1 is 0.755 and 0.682. The micro-averaged-F1, or the overall accuracy, is 0.747 and 0.672. Both models perform noticeably better on identifying the "Used-For" relation, likely due to its large proportion in the dataset (Table 4.5). Our method improves upon traditional models by incorporating Dependency Parse information, relation-term attention, and specialized word embeddings. Our model outperformed a traditional model by up to 0.073 weighted-average F1.

CHAPTER 5

LARGE LANGUAGE MODELS FOR SCIENCE

Transformer-based large language models (LLMs) such as BERT [25] have delivered impressive results for a variety of tasks. These models show particular promise for a range of scientific tasks, such as automating extraction of facts from literature, classifying and tagging abstracts, and deriving relationships between scientific entities. As a result, there have been several domain-specific (e.g., PubMedBERT [41], MatSciBERT) and general science (e.g., SciBERT) models trained by the community. These models show impressive performance on many NLP tasks; however, each model is limited with respect to the breadth of its focus, including the SciBERT model that is primarily trained on computer science and biomedical data.

We train a series of scientific language models, called ScholarBERT, on a large, multidisciplinary scientific corpus consisting of 225B tokens to understand the effects of model size, data size, as well as pretraining and finetuning epochs on downstream task performance. We find that for information extraction tasks, the primary application for scientific language models, the performance gains by training a larger model for longer with more data are not robust—they are highly dependent on the individual tasks.

5.1 Data

5.1.1 The Public Resource Dataset

We pretrain the ScholarBERT models on a dataset provided by Public.Resource.Org, Inc. ("Public Resource"), a nonprofit organization based in California. This dataset was constructed from a corpus of 85M journal article PDF files, from which the Grobid tool, version 0.5.5, was used to extract text Grobid is "a machine learning library for extracting, parsing and re-structuring raw documents such as PDF into structured XML/TEI encoded documents with a particular focus on technical and scientific publications" [39]. We used the XML/TEI encoded documents, not the underlying PDF files. Not all extractions were successful, because of corrupted or badly encoded PDF files. We successfully extracted text from 75 496 055 articles from 178 928 journals in the Dataset. We work here with text from ∼75M articles in this dataset, For most articles, we also extracted metadata, such as the title, authors, a DOI, and other descriptive information. and categorize them using their journal titles to obtain the distribution of the articles by domain – 45.3% biomedicine, 23.1% technology, 20.0% physical sciences, 8.4% social sciences, and 3.1% arts & humanities.

5.1.2 The PILE dataset

The Pile is a vast English text dataset, consisting of 22 high-quality datasets, designed to train large-scale language models. It includes both established natural language processing datasets and newly introduced ones. The Pile can be used not only to train large language models but also as a benchmark for the generalization ability and cross-domain knowledge of language models. The dataset's size is 825.18 GiB, making it an excellent resource for language model development [36]. Notable models trained on the PILE dataset include GPT-2 and GPT-3 from OpenAI. Among its 22 datasets, a number of them are primarily science-focused, such as ArXiv and PubMed.

5.1.3 Deduplication of the Training Corpora

As a single corpus is limited in size and domain, having just one corpus is often not enough to train a large language model with billions of parameters. Therefore, combining texts from multiple sources is a common technique to efficiently collect a vast and diverse training corpus. However, as the corpora are collected from a variety of sources, they may contain duplicates of the same article or articles that share large portions of their text (such as preprints and published articles). Having unknown duplicates in the training corpus will bias the model towards the overrepresented data, consume unnecessary compute cycles and memory, and produce overfit models that end up memorizing those specific instances rather than learning the underlying patterns. Therefore, deduplication is a necessary step before the corpus can be used in training. Even the same article may appear slightly differently in various sources due to variations in typesetting, publisher copyright statements, etc. Strict string matching will fail to detect such duplicates. A better way to measure the “similarity” of two documents is the Jaccard Similarity Score, which is defined as

$$Jaccard(S,T) = \frac{\mid S \cap T \mid}{\mid S \cup T \mid}$$

where S and T are the sets of tokens (i.e., words) of two input texts, respectively. With the Jaccard similarity score, we can set a threshold above which two documents will be considered to be duplicates. However, a brute-force method comparing every pair of documents will take $O(N^2)$ time, so it is not computationally feasible for a large collection of texts.

To detect duplicated texts effectively and efficiently, we applied a technique called ”minhashing”, which compresses large sets in such a way that we can still deduce the similarity of the underlying sets from their compressed versions. By combining minhashing with ”locality-sensitive hashing (LSH)”, which focuses on pairs that are likely to be similar without investigating every pair, we can achieve sub-linear performance, making deduplicating a corpus as large as PILE and PRD a possibility [65].

The overall process consists of three main steps: 1) precompute minhash signatures for every document, 2) build an LSH index for the corpus, and 3) query the index for potential duplicates.

Step 1. Precompute minhash signatures

To compute the minhash for every document, we first construct a characteristic matrix, as shown in Table 5.1, where the columns are the documents and the rows are the tokens in the documents. If a token appears in a document, it will have a ‘1’ in the corresponding location in the matrix, otherwise it will be ‘0’. After a random permutation of the rows,

Table 5.1: A minhash characteristic matrix for 3 documents T_1, T_2, T_3. a – d are the tokens in the documents. The rows are randomly permutated. A '1' in the matrix means the token exists in that document, whereas a '0' indicates the opposite.

	T_1	T_2	T_3
b	1	0	0
a	0	1	0
d	1	0	1
c	1	1	1

the minhash value for a document is the number of the first row in which the column has a '1'. It can be formally proved that the probability that the minhash function for a random permutation of rows produces the same value for two documents equals the Jaccard similarity of those documents [65].

Computing the minhashes involves randomly permutating the matrix, but in practice the number of rows in the matrix will be in the millions, equal to the number of distinct tokens in the corpus. Picking a random permutation and sorting the rows according to the permutation are computationally infeasible. We can simulate a permutation by a random hash function that maps row numbers to as many buckets as there are rows. To calculate the probability that the minhash function for a random permutation of rows produces the same value for two documents, we pick n random hash functions instead of n random permutations.

Computing the minhashes is a time-consuming process due to heavy I/O (i.e., reading every file in the corpus). To speed up the process, our implementation is parallelized such that the documents are distributed to all CPU cores to compute their minhash signatures, and the main process will save all the returned signatures in a pickle file.

Each minhash signature is stored together with a key that provides a unique identifier for a document in the corpus. In our implementation, the key consists of the name of the source file where the document is stored and the line number that indicates where the document starts within that source file.

Step 2. Build the LSH index

With the minhash characteristic matrix built, computing Jaccard similarities for every pair of documents still takes $O(N^2)$ time because minhashing does not reduce the time complexity for the pairwise comparison. Intuitively, we want to compare only the pairs that are above some lower bound in similarity, without investigating every pair. This is called locality-sensitive hashing (LSH) or near-neighbor search. We can divide the signature matrix into b bands consisting of r rows each. We shall normally assume that two vectors hash to the same bucket if and only if they are identical. We consider a pair of documents to be potential duplicates if and only if the signatures agree in all the rows of at least one band, and we will only compute the Jaccard Similarities for pairs of documents that meet this requirement. In this way, detecting duplicates in a large corpus with N documents can be achieved in less than $O(N)$ (sublinear) time.

The threshold parameter controls the lower bound of similarity above which a pair of documents would be considered duplicates. Note that the threshold is decided when building the index and it is not possible to change the threshold at query time without building another index. Therefore, selecting an appropriate threshold is the key to minimize false positive and false negative detection. We experimented with threshold values of 0.6, 0.7, 0.8 and 0.9. As an evaluation, we randomly checked 20 pairs of documents predicted to be duplicates with each threshold value. With the threshold at 0.6, the false detection rate (FDR) was 45%; as the threshold increased to 0.7, the FDR dropped to 15%. When the threshold was set to 0.8 and 0.9, the FDR is 0%, but at 0.9, all the detected pairs were exact matches, which defeated the purpose of applying Jaccard Similarity in the first place. Therefore, the threshold is chosen to be 0.8.

Technically, the index can be stored as an in-memory object. However, to enable parallelization and data persistency, Redis is used as the storage layer for the LSH index. The index will consume roughly twice as much memory as the minhash signatures do, so the index for a very large corpus might exceed the available RAM of a compute node, in which case the signatures need to be partitioned and saved into different indices on different nodes. When Redis is used as the storage backend instead of an in-memory data structure, it is important to set a specific basename for the LSH index. This is necessary because multiple indices may be stored in the same Redis database. To query an index stored in Redis, the same basename must be used to initialize the LSH object. Otherwise, queries will always result in empty responses.

Step 3: Query for duplicates

In the last step, we read minhash signatures from Step 1 and query them against the LSH index built in Step 2.

The code is parallelized so that the signatures are distributed to all CPU cores. The query returns duplicated documents as tuples, in which the first element is always the key of the minhash signature used to query the index, and the second element is the key of a duplicate document found in the index.

Result: Deduplication of PRD and PILE

The deduplication pipeline was tested on the PRD dataset and PILE dataset. PILE contains documents from a variety of sources, many of which are not scientific articles and are unlikely to find duplicates in the ArXiv dataset. Therefore, only the "ArXiv" and "PubMed Central" subsets of PILE were searched for deduplication with ARXIV.

There are 85 million files in the PRD dataset, and 8 million files in the science subsets of PILE. 93 204 documents in ARXIV were found to have at least 1 duplicate in the ArXiv and PubMed Central subsets of PILE. 70 788 documents in the ‘ArXiv‘ and ‘PubMed Central‘ subsets of PILE were found to have at least 1 duplicate in PRD. In total, there are 90 296 pairs of duplicated documents.

5.1.4 ScholarBERT Pretraining

We randomly sample 1%, 10%, and 100% of the Public Resource dataset to create PRD_1, PRD_10, and PRD_100. We pretrain ScholarBERT models on these PRD subsets by using the RoBERTa pretraining procedure, which has been shown to produce better downstream task performance in a variety of domains [70].

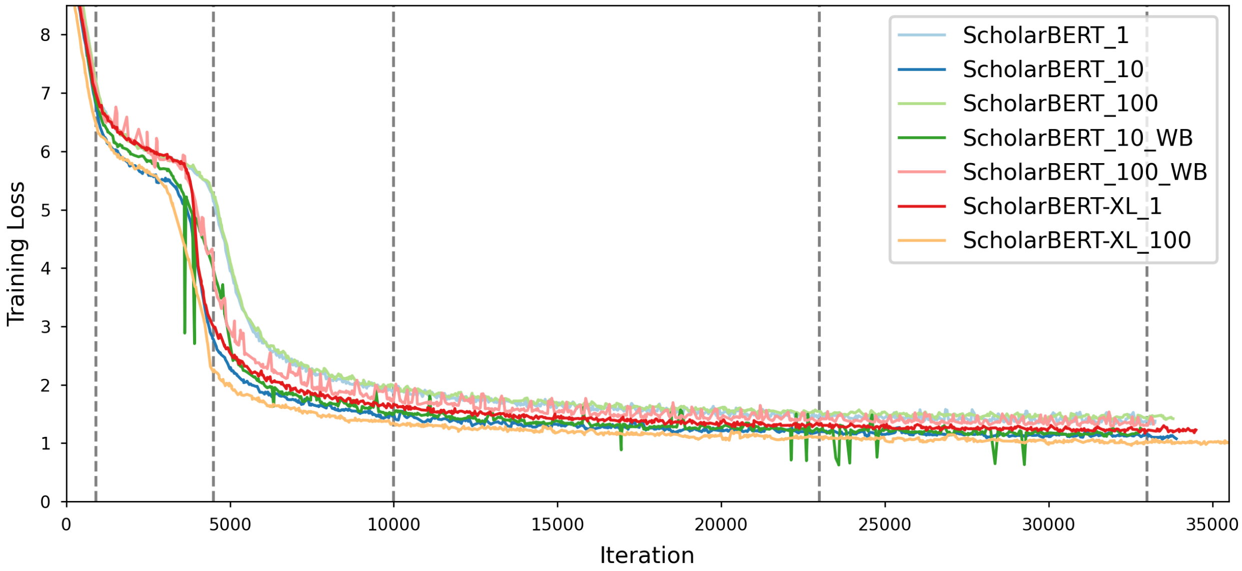


Figure 5.1: Pretraining loss plots for the ScholarBERT models listed in Table 5.5. The vertical dashed lines indicate the approximate locations of the iteration checkpoints selected for evaluation.

Tokenization

The vocabularies generated for PRD_1 and PRD_10 differed only in 1–2% of the tokens; however, in an initial study, the PRD_100 vocabulary differed from that of PRD_10 by 15%. A manual inspection of the PRD_100 vocabulary revealed that many common English words such as "is," "for," and "the" were missing. We determined that these omissions were an artifact of PRD_100 being sufficiently large to cause integer overflows in the unsigned 32-bit-integer token frequency counts used by HuggingFace's tokenizers library. For example, "the" was not in the final vocabulary because the token "th" overflowed. Because WordPiece

Table 5.2: Pretraining corpora used by models in this study. The domains are Bio=biomedicine, CS=computer science, Gen=general, Materials=materials science and engineering and Sci=broad scientific.

Name	**Description**	**Domain**	**Tokens**
`Wiki`	English-language Wikipedia articles [51]	Gen	2.5B
`Books`	BookCorpus [141, 51]: Full text of 11038 books	Gen	0.8B
`SemSchol`	1.14M papers from Semantic Scholar [19], 18% in CS, 82% in Bio	Bio, CS	3.1B
$\texttt{PubMed}_\texttt{A}$	Biomedical abstracts sampled from PubMed [41]	Bio	3.1B
$\texttt{PubMed}_\texttt{B}$	Biomedical abstracts sampled from PubMed [64]	Bio	4.5B
`PMC`	Full-text biomedical articles sampled from PubMedCentral [41]	Bio	13.7B
`MatSci`	2M peer-reviewed materials science journal articles [125]	Materials	8.8B
`Battery`	0.4M battery-related publications [49]	Materials	5.2B
`PRD_1`	1% of the English-language research articles from the Public Resource dataset	Sci	2.2B
`PRD_10`	10% of the English-language research articles from the Public Resource dataset	Sci	22B
`PRD_100`	100% of the English-language research articles from the Public Resource dataset	Sci	221B

iteratively merges smaller tokens to create larger ones, the absence of tokens like "th" or "##he" means that "the" could not appear in the final vocabulary.

We modified the tokenizers library to use unsigned 64-bit integers for all frequency counts, and recreated a correct vocabulary for PRD_100. Interestingly, models trained on the PRD_100 subset with the incorrect and correct vocabularies exhibited comparable performance on downstream tasks.

BERT Pretraining

BERT pre-training is performed in two phases that train with maximum sequence lengths of 128 and 512, respectively [25]. Typically, the first phase is used for the majority of pre-training steps (90% in the original BERT) because longer sequences are more expensive due to the quadratic complexity of the attention mechanism. WordPiece embedding is used with a vocabulary of 30 000. The BERT pre-training processes optimizes for two tasks, masked language modeling (MLM) and next sentence prediction (NSP). In MLM, a small percentage, generally 15%, of tokens in the input sequence are randomly masked with the `[MASK]` token and the model is asked to predict the masked tokens. In downstream tasks, however, `[MASK]` is not a token that appears, so for the tokens selected for masking, 80% are actually masked with the `[MASK]` token, 10% are replaced by a random token in the vocabulary, and 10% are not changed. Each input sequence contains two sentences where there is a 50% chance the second sentence is the actual next sentence in the document otherwise the second sentence is a random sentence from the training corpus. For the NSP tasks, the model must predict if the second sentence is the actual next sentence or a random one. The NSP task aims to improve performance in downstream tasks that use the relationship between sentences, such as question answering.

RoBERTa Optimizations

RoBERTa introduces the following optimizations for improving BERT pretraining performance [70]. 1) It uses a single phase training approach whereby all training is performed with a maximum sequence length of 512. 2) Unlike BERT which randomly introduces a small percentage of shortened sequence lengths into the training data, RoBERTa does not randomly use shortened sequences. 3) RoBERTa uses dynamic masking, meaning that each time a batch of training samples is selected at runtime, a new random set of masked tokens is selected; in contrast, BERT uses static masking, pre-masking the training samples prior to training. BERT duplicates the training data 10 times each with a different random, static masking. 4) RoBERTa does not perform Next Sentence Prediction during training. 5) RoBERTa takes sentences contiguously from one or more documents until the maximum sequence length is met. 6) RoBERTa uses a larger batch size of 8192. 7) RoBERTa uses byte-pair encoding (BPE) rather than WordPiece. 8) RoBERTa uses an increased vocabulary size of 50 000, 67% larger than BERT. 9) RoBERTa trains for more iterations (up to 500 000) than does BERT-Base (31 000).

We adopt RoBERTa training methods, with three key exceptions. 1) Unlike RoBERTa, we randomly introduce smaller length samples because many of our downstream tasks use sequence lengths much smaller than the maximum sequence length of 512 that we pretrain with. 2) We pack training samples with sentences drawn from a single document, as the RoBERTa authors note that this results in slightly better performance. 3) We use WordPiece encoding rather than BPE, as the RoBERTa authors note that BPE can result in slightly worse downstream performance.

Hardware and Software Stack

Table 5.3: Pretraining hyperparameters. All ScholarBERT variants use the same pretraining hyperparameters.

Hyperparameter	**Value**
Steps	33 000
Optimizer	LAMB
LR	0.0004
LR Decay	Linear
LR Warmup Steps	0.06%
Batch Size	32 768
Precision	FP16
Weight Decay	0.01
Attention Dropout	10%
Hidden Dropout	10%
Hidden Activation	GELU

We perform data-parallel pretraining on a cluster with 24 nodes, each containing eight 40 GB NVIDIA A100 GPUs. In data-parallel distributed training, a copy of the model is replicated on each GPU, and, in each iteration, each GPU computes on a unique local mini-batch. At the end of the iteration, the local gradients of each model replica are averaged to keep each model replica in sync. We perform data-parallel training of ScholarBERT models using PyTorch's distributed data-parallel model wrapper and 16 A100 GPUs. For the larger ScholarBERT-XL models, we use the DeepSpeed data-parallel model wrapper and 32 A100 GPUs. The DeepSpeed library incorporates a number of optimizations that improve training time and reduced memory usage, enabling us to train the larger model in roughly the same amount of time as the smaller model.

We train in FP16 with a batch size of 32 768 for ∼33 000 iterations (Table 5.3). To achieve training with larger batch sizes, we employ NVIDIA Apex's FusedLAMB [86] optimizer, with an initial learning rate of 0.0004. The learning rate is warmed up for the first 6% of iterations and then linearly decayed for the remaining iterations. We use the same masked token percentages as are used for BERT. Training each model requires roughly 1000 node-hours, or 8000 GPU-hours.

Fig. 5.1 depicts the pretraining loss for each ScholarBERT model. We train each model past the point of convergence and take checkpoints throughout training to evaluate model performance as a function of training time.

5.2 Vector space analysis of models prior to fine tuning

Fine-tuning steps are necessary for adapting models to various downstream tasks. However, these steps also modify the models by introducing new, out-of-sample data that were not considered during pre-training. We designed experiments to evaluate the extent to which BERT-based model representations of scientific and scholastic contexts match our intuitions about the relationship between entities and disciplines.

We introduce a novel method for performing this evaluation. This method leverages vector representations of entities generated by BERT-based models prior to fine tuning. For a particular model, we can generate such vectors for all entities that appear in the training corpus. The resulting vector space is then the given model's representation of that scientific literature. Similarly, the vectors for entities *for a particular scientific domain* (e.g., biomedicine) constitute the model's representation of that domain. Furthermore, the average of those vectors yields a *domain vector* for that domain.

As described by Kozlowski et al. [60], a *difference vector* can be constructed for any two domains by subtracting one domain vector from the other. We can then evaluate the model's understanding of scientific entities by projecting each entity vector in the two domains onto the difference vector. If the projected vectors clustered around their respective end of the difference vector, then we can conclude that the the model's representation accords with our scientific and scholastic intuitions. Compared to traditional dimension reduction and clustering methods, such as PCA with K-Means, this evaluation method is more accurate because the semantics of the reduced dimension (i.e., the difference vector) is known to us, so that it is possible to evaluate not only whether the model can tell entities from different domains apart, but also whether the model maps each entity to the correct domain.

We now provide details on how we construct and apply the entity vectors, domain vectors, and difference vectors referred to above.

We extract, from the pre-trained BERT models without fine-tuning, an *entity vector* for each named entity in annotated NER datasets in Biomedical, Chemistry, Materials Science, Computer Science, and Social Science. We generate each such entity vector in a hierarchical fashion. First, we generate a *word vector* for each token in the text corpus. A variety of methods have been used to extract "word vectors" from BERT: for example, by using the average of the last two to four hidden layers [25]. In our experiments, we take the average of the last four hidden layers as the vector for a token, yielding vectors of dimension 1024.

Given that BERT vectors are contextual by design, different occurrences of the same token will often be represented by distinct word vectors. Thus, for each occurrence of a specific named entity, we generate an *entity occurrence vector* by averaging the word vectors of the one or more tokens that make up the named entity. We average the entity occurrence vectors for the zero or more occurrences of the named entity in the corpus to obtain the final *entity vector*. In the end, a total of 34 019 biomedical, 19 980 chemistry, 2508 computer science, and 120 sociology entity vectors are collected.

Having computed entity vectors for the entities, we then average the entity vectors for appropriate subsets of the associated entities to create a *domain vector* for each of four domains: Biomedicine (BC5CDR + JNLPBA + NCBI-Disease), Computer science (SciERC), Chemistry (ChemdNER), and Sociology (Coleridge).

To evaluate the representational fidelity of the various BERT-based models, we draw inspiration from Kozlowski et al. [60], who evaluate cultural associations of named entities in large corpora encoded in word embedding models. Analogously, in this paper we evaluate the domain associations of named entities. To do this, we construct a *difference vector* for each pair of domains, A and B, as follows:

$$\overrightarrow{diff_{A,B}} = \overrightarrow{domainA} - \overrightarrow{domainB} \tag{5.1}$$

The high-dimensional *entity vectors* and *domain vectors* that we obtain from our BERT models make it difficult to visualize and compare the representations of different domains, but the proximity of any pair of vectors can be represented by the angle between them. All word vectors are normalized so that they lie on the surface of the same hypersphere, and their distance on the hypersphere, a scalar, is proportional to the the angle between them [66, 60]. This reducing of proximity to a scalar value permits visualization and evaluation of differences between domains, as we show in the following.

For each difference vector $\mathtt{d} = \overrightarrow{diff_{A,B}}$ for a pair of domains A and B, we then compute, for each entity in A and B, a vector projection of the entity's entity vector $\mathbf{e}$ onto $\mathbf{d}$:

$$\mathrm{proj}_{\mathbf{de}} = \left(\frac{\mathbf{e} \cdot \mathbf{d}}{|\mathbf{d}|^2}\right)\mathbf{d} = \left(\frac{\mathbf{e} \cdot \mathbf{d}}{|\mathbf{d}|}\right)\frac{\mathbf{d}}{|\mathbf{d}|}$$

This projection produces a vector with dimension 1024. Importantly, for a given $\mathbf{d}$, the vectors produced via this projection for all entities point in the same direction such that only their magnitudes, defined as follows, differ:

$$\mathrm{proj\_scalar}_{\mathbf{de}} = \frac{\mathbf{e} \cdot \mathbf{d}}{|\mathbf{d}|}$$

All vectors are then normalized so that they are on the same hypersphere, and the scalar projection of two vectors is equal to their cosine similarity such that:

$$\text{similarity}(\mathbf{d}, \mathbf{e}) = \frac{\mathbf{d} \cdot \mathbf{e}}{|\mathbf{d}| \times |\mathbf{e}|}$$

Figures 5.2 display results obtained when this method is applied to each of the 12 domain pairs. In each case, we expect that if the model represents domain entity associations with fidelity to our scientific intuitions, then domain A entities should fall on the positive range, whereas the entities in domain B should fall on the negative. (The opposite would be the case if domain vectors A and B are switched in computing the difference vector.)

One reasonable concern might be that, because a difference vector $\mathbf{d}$ is computed as the average of a set that includes the entity vectors, the position of each entity vector $\mathbf{e}$ on $\mathtt{d}$ would be a function of its distance from itself. To avoid this concern, we define an alternative difference vector, $\overrightarrow{\text{diff}_{A,B,e}}$, that removes the entity being projected from the difference vector calculation so that the difference vector is independent of the current entity vector:

$$\overrightarrow{\text{diff}_{A,B,e}} = \begin{cases} \text{mean}\left(\left\{\overrightarrow{\text{DomainA}}\right\} \setminus \{\overrightarrow{\text{e}}\}\right) - \text{mean}\left(\left\{\overrightarrow{\text{DomainB}}\right\}\right), \text{ if e} \in \text{DomainA} \\ \text{mean}\left(\left\{\overrightarrow{\text{DomainA}}\right\}\right) - \text{mean}\left(\left\{\overrightarrow{\text{DomainB}}\right\} \setminus \{\overrightarrow{\text{e}}\}\right), \text{ if e} \in \text{DomainB} \end{cases} \tag{5.2}$$

We show results obtained with this alternative definition in Figures 5.3. For ScholarBERT models, we see similar results to Figures 5.2, which is intuitive since if all vectors in a domain point largely to the same direction, then removing one vector will not significantly move the average vector and thus the difference vector and the projection results will stay about the same. For other models, the dots in Figures 5.3 are shifted by some distance compared to Figures 5.2. This happens when different vectors in a domain point in different directions; thus removing one vector moves the average vector by some degree and thus shifts the projection (i.e., cosine similarity between each vector and the difference vector). This

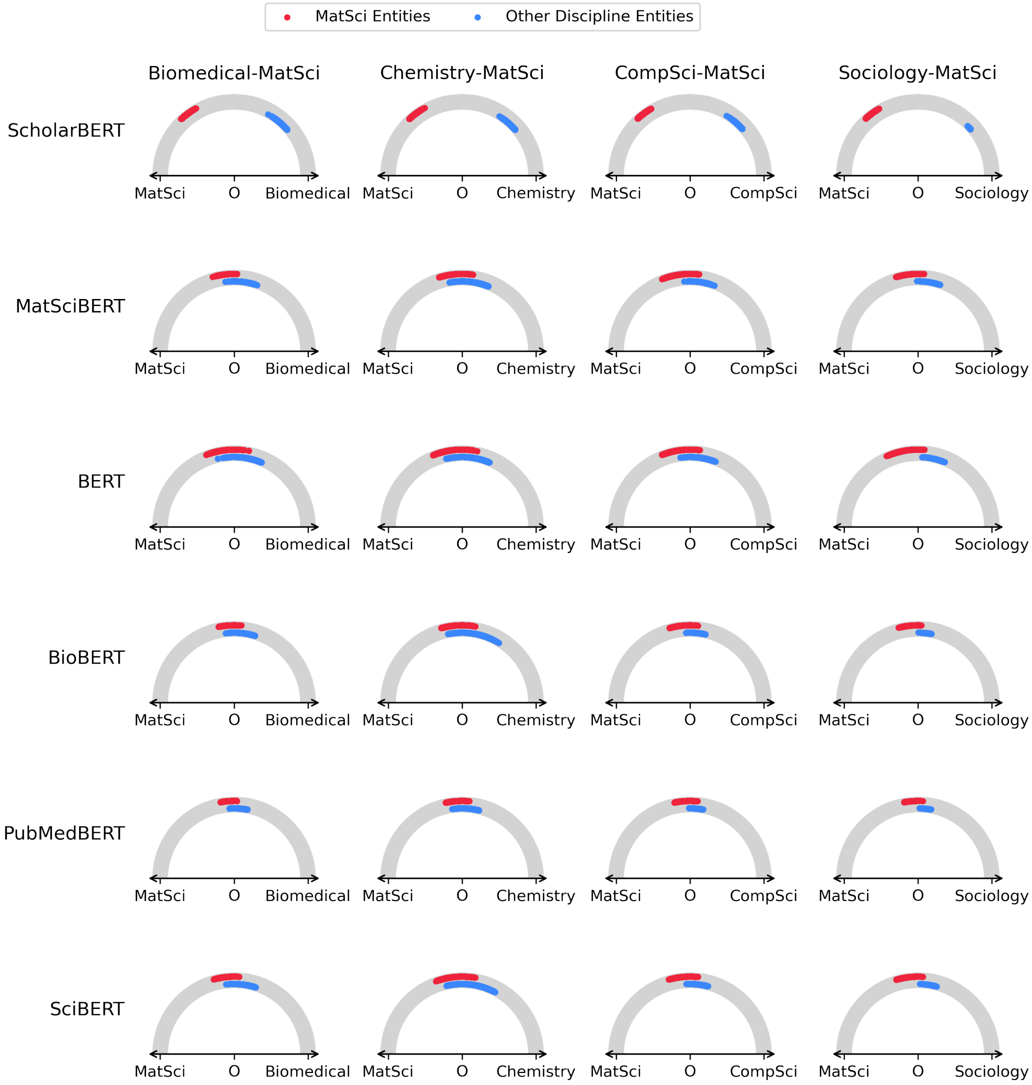


Figure 5.2: Difference vector = avg(domainA entities) – avg(chemistry entities)

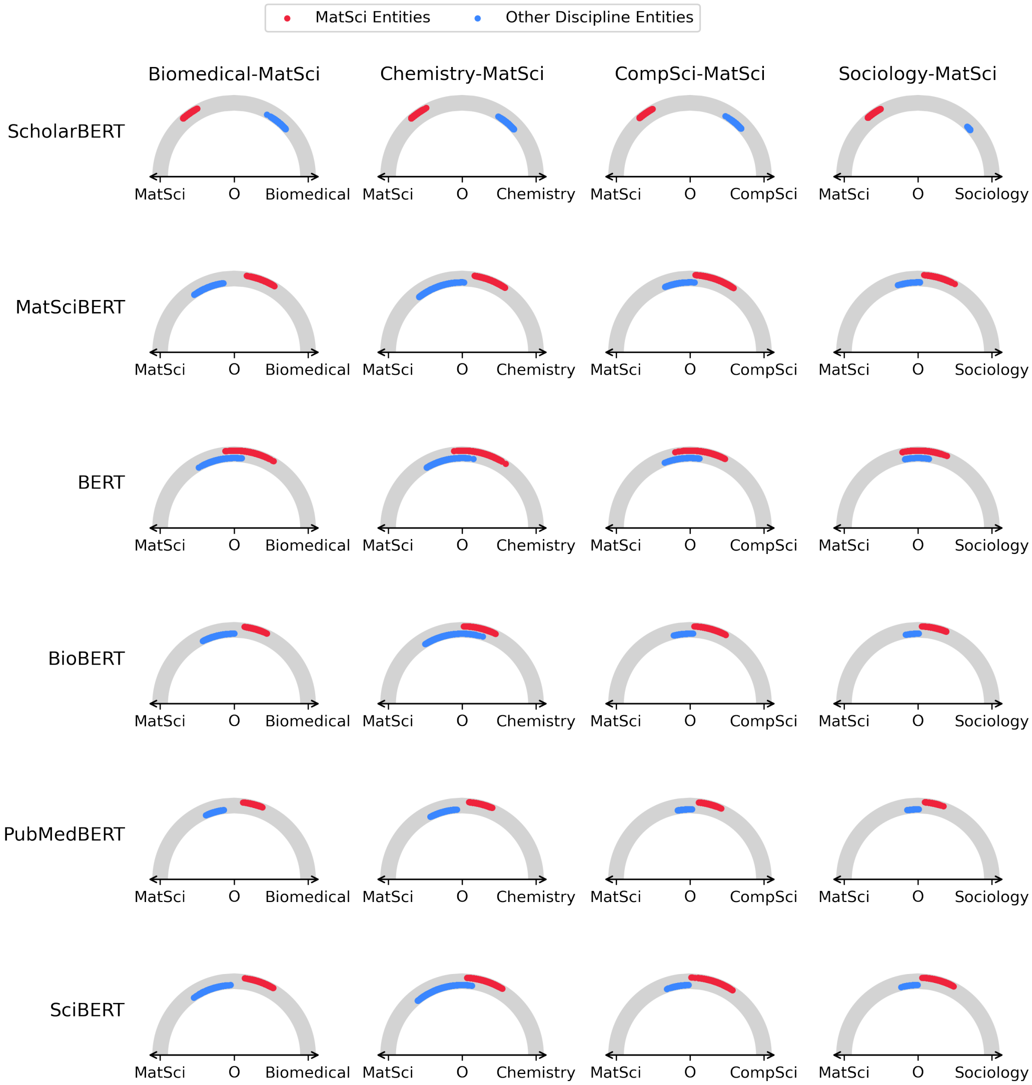


Figure 5.3: As Fig. 5.2, except that the similarity for each entity e is with the difference vector $\overrightarrow{diff_{X,\ Chemistry,\ e}}$ as computed with Eq. 5.2, i.e., one in which e is not considered.

comparison further demonstrates the advantage of ScholarBERT models in understanding and creating proper representations for scientific entities.

Upon further investigation of the various training parameters of ScholarBERT, we have discovered that the selection of the optimizer plays a vital role in the model's ability to discern the entity domains. Replacing the LAMB optimizer with the traditional Adam optimizer led to significantly decreased separations between entities in different domains. The discovery lends further support to the notion that LAMB is better than Adam or AdamW for training large models and demonstrates the importance of appropriate parameter selection in training an effective LLM.

5.3 ScholarBERT Performance on Downstream Scientific Tasks

We first perform sensitivity analysis across ScholarBERT pretraining dimensions to determine the trade-off between time spent in pretraining versus finetuning. We also compare the downstream task performance of SCHOLARBERT to that achieved with other BERT models. Details of each evaluation task are in Section 5.3.2.

5.3.1 Sensitivity Analysis

We save checkpoints periodically while pretraining each SCHOLARBERT(-XL) model. In this analysis, we checkpoint at ∼0.9k, 5k, 10k, 23k, and 33k iterations based on the decrease of training loss between iterations. We observe that pretraining loss decreases rapidly until around 10 000 iterations, which marks the end of the "rapid learning" phase. further training to convergence (roughly 33 000 iterations) yields only small decreases of training loss: see Fig. 5.1.

To measure how downstream task performance is impacted by pretraining and finetuning time, we finetune each of the checkpointed models for 5 and 75 epochs. We observe that: (1) The under-trained 0.9k-iteration model sees the biggest boost in the F1 scores of downstream tasks (+8%) with more finetuning, but even with 75 epochs of finetuning the 0.9k-iteration models' average F1 score is still 19.9 percentage points less than that of the 33k-iteration model with 5 epochs of finetuning. (2) For subsequent checkpoints, the performance gains from more finetuning decreases as the number of pretraining iterations increases. The average downstream task performance of the 33k-iteration model is only 0.39 percentage points higher with 75 epochs of finetuning than with 5 epochs. Therefore, in the remaining experiments, we use the ScholarBERT(-XL) model that was pretrained for 33k iterations and finetuned for 5 epochs.

Table 5.4: NER F1 scores for each model. Models are finetuned five times for each dataset and the average result is presented. Underlined results represent the F1-scores of models trained on in-distribution data for the given task, and bolded results indicate the best performing model on that task. `SB` = ScholarBERT. Task domain: Biomedical = {BC5CDR, JNLPBA, NCBI-Disease, ChemDNER}, CS = {SciERC}, Materials = {MatSciNER}, Sociology = {Coleridge}, Multi-Domain = {ScienceExam}.

Domain	**Biomedical**				**CS**	**Materials**	**Multi-Domain**	**Sociology**	
Dataset	**BC5CDR**	**JNLPBA**	**NCBI-Disease**	**ChemDNER**	**SciERC**	**MatSciNER**	**ScienceExam**	**Coleridge**	**Mean**
`BERT-Base`	85.36	72.15	84.28	84.84	56.73	78.51	78.37	57.75	74.75
`BERT-Large`	86.86	72.80	84.91	85.83	59.20	82.16	82.32	57.46	76.44
`SciBERT`	88.43	73.24	86.95	85.76	**59.36**	82.64	78.83	54.07	76.16
`PubMedBERT`	**89.34**	**74.53**	**87.91**	**87.96**	59.03	82.63	69.73	57.71	76.11
`BioBERT`	88.01	73.09	87.84	85.53	58.24	81.76	78.60	57.04	76.26
`MatBERT`	86.44	72.56	84.94	86.09	58.52	**83.35**	80.01	56.91	76.10
`BatteryBERT`	87.42	72.78	87.04	86.49	59.00	82.94	78.14	**59.87**	**76.71**
`SB_1`	87.27	73.06	85.49	85.25	58.62	80.87	82.75	55.34	76.08
`SB_10`	87.69	73.03	85.65	85.80	58.39	80.61	**83.24**	53.41	75.98
`SB_100`	87.84	73.47	85.92	85.90	58.37	82.09	83.12	54.93	76.46
`SB_10_WB`	86.68	72.67	84.51	83.94	57.34	78.98	83.00	54.29	75.18
`SB_100_WB`	86.89	73.16	84.88	84.31	58.43	80.84	82.43	54.00	75.62
`SB-XL_1`	87.09	73.14	84.61	85.81	58.45	82.84	81.09	55.94	76.12
`SB-XL_100`	87.46	73.25	84.73	85.73	57.26	81.75	80.72	54.54	75.68

5.3.2 Finetuning

We finetuned the ScholarBERT models and the state-of-the-art scientific models listed in Table 5.5 on NER, relation extraction, and sentence classification tasks.

The extant BERT-based models used for evaluations include:

1. BERT-Base and BERT-Large [25], with ~110M and ~340M parameters, as transformer-based masked language models conditioned on both the left and right contexts. Both are pretrained on the English Wikipedia + BooksCorpus datasets.

Table 5.5: Characteristics of the 14 BERT models considered in this study. The BERT-Base architecture has 12 layers, hidden size of 768, and 12 heads; BERT-Large has 24 layers, hidden size of 1024, and 16 heads [25]; the BERT-XL architecture has 36 layers, hidden size of 1280, and 20 heads. Details of the pretraining corpora are in Table 5.2. The domains are Bio=biomedicine, CS=computer science, Gen=general, Mat=materials science and engineering, and Sci=broad scientific.

Model	**Architecture**	**Pretraining Method**	**Casing**	**Pretraining Corpus**	**Domain**	**Tokens**
BERT_Base	BERT-Base	BERT	Cased	Wiki + Books	Gen	3.3B
SciBERT	BERT-Base	BERT	Cased	SemSchol	Bio, CS	3.1B
PubMedBERT	BERT-Base	BERT	Uncased	PubMed$_\text{A}$ + PMC	Bio	16.8B
BioBERT_1.2	BERT-Base	BERT	Cased	PubMed$_\text{B}$ + Wiki + Books	Bio, Gen	7.8B
MatBERT	BERT-Base	BERT	Cased	MatSci	Mat	8.8B
BatteryBERT	BERT-Base	BERT	Cased	Battery	Mat	5.2B
BERT_Large	BERT-Large	BERT	Cased	Wiki + Books	Gen	3.3B
ScholarBERT_1	BERT-Large	RoBERTa-like	Cased	PRD_1	Sci	2.2B
ScholarBERT_10	BERT-Large	RoBERTa-like	Cased	PRD_10	Sci	22B
ScholarBERT_100	BERT-Large	RoBERTa-like	Cased	PRD_100	Sci	221B
ScholarBERT_10_WB	BERT-Large	RoBERTa-like	Cased	PRD_10 + Wiki + Books	Sci, Gen	25.3B
ScholarBERT_100_WB	BERT-Large	RoBERTa-like	Cased	PRD_100 + Wiki + Books	Sci, Gen	224.3B
ScholarBERT-XL_1	BERT-XL	RoBERTa-like	Cased	PRD_1	Sci	2.2B
ScholarBERT-XL_100	BERT-XL	RoBERTa-like	Cased	PRD_100	Sci	221B

2. SciBERT [9] follows the BERT-Base architecture and is pretrained on data from two domains, namely, biomedical science and computer science. SciBERT outperforms BERT-Base on finetuning tasks by an average of 1.66% and 3.55% on biomedical tasks and computer science tasks, respectively.

3. BioBERT [64] is a BERT-Base model with a pretraining corpus from PubMed abstracts and full-text PubMedCentral articles. Compared to BERT-Base, BioBERT achieves improvements of 0.62%, 2.80%, and 12.24% on biomedical NER, biomedical relation extraction, and biomedical question answering, respectively.

4. PubMedBERT [41], another BERT-Base model targeting the biomedical domain, is also pretrained on PubMed and PubMedCentral text. However, unlike BioBERT, PubMedBERT is trained as a new BERT-Base model, using text drawn exclusively from PubMed and PubMedCentral. As a result, the vocabulary used in PubMedBERT varies significantly from that used in BERT and BioBERT. Its pretraining corpus contains 3.1B words from PubMed abstracts and 13.7B words from PubMedCentral articles. PubMedBERT achieves state-of-the-art performance on the Biomedical Language Un-

derstanding and Reasoning Benchmark, outperforming BERT-Base by 1.16% [41].

5. MatBERT [125] is a materials science-specific model pretrained on 2M journal articles (8.8B tokens). It consistently outperforms BERT-Base and SciBERT in recognizing materials science entities related to solid states, doped materials, and gold nanoparticles, with ∼10% increase in F1 score compared to BERT-Base, and a 1% to 2% improvement compared to SciBERT.

6. BatteryBERT [49] is a model pretrained on 400 366 battery-related publications (5.2B tokens). BatteryBERT has been shown to outperform BERT-Base by less than 1% on the SQuAD question answering task. For battery-specific question-answering tasks, its F1 score is around 5% higher than that of BERT-base.

NER is a common token-level task in SciIE. The NER tasks used for evaluations are listed below.

1. BC5CDR [67]: An NER dataset identifying diseases, chemicals, and their interactions, generated from the abstracts of 1500 PubMed articles containing 4409 annotated chemicals, 5818 diseases, and 3116 chemical-disease interactions, totaling 6283 unique entities.

2. JNLPBA [59]: A bio-entity recognition dataset of molecular biology concepts from 2404 MEDLINE abstracts, consisting of 21 800 unique entities.

3. SciERC [72]: A dataset annotating entities, relations, and coreference clusters in 500 abstracts from 12 AI conference/workshop proceedings. It contains 5714 distinct named entities.

4. NCBI-Disease [28]: Annotations for 793 PubMed abstracts: 6893 disease mentions, of which 2134 are unique.

5. ChemDNER [61]: A chemical entity recognition dataset derived from 10 000 abstracts containing 19 980 unique chemical entity mentions.

6. MatSciNER [125]: 800 annotated abstracts from solid state materials publications sourced via Elsevier's Scopus/ScienceDirect, Springer-Nature, Royal Society of Chemistry, and Electrochemical Society. Seven types of entities are labeled: inorganic materials (MAT), symmetry/phase labels (SPL), sample descriptors (DSC), material properties (PRO), material applications (APL), synthesis methods (SMT), and characterization methods (CMT).

7. ScienceExam [111]: 133K entities from the Aristo Reasoning Challenge Corpus of 3rd to 9th grade science exam questions.

8. Coleridge [21]: 13 588 entities from sociology articles indexed by the Inter-university Consortium for Political and Social Research (ICPSR).

The sentence-level downstream tasks are relation extraction on the ChemProt (biology) and SciERC (computer science) datasets, and sentence classification on the Paper Field (multidisciplinary) and Battery (materials) dataset:

1. ChemProt consists of 1820 PubMed abstracts with chemical-protein interactions annotated by domain experts [91].

2. SciERC, introduced above, provides 4716 relations [72].

3. The Paper Field dataset [9], built from the Microsoft Academic Graph [110], maps paper titles to one of seven fields of study (geography, politics, economics, business, sociology, medicine, and psychology), with each field of study having around 12K training examples.

4. The Battery Document Classification dataset [49] includes 46 663 paper abstracts, of which 29 472 are labeled as battery and the other 17 191 as non-battery. The labeling is performed in a semi-automated manner. Abstracts are selected from 14 battery journals and 1044 non-battery journals, with the former labeled "battery" and the latter "non-battery."

F1 scores for each model-task pair, averaged over five runs, are shown in Tables 5.4 and 5.6. For NER tasks, we use the CoNLL NER evaluation Perl script [106] to compute F1 scores for each test.

Table 5.6: F1 scores for each model on Relation Extraction (SciERC, ChemProt) and Sentence Classification (PaperField, Battery) tasks. Models are finetuned five times for each dataset and the average result is presented. Underlined results represent the F1-scores of models trained on in-distribution data for the given task, and bolded results indicate the best performing model on that task. `SB` = ScholarBERT. Task domain: Biomedical = {ChemProt}, CS = {SciERC}, Materials = {Battery}, Multi-Domain = {Paperfield}.

Domain	**CS**	**Biomedical**	**Multi-Domain**	**Materials**	
Dataset	**SciERC**	**ChemProt**	**PaperField**	**Battery**	**Mean**
`BERT-Base`	74.95	83.70	72.83	96.31	81.95
`BERT-Large`	80.14	88.06	73.12	**96.90**	84.56
`SciBERT`	79.26	89.80	73.19	96.38	84.66
`PubMedBERT`	77.45	**91.78**	+73.93-	96.58	**84.94**
`BioBERT`	80.12	89.27	73.07	96.06	84.63
`MatBERT`	79.85	88.15	71.50	96.33	83.96
`BatteryBERT`	78.14	88.33	73.28	96.06	83.95
`ScholarBERT_1`	73.01	83.04	72.77	94.67	80.87
`ScholarBERT_10`	75.95	82.92	72.94	92.83	81.16
`ScholarBERT_100`	76.19	87.60	73.14	92.38	82.33
`ScholarBERT_10_WB`	73.17	81.48	72.37	93.15	80.04
`ScholarBERT_100_WB`	76.71	83.98	72.29	95.55	82.13
`ScholarBERT-XL_1`	74.85	90.60	73.22	88.75	81.86
`ScholarBERT-XL_100`	**80.99**	89.18	73.66	95.44	84.82

Tables 5.4 and 5.6 show the results, from which we observe: (1) With the same training data, a larger model does not always achieve significant performance improvements. BERT-Base achieved F1 scores within 1 percentage point of BERT-Large on 6/12 tasks; SB_1 achieved F1 scores within 1 percentage point of SB-XL_1 on 7/12 tasks; SB_100 achieved F1 scores within 1 percentage point of SB-XL_100 on 6/12 tasks. (2) With the same model size, a model pretrained on more data cannot guarantee significant performance improvements. SB_1 achieved F1 scores within 1 percentage point of SB_100 on 8/12 tasks; SB_10_WB achieved F1 scores within 1 percentage point of SB_100_WB on 7/12 tasks; SB-XL_1 achieved F1 scores within 1 percentage point of SB-XL_100 on 10/12 tasks. (3) Domain-specific pretraining cannot guarantee significant performance improvements. The Biomedical domain is the only domain where we see the on-domain model (i.e., pretrained for the associated

domain; marked with underlines; in this case, PubMedBERT) consistently outperformed models pretrained on off-domain or more general corpora by more than 1 percentage point F1. The same cannot be said for CS, Materials, or Multi-Domain tasks.

Table 5.7 shows average F1 scores with standard deviations for the NER tasks, each computed over five runs; Fig. 5.4 presents the same data, with standard deviations represented by error bars. Table 5.8 and Fig. 5.5 show the same for sentence classification tasks. The significant overlaps of error bars for NCBI-Disease, SciERC NER, Coleridge, SciERC Sentence Classification, and ChemProt corroborate our observation in Section 5.3 that on-domain pretraining provides only marginal advantage for downstream prediction over pretraining on a different domain or a general corpus.

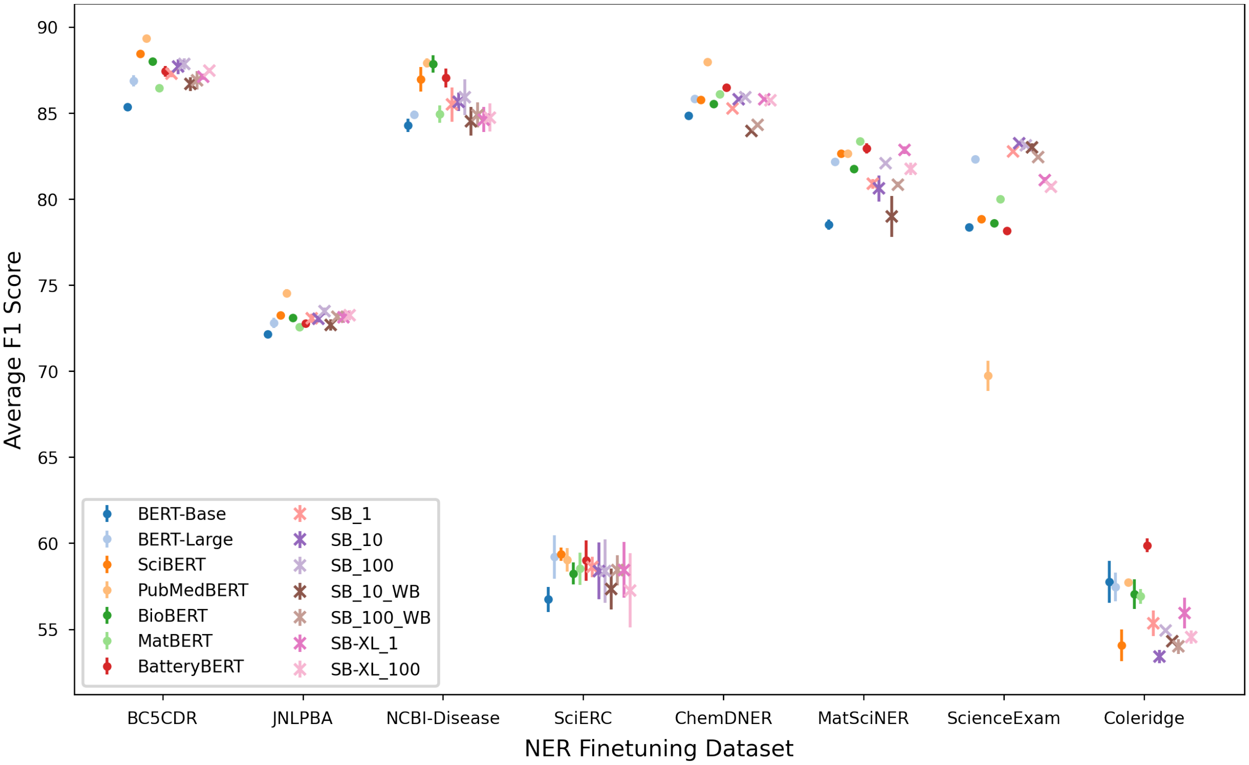


Figure 5.4: NER F1 scores with standard deviations represented by error bars.

Table 5.7: NER F1 scores for each of 14 models (rows), when the model is finetuned on eight different domain datasets and the resulting finetuned model applied to that dataset's associated NER task (columns). In each case, we give the average value and its standard deviation over five runs.

	BC5CDR	**JNLPBA**	**NCBI-Disease**	**SciERC**
`BERT-Base`	85.36 ± 0.189	72.15 ± 0.118	84.28 ± 0.388	56.73 ± 0.716
`BERT-Large`	86.86 ± 0.321	72.80 ± 0.299	84.91 ± 0.229	59.20 ± 1.260
`SciBERT`	88.43 ± 0.112	73.24 ± 0.184	86.95 ± 0.714	59.36 ± 0.390
`PubMedBERT`	89.34 ± 0.185	74.53 ± 0.220	87.91 ± 0.267	59.03 ± 0.688
`BioBERT`	88.01 ± 0.133	73.09 ± 0.230	87.84 ± 0.513	58.24 ± 0.631
`MatBERT`	86.44 ± 0.156	72.56 ± 0.162	84.94 ± 0.504	58.52 ± 0.933
`BatteryBERT`	87.42 ± 0.308	72.78 ± 0.190	87.04 ± 0.553	59.00 ± 1.174
`ScholarBERT_1`	87.27 ± 0.189	73.06 ± 0.265	85.49 ± 0.998	58.62 ± 0.602
`ScholarBERT_10`	87.69 ± 0.433	73.03 ± 0.187	85.65 ± 0.544	58.39 ± 1.643
`ScholarBERT_100`	87.84 ± 0.329	73.47 ± 0.210	85.92 ± 1.040	58.37 ± 1.845
`ScholarBERT_10_WB`	86.68 ± 0.397	72.67 ± 0.329	84.51 ± 0.838	57.34 ± 1.199
`ScholarBERT_100_WB`	86.89 ± 0.543	73.16 ± 0.211	84.88 ± 0.729	58.43 ± 0.881
`ScholarBERT-XL_1`	87.09 ± 0.179	73.14 ± 0.352	84.61 ± 0.730	58.45 ± 1.614
`ScholarBERT-XL_100`	87.46 ± 0.142	73.25 ± 0.300	84.73 ± 0.817	57.26 ± 2.146

	ChemDNER	**MatSciNER**	**ScienceExam**	**Coleridge**
`BERT-Base`	84.84 ± 0.004	78.51 ± 0.300	78.37 ± 0.004	57.75 ± 1.230
`BERT-Large`	85.83 ± 0.022	82.16 ± 0.040	82.32 ± 0.072	57.46 ± 0.818
`SciBERT`	85.76 ± 0.089	82.64 ± 0.054	78.83 ± 0.004	54.07 ± 0.930
`PubMedBERT`	87.96 ± 0.094	82.63 ± 0.045	69.73 ± 0.872	57.71 ± 0.107
`BioBERT`	85.53 ± 0.130	81.76 ± 0.094	78.60 ± 0.072	57.04 ± 0.868
`MatBERT`	86.09 ± 0.170	83.35 ± 0.085	80.01 ± 0.027	56.91 ± 0.434
`BatteryBERT`	86.49 ± 0.085	82.94 ± 0.309	78.14 ± 0.103	59.87 ± 0.398
`ScholarBERT_1`	85.25 ± 0.063	80.87 ± 0.282	82.75 ± 0.049	55.34 ± 0.742
`ScholarBERT_10`	85.80 ± 0.094	80.61 ± 0.747	83.24 ± 0.063	53.41 ± 0.380
`ScholarBERT_100`	85.90 ± 0.063	82.09 ± 0.022	83.12 ± 0.085	54.93 ± 0.063
`ScholarBERT_10_WB`	83.94 ± 0.058	78.98 ± 1.190	83.00 ± 0.250	54.29 ± 0.080
`ScholarBERT_100_WB`	84.31 ± 0.080	80.84 ± 0.161	82.43 ± 0.031	54.00 ± 0.425
`ScholarBERT-XL_1`	85.81 ± 0.054	82.84 ± 0.228	81.09 ± 0.170	55.94 ± 0.899
`ScholarBERT-XL_100`	85.73 ± 0.058	81.75 ± 0.367	80.72 ± 0.174	54.54 ± 0.389

5.3.3 Discussion

Here we offer possible explanations for the three observations above. (1) The nature of the task is more indicative of task performance than the size of the model. In particular, with the same training data, a larger model size impacts performance only for relation extraction tasks, which consistently saw F1 scores increase by more than 1 percentage point when going

Table 5.8: Sentence classification F1 scores for each of 14 models (rows), when the model is finetuned on one of four different domain datasets and the finetuned model is applied to that dataset's associated sentence classification task (columns). In each case, we give the average value and its standard deviation over five runs.

	SciERC	**ChemProt**	**PaperField**	**Battery**
BERT-Base	74.95 ± 1.596	83.70 ± 0.472	72.83 ± 0.082	96.31 ± 0.087
BERT-Large	80.14 ± 2.266	88.06 ± 0.353	73.12 ± 0.125	96.90 ± 0.156
SciBERT	79.26 ± 0.498	89.80 ± 0.263	73.19 ± 0.046	96.38 ± 0.153
PubMedBERT	77.45 ± 0.964	91.78 ± 0.096	73.93 ± 0.099	96.58 ± 0.148
BioBERT	80.12 ± 0.179	89.27 ± 0.281	73.07 ± 0.074	96.06 ± 0.200
MatBERT	79.85 ± 0.121	88.15 ± 0.026	71.50 ± 0.135	96.33 ± 0.106
BatteryBERT	78.14 ± 0.550	88.33 ± 0.939	73.28 ± 0.022	96.06 ± 0.437
ScholarBERT_1	73.01 ± 0.248	83.04 ± 0.150	72.77 ± 0.060	94.67 ± 0.671
ScholarBERT_10	75.95 ± 0.203	82.92 ± 0.792	72.94 ± 0.182	92.83 ± 3.758
ScholarBERT_100	76.19 ± 1.592	87.60 ± 0.324	73.14 ± 0.085	92.38 ± 5.789
ScholarBERT_10_WB	73.17 ± 1.254	81.48 ± 1.705	72.37 ± 0.115	93.15 ± 1.763
ScholarBERT_100_WB	76.71 ± 2.114	83.98 ± 0.252	72.29 ± 0.048	95.55 ± 0.272
ScholarBERT-XL_1	74.85 ± 1.497	90.60 ± 0.246	73.22 ± 0.009	88.75 ± 4.035
ScholarBERT-XL_100	80.99 ± 0.900	89.18 ± 0.499	73.66 ± 0.113	95.44 ± 0.100

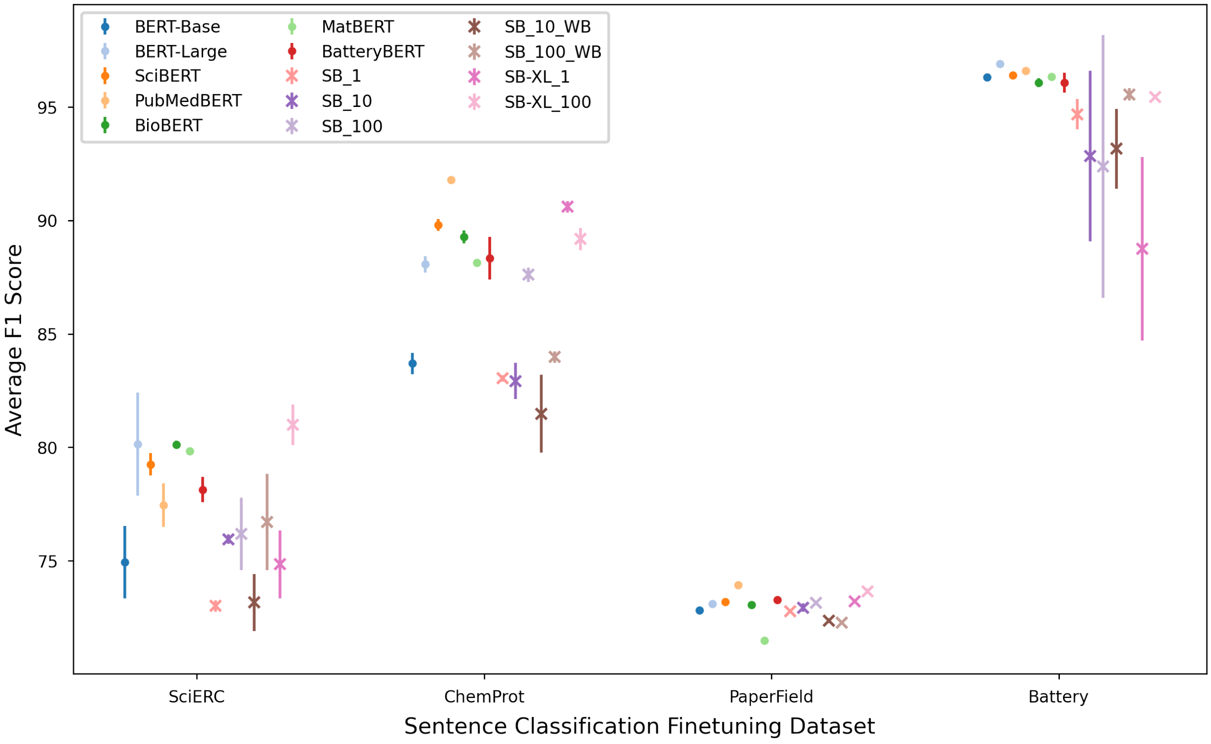


Figure 5.5: Sentence classification F1 scores from Table 5.8, with standard deviations represented by error bars.

from smaller models to larger models (i.e., BERT-Base to BERT-Large, SB_1 to SB-XL_1, SB_100 to SB-XL_100). In contrast, the NER and sentence classification tasks did not see such consistent significant improvements. (2) Our biggest model, ScholarBERT-XL, is only twice as large as the original BERT-Large, but its pretraining corpus is 100X larger. The training loss of the ScholarBERT-XL_100 model dropped rapidly only in the first ∼10k iterations (Fig. 5.1), which covered the first 1/3 of the PRD corpus, thus it is possible that the PRD corpus can saturate even our biggest model. [55, 46]. (3) Finetuning can compensate for missing domain-specific knowledge in pretraining data. While pretraining language models on a specific domain can help learn domain-specific concepts, finetuning can also fill holes in the pretraining corpora's domain knowledge, as long as the pretraining corpus incorporates the characteristics specific to the finetuning dataset.

5.4 Combining ScholarBERT and Graph Models for Semi-structured Extraction

Knowledge bases (KBs), such as relational databases and knowledge graphs, are used in many applications to embed human knowledge, recommendation systems [29] and digital assistants [50, 140] are a few examples. However, building KBs manually is expensive, so extracting relations from semi-structured pages on the web can be a more efficient method for building extensive KBs [71]. The Internet Movie Database, for example, has over 10 million pages with detailed information on films and TV episodes [52]. In the science domain, online KBs such as PubMed [100], PolyInfo [90] and Polymer Property Predictor and Database [122] provide useful knowledge in their respective domains. These pages are "semi-structured" because they are automatically populated into an HTML template from a database, and the page structure conveys semantic information. Although the data is often presented in a homogeneous structure within each KB, extracting information from a large number of KBs is still a challenging task because of the heterogeneous structures defined by various KBs.

Relation extraction is the task of extracting triples of *(subject, predicate, object)* such as *(Polystyrene, glass transition temperature,* $100°C$*)* from texts. RE methods can be classified as either ClosedIE or OpenIE. ClosedIE methods only extract objects for pre-defined predicates, while OpenIE methods extract predicates from the text. However, OpenIE outputs can be noisy due to language ambiguity and variability. The "schema alignment" problem, or the difficulty of matching extracted predicates to predefined relations, has limited the use of OpenIE in production. Our goal is to solve schema alignment while extracting tuples. Existing methods rely on domain-specific training data or produce noisy outputs. We focus here on extracting targeted relations from semi-structured web pages given only a short description of the relation. Semi-structured pages are a rich resource of such information since they are most likely populated from a back-end database into a front-end template. amassed vast amounts of human knowledge as it tightly integrates into every aspect of our life, yet the data exists in a chaotic and heterogeneous manner. Extracting relations from semi-structured pages poses many challenges. Manual curation is laborious and costly. Automated methods rely on domain-specific training data or produce noisy outputs that require post-processing to reconcile. We present GraphScholarBERT, an open-domain information extraction method based on a joint graph and language model structure. GraphScholarBERT can generalize to previously unseen domains without additional data or training and produces only clean extraction results matched to the search keyword.

Large Language Models (LLMs) have shown exceptional performance on many IE tasks [25, 24, 102], but they are limited to unstructured text. In this section, we propose a novel OpenIE model, GraphScholarBERT, that combines the graph model's strength in recognizing structures with the LLM's ability to understand semantics. GraphScholarBERT takes a target relation keyword and a short textual description of the relation as inputs, and extracts ¡relation, value¿ tuples matching the target relation from semi-structured webpages in a zero-shot manner. GraphScholarBERT can scale to unknown relations without any retraining or finetuning, and it does the extraction and schema alignment in one step to produce clean results. Experiments show that GraphScholarBERT can improve extraction F1 scores by as much as 34.8% compared to previous work in a zero-shot domain and zero-shot website setting.

5.4.1 Problem Definition

$\mathcal{W}$ is a semi-structured webpage on an entity $\mathcal{T}$ (which we assume has already been identified, for example, from the `<title>` tag of an HTML webpage). Given $\mathcal{W}$ and a short textual description $\mathcal{D}$ of a target relation $\mathcal{K}$, extract all relations of $\mathcal{T}$ as tuples $\mathcal{V} = \{(r_1, v_1), (r_2, v_2), \ldots\}$ from $\mathcal{W}$, where r_i is a relation (predicate) that is or is a synonym of $\mathcal{K}$ and v_i is a value (object) corresponding to r_i.

5.4.2 Methodology

On a semi-structured webpage, the semantics and the structure of the text are equally important indicators of potential relations. Therefore, an effective solution must be capable of handling both parts equally well.

As shown in Fig. 5.6, GraphScholarBERT consists of two components: a graph model (left) and an LLM (right). The LLM takes the concatenation of $\mathcal{K}, \mathcal{D}, r_i, v_i$ as inputs. It is good at capturing the semantics of the text, but it also needs the layout of the webpage to predict if (r_i, v_i) is a valid pair. Therefore, we augment the LLM with a graph model, which takes a web page's Document Object Model (DOM) tree as input and extracts DOM features and visual features for every text node and supplies them to the classification head of the LLM. Text embeddings are not used in the graph model because its message propagation process would dilute and confuse the semantics of the text. Instead, the semantic features are used by the LLM in combination with the graph features to predict whether the extracted value v is valid for the extracted relation and whether the extracted relation matches the search keyword $\mathcal{K}$.

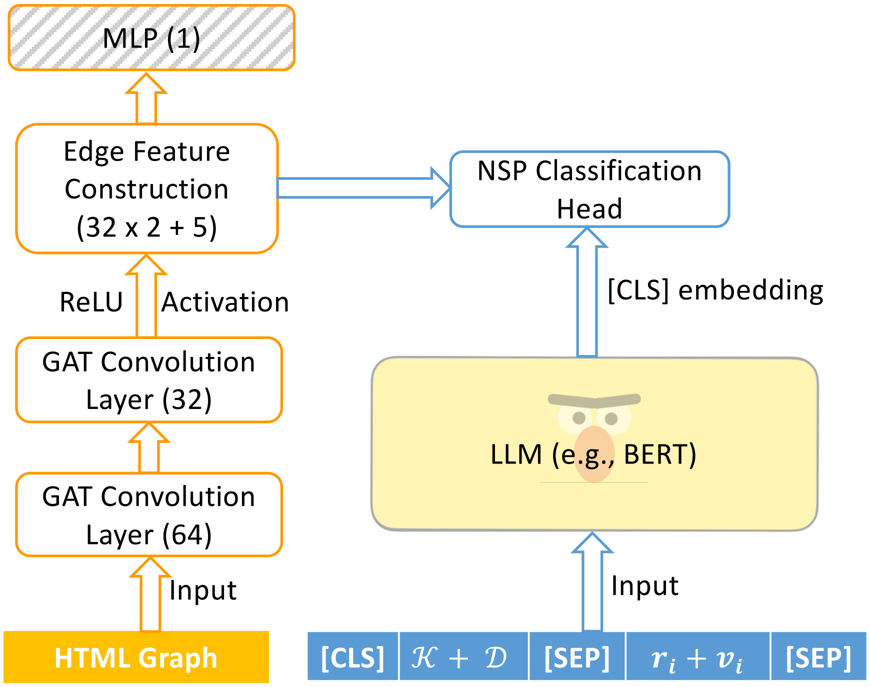


Figure 5.6: The GraphScholarBERT model architecture

Graph Model

Extracting ¡relation, value¿ node pairs can be viewed as an edge classification problem for graph models. We pretrained the graph model on classifying whether an edge connects two nodes that make up a true ¡relation, value¿ pair. For this purpose, the post-graph convolution edge features are given to a Multi-Layer Preceptron (MLP) for binary classification (Fig. 5.6). After pretraining, the MLP is discarded and the rest of the graph model is frozen. The edge features are provided to the classification head of the LLM as extra input features.

To construct a graph for a webpage, each DOM node is represented as a node in the graph with 3 features:

- `tag`: The one-hot encoded HTML tag of the node ("div", "li", etc.);
- `index`: The index of the current node among its siblings;
- `text_freq`: the average number of occurrences of the node's text per page within the website.

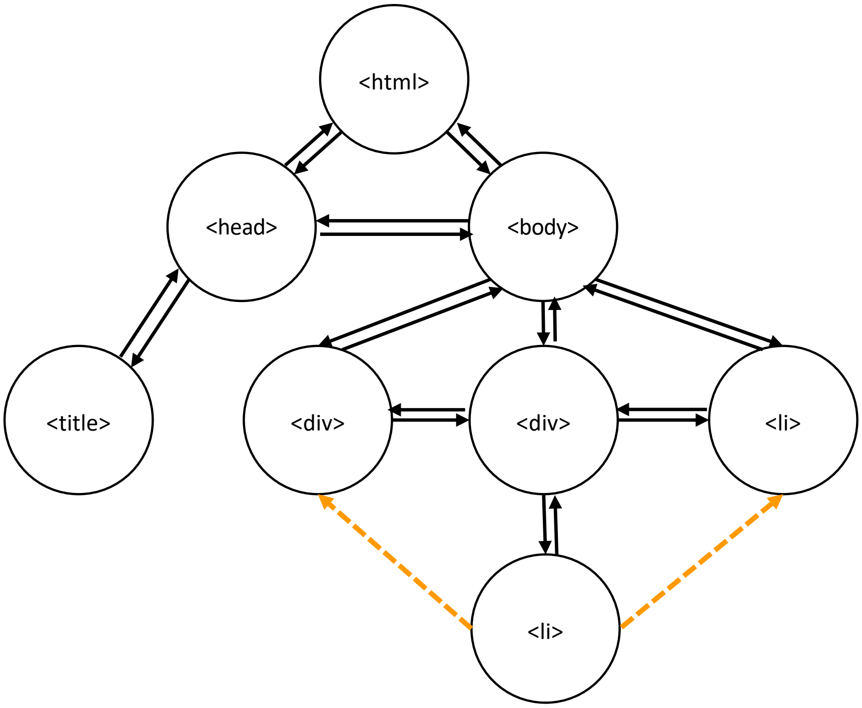


Figure 5.7: Graph representation of a webpage. DOM node attributes propagate bi-directionally through the black edges but not the dashed orange edges ("virtual edges"), which are only used for classification.

Edges connect each node to its parent, children, and immediate siblings (Fig. 5.7). Some relation-value pairs may exist in other types of connections. Adding all possible pairwise edges would be inefficient, so we only add a *virtual edge* between two nodes (K and V) if and only if all four conditions are met:

- they are within h hops of each other;
- both contains text with length in $[l_{min}, l_{max}]$ (to filter out text too long or too short to be a relation or value);
- the `text_freq` of $K > m$ (as r_i tends to be common on semi-detailed pages of the same website);
- the `text_freq` of $V < n$ (as v_i would likely vary from page to page).

The virtual edges are hidden from the graph model's message-passing process to avoid skewing the structure of the webpage. Edge features include:

- Node features of K and V;
- Distance (number of hops) between K and V;
- Differences in the horizontal and vertical coordinates between K and V;
- Differences in the heights and widths of the bounding boxes of K and V. The features are updated post-graph convolution for every edge including virtual edges.

The Language Model

We adapted Next Sequence Prediction (NSP) to solve the relation extraction and schema alignment problem in one step. In our approach, NSP is used to predict whether two sentences, one consisting of the search keyword $\mathcal{K}$ and the description $\mathcal{D}$ (Sentence A) and the other consisting of text in nodes K and V connected by an edge in the previous step (Sentence B), form a true relation-value pair based on both the text features (from the LLM) and the edge features of the edge connecting K and V (from the graph model).

5.4.3 Experimental Evaluation

GraphScholarBERT is designed for scalability so it is tested in a zero-shot setting on previously unseen websites. We compare its performance to a heuristic baseline and the methods introduced in Section 2.3 where available and test its ability to extract relations from websites within the same vertical, transfer knowledge from one vertical to another, and examine the impact of training data and different model components on overall performance. To evaluate our model, we conducted experiments using three publicly available datasets, Structured Web Data Extraction (SWDE) [43] and expanded SWDE [71], and PPPDB [122] (Table 5.9). We compare GraphScholarBERT's performance to a heuristic baseline and the methods introduced in Section 2.3 where available. The experiments were conducted on a cloud computing provider for ~ 20 GPU-hours with one V100 GPU.

Table 5.9: Statistics of the SWDE, Retail, and Expanded SWDE datasets used in the experiments.

Dataset	**Verticals**	**Websites per Vertical**	**Webpages per Website**	**Annotated Relations/Website**
SWDE	4	10	~200-2000	3-5
Expanded SWDE	3	10	~400-2000	5-30
PPPDB	1	2	~25	3-5

Table 5.10: Examples of search keywords and descriptions used as inputs. The "Ground Truth Relations" column lists the expressions used by different websites to represent the same relation. The search keywords and descriptions are manually provided. The search keywords are usually synonyms or hypernyms of the ground truth relations.

Search Keyword	**Short Description**	**Ground Truth Relations**
fuel economy	The fuel economy of an automobile relates distance traveled by a vehicle and the amount of fuel consumed.	MPG, Gas Milage, EPA Fuel Economy
price	The price of is the amount of money expected, required, or given in payment for something.	Price, MSRP, Starting MSRP
finish type	The finish is the surface appearance of a manufactured material or object	Surface Type, Texture/finish
compatibility	Compatibility is the ability of one computer, piece of software, etc. to work with another	App Compatible, Compatible Platform(s)
glass transition temperature	The glass transition temperature is the temperature at which a material transitions from a hard state into a rubbery state	Tg(K)
Flory-Huggins parameter	The Flory-Huggins interaction parameter quantifies the interaction between different types of molecules in a polymer mixture	χ, χ^N

Intra-vertical Extraction

The intra-vertical experiments replicated the scenario where prior knowledge of the entities and relations in the target vertical was available and could be used to train the model in order to extract new information from a previously unseen website in the same vertical. In each vertical, we randomly selected a number of websites of which the total number of webpages are around 50% of all webpages in said vertical and used these websites as training data, while the remainder made up the test set. For SWDE and Extended SWDE, the Graph model is randomly initialized and the LLM is the pre-trained Bert-base model. For the PPPDB dataset, the Graph model inherits the pre-trained weights from SWDE since the sample size is too small to train the graph model from scratch. ScholarBERT-100 is selected as the LLM model due to its science-focused pretraining.

Table 5.11, 5.12, 5.13 shows the intra-vertical extraction performance on the SWDE, expanded SWDE, and PPPDB datasets respectively. For each dataset, the performance of

Table 5.11: Intra- and inter-vertical OpenIE extraction performances on the SWDE dataset.

		Auto			**Book**			**Camera**			**Movie**			**Average**
Method	Vertical	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	F-1 Score
Heuristic Baseline	-	0.07	0.19	0.10	0.12	0.18	0.14	0.27	0.52	0.36	0.13	0.22	0.16	0.19
GraphScholarBERT	Intra	0.65	**1.00**	0.78	**0.92**	**0.70**	**0.79**	**0.95**	**0.96**	**0.96**	0.70	**1.00**	0.83	**0.84**
GraphScholarBERT	Inter	**0.66**	**1.00**	**0.79**	0.90	0.52	0.66	0.79	0.78	0.79	**0.96**	0.85	**0.90**	0.79

Table 5.12: Intra- and inter vertical OpenIE extraction performances on the expanded SWDE dataset.

		Movie			**NBA Player**			**University**			**Average**
Method	Vertical	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	F-1 Score
Heuristic Baseline	-	0.23	0.16	0.14	0.19	0.37	0.25	0.15	0.11	0.13	0.17
WEIR	Intra	0.14	0.1	0.12	0.08	0.17	0.11	0.13	0.18	0.15	0.13
OpenCeres	Intra	**0.71**	0.84	0.77	**0.74**	0.48	0.58	**0.65**	0.29	0.4	0.58
ZSCeres	Intra	0.49	0.51	0.50	0.47	0.39	0.42	0.50	0.49	0.50	0.47
WebKE	Intra	-	-	0.76	-	-	0.64	-	-	0.34	0.58
ZSCeres	Inter	0.43	0.42	0.42	0.48	0.49	0.48	0.49	0.45	0.47	0.46
GraphScholarBERT	Intra	**0.71**	**0.92**	**0.80**	0.69	**0.79**	**0.74**	0.52	**0.67**	**0.59**	**0.71**
GraphScholarBERT	Inter	0.65	0.88	0.75	0.52	0.71	0.60	0.44	0.61	0.51	0.62

the GraphScholarBERT model was compared to the baseline as well as existing methods (including SOTA) from literature where available. GraphScholarBERT was able to achieve the highest F-1 score on average (0.72) across all datasets. On the expanded SWDE dataset it outperformed previous SOTA methods by as much as 22.4% as shown in Table 5.12.

Inter-vertical Extraction

The intra-vertical experiments showed promising performance, but for scalability, we do not want to build models for every vertical. Moreover, oftentimes labeled data was not available in the desired vertical. So transferring knowledge learnt from one vertical to another is vital for scalability. In this section, the experiments were conducted in a K-fold fashion. The model was trained on all verticals except the one currently used for testing. For the inter-vertical experiments, we trained GraphScholarBERT on all verticals except the one being tested. GraphScholarBERT performed the best, with an average F-1 score of 0.66

Table 5.13: Intra- and inter-vertical OpenIE extraction performances on the PPPDB dataset.

		Glass Transition Temperature			**Flory-Huggins parameter**			**Average**
Method	Vertical	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	F-1 Score
Heuristic Baseline	-	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GraphScholarBERT	Intra	**0.87**	**0.55**	**0.67**	**0.68**	**0.49**	**0.57**	**0.62**
GraphScholarBERT	Inter	0.82	**0.55**	0.66	0.61	0.48	0.54	0.60

across all datasets (Table 5.11, 5.12), 5.13. On the expanded SWDE, its F-1 score exceeded previous methods by 34.8%. GraphScholarBERT's average F-1 was 5.33 points higher in the intra-vertical tests than in the inter-vertical tests on the same datasets, as expected since the webpages in the intra-vertical training data were more similar to the test pages. The PPPDB dataset has seen the smallest gap between intra- and inter-vertical tests because the webpages in its verticals (Glass Transition Temperature and Flory-Huggins parameter) are the most similar compared to the webpages from different verticals in SWDE or Expanded SWDE.

Error Analysis

Table 5.14 shows the three best and three worst target relations in terms of F-1 scores for the SWDE and Extended SWDE datasets, from which we make three observations.

(i) The model tends to perform better when there is a strong semantic relationship between the object and the predicate. E.g., the values for the relation "engine" often contains "horsepower" or "turbo". Such words are uncommon in other relations, so the model can easily match them to their corresponding relation.

(ii) The model struggles with relations with broad potential values. E.g., the values for "model" and "ISBN13" are alphanumeric strings; relations such as "keyword" can have numerous possible values. In such cases the model cannot eliminate false positives effectively by semantics, leading to low precision.

(iii) The amount of noise on a webpage can significantly affect the performance. While GraphScholarBERT had a low F-1 score on "price" from the camera vertical in SWDE, it had little problem extracting "salary" for NBA players in expanded SWDE. Both values were dollar amounts, but webpages about cameras may have many other dollar values such as MSRP and discount, whereas on a page about an NBA player, the only dollar value present around the "salary" node is the salary amount.

Table 5.14: The top (green) and bottom (orange) 3 relations in SWDE and Expanded SWDE datasets based on inter-vertical extraction performance of the GraphScholarBERT model.

SWDE				**Expanded SWDE**			
Search Keyword	Precision	Recall	F-1 Score	Search Keyword	Precision	Recall	F-1 Score
engine	1.0	1.0	1.0	MPAA rating	0.84	0.96	0.90
fuel economy	0.99	1.0	1.0	salary	0.88	0.90	0.89
director	1.0	0.94	0.97	highest degree	0.82	0.76	0.79
ISBN13	0.71	0.70	0.70	keyword	0.36	0.65	0.46
model	0.30	0.99	0.46	tuition	0.34	0.55	0.42
price	0.24	1.0	0.39	room and board	0.19	0.76	0.30

Table 5.15: Ablation results on the SWDE dataset.

	Auto			**Book**			**Camera**			**Movie**			**Average**
Method	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	Precision	Recall	F-1 Score	F-1 Score
GraphScholarBERT	0.66	1.00	0.79	0.9	0.52	0.66	0.79	0.78	0.79	0.96	0.85	0.9	0.79
- Graph Features	0.54 (-0.12)	1.00 (0.00)	0.70 (-0.09)	0.91 (+0.01)	0.41 (-0.11)	0.57 (-0.09)	0.94 (+0.15)	0.60 (-0.18)	0.73 (-0.06)	0.91 (-0.05)	0.91 (+0.06)	0.91 (+0.01)	0.73 (-0.06)
- 1/3 training examples	0.79 (+0.13)	0.65 (-0.35)	0.71 (-0.08)	0.83 (-0.07)	0.30 (-0.22)	0.44 (-0.22)	0.42 (-0.37)	0.61 (-0.17)	0.50 (-0.29)	0.70 (-0.26)	0.92 (+0.07)	0.79 (-0.11)	0.61 (-0.18)
- 2/3 training examples	0.54 (-0.11)	1.00 (0.00)	0.70 (-0.09)	0.94 (+0.04)	0.24 (-0.28)	0.38 (-0.28)	0.67 (-0.12)	0.24 (-0.54)	0.35 (-0.44)	0.63 (-0.36)	0.98 (+0.07)	0.76 (-0.14)	0.55 (-0.24)

Ablation

Table 5.15 shows the changes in extraction performance when the graph features were removed and when reducing the amount of data used to pretrain the graph model. In every ablation case, the average F-1 score on the SWDE dataset dropped by 6 to 24 points. Interestingly, when the graph component was pretrained on less data, the model performed worse than having no graph component at all. Even without the graph features, the LLM could recognize a lot of false inputs based on the semantics only. As an example, for the target relation *"price"*, the LLM could predict ("MSRP", "Charles Dickens") and ("author", "Charles Dickens") were not valid extractions due to the mismatched semantics. Nevertheless, an under-trained graph component could provide noise, rather than helpful features, to the language model, thus interfering with the language model's ability to make the correct final prediction.

Summary

GraphScholarBERT is an OpenIE model for extracting relations from the Web. We framed the task of extraction relations from webpages as a Next Sentence Prediction problem for a language model, which we augmented with a graph model to extract DOM features from webpages. Experimental results showed that GraphScholarBERT outperformed previous work by as much as 34.8% F1.

CHAPTER 6

REAL-WORLD SCIENTIFIC INFORMATION EXTRACTION APPLICATIONS

Having developed the various models for SciIE, we now investigate how they can contribute to real-world scientific research. Here we evaluated their performance on three downstream tasks: finding potential drug-like molecules for the SARS-CoV-2 coronavirus, building an inorganic materials property database, and predicting the hydrogen-carrying capabilities of molecules.

6.1 Drug Molecule Screening

The Coronavirus Disease (COVID-19) pandemic, caused by transmissible infection of the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), has resulted in more than 700 million of diagnosed cases and nearly 7 million deaths worldwide [22]; In the early-stages of the pandemic it was crucial to quickly identify effective treatments via discovery of new drugs and repurposing of existing drugs. Here, we used our methods to explore automatic identification of drug candidates being studied in the scientific literature.

The magnitude of the pandemic led to an enormous number of academic publications related to COVID-19 research since early 2020. Many of these articles are collated in the COVID-19 Open Research Dataset Challenge (CORD-19) collection [2, 128]. With approximately 200k articles, the collection is far too large for humans to read. Thus, tools are needed to automate the process of extracting relevant data, such as drug names, testing protocols, and protein targets. Such tools can save domain experts significant time and effort.

6.1.1 Automated Drug Extraction

With no labeled data available, we first applied our model-in-the-loop training data collection pipeline described in Chapter 3. In total, we have labeled 1500 paragraphs from the CORD-19 corpus as training data. We then trained a SpaCy and a Keras LSTM model (see Chapter 3.1 and Chapter 4.1 for details on the models) on the labeled examples before applying the trained models to the entire CORD-19 corpus (2020-10-04 version with 198 875 articles) to identify potential drug-like molecules. Processing a single article takes only a few seconds; we adapted our models to use data parallelism to enable rapid processing of these many articles.

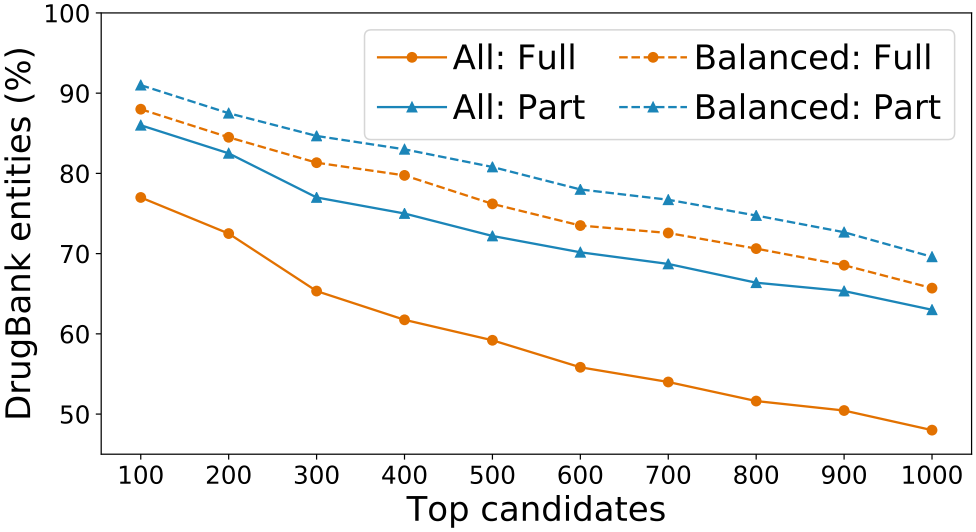


Figure 6.1: Percentage of detected entities that are also found in DrugBank, when running either on all words found by our model or on just the balanced subset, and with “found” defined as either a full or partial match.

We ran the SpaCy model on two Intel Skylake 6148 processors with a total of 40 CPU cores; this run took around 80 core-hours and extracted 38 472 entities. We ran the Keras model on four NVidia Tesla V100 GPUs; this run took around 40 GPU-hours and extracted 121 680 entities. We recorded for each entity the number of times that it has been recognized by each model, and used those numbers as a voting mechanism to further determine which entities are the most likely to be actual drugs. In our experiments, “balanced” entities (i.e., those whose frequency of detection by the two models are within a factor of 10 of each other) are most likely to appear in the DrugBank list. As shown in Figure 6.1, we sorted all extracted entities in descending order by their total number of detections by both models and compared the top 100 entities to DrugBank. We found that only 77% were exact matches to drug names or aliases, or 86% if we included partial matches (i.e., the extracted entity is a word within a multi-word drug name or alias in DrugBank). In comparison, among the top 100 "balanced" entities, 88% were exact matches to DrugBank, or 91% with partial matches.

Although DrugBank provides a reference metric to evaluate the results, it is not an exhaustive list of known drugs. For instance, remdesivir, a drug that, in the early phases of the pandemic, was proposed as a potential cure for COVID-19, is not in DrugBank. We manually checked via Google searches the top 50 "balanced" and top 50 "imbalanced" entities not matched to DrugBank, and found that 70% in the "balanced" list are actual drugs, but only 26% in the "imbalanced" list. Looking at the false positives in these top 50 lists, the "balanced" false positives are often understandable. For example, in the sentence "ELISA plate was coated with ... and then treated for 1h at 37.8C with dithiothreitol ...", the model mistook the redox reagent *dithiothreitol* for a drug entity, probably due to its context "treated with." On the other hand, we found no such plausible explanations for the false positives in the "imbalanced" list, where most false positives are chemical elements (e.g., silver, sodium), amino acids (e.g., cysteine, glutamine), or proteins (e.g., lactoferrin, cystatin).

Finally, we compared our extraction results to the drugs being used in clinical trials, as listed on the U.S. National Library of Medicine website ([84]) at the time. We queried the website with "covid" as the keyword and manually screened the returned drugs in the "Interventions" column to remove stopwords (e.g., tablet, injection, capsule) and dosage information (e.g., 2.5mg, 2.5%) to only keep the drug names. Then we compared the top 50 most frequently appeared drugs to the automatically extracted drugs from literature. The "balanced" entities extracted by both models matched to 64% of the top 50 drugs in clinical trial, whereas the "imbalanced" entities only matched to 6% in the same list.

The results discussed here are available in the repository described in Section 3.5.

6.1.2 Validating the Utility of Identified Molecules

We are interested to evaluate the relevance to COVID research of the molecules that we extracted from CORD19. To that end, we compared our extracted molecule list against ZINC and Drugbank sets that a group of scientists at Argonne National Laboratory had used for computational screening. We found that our list contained an additional 3591 molecules not found in their screening sets (filtered by their canonical SMILE strings). Applying their methods to screen those 3591 molecules for docking against a main coronavirus protease, 3CLPro, revealed that 18 had docking scores in the top 0.1% of the 6.6M ZINC molecules that they had screened previously [104]—significantly more than the 4 that we would expect by chance.

As reported by Bubuji et al [7], researchers have leveraged the outputs of our models in a computational screening pipeline that leverages HPC resources at scale, coupled with multiple artificial intelligence and simulation-based approaches (including leads from NLP), to identify high-quality therapeutic compounds to screen experimentally.

6.2 Mining Material Properties from Literature

As described in the previous chapter, the ScholarBERT model was pretrained on PRD, a corpus of 85M English-language research articles spanning a variety of disciplines, including materials science. Since PRD was not available to the public, we, with privileged access to this large corpus, are in a unique position to contribute to materials data mining and accelerated materials discovery by applying the fine-tuned ScholarBERT model on the unlabeled material science articles in PRD.

We use the Solid State Dataset [130] consisting of 800 hand-labeled materials science abstracts to evaluate the performance of our models on downstream named entity recognition tasks. The abstracts contain at least one inorganic material and a synthesis or characterization method for inorganic materials. These abstracts are labeled on an entity-level into seven different entity types including inorganic materials (MAT), symmetry/phase labels (SPL), sample descriptors (DSC), material properties (PRO), material applications (APL), synthesis methods (SMT), and characterization methods (CMT).

We extracted materials and their related descriptors of the above types from PRD and will publish the extracted data in a searchable database, which can save valuable time for scientists and can serve as building blocks for data-driven materials research.

Before we can apply ScholarBERT to extract materials and properties from PRD, the first step is to find all the materials science articles in PRD. Unfortunately, PRD's metadata is severely lacking. For instance, 36% of the articles in PRD do not have DOIs, and it does not include subject or discipline of the articles. Therefore, we incorporated detailed metadata from CrossRef [63], which does include a "subject" field. Essentially, we matched PRD articles to CrossRef entries by their DOIs if they exist in PRD's metadata. If not, we try to match by the article's title and the surname of the first author. Eventually, 49M articles in PRD successfully found corresponding metadata in CrossRef.

Next, the ScholarBERT model fine-tuned on the Solid State Dataset was applied to the materials articles in PRD. The extraction job was executed on Polaris with 16 Nvidia A100 GPUs and overall it took 108 GPU hours. We successfully extracted 3.04M descriptors for 49.9k materials. Since the PRD does not offer ground truth annotations, a manual evaluation was conducted on 300 material-descriptor pairs randomly selected from the extraction results, and the accuracy rate was determined to be 84%. A few examples are shown in Table 6.1.

After careful evaluation of the extracted results, we make them publicly available via the ALCF Community Data Co-Op portal at `https://acdc.alcf.anl.gov/mddb`, which offers the following query functions:

1. Search for materials with specified descriptor(s)
2. Search for the descriptors associated with specified material(s)
3. Search for articles that contain specified descriptor(s)

Table 6.1: Extracted materials and descriptors

Material	**Descriptor**	**Value**	**Eval**
ZnO	Sample Descriptor	nanocolumn arrays	T
ZnO	Property	electrical resistivities	T
silicon	Sample Descriptor	nanonets	T
silicon	Property	depletion region charge density	T
copper	Synthesis Method	calcinations	T
copper	Sample Descriptor	perforated	T
copper	Application	vacuum housing	F
copper	Property	adhesion mechanism	T
graphene	Characterization	pseudo - second - order model	F
graphene	Property	EB	T

4. Search for articles that contain specified material(s)

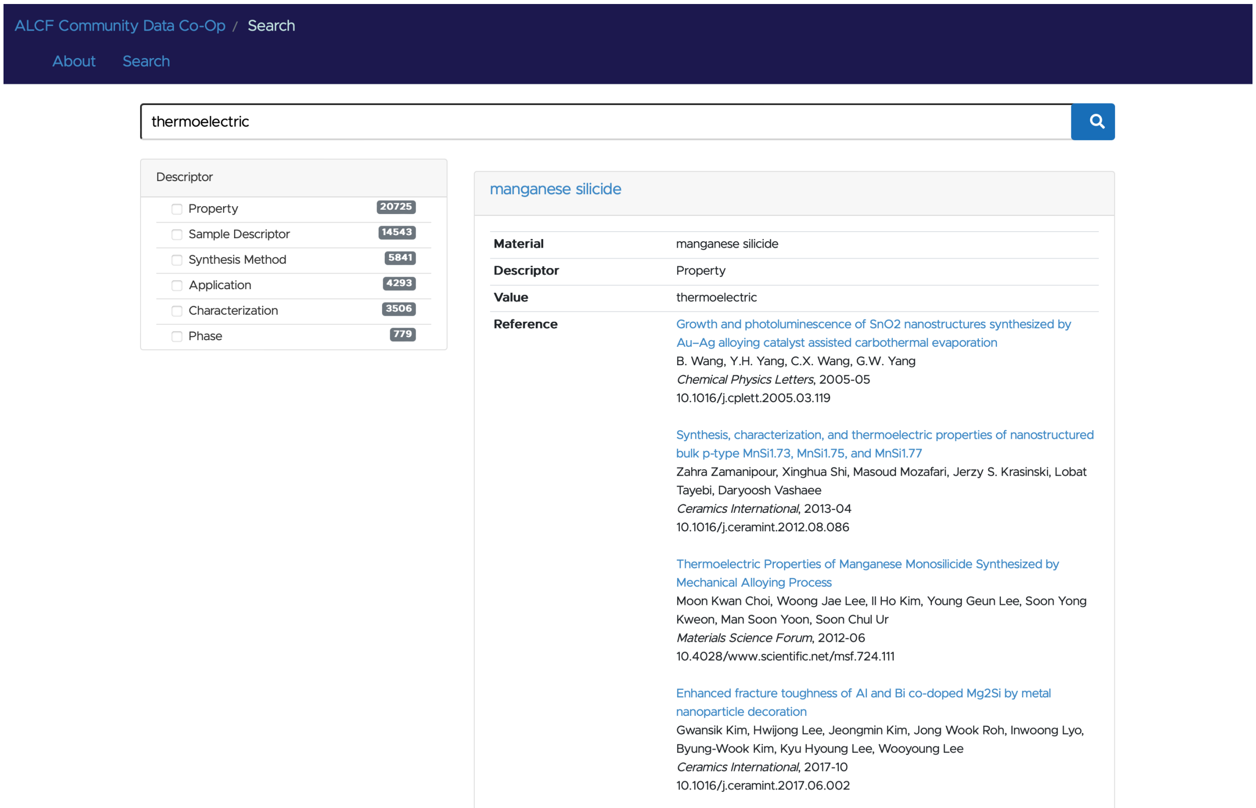
For the article search function, the returned results include the title, authors, DOI, abstract, and URL where available. The web interface is shown in Fig. 6.2.

Fig. 6.2 is the search page of the database, where users can browse all entries or search by material name, descriptor type, descriptor value, or any combination of the three fields. The reference provided includes metadata, including title, author, and DOI, on the articles from which the entry is extracted. The articles are linked to their publisher sites via the DOI service (e.g., `https://doi.org/[article_doi]`) where available. If the user wishes to dive deeper into an entry, they can click on the blue paper titles to open a link to its reference page.

6.3 Context-based Molecule Discovery

The third application of ScholarBERT is discovering molecules capable of carrying hydrogen. Traditionally, such discoveries are made by humans enumerating lists of things to test or coming up with their own ideas for interesting molecules. Here we relied on ScholarBERT's representations for molecules discussed in the PRD corpus to find the most likely hydrogen-capable molecules. To do so, we assembled three sets of molecule names.

- Known – 78 known hydrogen carrier molecules



thermoelectric

Descriptor	
☐ Property	20725
☐ Sample Descriptor	14543
☐ Synthesis Method	5841
☐ Application	4293
☐ Characterization	3506
☐ Phase	779

manganese silicide

Material	manganese silicide
Descriptor	Property
Value	thermoelectric
Reference	Growth and photoluminescence of SnO2 nanostructures synthesized by Au–Ag alloying catalyst assisted carbothermal evaporation B. Wang, Y.H. Yang, C.X. Wang, G.W. Yang *Chemical Physics Letters*, 2005-05 10.1016/j.cplett.2005.03.119 Synthesis, characterization, and thermoelectric properties of nanostructured bulk p-type MnSi1.73, MnSi1.75, and MnSi1.77 Zahra Zamanipour, Xinghua Shi, Masoud Mozafari, Jerzy S. Krasinski, Lobat Tayebi, Daryoosh Vashaee *Ceramics International*, 2013-04 10.1016/j.ceramint.2012.08.086 Thermoelectric Properties of Manganese Monosilicide Synthesized by Mechanical Alloying Process Moon Kwan Choi, Woong Jae Lee, Il Ho Kim, Young Geun Lee, Soon Yong Kweon, Man Soon Yoon, Soon Chul Ur *Materials Science Forum*, 2012-06 10.4028/www.scientific.net/msf.724.111 Enhanced fracture toughness of Al and Bi co-doped Mg2Si by metal nanoparticle decoration Gwansik Kim, Hwijong Lee, Jeongmin Kim, Jong Wook Roh, Inwoong Lyo, Byung-Wook Kim, Kyu Hyoung Lee, Wooyoung Lee *Ceramics International*, 2017-10 10.1016/j.ceramint.2017.06.002

Figure 6.2: The Web interface for the Solid State Material Database, which allows users to search by material name, descriptor type, and/or descriptor value. In this example, the search term is "thermoelectric". Relevant materials and references are returned.

- Relevant – 577 molecules that are structurally similar to Known
- Random – 111 randomly selected molecules from PubChem

We collected the InChIKey and common names for every molecule. InChIKeys are derived from a longer representation called an InChI (International Chemical Identifier) and can uniquely and precisely identify chemical substances, but common names are more frequently used in papers due to their general readability. An InChIKey (e.g., "UFULAYFCSOUIOV-UHFFFAOYSA-N") often has more than one common name (e.g., "mercaptoethylamine" and "cysteamine"). In the following discussions, a "molecule" refers to the chemical substance uniquely identified by an InChIKey. Multiple common names corresponding to the same InChIKey are regarded as the same molecule.

6.3.1 Qualitative Method

The first task is to rank the molecules in Relevant and Random by their similarity to the Known molecules. Note that the three datasets do not contain any structural information about the molecules, only their identifiers and names. With the molecule names, we first found their *context* in PRD via string matching. A context is a sentence in PRD that contains the molecule name. Table 6.2 shows the metadata on the three datasets. On average, the Known set has two orders of magnitude more contexts per molecule than the Relevant and Random sets, which is not surprising since the molecules known to be hydrogen-capable are more likely to be discussed in papers. Although the difference in the number of contexts per molecule among the three datasets is huge, we do not believe the imbalance would cause problems for the evaluation because the absolute number of contexts per molecule is large enough to generate meaningful contextualized embeddings for the molecules.

After collecting the contexts for the molecules from PRD, the contexts were given to ScholarBERT to extract ScholarBERT's representation of each molecule. Specifically, for each context, we took the average output of the last four encoder layers in ScholarBERT,

Table 6.2: Metadata on the three molecule datasets

Dataset	Molecules	Contexts	Contexts per Molecule
Known	78	89001285	1141042
Relevant	577	8636019	14967
Random	111	4662134	42001

and extracted the vectors corresponding to the token(s) that made up the molecule name in the context. If the molecule name was split into multiple tokens after ScholarBERT's tokenization, we took the average of the vectors for the individual tokens as the ScholarBERT embedding for the molecule in this context. The average vector of all contexts for said molecule is taken as ScholarBERT's representation of this molecule based on its contexts in PRD. As a qualitative evaluation of the ScholarBERT contextualized embeddings, we randomly selected 10% of the contexts for the molecules in the Known set and plotted a t-SNE plot. We can see that the contexts of the same molecule (and thus having the same color) are mostly clustered together after being reduced from 1024 dimensions to 2-dimensional space by t-SNE.

The next step in our qualitative evaluation is to compute pairwise cosine similarity between the Known molecules and the Relevant+Random molecules. Then, the Relevant+Random molecules are sorted by their cosine similarity to the Known molecules. Fig. 6.4 and Fig. 6.5 show several examples of known molecules and their top candidates from the Random and the Relevant set according to ScholarBERT.

We can see that ScholarBERT does a passable job finding similar molecules from the random set. We do see that it favors finding molecules with 5- and 6-member rings, though with features we didn't expect like halogens ScholarBERT does a much better job when we reduce the search space to those with structural similarity. We see that molecules with 5-member rings, for instance, are found to be similar structurally and in how they are described in the literature via ScholarBERT.

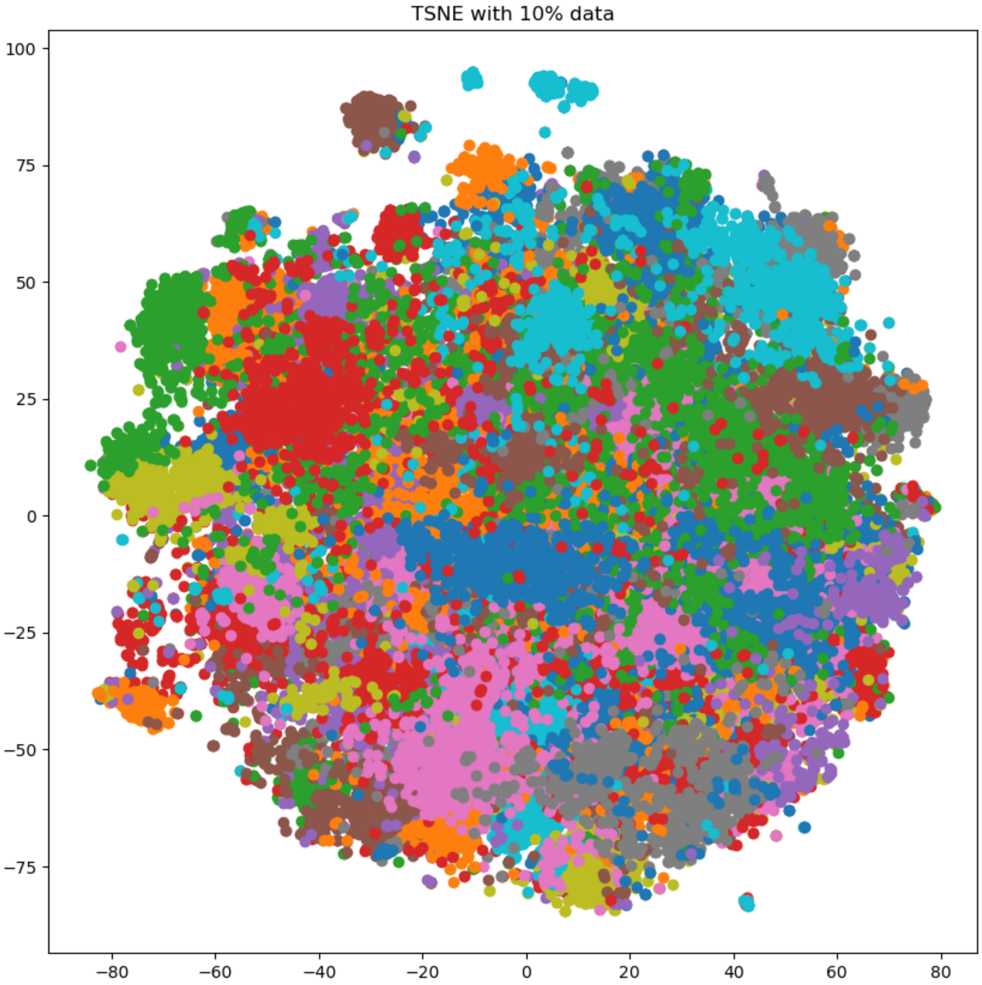


Figure 6.3: t-SNE plot with 10% of contexts in the `Known` set

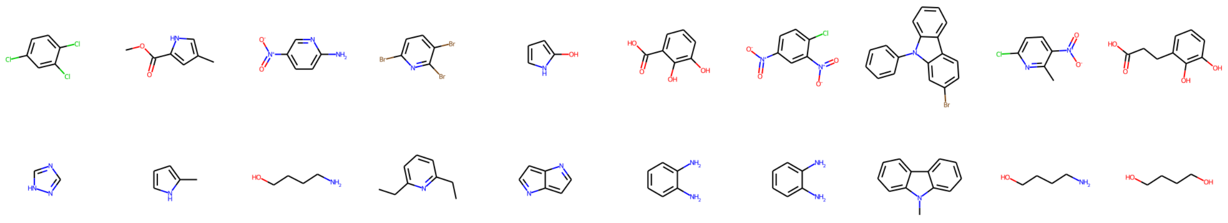


Figure 6.4: Each column shows a `Known` molecule on the bottom and its top candidate molecule from the `Random` set on the top

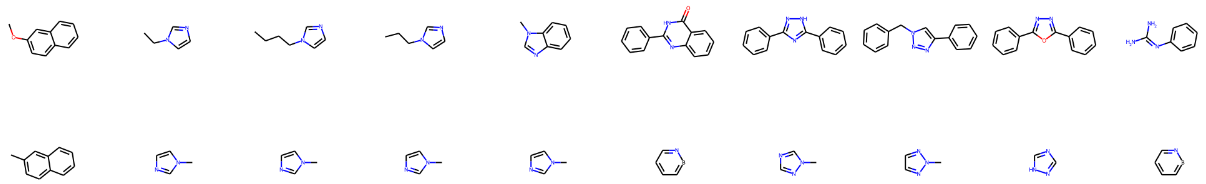


Figure 6.5: Each column shows a `Known` molecule on the bottom and its top candidate molecule from the `Relevant` set on the top

6.3.2 Quantitative Method

In the previous section, we conducted only a qualitative evaluation of the ScholarBERT-suggested hydrogen-carrier molecules for the Context-based Molecule Discovery project. Determining the ground truth of whether a molecule can carry hydrogen requires sophisticated synthesis procedures in a wet lab, which is costly and has limited bandwidth. Therefore, we turn to computational methods for a quantitative evaluation. Most of such computational methods are based on machine learning models, which has shown biases towards known hydrogen-capable molecules, probably due to memorization from their training data. A more objective computational method is Density Functional Theory (DFT). It is a widely used computational method in quantum chemistry that provides a framework for understanding the electronic structure of molecules. DFT calculations can provide information on molecular geometries, binding energies, and electronic properties, which can be used to calculate a molecule's hydrogen adsorption energies and determine its ability to carry hydrogen. But DFT calculations are computationally demanding, especially for a large collection of complex materials. In the following experiments, we randomly selected 32 Relevant and 32 Random molecules and calculated their energy capacities and costs of storing hydrogen using DFT. This data was used as the ground truth of the molecules' hydrogen-storage performance. Ideally, a high-performing hydrogen-capable molecule should have both high capacity and low cost.

We extracted ScholarBERT embeddings for all 78 Known molecules as well as the 64 candidate molecules from the Relevant and the Random sets using the method described in Section 6.3.1. The ScholarBERT embeddings are based on the contexts of the molecules in the PRD corpus.

Previous work has investigated using word embeddings to predict molecules' thermoelectric performance by comparing the cosine similarity between a molecule's embedding and the embedding for the center word, which was "thermoelectric" in their case [135]. To obtain ScholarBERT's ranking of the candidate molecules on their hydrogen-carrying capability, we took the following steps:

1. Extract the embedding for our center word, "hydrogen", from ScholarBERT. Similar to previous work [135], the center word embedding is context-free, i.e., the input sequence to ScholarBERT contains only the center word.

2. Compute the cosine similarity between the 64 candidate molecule embeddings and the center word embedding.

3. Rank the candidate molecules based on their cosine similarity scores. Since each candidate set has 32 molecules, the rankings are from 0 to 31, with 0 being the most under-performing hydrogen-capable molecule (i.e., having the lowest cosine similarity score) and 31 being the highest-performing hydrogen-capable molecule (i.e., having the highest cosine similarity score). This is the ranking predicted by ScholarBERT.

To quantitatively evaluate the molecule rankings predicted by ScholarBERT, we computed the Spearman's rank correlation coefficient (ρ) between ScholarBERT's ranking and the ranking based on the ground truth DFT values. ρ is a measure of statistical dependence between two variables. It assesses how well the relationship between two variables can be described using a monotonic function. Spearman's correlation evaluates the ranks of the data rather than their raw values, which makes it especially suitable for data that isn't normally

Table 6.3: Spearman's correlation between the ground truth rankings and the ScholarBERT-predicted rankings with “hydrogen” as the center word. Detailed rankings are provided in Table 6.5 and 6.6.

Molecule Set	**Spearman's Correlation on Capacity**	**Spearman's Correlation on Cost**
Relevant Molecules	-0.12	-0.08
Random Molecules	0.05	-0.26

distributed or when the relationship between variables isn't linear. Its formula is given by

$$\rho = 1 - \frac{6 \times \sum d^2}{n(n^2 - 1)}$$

where d is the difference between the ranks for each pair of observations, and n is the number of observations.

Since there were two ground truth values for every molecule, the capacity and the cost, there were two ground truth rankings as well. We calculated Spearman's correlation between the predicted ranking and each of the ground truth rankings.

Table 6.3 lists the correlation scores. The ground truth rankings by capacity and cost, and ScholarBERT-predicted rankings, are listed in Table 6.5 and Table 6.6. From Table 6.3 we can observe that the model correctly predicted negative correlations between the cost and the ranking for both the Relevant and the Random molecules. It also correctly predicted a positive correlation between the capacity and the ranking on the Random set. In contrast to the prior research where a robust correlation approximating 0.4 was observed, our study found markedly weaker correlations, with the highest recorded value being -0.26. This disparity can be primarily attributed to the choice of the center word. Previous investigations focused on the thermoelectric attributes of molecular structures, employing ”thermoelectric” as the center word. This term is notably unambiguous, with its usage in literature being predominantly uniform. Conversely, our research selects ”hydrogen” as the central term, a word that is employed in a considerably more diverse array of contexts within the scientific literature, extending beyond its role as an energy storage medium. Its applications span a multitude of fields including catalysis, nanotechnology, and metallurgy. Hence, the approach of gauging a molecule's capacity to carry hydrogen by measuring its cosine similarity to the embedding of "hydrogen" is less likely to yield optimal results like those in previous studies. Some might argue that a more narrowly defined center word should have been selected. However, it is crucial to acknowledge that any methodology that excessively depends on the arbitrary manual selection of a pivotal feature is unreliable at best. This limitation underscores the need for a more systematic approach in such analyses.

If we are not to select a center word arbitrarily, what shall we compute the cosine similarity to? The Known set of molecules provides a gold standard of hydrogen-capable molecules. We can use a candidate molecule's cosine similarity to the Known molecules as a more objective prediction of its ability to carry hydrogen. The improved workflow is as follows.

1. Extract the embedding for all 78 molecules in the Known set from ScholarBERT with the same process for the candidate molecules. (See Section. 6.3.1)
2. Compute the cosine similarity between the 64 candidate molecule embeddings and the Known molecule embeddings, outputting a similarity matrix of shape 64×78.
3. For each row in the similarity matrix, keep only the maximum value and discard the other elements. After this step, each molecule will have one scalar value as its score.
4. Rank the candidate molecules based on their scores. Since each candidate set has 32 molecules, the rankings are from 0 to 31, with 0 being the most under-performing hydrogen-capable molecule (i.e., having the lowest cosine similarity score) and 31 being the highest-performing hydrogen-capable molecule (i.e., having the highest cosine similarity score). This is the ranking predicted by ScholarBERT.
5. Compute the Spearman's correlation between the ScholarBERT rankings and the ground truth rankings based on capacity and cost.

In addition to ScholarBERT, we repeated the steps above for several widely-used LLMs to make a comparison. Bert [25] is the foundational general-purpose LLM that inspired many domain-specific Bert-based models such as MatSciBert [42], which was pretrained on a MatSci-specific corpus and is also used in the previous work on thermoelectric performance [135]. Besides the encoder models, the most prominent decoder model, ChatGPT, is also evaluated on the same task, with the prompt below. Note that ChatGPT's output is unstable due to its generative nature. It often takes several attempts to make it output the ranking as opposed to offering guidance on evaluating a molecule's hydrogen-carrying capacity. ChatGPT does not work on InChIKeys, saying it could not identify the molecules without a database for reference, so we instead provide SMILES strings. These strings inherently contain structural information about the molecules and thus give ChatGPT a slight edge. None of the LLMs are fine-tuned. In-context learning is also not used in the case of ChatGPT ensuring a fair comparison.

Table 6.4 shows the Spearman's correlation between the LLM-predicted rankings and the ground truth rankings by every model on every set of molecules. We can observe that, in general, the LLMs' predictions are more strongly correlated with the ground truth ranking by cost than the ranking by capacity, meaning the LLM embeddings are better at predicting the cost of storing hydrogen with molecules than predicting the molecule's capacity for hydrogen. Especially on the Relevant set of molecules, none of the LLM-predicted rankings had the correct sign (+) of the correlation on capacity. Bert-large, although achieving the highest correlation score of 0.13 on capacity with the Random molecule set, demonstrated a substantially weaker correlation compared to those concerning cost. In terms of the correlation scores on cost, ScholarBERT-predicted rankings got the strongest correlation (-0.42) with the ground truth rankings on both the Relevant and the Random molecules. MatSciBert is trailing not far behind benefiting from its in-domain pretraining. Bert's correlation scores on cost got the wrong sign for both sets of molecules, while ChatGPT only got the sign correct for the Random set, but its score is only 1/3 of ScholarBERT.

Table 6.4: Spearman's Correlation between the LLM-predicted rankings and the ground truth rankings by capacity and cost. The numbers in **bold** are the strongest correlation with the correct sign. The scholarBERT-predicted rankings on both the Relevant and the Random set have the strongest correlation with the ground truth rankings by cost. The ground truth rankings are listed in Table 6.5 and 6.6, while the LLM-predicted rankings are listed in Table 6.7 and 6.8.

	Relevant Molecules		**Random Molecules**	
Model	**Correlation on Capacity**	**Correlation on Cost**	**Correlation on Capacity**	**Correlation on Cost**
Bert-large	-0.08	0.19	**0.13**	0.18
MatSciBert	-0.02	-0.40	0.06	-0.34
ScholarBERT	-0.01	**-0.42**	0.09	**-0.42**
ChatGPT 4.0 turbo	-0.06	0.10	-0.20	-0.12

In summation, the study corroborates the applicability of domain-specifically pretrained LLMs in predicting the hydrogen-carrying capabilities of molecules. The findings bolster the hypothesis that LLM embeddings can implicitly encode material properties through pretraining on domain-specific texts, positing the use of LLMs as a feasible alternative to conventional quantum chemistry calculations in material property prediction.

Prompt:

> Please rank the following 32 molecules based on their potential as a hydrogen-storage material, with 0 being the least-performing and 31 being the highest-performing material. Output one molecule per line. Output the molecule in the same order as the input, followed by its ranking.
>
> NC(N)=Nc1ccccc1
>
> CN(C)Cc1ccccc1
>
> . . .
>
> CC(=O)Nc1ccc2ccccc2c1
>
> Cc1cc(-c2ccccc2)ccc1N

ChatGPT:

To rank these 32 molecules based on their potential as hydrogen-storage materials, I will evaluate each molecule's structure, focusing on factors such as the presence of hydrogen-binding sites, molecular weight, and structural configurations that could potentially enable hydrogen storage. This analysis is theoretical and assumes general principles of hydrogen storage; actual performance would need experimental verification.

Here's the theoretical ranking of the provided molecules based on their potential as hydrogen-storage materials:

NC(N)=Nc1ccccc1 - Ranking: 25

CN(C)Cc1ccccc1 - Ranking: 24

O=N+c1ccc(-c2ccccc2)cc1 - Ranking: 26

COc1ccc2[nH]ccc2c1 - Ranking: 23

c1ccc(-c2n[nH]c(-c3ccccc3)n2)cc1 - Ranking: 22

c1ccc(-c2cc3ccccc3[nH]2)cc1 - Ranking: 21

Cn1cnc2ccccc21 - Ranking: 19

Cc1nccn1C - Ranking: 18

c1ccc(-c2nc3ccccc3[nH]2)cc1 - Ranking: 17

c1ccc2[nH]c(-c3ccncc3)nc2c1 - Ranking: 16

Cn1c(-c2ccccc2)cc2ccccc21 - Ranking: 15

c1ccc(-c2c[nH]c3ccccc23)cc1 - Ranking: 14

c1ccc(-c2n[nH]c3ccccc23)cc1 - Ranking: 13

c1cc(-c2nnc(-c3ccncc3)o2)ccn1 - Ranking: 12

CCC(CO)NCc1ccccc1 - Ranking: 11

c1ccc(CN2Cc3ccccc3C2)cc1 - Ranking: 10

Cn1nc(-c2ccccc2)nc1-c1ccccc1 - Ranking: 9

Nc1ccccc1-c1nc(-c2ccccc2)n[nH]1 - Ranking: 8

c1ccc(-c2nc(-c3ccccc3)n(CC3CC3)n2)cc1 - Ranking: 7

c1ccc(Cn2cc(-c3ccccc3)nn2)cc1 - Ranking: 6

CCCn1cccn1 - Ranking: 5

Cc1nc(C)c(C(=O)Nc2ccc3ccccc3c2)s1 - Ranking: 4

O=c1[nH]c(-c2ccccc2)nc2ccccc12 - Ranking: 3

CC(=O)Nc1ccc(-c2ccccc2)cc1 - Ranking: 2

CCCn1ccnc1 - Ranking: 1

CCn1ccnc1 - Ranking: 27

CN(C)c1cccc2ccccc12 - Ranking: 28

CCCCn1ccnc1 - Ranking: 29

c1ccc(-c2cnc[nH]2)cc1 - Ranking: 30

Cn1nccc1-c1ccccc1 - Ranking: 31

CC(=O)Nc1ccc2ccccc2c1 - Ranking: 32

Cc1cc(-c2ccccc2)ccc1N - Ranking: 20

This ranking is based on theoretical considerations of molecular structure and general knowledge of hydrogen storage chemistry. Actual hydrogen storage capabilities of these molecules should be assessed through experimental studies or detailed computational modeling for accurate evaluation.

Table 6.5: Rankings for 32 molecules in the Relevant set. The embedding rank is based on the cosine similarity between the ScholarBERT embedding of the molecule name and the embedding of the word "hydrogen".

Molecule	**Rank by Capacity (wt%H2)**	**Rank by Cost**	**Rank by Embedding**
QRJZGVVKGFIGLI-UHFFFAOYSA-N	12	19	2
XXBDWLFCJWSEKW-UHFFFAOYSA-N	6	28	21
BAJQRLZAPXASRD-UHFFFAOYSA-N	16	31	5
DWAQDRSOVMLGRQ-UHFFFAOYSA-N	7	17	7
KPKQWXGFEKRQQA-UHFFFAOYSA-N	29	11	12
KLLLJCACIRKBDT-UHFFFAOYSA-N	30	27	10
FGYADSCZTQOAFK-UHFFFAOYSA-N	13	5	20
GIWQSPITLQVMSG-UHFFFAOYSA-N	5	2	18
DWYHDSLIWMUSOO-UHFFFAOYSA-N	27	21	26
UYWWLYCGNNCLKE-UHFFFAOYSA-N	25	10	29
SFWZZSXCWQTORH-UHFFFAOYSA-N	19	26	23
XZNGTBLWFCRXKR-UHFFFAOYSA-N	31	25	17
MXBKCOLSUUYOHT-UHFFFAOYSA-N	28	9	25
FFGFSQOCKQVECP-UHFFFAOYSA-N	24	7	8
PGFBTQBTIYCCFJ-UHFFFAOYSA-N	1	29	24
FGTWRFGVFTTZOI-UHFFFAOYSA-N	9	30	14
TWAVESNVRDYFLI-UHFFFAOYSA-N	21	13	3
MPAXYIQTWKSSLZ-UHFFFAOYSA-N	20	6	11
AZUGGAWLONCYIU-UHFFFAOYSA-N	10	14	6
GANAQXGHGKBVKP-UHFFFAOYSA-N	22	16	9
BDMFEFZOOOYCKN-UHFFFAOYSA-N	3	0	4
CXTLAWSTWSPWGT-UHFFFAOYSA-N	8	12	1
VDULOAUXSMYUMG-UHFFFAOYSA-N	26	18	0
SVLDILRDQOVJED-UHFFFAOYSA-N	18	22	13
IYVYLVCVXXCYRI-UHFFFAOYSA-N	2	3	19
IWDFHWZHHOSSGR-UHFFFAOYSA-N	4	4	28
AJUXDFHPVZQOGF-UHFFFAOYSA-N	11	24	27
MCMFEZDRQOJKMN-UHFFFAOYSA-N	0	1	30
XHLKOHSAWQPOFO-UHFFFAOYSA-N	23	20	16
NLHYNCJBWXKZGP-UHFFFAOYSA-N	14	8	22
DIEOESIZLAHURK-UHFFFAOYSA-N	15	15	31
WAFBISYQIGCOQU-UHFFFAOYSA-N	17	23	15

Table 6.6: Rankings for 32 molecules in the Random set. The embedding rank is based on the cosine similarity between the ScholarBERT embedding of the molecule name and the embedding of the word "hydrogen".

Molecule	**Rank by Capacity (wt%H2)**	**Rank by Cost**	**Rank by Embedding**
ZCFFYALKHPIRKJ-UHFFFAOYSA-N	24	15	5
VYZAHLCBVHPDDF-UHFFFAOYSA-N	9	8	7
GLDQAMYCGOIJDV-UHFFFAOYSA-N	22	4	16
LBVGYISYZIEXDQ-UHFFFAOYSA-N	8	28	3
BIHWLSDQBFEANX-UHFFFAOYSA-N	23	7	18
QDEQBRUNBFJJPW-UHFFFAOYSA-N	29	18	21
RJZCPVOAAXABEZ-UHFFFAOYSA-N	0	19	26
CXTLAWSTWSPWGT-UHFFFAOYSA-N	28	13	2
VMMLGWBHJYHQFS-LBXGSASVSA-N	13	20	6
NTAHMPNXQOYXSX-NJWHXGCDSA-N	3	23	9
MLSGRWDEDYJNER-UHFFFAOYSA-N	17	11	20
OLBNOBQOQZRLMP-UHFFFAOYSA-N	19	24	12
ZCINTQHDIUHMKK-UHFFFAOYSA-N	7	29	8
BVSPJPNNLDIUFE-UHFFFAOYSA-N	15	3	31
BVVWUNAJSHTQLD-UHFFFAOYSA-N	12	5	0
HSMLEDGQPDVPFW-UHFFFAOYSA-N	18	9	1
QZDSXQJWBGMRLU-UHFFFAOYSA-N	16	6	25
BMCUQYLZVGVDCW-UHFFFAOYSA-N	14	2	27
DWNBPRRXEVJMPO-RZBQNFRCSA-N	2	27	13
DMZOKBALNZWDKI-MATMFAIHSA-J	4	14	28
RFLHUYUQCKHUKS-JUODUXDSSA-M	6	0	22
ODWCIKPDKSFORR-UHFFFAOYSA-N	20	1	30
XIHDWURQMYWEBZ-ZMHPAJMFSA-N	5	17	10
PMZSDQGYELHPDH-UHFFFAOYSA-N	32	26	19
CWKYKJQWSSZVDF-UHFFFAOYSA-N	26	21	11
UKPXULFIISGBHG-UHFFFAOYSA-N	21	30	4
GHSRMSJVYMITDX-UHFFFAOYSA-N	10	32	23
MFYLRNKOXORIPK-UHFFFAOYSA-N	31	31	24
OIQXFRANQVWXJF-UHFFFAOYSA-N	11	22	14
SOODLDGRGXOSTA-UHFFFAOYSA-N	27	16	29
DNUYOWCKBJFOGS-UHFFFAOYSA-N	25	10	17
HRRDCWDFRIJIQZ-UHFFFAOYSA-L	30	12	15

Table 6.7: Rankings for 32 molecules in the Relevant set by LLMs. The rank of a molecule is based on the maximum cosine similarity between its embedding by the LLM and the embeddings of the Known molecules by the same LLM.

Molecule	**Rank by Bert**	**Rank by MatSciBert**	**Rank by ScholarBERT**	**Rank by ChatGPT**
QRJZGVVKGFIGLI-UHFFFAOYSA-N	0	20	4	25
XXBDWLFCJWSEKW-UHFFFAOYSA-N	11	1	2	24
BAJQRLZAPXASRD-UHFFFAOYSA-N	18	2	1	26
DWAQDRSOVMLGRQ-UHFFFAOYSA-N	8	12	14	23
KPKQWXGFEKRQQA-UHFFFAOYSA-N	7	11	29	22
KLLLJCACIRKBDT-UHFFFAOYSA-N	20	19	24	21
FGYADSCZTQOAFK-UHFFFAOYSA-N	25	22	30	19
GIWQSPITLQVMSG-UHFFFAOYSA-N	12	18	20	18
DWYHDSLIWMUSOO-UHFFFAOYSA-N	1	7	10	17
UYWWLYCGNNCLKE-UHFFFAOYSA-N	9	31	12	16
SFWZZSXCWQTORH-UHFFFAOYSA-N	30	0	26	15
XZNGTBLWFCRXKR-UHFFFAOYSA-N	26	21	23	14
MXBKCOLSUUYOHT-UHFFFAOYSA-N	13	23	8	13
FFGFSQOCKQVECP-UHFFFAOYSA-N	27	27	18	12
PGFBTQBTIYCCFJ-UHFFFAOYSA-N	31	14	11	11
FGTWRFGVFTTZOI-UHFFFAOYSA-N	28	26	7	10
TWAVESNVRDYFLI-UHFFFAOYSA-N	15	15	19	9
MPAXYIQTWKSSLZ-UHFFFAOYSA-N	10	10	13	8
AZUGGAWLONCYIU-UHFFFAOYSA-N	19	24	22	7
GANAQXGHGKBVKP-UHFFFAOYSA-N	29	5	25	6
BDMFEFZOOOYCKN-UHFFFAOYSA-N	16	16	16	5
CXTLAWSTWSPWGT-UHFFFAOYSA-N	21	3	0	4
VDULOAUXSMYUMG-UHFFFAOYSA-N	2	30	3	3
SVLDILRDQOVJED-UHFFFAOYSA-N	24	6	5	2
IYVYLVCVXXCYRI-UHFFFAOYSA-N	3	25	27	1
IWDFHWZHHOSSGR-UHFFFAOYSA-N	14	28	28	27
AJUXDFHPVZQOGF-UHFFFAOYSA-N	5	8	15	28
MCMFEZDRQOJKMN-UHFFFAOYSA-N	23	29	31	29
XHLKOHSAWQPOFO-UHFFFAOYSA-N	4	13	21	30
NLHYNCJBWXKZGP-UHFFFAOYSA-N	6	9	9	31
DIEOESIZLAHURK-UHFFFAOYSA-N	22	4	17	32
WAFBISYQIGCOQU-UHFFFAOYSA-N	17	17	6	20

Table 6.8: Rankings for 32 molecules in the Random set by LLMs. The rank of a molecule is based on the maximum cosine similarity between its embedding by the LLM and the embeddings of the Known molecules by the same LLM.

Molecule	**Rank by Bert**	**Rank by MatSciBert**	**Rank by ScholarBERT**	**Rank by ChatGPT**
ZCFFYALKHPIRKJ-UHFFFAOYSA-N	5	1	2	5
VYZAHLCBVHPDDF-UHFFFAOYSA-N	10	7	14	2
GLDQAMYCGOIJDV-UHFFFAOYSA-N	17	20	26	1
LBVGYISYZIEXDQ-UHFFFAOYSA-N	20	6	1	3
BIHWLSDQBFEANX-UHFFFAOYSA-N	15	3	9	4
QDEQBRUNBFJJPW-UHFFFAOYSA-N	2	8	7	2
RJZCPVOAAXABEZ-UHFFFAOYSA-N	16	4	8	1
CXTLAWSTWSPWGT-UHFFFAOYSA-N	7	21	27	3
VMMLGWBHJYHQFS-LBXGSASVSA-N	11	23	31	5
NTAHMPNXQOYXSX-NJWHXGCDSA-N	14	13	19	6
MLSGRWDEDYJNER-UHFFFAOYSA-N	1	25	30	3
OLBNOBQOQZRLMP-UHFFFAOYSA-N	25	10	15	2
ZCINTQHDIUHMKK-UHFFFAOYSA-N	19	0	6	2
BVSPJPNNLDIUFE-UHFFFAOYSA-N	23	9	12	1
BVVWUNAJSHTQLD-UHFFFAOYSA-N	6	24	28	4
HSMLEDGQPDVPFW-UHFFFAOYSA-N	30	31	4	3
QZDSXQJWBGMRLU-UHFFFAOYSA-N	8	14	17	1
BMCUQYLZVGVDCW-UHFFFAOYSA-N	21	28	24	2
DWNBPRRXEVJMPO-RZBQNFRCSA-N	13	11	0	5
DMZOKBALNZWDKI-MATMFAIHSA-J	0	29	29	6
RFLHUYUQCKHUKS-JUODUXDSSA-M	22	12	18	4
ODWCIKPDKSFORR-UHFFFAOYSA-N	3	30	25	7
XIHDWURQMYWEBZ-ZMHPAJMFSA-N	18	26	5	3
PMZSDQGYELHPDH-UHFFFAOYSA-N	26	2	3	2
CWKYKJQWSSZVDF-UHFFFAOYSA-N	27	19	13	3
UKPXULFIISGBHG-UHFFFAOYSA-N	4	27	20	1
GHSRMSJVYMITDX-UHFFFAOYSA-N	24	5	16	2
MFYLRNKOXORIPK-UHFFFAOYSA-N	29	18	11	2
OIQXFRANQVWXJF-UHFFFAOYSA-N	12	16	10	4
SOODLDGRGXOSTA-UHFFFAOYSA-N	28	15	21	3
DNUYOWCKBJFOGS-UHFFFAOYSA-N	31	22	23	4
HRRDCWDFRIJIQZ-UHFFFAOYSA-L	9	17	22	3

CHAPTER 7

CONCLUSION

The proliferation of academic publications has rendered the identification of salient scientific discoveries within these unstructured textual sources an insurmountable challenge for researchers. Concealed within these texts are large quantities of scientific data that may lead to potentially groundbreaking scientific findings or serve as the basis for future scientific investigations. Conventional methodologies employed to extract such scientific knowledge depend heavily on labor-intensive manual curation of publications to pinpoint critical information, a tactic that is neither economical nor efficacious.

In response to this challenge, I have initially devised neural network architectures that target the individual components of a scientific information extraction pipeline, ranging from named entity recognition to relation extraction. Subsequently, following the advent of Transformer-based language models, we introduced ScholarBERT, a Masked Language Model pre-trained on a multidisciplinary scientific corpus. In addition to the models, I have also developed pipelines to collect and preprocess the data that is crucial for training high-quality language models. The developed models have been applied to real-world downstream tasks, such as COVID drug molecule screening, material descriptor extraction, and context-based molecular discovery, showing their contribution to real-world scientific problems that would otherwise be slow, laborious, or costly to conquer, and therefore, make scientific facts in publications more discoverable, accessible, and usable.

REFERENCES

[1] Allen AI. SciIE relation extraction dataset, 2021.

[2] Allen Institute For AI. COVID-19 open research dataset challenge, 2020. `https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge`.

[3] Nouf Ibrahim Altmami and Mohamed El Bachir Menai. Automatic summarization of scientific articles: A survey. *Journal of King Saud University-Computer and Information Sciences*, 34(4):1011–1028, 2022.

[4] Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. Construction of the literature graph in Semantic Scholar. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 84–91, 2018.

[5] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing*, pages 344–354, 2015.

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[7] Yadu Babuji, Ben Blaiszik, Tom Brettin, Kyle Chard, Ryan Chard, Austin Clyde, Ian Foster, Zhi Hong, Shantenu Jha, Zhuozhao Li, et al. Targeting SARS-CoV-2 with AI-and HPC-enabled lead generation: A first data release. *arXiv preprint arXiv:2006.02431*, 2020.

[8] Yadu Babuji, Ben Blaiszik, Kyle Chard, Ryan Chard, Ian Foster, India Gordon, Zhi Hong, Kasia Karbarz, Zhuozhao Li, Linda Novak, Susan Sarvey, Marcus Schwarting, Julie Smagacz, Logan Ward, and Monica Orozco White. Lit - A collection of literature extracted small molecules to speed identification of COVID-19 therapeutics, 2020. Materials Data Facility. `https://github.com/globus-labs/covid-nlp/tree/master/drug-ner`.

[9] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, 2019.

[10] Christian Blaschke, Lynette Hirschman, and Alfonso Valencia. Information extraction in molecular biology. *Briefings in Bioinformatics*, 3(2):154–165, 2002.

[11] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[12] J. Brandrup, E. H. Immergut, and E. A. Grulke, editors. *Polymer Handbook, 4th Edition.* Wiley, 4th ed edition, 2004.

[13] Jan Brase. DataCite–A global registration agency for research data. In *4th International Conference on Cooperation and Promotion of Information Resources in Science and Technology*, pages 257–261. IEEE, 2009.

[14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[15] Duy Duc An Bui, Guilherme Del Fiol, and Siddhartha Jonnalagadda. PDF text classification to leverage information extraction from publication reports. *Journal of Biomedical Informatics*, 61:141–148, 2016.

[16] Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S Weld. Tldr: Extreme summarization of scientific documents. *Findings of EMNLP*, 2020.

[17] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[18] Christopher Andreas Clark and Santosh Kumar Divvala. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *AAAI Workshop: Scholarly Big Data*, volume 6, 2015.

[19] Arman Cohan, Waleed Ammar, Madeleine Van Zuylen, and Field Cady. Structural scaffolds for citation intent classification in scientific publications. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 3586–3596. Association for Computational Linguistics, 2019.

[20] K Bretonnel Cohen, Karin Verspoor, Helen L Johnson, Christophe Roeder, Philip Ogren, William A Baumgartner Jr, Elizabeth White, and Lawrence Hunter. High-precision biological event extraction with a concept recognizer. In *BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 50–58, 2009.

[21] Coleridge Initiative, 2020. `https://www.kaggle.com/c/coleridgeinitiative-show-us-the-data`.

[22] WHO COVID. COVID Dashboard. *Geneva: World Health Organization*, 2023. `https://data.who.int/dashboards/covid19/`.

[23] Dimitrije Curcic. Number of academic papers published per year, 2023. `https://wordsrated.com/number-of-academic-papers-published-per-year/`.

[24] Debasmita Das, Yatin Katyal, Janu Verma, Shashank Dubey, AakashDeep Singh, Kushagra Agarwal, Sourojit Bhaduri, and RajeshKumar Ranjan. Information retrieval and extraction on covid-19 clinical articles using graph community detection and biobert embeddings. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, 2020.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pretraining of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pretraining of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics, 2019.

[27] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[28] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10, 2014.

[29] Xin Luna Dong. Building a broad knowledge graph for products. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 25–25. IEEE, 2019.

[30] Matthew Duggan. System and method for generating unique and persistent identifiers, Jan 10 2008. US Patent App. 11/444,887.

[31] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-REx: A large scale alignment of natural language with knowledge base triples. In *11th International Conference on Language Resources and Evaluation*. European Language Resources Association, 2018.

[32] Santo Fortunato, Carl T. Bergstrom, Katy Börner, James A. Evans, Dirk Helbing, Staša Milojević, Alexander M. Petersen, Filippo Radicchi, Roberta Sinatra, Brian Uzzi, Alessandro Vespignani, Ludo Waltman, Dashun Wang, and Albert-László Barabási. Science of science. *Science*, 359(6379), 2018.

[33] Carol Friedman, Pauline Kra, Hong Yu, Michael Krauthammer, and Andrey Rzhetsky. GENIES: A natural-language processing system for the extraction of molecular pathways from journal articles. In *ISMB (supplement of bioinformatics)*, pages 74–82, 2001.

[34] Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[35] Kuzman Ganchev, Fernando Pereira, Mark Mandel, Steven Carroll, and Peter White. Semi-automated named entity annotation. In *Proceedings of the linguistic annotation workshop*, pages 53–56, 2007.

[36] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[37] Basilios Gatos, Dimitrios Danatsas, Ioannis Pratikakis, and Stavros J Perantonis. Automatic table detection in document images. In *International Conference on Pattern Recognition and Image Analysis*, pages 609–618. Springer, 2005.

[38] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Workshop on Automatic Speech Recognition and Understanding*, pages 273–278. IEEE, 2013.

[39] GROBID. `https://github.com/kermitt2/grobid`, 2008–2022.

[40] Paul Groth, Michael Lauruhn, Antony Scerri, and Ron Daniel Jr. Open information extraction on scientific text: An evaluation. *arXiv preprint arXiv:1802.05574*, 2018.

[41] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23, 2021.

[42] Tanishq Gupta, Mohd Zaki, NM Anoop Krishnan, and Mausam. Matscibert: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1):102, 2022.

[43] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 775–784, 2011.

[44] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *5th International Workshop on Semantic Evaluation*, pages 33–38. Association for Computational Linguistics, 2010.

[45] Florian Hillen and Bernhard Höfle. Geo-reCAPTCHA: Crowdsourcing large amounts of geographic information from earth observation data. *International Journal of Applied Earth Observation and Geoinformation*, 40:29–38, 2015.

[46] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl,

Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[47] Zhi Hong, J Gregory Pauloski, Logan Ward, Kyle Chard, Ben Blaiszik, and Ian Foster. AI-and HPC-enabled lead generation for SARS-CoV-2: Models and processes to extract druglike molecules contained in natural language text. *arXiv preprint arXiv:2101.04617*, 2021.

[48] Matthew Honnibal and Ines Montani. SpaCy pretrained English models, 2020. `https://spacy.io/models/en`.

[49] Shu Huang and Jacqueline M Cole. Batterybert: A pretrained language model for battery database enhancement. *Journal of Chemical Information and Modeling*, 62(24):6365–6377, 2022.

[50] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 105–113, 2019.

[51] HuggingFace. English wikipedia corpus. `https://huggingface.co/datasets/wikipedia`, 2020. [Online; accessed 08-January-2022].

[52] IMDb. Imdb statistics. `https://www.imdb.com/pressroom/stats/`, 2022. [Online; accessed Oct-06-2022].

[53] Hideki Isozaki. Japanese named entity recognition based on a simple rule generator and decision tree learning. In *39th Annual Meeting on Association for Computational Linguistics*, pages 314–321. Association for Computational Linguistics, 2001.

[54] Yohan Jo, Elijah Mayfield, Chris Reed, and Eduard Hovy. Machine-aided annotation for fine-grained proposition types in argumentation. In *12th Language Resources and Evaluation Conference*, pages 1008–1018, 2020.

[55] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[56] Isaak Kavasidis, Carmelo Pino, Simone Palazzo, Francesco Rundo, Daniela Giordano, P Messina, and Concetto Spampinato. A saliency-based convolutional neural network for table and chart detection in digitized documents. In *International Conference on Image Analysis and Processing*, pages 292–302. Springer, 2019.

[57] Edward Kim, Kevin Huang, Alex Tomala, Sara Matthews, Emma Strubell, Adam Saunders, Andrew McCallum, and Elsa Olivetti. Machine-learned and codified synthesis parameters of oxide materials. *Scientific Data*, 4(1):1–9, 2017.

[58] Edward Kim, Zach Jensen, Alexander van Grootel, Kevin Huang, Matthew Staib, Sheshera Mysore, Haw-Shiuan Chang, Emma Strubell, Andrew McCallum, Stefanie Jegelka, and Elsa Olivetti. Inorganic materials synthesis planning with literature-trained neural networks. *Journal of Chemical Information and Modeling*, 60(3):1194–1201, 2020.

[59] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at JNLPBA. In *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75, 2004.

[60] Austin C Kozlowski, Matt Taddy, and James A Evans. The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review*, 84(5):905–949, 2019.

[61] Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. CHEMDNER: The drugs and chemical names extraction challenge. *Journal of Cheminformatics*, 7(1):S1, 2015.

[62] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. SystemT: A system for declarative information extraction. *ACM SIGMOD Record*, 37(4):7–13, 2009.

[63] Rachael Lammey. Crossref text and data mining services. *Science Editing*, 2(1):22–27, 2015.

[64] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[65] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020.

[66] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[67] Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. BioCreative V CDR task corpus: A resource for chemical disease relation extraction. *Database*, 2016, 2016.

[68] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[69] Ying Liu, Kun Bai, Prasenjit Mitra, and C Lee Giles. Tableseer: Automatic table metadata extraction and searching in digital libraries. In *7th ACM/IEEE-CS Joint Conference on Digital libraries*, pages 91–100, 2007.

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[71] Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. Zeroshotceres: Zero-shot relation extraction from semi-structured webpages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8105–8117, 2020.

[72] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232. Association for Computational Linguistics, 2018.

[73] Kevin Lybarger, Mari Ostendorf, and Meliha Yetisgen. Annotating social determinants of health using active learning, and characterizing determinants using neural event extraction. *Journal of Biomedical Informatics*, 113:103631, 2021.

[74] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, 2011.

[75] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. Technical Report MS-CIS-93-8, University of Pennsylvania, Department of Computer and Information Science, 1993.

[76] Brigitte Mathiak and Katarina Boland. Challenges in matching dataset citation strings to datasets in social science. *D-Lib Magazine*, 21(1/2):23–28, 2015.

[77] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[78] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *International Conference on Language Resources and Evaluation*, 2018.

[79] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[80] Marcin Michał Mirończuk. The BigGrams: The semi-supervised information extraction system from HTML: An improvement in the wrapper induction. *Knowledge and Information Systems*, 54(3):711–776, 2018.

[81] Erin Murphy. *Ensemble labeling towards scientific information extraction (ELSIE).* PhD thesis, College of Computing and Digital Media, 2020.

[82] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[83] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Information extraction from scientific articles: a survey. *Scientometrics*, 117:1931–1990, 2018.

[84] National Institutes of Health. U.S. COVID-19 Clinical Trials, 2020. `https://clinicaltrials.gov/ct2/results?cond=covid&cntry=US`.

[85] Quang Long Nguyen, Domonkos Tikk, and Ulf Leser. Simple tricks for improving pattern-based information extraction from the biomedical literature. *Journal of Biomedical Semantics*, 1(1):1–17, 2010.

[86] NVIDIA. NVIDIA Apex (a PyTorch extension), 2017. `https://github.com/NVIDIA/apex`.

[87] Elsa A Olivetti, Jacqueline M Cole, Edward Kim, Olga Kononova, Gerbrand Ceder, Thomas Yong-Jin Han, and Anna M Hiszpanski. Data-driven materials research enabled by natural language processing and information extraction. *Applied Physics Reviews*, 7(4):041317, 2020.

[88] Toshihide Ono, Haretsugu Hishigaki, Akira Tanigami, and Toshihisa Takagi. Automated extraction of information on protein–protein interactions from the biological literature. *Bioinformatics*, 17(2):155–161, 2001.

[89] OpenAI. Gpt-4 technical report, 2023.

[90] Shingo Otsuka, Isao Kuwajima, Junko Hosoya, Yibin Xu, and Masayoshi Yamazaki. Polyinfo: Polymer database for polymeric materials design. In *2011 International Conference on Emerging Intelligent Data and Web Technologies*, pages 22–29. IEEE, 2011.

[91] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *18th BioNLP Workshop and Shared Task*, pages 58–65. Association for Computational Linguistics, 2019.

[92] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[93] Violaine Pillet. *Méthodologie d'extraction automatique d'information à partir de la littérature scientifique en vue d'alimenter un nouveau système d'information: application à la génétique moléculaire pour l'extraction d'information sur les interactions*. PhD thesis, Univ. d'Aix-Marseille 3, 2000.

[94] Prodi.gy. Prodi.gy: An annotation tool for AI, Machine Learning, and NLP. `https://prodi.gy`, 2021. Online; accessed 02-May-2021.

[95] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[96] Cartic Ramakrishnan, Abhishek Patnia, Eduard Hovy, and Gully APC Burns. Layout-aware text extraction from full-text PDF of scientific articles. *Source Code for Biology and Medicine*, 7(1):1–10, 2012.

[97] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *11th National Conference on Artificial intelligence*, pages 811–816, 1993.

[98] Ellen Riloff, Janyce Wiebe, and William Phillips. Exploiting subjectivity classification to improve information extraction. In *AAAI*, pages 1106–1111, 2005.

[99] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *7th Conference on Natural Language Learning*, pages 25–32, 2003.

[100] Richard J. Roberts. PubMed Central: The GenBank of the published literature. *Proceedings of the National Academy of Sciences*, 98(2):381–382, 2001.

[101] Corby Rosset. Turing-NLG: A 17-billion-parameter language model by Microsoft. *Microsoft Research Blog*, feb 2020. `https://bit.ly/3eF1coS`.

[102] Arpita Roy and Shimei Pan. Incorporating medical knowledge in bert for clinical relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5357–5366, 2021.

[103] Andrey Rzhetsky, Ivan Iossifov, Tomohiro Koike, Michael Krauthammer, Pauline Kra, Mitzi Morris, Hong Yu, Pablo Ariel Duboué, Wubin Weng, W John Wilbur, et al. GeneWays: A system for extracting, analyzing, visualizing, and integrating molecular pathway data. *Journal of biomedical informatics*, 37(1):43–53, 2004.

[104] Aymen Al Saadi, Dario Alfe, Yadu Babuji, Agastya Bhati, Ben Blaiszik, Thomas Brettin, Kyle Chard, Ryan Chard, Peter Coveney, Anda Trifan, et al. IMPECCABLE: Integrated modeling pipeline for COVID cure by assessing better leads. *arXiv preprint arXiv:2010.06574*, 2020.

[105] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for Twitter sentiment analysis: A survey and a new dataset, the STS-Gold. In *1st International Workshop on Emotion and Sentiment in Social and Expressive Media: Approaches and Perspectives from AI*, 2013.

[106] Erik F Sang and Fien De Meulder. CoNLL eval script, 2003. `https://www.clips.uantwerpen.be/conll2000/chunking/output.html`.

[107] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

[108] Khaled Shaalan and Hafsa Raza. Arabic named entity recognition from diverse text types. In *International Conference on Natural Language Processing*, pages 440–451. Springer, 2008.

[109] Yatian Shen and Xuan-Jing Huang. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2526–2536, 2016.

[110] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of Microsoft Academic Service (MAS) and applications. In *24th International Conference on World Wide Web*, pages 243–246, 2015.

[111] Hannah Smith, Zeyu Zhang, John Culnan, and Peter Jansen. ScienceExamCER: A high-density fine-grained science-domain corpus for common entity recognition. In *12th Language Resources and Evaluation Conference*, pages 4529–4546. European Language Resources Association, 2019.

[112] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272, 1999.

[113] Daniel Sui, Sarah Elwood, and Michael Goodchild. *Crowdsourcing Geographic Knowledge: Colunteered Geographic Information (VGI) in Theory and Practice.* Springer Science & Business Media, 2012.

[114] Weiwei Sun, Xiaochang Peng, and Xiaojun Wan. Capturing long-distance dependencies in sequence models: A case study of chinese part-of-speech tagging. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 180–188, 2013.

[115] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *13th Annual Conference of the International Speech Communication Association*, 2012.

[116] Matthew C Swain and Jacqueline M Cole. ChemDataExtractor: A toolkit for automated extraction of chemical information from the scientific literature. *Journal of Chemical Information and Modeling*, 56(10):1894–1904, 2016.

[117] Roselyne Tchoua, Aswathy Ajith, Zhi Hong, Logan Ward, Kyle Chard, Debra Audus, Shrayesh Patel, Juan de Pablo, and Ian Foster. Active learning yields better training data for scientific named entity recognition. In *15th International Conference on eScience*, pages 126–135. IEEE, 2019.

[118] Roselyne Tchoua, Zhi Hong, Debra Audus, Shrayesh Patel, Logan Ward, Kyle Chard, Juan De Pablo, and Ian Foster. Developing databases for polymer informatics. *Bulletin of the American Physical Society*, 65, 2020.

[119] Roselyne B Tchoua, Aswathy Ajith, Zhi Hong, Logan T Ward, Kyle Chard, Alexander Belikov, Debra J Audus, Shrayesh Patel, Juan J de Pablo, and Ian T Foster. Creating

training data for scientific named entity recognition with minimal human effort. In *International Conference on Computational Science*, pages 398–411. Springer, 2019.

[120] Roselyne B. Tchoua, Kyle Chard, Debra Audus, Jian Qin, Juan de Pablo, and Ian Foster. A hybrid human-computer approach to the extraction of scientific facts from the literature. *Procedia Computer Science*, 80:386–397, 2016.

[121] Roselyne B Tchoua, Kyle Chard, Debra J Audus, Logan T Ward, Joshua Lequieu, Juan J De Pablo, and Ian T Foster. Towards a hybrid human-computer scientific information extraction pipeline. In *IEEE 13th International Conference on e-Science*, pages 109–118. IEEE, 2017.

[122] Roselyne B Tchoua, Jian Qin, Debra J Audus, Kyle Chard, Ian T Foster, and Juan de Pablo. Blending education and polymer science: Semiautomated creation of a thermodynamic property database. *Journal of Chemical Education*, 93(9):1561–1568, 2016.

[123] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *7th Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[124] Andrew Trask, Phil Michalak, and John Liu. sense2vec–a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*, 2015.

[125] Amalie Trewartha, Nicholas Walker, Haoyan Huo, Sanghoon Lee, Kevin Cruse, John Dagdelen, Alexander Dunn, Kristin A. Persson, Gerbrand Ceder, and Anubhav Jain. Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. *Patterns*, 3(4):100488, 2022.

[126] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[127] Nicholas Walker, Amalie Trewartha, Haoyan Huo, Sanghoon Lee, Kevin Cruse, John Dagdelen, Alexander Dunn, Kristin Persson, Gerbrand Ceder, and Anubhav Jain. The impact of domain-specific pre-training on named entity recognition tasks in materials science. *Available at SSRN 3950755*, 2021.

[128] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. CORD-19: The COVID-19 open research dataset. Preprint 2004.10706, ArXiv, 2020.

[129] Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al.

Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23, 2013.

[130] Leigh Weston, Vahe Tshitoyan, John Dagdelen, Olga Kononova, Amalie Trewartha, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Named entity recognition and normalization applied to large-scale information extraction from the materials science literature. *Journal of chemical information and modeling*, 59(9):3692–3702, 2019.

[131] Karen White. Publications output: US trends and international comparisons. Technical report, National Science Foundation, 2019. `https://ncses.nsf.gov/pubs/nsb20206/`.

[132] Janyce Wiebe and Ellen Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 486–497. Springer, 2005.

[133] Janyce Wiebe and Ellen Riloff. Finding mutual benefit between subjectivity analysis and information extraction. *IEEE Transactions on Affective Computing*, 2(4):175–191, 2011.

[134] Andrea Wiggins and Kevin Crowston. From conservation to crowdsourcing: A typology of citizen science. In *44th Hawaii international conference on system sciences*, pages 1–10. IEEE, 2011.

[135] Tong Xie, Yuwei Wan, Ke Lu, Wenjie Zhang, Chunyu Kit, and Bram Hoex. Tokenizer effect on functional material prediction: Investigating contextual word embeddings for knowledge discovery. In *AI for Accelerated Materials Design-NeurIPS 2023 Workshop*, 2023.

[136] S. Yan, W. S. Spangler, and Y. Chen. Chemical name extraction based on automatic training data generation and rich feature set. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(5):1218–1233, 2013.

[137] Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. Textrunner: Open information extraction on the web. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 25–26, 2007.

[138] Antonio Jimeno Yepes, Andrew MacKinlay, Natalie Gunn, Christine Schieber, Noel Faux, Matthew Downton, Benjamin Goudey, and Richard L Martin. A hybrid approach for automated mutation annotation of the extended human mutation landscape in scientific literature. In *AMIA Annual Symposium Proceedings*, volume 2018, page 616. American Medical Informatics Association, 2018.

[139] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Conference on Empirical Methods in Natural Language Processing*, pages 35–45, 2017.

[140] Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, and Kangfei Zhao. Natural language question/answering: Let users talk with the knowledge graph. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 217–226, 2017.

[141] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *IEEE International Conference on Computer Vision*, pages 19–27, 2015.