INFORMATION EXTRACTION ON SCIENTIFIC LITERATURE UNDER LIMITED SUPERVISION

A Dissertation
Presented to
The Academic Faculty

By

Fan Bai

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

December 2023

INFORMATION EXTRACTION ON SCIENTIFIC LITERATURE UNDER LIMITED SUPERVISION

Thesis committee:

Dr. Alan Ritter, advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Gabriel Stanovsky
School of Computer Science and Engineering
Hebrew University of Jerusalem

Dr. Wei Xu
School of Interactive Computing
Georgia Institute of Technology

Dr. Hoifung Poon
Microsoft Health Futures

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Date approved: October 30, 2023

Dedicated to my parents.

ACKNOWLEDGMENTS

As with all the Ph.D. stories, my Ph.D. journey has its ups and downs, and will be a forever cherished memory for the rest of my life.

First and foremost, I would like to give special thanks to my advisor, Professor Alan Ritter. Alan not only introduced me to the fascinating world of Natural Language Processing, particularly information extraction, which I deeply cherish, but also meticulously guided me through the nuances of research - from designing experiments to presenting papers. His mentorship has been indispensable to the completion of my Ph.D. I would also like to thank the members of my thesis committee – Professor Gabriel Stanovsky, Professor Wei Xu, Professor Zsolt Kira, and Dr. Hoifung Poon – for their time and insightful feedback which helped improve my dissertation. I extend my gratitude to my collaborators: Dayne Freitag, Ronen Tamari, and Peter Madrid. Collaborating with these exceptional minds during my Ph.D. has been a privilege, and their insights have been invaluable.

My labmates – Junmo Kang, Max Le, Yang Chen, Yao Dou, Ruohao Guo, Jiang Chao, Mounica Maddela, Tarek Naous, Duong Le, Jeniya Tabassum, and Wuwei Lan – have been more than just colleagues; they were friends who shared both the good times and the tough moments of Ph.D. life. I hope all other group members who are still pursuing their Ph.D. will make steady research progress and graduate smoothly. I am also grateful to all my friends from Ohio State and Georgia Tech: Zhen Wang, Cheng Zhang, Ashutosh Baheti, Shuheng Liu, Chang Liu, Yingjun Mou, Qi Wang, Xianxing Zhang, and Jeremiah Coholich. They provided strong support and made my graduate life feel like home. I hope our friendship lasts a lifetime.

Finally, I want to express my deepest thanks to my parents, Jinyu Xie and Mingyu Bai. They have always been there for me, supporting and encouraging me to pursue my dreams. Without them, I wouldn't have been able to complete my Ph.D. or be who I am today. I hope to continue to make them proud.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The exponential growth of scientific literature presents both challenges and opportunities for researchers across various disciplines. Effectively extracting pertinent information from this extensive corpus is crucial for advancing knowledge, enhancing collaboration, and driving innovation. However, manual extraction is a laborious and time-consuming process, underscoring the demand for automated solutions. Information extraction (IE), a subfield of natural language processing (NLP) focused on automatically extracting structured information from unstructured data sources, plays a crucial role in addressing this challenge. Despite their success, many IE methods often require substantial human-annotated data, which might not be easily accessible, particularly in specialized scientific domains. This highlights the need for adaptable and robust techniques capable of functioning with limited supervision.

In this thesis, we study the task of *information extraction on scientific literature*, particularly addressing the challenge of *limited (human) supervision*. Specifically, our work has delved into four key dimensions of this problem. First, we explore the potential of harnessing easily accessible resources, like knowledge bases, to develop IE systems without direct human supervision. Second, we examine the use of pre-trained language models to create effective and efficient scientific IE systems, experimenting with various fine-tuning architectures and learning strategies. Next, we investigate the balance between the labor expenditure of human annotation and the computational cost linked with domain-specific pre-training, to achieve optimal performance under the budget constraints. Lastly, we capitalize on the emerging capabilities of large pre-trained language models by showcasing how information extraction can be achieved solely based on a human-crafted data schema. Through these explorations, this thesis aims to lay a solid foundation for the continued advancement of scientific IE under limited supervision.

CHAPTER 1
INTRODUCTION

The rapidly growing volume of scientific literature presents both challenges and opportunities for researchers and stakeholders in various disciplines [1, 2]. Efficiently extracting relevant and valuable information from this vast corpus is crucial for advancing knowledge, facilitating collaboration, and driving innovation. However, manual extraction of such information is both time-consuming and labor-intensive, highlighting the importance of automated solutions to alleviate the workload and expedite the discovery of novel insights.

Information extraction (IE), a sub-field of natural language processing (NLP) that involves the automatic extraction of structured information from unstructured data sources, plays a crucial role in addressing this challenge [3, 4, 5]. Although many IE methods have achieved impressive success in various information extraction tasks within scientific literature [6, 7], they often demand a significant quantity of human-annotated data. Unfortunately, such data may not always be accessible for every domain or context, particularly in specialized scientific fields. This constraint emphasizes the need for developing more adaptable and robust techniques capable of functioning under limited supervision.

In this thesis, we focus on the task of *information extraction on scientific literature* with an emphasis on addressing the challenge of *limited (human) supervision*. Our work has delved into four distinct aspects of this problem: 1) harnessing easily accessible sources, such as knowledge bases, to learn IE systems without direct human supervision; 2) exploring the use of pre-trained language models to create effective and efficient scientific IE systems; 3) examining the trade-off between the cost of human annotation and the advantages of self-supervised domain-specific pre-training in order to achieve optimal performance; and 4) leveraging the emerging capabilities of large pre-trained language models to perform information extraction with only a human-authored extraction schema.

1.1 Thesis Structure

Chapter 2 presents DNMAR, a novel approach for minimally supervised relation extraction that aims to predict sentence-level relation mentions using only proposition-level supervision from a knowledge base (KB). By explicitly reasoning about missing data during learning, the proposed approach enables large-scale training of neural networks while mitigating the issue of label noise inherent in distant supervision. DNMAR achieves state-of-the-art results on minimally supervised sentential relation extraction, outperforming numerous baselines, including a competitive approach that uses the attention layer of a purely neural model.

Chapter 3 delves into leveraging pre-trained language models for developing efficient and effective scientific information extraction systems. Our experiments encompass two scenarios: standard supervised learning and few-shot learning. In the former, we focus on task of process execution graph extraction, comparing the efficacy of pipeline-based and end-to-end multi-task fine-tuning architectures. The end-to-end approach exhibits superior performance, especially in minimizing error propagation. In the few-shot learning setting, we investigate anaphora resolution within scientific protocols, a task typically hampered by insufficient annotated data. Our exploration confirms the practicality of distilling robust, compact models from larger in-context learned models, marking a significant stride towards efficient scientific information extraction with limited supervision.

In Chapter 4, we explore the crucial question: "When adapting to a new scientific domain: given a fixed budget, what combination of data annotation and pre-training should be selected to maximize performance?" We frame domain adaptation with a constrained budget as a consumer choice problem, aiming to identify the optimal mix of data annotation and pre-training under varying budget constraints. We measure the annotation costs of three procedural text datasets and the pre-training costs of several in-domain language models to assess the utility of different combinations. The findings reveal that for small budgets,

allocating all funds to annotation leads to the best performance. However, once the budget increases, a combination of data annotation and in-domain pre-training yields better results. Additionally, built upon our created datasets and pre-trained models, we present SYNKB, an open-source system for large-scale extraction and querying of chemical synthesis procedures. SYNKB provides efficient searches against semantic action graphs and chemical reaction slots derived from 6 million chemical synthesis procedures, rivaling Reaxys, one of the leading commercial databases of reaction information.

Finally, in Chapter 5, we propose *schema-driven information extraction*, a new task that transforms tabular data into structured records, following a human-authored schema. To assess various LLM's capabilities on this task, we develop a benchmark composed of tables from four diverse domains: machine learning papers, chemistry literature, material science journals, and webpages. Accompanying the benchmark, we present INSTRUCTE, an extraction method based on instruction-tuned LLMs. INSTRUCTE requires only a human-authored extraction schema, removing the need for custom pipelines for each new domain or data format. INSTRUCTE demonstrates competitive performance without task-specific labels, achieving F_1 scores ranging from 74.2 to 96.1. Moreover, we validate the feasibility of distilling compact models from INSTRUCTE to minimize extraction costs and reduce API reliance.

CHAPTER 2

STRUCTURED MINIMALLY SUPERVISED LEARNING FOR NEURAL RELATION EXTRACTION

This chapter contains material that was originally published in [8].

Traditional datasets for information extraction are based on expensive and time-consuming human annotation [9] and are therefore relatively small. Distant supervision [10], a technique that uses existing knowledge bases such as Freebase or Wikipedia as a source of weak supervision, enables learning from large quantities of unlabeled text and is a promising approach for scaling up, especially in specialized arenas, such as scientific domains. Previous work has shown promising results from large-scale training of neural networks for relation extraction [11, 12]. There are, however, significant challenges due to the inherent noise in distant supervision. For example, Riedel et al. [13] showed that, when learning using distant supervision from a knowledge base, the portion of mis-labeled examples can vary from 13% to 31%. To address this issue, another line of work has explored *structured* learning methods that introduce latent variables. An example is MultiR [14], which is based on a joint model of relations between entities in a knowledge base and those mentioned in text. This structured learning approach has a number of advantages; for example, by integrating inference into the learning procedure it has the potential to overcome the challenge of missing facts by ignoring the knowledge base when mention-level classifiers have high confidence [15, 16]. Prior work on structured learning from minimal supervision has leveraged sparse feature representations, however, and has therefore not benefited from learned representations, which have recently achieved state-of-the-art results on a broad range of NLP tasks.

In this chapter, we present an approach that combines the benefits of structured and neural methods for minimally supervised relation extraction. Our proposed model learns

sentence representations that are computed by a 1D convolutional neural network [17] and are used to define potentials over latent relation mention variables. These mention-level variables are related to observed facts in a KB using a set of deterministic factors, followed by pairwise potentials that encourage agreement between extracted propositions and observed facts, but also enable inference to override these soft constraints during learning, allowing for the possibility of missing information. Because marginal inference is intractable in this model, a MAP-based approach to learning is applied [18].

Our approach is related to recent work structured learning with end-to-end learned representations, including Structured Prediction Energy Networks (SPENs) [19]; the key differences are the application to minimally supervised relation extraction and the inclusion of latent variables with deterministic factors, which we demonstrate enables effective learning in the presence of missing data in distant supervision. Our proposed method achieves state-of-the-art results on minimally supervised sentential relation extraction, outperforming a number of baselines including one that leverages the attention layer of a purely neural model [20].

2.1 A Latent Variable Model for Neural Relation Extraction

In this section we present our model, which combines continuous representations with structured learning. We first review the problem setting and introduce notation, next we present our approach to extracting feature representations which is based on the piecewise convolutional neural network (PCNN) model of Zeng et. al. [12] and includes positional embeddings [17]. Finally we describe how this can be combined with structured latent variable models that reason about overlapping relations and missing data during learning.

2.1.1 Assumptions and Problem Formulation

Given a set of sentences, $\mathbf{s} = s_1, s_2 \ldots, s_n$ that mention a pair of knowledge base entities e_1 and e_2 (dyad), our goal is to predict which relation, r, is mentioned between e_1 and e_2

in the context of each sentence, represented by a set of hidden variables, $\mathbf{z} = z_1, z_2, \ldots z_n$. Relations are selected from a fixed set drawn from a knowledge base, in addition to *NA* (no relation). Minimally supervised learning is more difficult than supervised relation extraction, because we do not have direct access to relation labels on the training sentences. Instead, during learning, we are only provided with information about what relations hold between e_1 and e_2 according to the KB. The problem is further complicated by the fact that most KBs are highly incomplete (this is the reason we want to extend them by extracting information from text in the first place), which effectively leads to false-negatives during learning. Furthermore, there are many overlapping relations between dyads, so it is easy for a model trained using minimal supervision from a KB to confuse these relationships. All of these issues are addressed to some degree by the structured learning approach that we present in subsection 2.1.3. First, however we present our approach to feature representation based on convolutional neural networks.

2.1.2 Mention Representation

In the following section we review the Piecewise CNN (PCNN) architecture, first proposed by Zeng et. al. [12], which is used as the basis for our feature representation.

Input Representation: A sentence, s_i consisting of l words is represented by two types of embeddings: word embeddings, E_i, and position embeddings, P_i relative to the entity pair. Following Lin et. al. [20], word embeddings were initialized by running Word2Vec on the New York Times corpus and later fine-tuned; position embeddings encode the position of the word relative to KB entities, e_1 and e_2, mentioned in the sentence. The form of input sentence representation is $w_1, w_2, \cdots, w_l$, where $w_i \in \mathbb{R}^d$. The dimension of embedding at each word position is equal to the word embedding dimension plus two times the position embedding size (one position is encoded for each entity).

Convolution: Given an input sentence representation, we perform 1D convolution within a window of length l to extract local features. Assume we have d_f convolutional filters

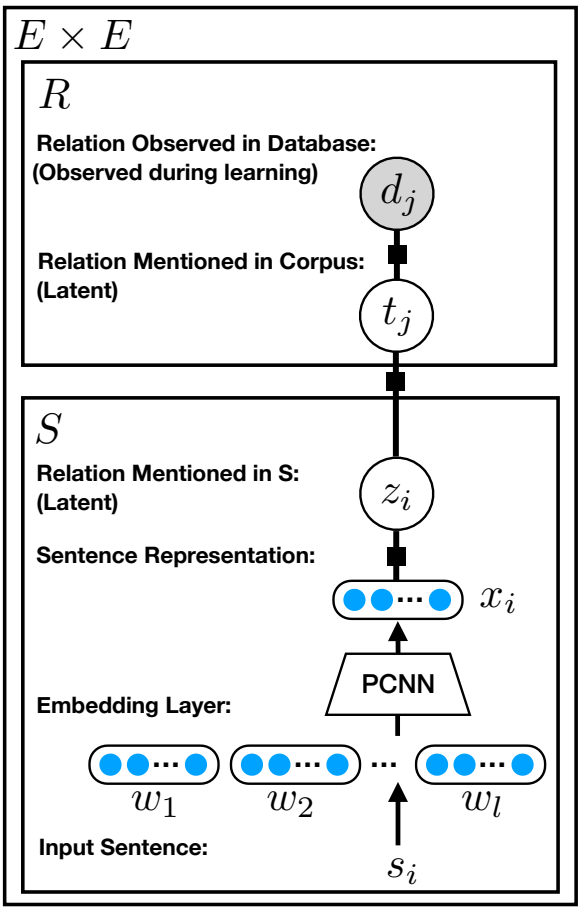


Figure 2.1: Plate representation of our proposed model. Plates represent replication; $E \times E$ is the number of entity pairs in the dataset, S is the number of sentences mentioning each entity pair and R is the number of relations. Arrows represent functions from input to output. Latent variables are represented as unshaded nodes. Factors over variables are represented as boxes.

$(F = \{f_1, f_2, \cdots, f_{d_f}\}, f_i \in \mathbb{R}^{l \times d})$. The output of the i-th convolutional filter within the j-th window is:

$$c_{ij} = f_i \cdot w_{j-l+1:j} + b \quad (1 \leqslant j \leqslant m + l - 1)$$

Where b is a bias term. We use zero padding when the window slides out of the sentence boundaries.

Piecewise Max Pooling: The output of the convolutional layer c_i is separated into three parts (c_{i1}, c_{i2}, c_{i3}) using the positions of the two entities in the sentence. Max pooling over

time is then applied to each of these parts, followed by an elementwise tanh. The final sentence vector is defined as follows:

$$[x]_{ik} = \tanh(\max_j(c_{ikj})) \quad (1 \leqslant i \leqslant d_f, 1 \leqslant k \leqslant 3)$$

2.1.3 Structured Minimally Supervised Learning

Our proposed model is based on the PCNN representations described above, in addition to a latent variable model that reasons about missing data and ambiguous relations during learning and is illustrated in Figure 2.1. The embedding for sentence i, is used to define a factor over the ith input sentence and latent relation mention variable z_i:

$$\phi_{\text{PCNN}}(s_i, z_i) = e^{x_i \cdot \theta_{z_i}}$$

where x_i is the representation for sentence s_i, as encoded by the piecewise CNN.

Another set of factors, ϕ_{OR}, link the sentence-level mention variables, z_i, to aggregate-level variables t_j, representing whether relation j is mentioned between e_1 and e_2 in text. This is modeled using a deterministic OR:

$$\phi_{\text{OR}}(\mathbf{z}, t_j) = \mathbf{1}_{\neg t_j \oplus \exists i: j = z_i}$$

where $\mathbf{1}_x$ is an indicator function that takes the value 1 when x is true. The choice of deterministic OR can be interpreted intuitively as follows: if a proposition is true according to t_j, then it must be extracted from at least one sentence in the training corpus, on the other hand, if it is false, no sentences in the corpus can mention it.

Finally, we incorporate a set of factors that penalize disagreement between observed relations in the KB, d_j, and latent variables t_j, which represent whether relation j was extracted from the text. The penalties for disagreement with the KB are hyperparameters that are adjusted on held-out development data and incorporate entity frequency information

from the KB, to model the intuition that more popular entities are less likely to have missing facts:

$$\phi_{\text{A}}(t_j, d_j) = \begin{cases} e^{-\alpha_T}, & \text{if } t_j = 0 \text{ and } d_j = 1 \\ e^{-\alpha_D}, & \text{if } t_j = 1 \text{ and } d_j = 0 \\ 1, & \text{otherwise} \end{cases}$$

Putting everything together, the (unnormalized) joint distribution over $\mathbf{t}$, $\mathbf{d}$ and $\mathbf{z}$ conditioned on sentences $\mathbf{s}$ mentioning a dyad is defined as follows:

$$\begin{aligned} P(\mathbf{d}, \mathbf{t}, \mathbf{z}|\mathbf{s}) &\propto \prod_{i=1}^{|\mathbf{s}|} \phi_{\text{PCNN}}(s_i, z_i) \times \\ &\quad \Big(\prod_{j=1}^{|\mathbf{r}|} \phi_{\text{OR}}(\mathbf{z}, t_j) \phi_{\text{A}}(t_j, d_j) \Big)^{\mu} \\ &= \exp(S_\theta(\mathbf{s}, \mathbf{z}, \mathbf{t}, \mathbf{d})) \end{aligned} \tag{2.1}$$

Here, μ is a tunable hyperparameter to adjust impact of the disagreement penalty, and $S_\theta(\cdot)$ is the model score for a joint configuration of variables, which corresponds to the log of the unnormalized probability.

A standard conditional random field (CRF) formulation would optimize model parameters, θ so as to maximize marginal probability of the observed KB relations, $\mathbf{d}$ conditioned on observed sentences, $\mathbf{s}$:

$$P(\mathbf{d}|\mathbf{s}) = \sum_{\mathbf{z},\mathbf{t}} P(\mathbf{d}, \mathbf{t}, \mathbf{z}|\mathbf{s})$$

Computing gradients with respect to $P(\mathbf{d}|\mathbf{s})$ (and marginalizing out $\mathbf{z}$ and $\mathbf{t}$) is computationally intractable, so instead we propose an approach that uses maximum-a-posteriori (MAP) parameter learning [18] and is inspired by the latent structured SVM [21].

Given a large text corpus in which a set of sentences, $\mathbf{s}$ mention a specific pair of entities (e_1, e_2) and a set of relations $\mathbf{d}$ hold between e_1 and e_2, our goal is to minimize the structured

hinge loss: $L_{\text{SH}}(\theta) =$

$$\max\left\{\begin{array}{c} \max\limits_{\mathbf{z}_e^*,\mathbf{t}_e^*,\mathbf{d}_e^*}\left[S_\theta(\mathbf{s},\mathbf{z}_e^*,\mathbf{t}_e^*,\mathbf{d}_e^*)+l_{\text{Ham}}(\mathbf{d}_e^*,\mathbf{d})\right] \\ -\max\limits_{\mathbf{z}_g^*,\mathbf{t}_g^*}\left[S_\theta(\mathbf{s},\mathbf{z}_g^*,\mathbf{t}_g^*,\mathbf{d})\right] \end{array}\right\} \tag{2.2}$$

Where $l_{\text{Ham}}(\mathbf{d}_e^*,\mathbf{d})$ is the Hamming distance between the bit vector corresponding to the set of observed relations holding between (e_1,e_2) in the KB and those predicted by the model. Minimizing $L_{\text{SH}}(\theta)$ can be understood intuitively as adjusting the parameters so that configurations consistent with observed relations in the KB, $\mathbf{d}$, achieve a higher model score than those with a large hamming distance from the observed configuration. $\mathbf{z}_e^*$ corresponds to the most confusing configuration of the sentence-level relation mention variables (i.e. one that has a large score and also a large Hamming loss) and $\mathbf{z}_g^*$ corresponds to the best configuration that is consistent with the observed relations in the KB.

This objective can be minimized using stochastic subgradient descent. Fixing $\mathbf{z}_g^*$ and $\mathbf{z}_e^*$ to their maximum values in Equation 2.2, subgradients with respect to the parameters can be computed as follows:

$$\nabla_\theta L_{\text{SH}}(\theta) = \begin{cases} \mathbf{0} & \text{if } L_{\text{SH}}(\theta)\leqslant 0, \\ \nabla_\theta S_\theta(\mathbf{s},\mathbf{z}_e^*,\mathbf{t}_e^*,\mathbf{d}_e^*) & \\ -\nabla_\theta S_\theta(\mathbf{s},\mathbf{z}_g^*,\mathbf{t}_g^*,\mathbf{d}) & \text{otherwise} \end{cases} \tag{2.3}$$

$$= \begin{cases} \mathbf{0} & \text{if } L_{\text{SH}}(\theta)\leqslant 0, \\ \sum_i \nabla_\theta \log\phi_{\text{PCNN}}(s_i,z_{e,i}^*) & \\ -\sum_i \nabla_\theta \log\phi_{\text{PCNN}}(s_i,z_{g,i}^*) & \text{otherwise} \end{cases} \tag{2.4}$$

Because the second factor of the product in Equation 2.1 does not depend on θ, it is straightforward to compute subgradients of the scoring function, $\nabla S_\theta(\cdot)$, with fixed values of $\mathbf{z}_g^*$ and $\mathbf{z}_e^*$ using backpropagation (Equation 2.4).

Inference: The two inference problems, corresponding to maximizing over hidden variables in Equation 2.2 can be solved using a variety of solutions; we experimented with A* search over left-to-right assignments of the hidden variables. An admissible heuristic is used to upper-bound the maximum score of each partial hypothesis by maximizing over the unassigned PCNN factors, ignoring inconsistencies. This approach is guaranteed to find an optimal solution, but can be slow and memory intensive for problems with many variables. In preliminary experiments on development data, we found that local-search [22] using both relation type and mention search operators [23, 15] usually finds an optimal solution and also scales up to large training datasets; we use local search with 30 random restarts to compute argmax assignments for the hidden variables, $\mathbf{z}_g^*$ and $\mathbf{z}_e^*$, in all our experiments.

Bag-Size Weighting Function: Since the search space of the MAP inference problem increases exponentially as the number of hidden variables goes up, it becomes more difficult to find the exact argmax solution using local search, leading to increased noise in the computed gradients. To mitigate the search-error problem in large bags of sentences, we introduce a weighting function based on the bag size as follows:

$$L_{\text{SH}}(\theta)' = f(s_i) \cdot L_{\text{SH}}(\theta)$$

$$f(s_i) = \begin{cases} 1, & \text{if } |\mathbf{s_i}| < \beta_1 \\ \frac{\beta_1}{|\mathbf{s}_i|}, & \text{if } \beta_1 \leqslant |\mathbf{s}_i| \leqslant \beta_2 \\ (\frac{\beta_1}{|\mathbf{s}_i|})^2, & \text{otherwise} \end{cases}$$

where $f(s_i)$ is the bag-size weight for ith training entity pair and β_1/β_2 are two tunable bag-size thresholds. In Table 2.3 and Table 2.4, we see that this strategy significantly improves performance, especially when training on the larger NYTFB-280K dataset. We also experimented with this method for PCNN+ATT, but found that its performance did not improve.

Table 2.1: Number of entity pairs and sentences in the training portion of Riedel's HELDOUT dataset (NYTFB-68K) and Lin's dataset (NYTFB-280K).

Dataset	NYTFB-68K [13]	NYTFB-280K [20]
Entity pairs	67,946	280,275
Sentences	126,184	523,312

2.2 Experiments

In Section 2, we presented an approach that combines the benefits of PCNN representations and structured learning with latent variables for minimally supervised relation extraction. In this section we present the details of our evaluation methodology and experimental results.

Datasets: We evaluate our models on the NYT-Freebase dataset [13] which was created by aligning relational facts from Freebase with the New York Times corpus, and has been used in a broad range of prior work on minimally supervised relation extraction. Several versions of this dataset have been used in prior work; to facilitate the reproduction of prior results, we experiment with two versions of the dataset used by Riedel et. al. [13] (henceforth NYTFB-68K) and Lin et. al. [20] (NYTFB-280K). Statistics of these datasets are presented in Table A.1. A more detailed discussion about the differences between datasets used in prior work is also presented in Appendix .

Hyperparameters: Following Lin et. al. [20], we utilize word embeddings pre-trained on the NYT corpus using the word2vec tool, other parameters are initialized using the method described by Glorot and Bengio [24]. The Hoffmann et. al. sentential evaluation dataset is split into a development and test set and grid search on the development set was used to determine optimal values for the learning rate λ among $\{0.001, 0.01\}$, KB disagreement penalty scalar μ among $\{100, 200, \cdots, 2000\}$ and β_1/β_2 bag size threshold for the weighting function among $\{10, 15, \cdots, 40\}$. Other hyperparameters with fixed values are presented in Table 2.2.

Neural Baselines: To demonstrate the effectiveness of the our approach, we compare

Table 2.2: Untuned hyperparameters in our experiments.

Window length l	3
Number of convolutional filters d_f	230
Word embedding dimension d_w	50
Position embedding dimension d_p	5
Batch size B	1

against col-less universal schema [25] in addition to the PCNN+ATT model of Lin et. al. [20]. After training the Lin et. al. model to predict observed facts in the KB, we use its attention layer to make mention-level predictions as follows:

$$p(r_j|x_i) = \frac{\exp(r_j \cdot x_i)}{\sum_{k=1}^{n_r} \exp(r_k \cdot x_i)}$$

Where r_j indicates the vector representation of the jth relation.

Structured Baselines: In addition to initializing convolutional filters used in the $\phi_{\text{PCNN}}(\cdot)$ factors randomly and performing structured learning of representations as in Equation 2.4, we also experimented with variants of MultiR and DNMAR, which are based on the structured perceptron [26], using fixed sentence representations: both traditional sparse feature representations, in addition to pre-trained continuous representations generated using our best-performing reimplementation of PCNN+ATT. For the structured perceptron baselines, we also experimented with variants based on MIRA [27], which we found to provide consistent improvements. More details are provided in Appendix section A.1.

2.2.1 Sentential Evaluation

In this work, we are primarily interested in mention-level relation extraction. For our first set of experiments (Table 2.3 and Table 2.4), we use the manually annotated dataset created by [14]. Note that sentences in the Hoffman et. al. dataset were selected from the output of systems used in their evaluation, so it is possible there are high confidence predictions made by our systems that are not present. Therefore, we further validate our findings, by

Table 2.3: AUC of sentential evaluation precision / recall curves for all models trained on NYTFB-68K. Continuous sentence representation works as well as human-engineered sentence representation, and MIRA consistently helps structured perceptron training. PCNN+ATT performs competitively while our PCNNNMAR (weighted) is statistically significantly better (p-value of bootstrap is less than 0.05)

Model	Name	DEV	TEST
Fixed Sentence Representations	MultiR_sparse [14]	66.2	63.2
	MultiR_sparse_MIRA	75.3	71.6
	MultiR_continuous	74.2	68.7
	MultiR_continuous_MIRA	80.3	72.5
	DNMAR_sparse [15]	77.9	70.1
	DNMAR_sparse_MIRA	77.5	72.1
	DNMAR_continuous	80.2	70.0
	DNMAR_continuous_MIRA	82.2	74.2
Jointly Learned Representations	PCNNNMAR	82.4	83.9
	PCNNNMAR (bag-size weighting function)	**85.4**	**86.0**
Baselines	col-less universal schema [25]	63.4	61.1
	PCNN+ATT (Lin et al. [20] code)	81.4	76.4
	PCNN+ATT (our reimplementation with parameter tuning)	83.6	78.4

performing a manual inspection of the highest confidence predictions in Table 2.5.

NYTFB-68K **Results:** As illustrated in Table 2.3, simply applying structured models (MultiR and DNMAR) with pre-trained sentence representations performs competitively. MIRA provides consistent improvements for both sparse and dense representations. PCNN+ATT outperforms most latent-variable models on the sentential evaluation, we found this result to be surprising as the model was designed for extracting proposition-level facts. Col-less universal schema does not perform very well in this evaluation; this is likely due to the fact that it was developed for the KBP slot filling evaluation [28], and only uses the part of a sentence between two entities as an input representation, which can remove important context. Our proposed model, which jointly learns sentence representations using a structured latent-variable model that allows for the possiblity of missing data, achieves the best overall performance; its improvements over all baselines were found to be statistically significant according to a paired bootstrap test [29, 30].[1]

[1] p-value is less than 0.05.

Table 2.4: AUC of sentential evaluation precision / recall curves for all models trained on NYTFB-280K. Our proposed PCNNNMAR (weighted) still performs the best, and the advantage over baselines is also statistically significant (p-value of bootstrap is less than 0.05).

Model	Name	DEV	TEST
Fixed Sentence Representations	MultiR_continuous	72.4	66.7
	MultiR_continuous_MIRA	74.6	73.4
	DNMAR_continuous	73.1	68.0
	DNMAR_continuous_MIRA	75.6	68.7
Jointly Learned Representations	PCNNNMAR	78.1	75.4
	PCNNNMAR (bag-size weighting function)	**82.9**	**83.1**
Baselines	col-less universal schema [25]	60.3	57.5
	PCNN+ATT (Lin et al. [20] code)	67.9	72.1
	PCNN+ATT (our reimplementation with parameter tuning)	78.2	74.8

NYTFB-280K **Results:** When training on the larger dataset provided by Lin et. al. [20], linguistic features are not available, so only neural representations are included in our evaluation. As illustrated in Table 2.4, PCNNNMAR also achieves the best performance when training on the larger dataset; its improvements over the baselines are statistically significant. The AUC of most models decreases on the Hoffmann et. al. sentential dataset when training on NYTFB-280K. This is not surprising, because the Hoffmann et. al. dataset is built by sampling sentences from positive predictions of models trained on NYTFB-68K; changing the training data causes a difference in the ranking of high-confidence predictions for each model, leading to the observed decline in performance against the Hoffmann et. al. dataset. To further validate our findings, we also manually inspect the models' top predictions as described below.

Manual Evaluation: Because the Hoffmann et. al. sentential dataset does not contain the highest confidence predictions, we also manually inspected each model's top 500 predictions for the most frequent 4 relations, and report precision @ N to further validate our results. As shown in Table 2.5, for NYTFB-68K, PCNN+ATT performs comparably on `/location/contains`[2] and `/person/company`, whereas our model has a considerable

[2] `/location/contains` is the most frequent relation in the Hoffmann et. al. dataset.

Table 2.5: Top: P@N of 4 most frequent relations for models trained on NYTFB-68K. Bottom: P@N of 4 most frequent relations for models trained on NYTFB-280K. Both models can perform well on `/location/contains` relation while PCNNNMAR (weighted) is consistently better over other relations.

Relation	N	PCNN+ATT	PCNNNMAR (weighted)
NYTFB-68K			
/location/contains	100	**1.00**	0.99
	500	0.97	**0.98**
/person/place_lived	100	0.76	**0.98**
	500	0.63	**0.78**
/person/nationality	100	0.62	**0.89**
	500	0.43	**0.54**
/person/company	100	0.98	0.98
	500	0.72	**0.78**
NYTFB-280K			
/location/contains	100	0.98	**0.99**
	500	0.82	**0.99**
/person/place_lived	100	0.58	**0.98**
	500	0.57	**0.84**
/person/nationality	100	0.70	**0.91**
	500	0.35	**0.56**
/person/company	100	0.59	**0.95**
	500	0.40	**0.68**

advantage on the other two relations. For NYTFB-280K, our model performs consistently better on all four relations compared with PCNN+ATT. When training on the larger NYTFB-280K dataset, we observe trend of increasing mention-level P@N for PCNNNMAR, however the performance of PCNN+ATT appears to decrease. We investigate this phenomenon further below.

Performance at Extracting New Facts: To explain PCNN+ATT's drop in mention-level performance after training on the larger NYTFB-280K dataset, our hypothesis is that the larger KB-supervised dataset not only contains more true positive training examples but also more false negative examples. This biases models toward predicting facts about popular entities, which are likely to exist in Freebase. To provide evidence in support of this hypothesis, we divide the manually annotated dataset from Hoffmann et. al. into two

Table 2.6: Top: Sentence distribution in Hoffmann et. al. [14] sentential evaluation DEV dataset. Bottom: Sentence distribution in Hoffmann et. al. [14] sentential evaluation TEST dataset. There are substantial Out-Of-Freebase mentions which are manually labeled as correct relational mentions.

Category	True	False	Total
DEV			
In-Freebase	102	180	282
Out-Of-Freebase	58	96	154
TEST			
In-Freebase	113	192	305
Out-Of-Freebase	41	99	140

Table 2.7: Top: Comparison of AUCs of In-Freebase and Out-Of-Freebase mentions on sentential DEV set for PCNN+ATT and PCNNNMAR (weighted) with two datasets. Bottom: Comparison of AUCs of In-Freebase and Out-Of-Freebase mentions on sentential TEST set for PCNN+ATT and PCNNNMAR (weighted) with two datasets. PCNN+ATT has significant drops about Out-Of-Freebase mentions on both sentential DEV and TEST set after training on the larger NYTFB-280K which explains why its overall AUC performances goes down while PCNNNMAR (weighted) does not have such problem.

Model	Dataset	InFB	OutFB
	DEV		
PCNN+ATT	NYTFB-68K	78.2	89.6
	NYTFB-280K	77.1	77.0
	Change	-1.1	**-12.6**
PCNNNMAR(weighted)	NYTFB-68K	81.3	90.4
	NYTFB-280K	77.7	90.6
	Change	**-3.6**	+0.2
	TEST		
PCNN+ATT	NYTFB-68K	78.7	75.9
	NYTFB-280K	81.9	56.8
	Change	+3.2	**-19.1**
PCNNNMAR(weighted)	NYTFB-68K	85.9	85.4
	NYTFB-280K	83.1	81.5
	Change	-2.8	**-3.9**

categories: mentions of facts found in Freebase, and those that are not; this distribution is presented in the Table 2.6. In Table 2.7, we present a breakdown of model performance on these two subsets. For PCNN+ATT, although the AUC of in-Freebase mentions on the test

set increases after training on the larger NYTFB-280K, its Out-Of-Freebase AUC on both dev and test sets drops significantly, which clearly illustrates the problem of increasing false negatives during training. In contrast, our model, which explicitly allows for the possibility of missing data in the KB during learning, has relatively stable performance on the two types of mentions, as the amount of weakly-supervised training data is increased.

2.2.2 Held-Out Evaluation

In subsection 3.2.5, we evaluated the results of minimally supervised approaches to relation extraction by comparing extracted mentions against human judgments. An alternative approach, which has been used in prior work, is to evaluate a model's performance by comparing predictions against held out facts from a KB. Taken in isolation, this approach to evaluation can be misleading, because it penalizes models that extract many new facts that do not already appear in the knowledge base. This is undesirable, because the whole point of an information extraction system is to extract *new* facts that are not already contained in a KB. Furthermore, sentential extraction has the benefit of providing clear provenance for extracted facts, which is crucial in many applications. Having mentioned these limitations of the held-out evaluation metrics, however, we now present results using this approach to facilitate comparison to prior work.

Figure 2.2 presents precision-recall curves against held out facts from Freebase comparing PCNNNMAR to several baselines and Figure 2.3 presents results on the larger NYTFB-280K dataset. All models perform better according to the held out evaluation metric when training on the larger dataset, which is consistent with our hypothesis, presented at the end of Table 2.2.1. Our structured model with learned representations, PCNNNMAR (weighted), has lower precision when recall is high. This also fits with our hypothesis, as systems that explicitly model missing data will extract many correct facts that do not appear in the KB, resulting in an under-estimate of precision according to this metric.

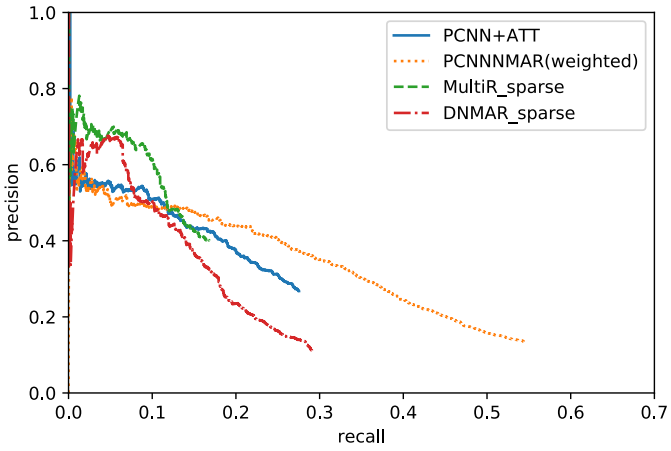


Figure 2.2: Held-out evaluation precision / recall curves for PCNN+ATT, MultiR, DNMAR and our proposed model PCNNNMAR (weighted) on NYTFB-68K.

2.3 Related Work

Knowledge Base Population: There is a long line of prior work on learning to extract relational information from text using minimal supervision. Early work on semantic bootstrapping [31, 32, 33, 34, 35, 36], applied an iterative procedure to extract lexical patterns and relation instances. These systems tend to suffer from the problem of semantic drift, which motivated work on distant supervision [37, 38, 39, 10], that explicitly minimizes standard loss functions, against observed facts in a knowledge base. The TAC KBP Knowledge Base Population task was a prominent shared evaluation of relation extraction systems [28, 40, 41, 42]. Recent work has explored a variety of new neural network architectures for relation extraction [43, 44, 45], experimenting with alternative sentence representations in our framework is an interesting direction for future work. Recent work has also shown improved performance by incorporating supervised training data on the sentence level [46, 47], in contrast our approach does not make use of any sentence-level labels during learning and therefore relies on less human supervision. Finally, prior work has explored a variety of methods to address the issue of noise introduced during distant supervision [48, 49, 50].

Another line of work has explored open-domain and unsupervised methods for IE [51,

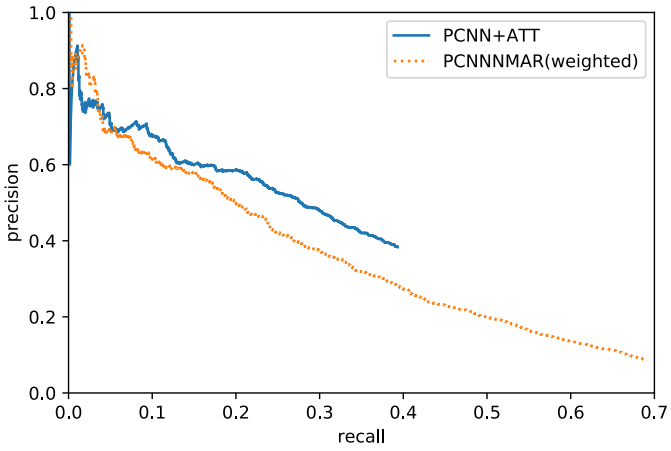


Figure 2.3: Held-out evaluation precision / recall curves for all NN-based models on NYTFB-280K.

52, 53, 54, 55]. Universal schemas [56] combine aspects of minimally supervised and unsupervised approaches to knowledge-base completion by applying matrix factorization techniques to multi-relational data [57, 58, 59]. Rows of the matrix typically model pairs of entities, and columns represent relations or syntactic patterns (i.e., syntactic dependency paths observed between the entities).

Structured Learning with Neural Representations: Prior work has investigated the combination of structured learning with learned representations for a number of NLP tasks, including parsing [60, 61, 62], named entity recognition [63, 64, 65] and stance detection [66]. We are not aware of any previous work that has explored this direction on the task of minimally supervised relation extraction; we believe structured learning is particularly crucial when learning from minimal supervision to help address the issues of missing data and overlapping relations.

2.4 Conclusions

In this chapter, we present a hybrid approach to minimally supervised relation extraction that combines the benefits of structured learning and learned representations. Extensive experiments show that by performing inference during the learning procedure to address

the issue of noise in distant supervision, our proposed model achieves state-of-the-art performance on minimally supervised mention-level relation extraction.

CHAPTER 3

EFFICIENT INFORMATION EXTRACTION ON SCIENTIFIC LITERATURE USING PRE-TRAINED LANGUAGE MODELS

This chapter contains materials that were originally published in [67] and [68].

The rapid expansion of scientific literature necessitates efficient extraction of key information to foster knowledge growth. However, the manual process is tedious and slow, emphasizing the need for automated solutions. Information extraction (IE), a branch of natural language processing (NLP) focused on automatically extracting structured information from unstructured data sources, plays a crucial role in addressing this challenge. Similar to other NLP tasks, scientific IE has benefited from the recent advances in deep learning, particularly the introduction of pre-trained language models (LMs) [69, 70], where the focus is to adapt pre-trained LMs to specific tasks rather than training models from the ground up. This chapter delves into the deployment of pre-trained LMs to develop effective and efficient scientific IE systems in both fully supervised and few-shot learning settings. We discuss two primary strategies: 1) task-specific fine-tuning, and 2) knowledge distillation from in-context learned LMs.

We begin by exploring task-specific fine-tuning for scientific IE in the supervised setting, specifically targeting Process Execution Graph (PEG) extraction. PEGs are directed, labeled, acyclic graph structures that capture the predicate-argument structure of a scientific protocol and can serve as a convenient scaffold to support downstream tasks such as text-to-code assistants [71]. We present two approaches for PEG prediction. First, we consider a pipelined approach, where different graph sub-components are predicted in isolation and then assembled to form a complete PEG prediction. In this approach, mention identification is treated as a sequence tagging problem, while temporal ordering is approached as a relation extraction task. Second, we present a graph-based multi-task framework based on DyGIE++

[72], which is capable of directly predicting the entire PEG. Our analysis reveals that while the pipelined approach excels in mention detection, the holistic modeling of mentions and relations in the multi-task framework can effectively reduce error propagation.

Next, we dive into applying knowledge distillation to in-context learned LMs for crafting efficient and compact scientific IE models, particularly in the few-shot learning setting. Our focus here is on anaphora resolution, a task that involves identifying the antecedents of anaphors (such as pronouns or noun phrases), which is particularly challenging and relevant in scientific protocols due to complex linguistic structures and a lack of extensive labeled data. In this context, we implement knowledge distillation via self-training, where the student model is trained on pseudo labels generated by the MICE in-context learning method (Le et al., 2022). We show that this approach enables the student model to approximate the performance of the in-context learned teacher model while offering substantial improvements in efficiency.

Overall, this chapter validates two distinct approaches for creating potent and efficient scientific IE systems using pre-trained LMs, applied to two critical and challenging tasks within scientific IE. We establish that task-specific fine-tuning is effective in standard supervised scenarios, particularly in PEG extraction where joint modeling of mentions and relations markedly reduces error propagation. Furthermore, we illustrate how knowledge distillation from in-context learned LMs in few-shot settings can lead to the development of high-performing, streamlined scientific IE models.

3.1 Task-Specific Fine-tuning

In this section, we explore various fine-tuning approaches for scientific IE, exemplifying with the task of process execution graph extraction - a particularly complex and unique IE challenge in the scientific domain.

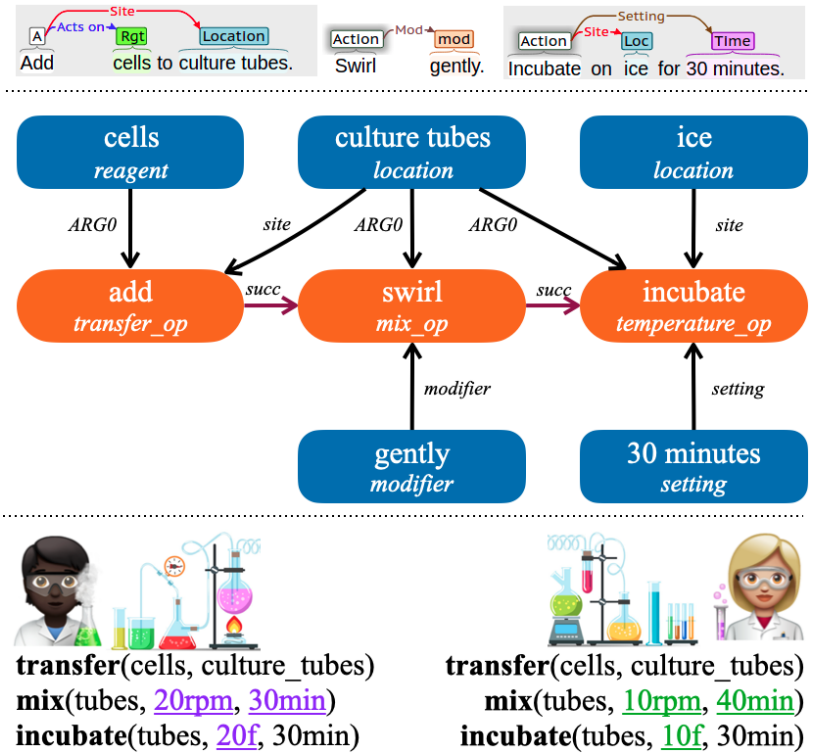


Figure 3.1: We develop a scaffold (center) between sentence-level lab procedure representations (top) and low-level, lab-specific instructions (bottom). The Process Execution Graph (PEG) captures document-level relations between procedures (orange rounded nodes) and their arguments (blue rectangular nodes).

3.1.1 Process Execution Graph Extraction

As shown in Figure 3.1, PEG is a directed, labeled, acyclic graph structure, which captures the predicate-argument structure of the protocol, allowing it to be more lenient than a programming language (for example, capturing that *gently* modifies `swirl`). Better suited to represent underspecified natural language, PEGs can serve as a convenient scaffold to support downstream tasks such as text-to-code assistants [71].

Nodes PEG nodes are triggered by explicit text spans in the protocol, e.g., “swirl", or “ice”. Nodes consist of two types: (1) *predicates*, marked in orange: denoting lab operations, such as `add` or `incubate`; and (2) *arguments*, marked in blue: representing physical lab objects (e.g., *culture tubes*, *cells*), exact quantities (*30 minutes*), or abstract instructions (e.g.,

gently).

Node grounding The PEG formulation above is motivated as a scaffold towards fully-executable lab programs employed in automatic lab environments. To achieve this, we introduce an ontology for each of the node types, based on the Autoprotocol specification [73], as indicated below each text span in Figure 3.1. For example, `swirl` corresponds to an Autoprotocol `mix` operation, a *culture tube* is of type *location*, and *30 minutes* is a *setting*. See Table 3.1 and Table 3.2 for details of predicate and argument types respectively, their frequencies in our data and example spans.

Edges Following PropBank notation [74], PEGs consist of three types of edges derived from the Autoprotocol ontology, and denoted by their labels: (1) core-roles (e.g., "ARG0", "ARG1"): indicating predicate-specific roles, aligning with Autoprotocol's ontology. For example, *ARG0* of `mix` assigns the element to be mixed; (2) non-core roles (e.g., "setting", "site", or "co-ref"): indicate predicate-agnostic relations. For example, the *site* argument always marks the location in which a predicate is taking place; and (3) temporal edges,

Table 3.1: Details of PEG predicate types, along with example frequent trigger spans and relative frequency in X-WLP.

Operation type	Frequent example spans	Count	Pct.
Transfer	add, transfer, place	1301	33.2
Temperature Treatment	incubate, store, thaw	503	12.8
General	Initiate, run, do not vortex	469	11.9
Mix	mix, vortex, inverting	346	8.8
Spin	spin, centrifuge, pellet	282	7.2
Create	prepare, make, set up	178	4.5
Destroy	discard, decant, pour off	170	4.3
Remove	remove, elute, extract	168	4.3
Measure	count, weigh, measure	149	3.8
Wash	wash, rinse, clean	146	3.7
Time	wait, sit, leave	114	2.9
Seal	cover, seal, cap	68	1.7
Convert	change, transform, changes	21	0.5

Table 3.2: Details of PEG argument types, along with example frequent trigger spans and relative frequency in X-WLP.

Argument type	Frequent example spans	Count	Pct.
Reagent	supernatant, dna, sample	3362	32.6
Measurement	1.5 mL, 595nm, 1pmol	1924	18.6
Setting	overnight, room temperature	1622	15.7
Location	tube, ice, plates	1373	13.3
Modifier	gently, carefully, clean	1070	10.3
Device	forceps, pipette tip	590	5.7
Method	dilutions, pipetting	271	2.6
Seal	lid, cap, aluminum foil	97	0.9

labeled with a special "succ" label: define a temporal transitive ordering between predicates. In Figure 3.1, `add` occurs before `swirl`, which occurs before `incubate`. See Table 3.3 for predicate-specific core-role semantics, and Table 3.4 for non-cores roles types and frequencies of all roles in X-WLP.

Reentrancies and cross-sentence relations While the PEG does not form directed cycles,[1] it does form non-directed cycles (or *reentrancies*) – where there exists nodes u, v such that there are two different paths from u to v. This occurs when an object participates in two or more temporally-dependent operations. For example, see *culture tubes*, which participates in all operations in Figure 3.1. In addition, edges (u, v) may be triggered either by *within-sentence relations*, when both u and v are triggered by spans in the same sentence, or by *cross-sentence relations*, when u and v are triggered by spans in different sentences. In the following section we will show that both reentrancies and cross-sentence relations, which are not captured by previous annotations, are abundant in our annotations.

3.1.2 Models

We present two approaches for PEG prediction. First, we design models for separate graph sub-component prediction, which are chained to form a pipeline PEG prediction model.

[1]This happens because the temporal relations define a partial ordering imposed by the linearity of the execution.

Table 3.3: Details of core role semantics for all operation types. The "Required" column specifies which roles must be filled for a given operation. ARG* is short for $\{\text{ARG0}, \text{ARG1}, \text{ARG2}\}$.

Operation	Role Semantics	Required
Spin	ARG0 centrifuged to produce solid phase ARG1 and/or liquid phase ARG2	ARG0
Convert	ARG0 converted to ARG1	ARG0, ARG1
Seal	ARG0 sealed with ARG1	ARG0
Create	ARG* are created	ARG0
General	-	ARG0
Destroy	ARG* discarded	ARG0
Measure	ARG* to be measured	ARG0
Mix	ARG* are mixed	ARG0
Remove	ARG0 removed from ARG1	ARG0
Temperature Treatment	ARG* to be heated/cooled	ARG0
Time	Wait after operation on ARG0	ARG0
Transfer	ARG* are sources, transferred to "site"	ARG0, site
Wash	ARG0 washed with ARG1	ARG0

Second, we present a model which directly predicts the entire PEG using a span-graph prediction approach.

Pipeline Model (PIPELINE) A full PEG representation as defined in subsection 3.1.1 can be obtained by chaining the following models which predict its sub-components. In all of these, we use SciBERT [75] which was trained on scientific texts similar to our domain.

Mention identification. Given a scientific protocol written in natural language, we begin by identifying all experiment-involved text spans mentioning lab operations (predicates) or entities and their traits (arguments), which are the building blocks for PEGs. We model this problem of mention identification as a sequence tagging problem. Specifically, we transfer span-level mention labels, which are annotated in the WLP corpus into token-level labels using the BIO tagging scheme, then fine-tune the SciBERT model for token classification.

Predicate grounding. Next, we ground predicate nodes into the operation ontology

Table 3.4: Breakdown of PEG relation types by frequency in X-WLP, showing counts of inter/intra-sentence relations. Re-entrancies are possible only for core and "site" arguments, and may be either inter or intra-sentence.

Relation	# Intra.	# Inter.	Total	# Re-entrancy
Core				
• ARG0	2962	952	3914	1645
• ARG1	560	127	687	3
• ARG2	84	123	207	77
Total (core)	3606	1202	4808	1725
Non-Core				
• site	1306	325	1631	360
• setting	3499	2	3501	-
• usage	1114	24	1138	-
• co-ref	129	1575	1704	-
• located-at	199	72	271	-
• measure	2936	18	2954	-
• modifier	1861	2	1863	-
• part-of	72	65	137	-
Total (non-core)	11116	2083	13199	360
Temporal	1218	788	2006	-
Grand Total	15940 (80%)	4073 (20%)	20013	2085

types discussed in subsection 3.1.1. Predicted mentions are marked using special start and end tokens (`[E-start]` and `[E-end]`), then fed as input to SciBERT. The contextual embedding of `[E-start]` is input to a linear softmax layer to predict the fine-grained operation type.

Operation argument role labeling. Once the operation type is identified, we predict its semantic arguments and their roles. Given an operation and an argument mention, four special tokens are used to specify the positions of their spans [3]. Type information is also encoded into the tokens, for example, when the types of the operator and its argument are `mix-op` and `reagent` respectively, four special tokens `[E1-mix-op-start]`, `[E1-mix-op-end]`, `[E2-rg-start]` and `[E2-rg-end]` are used to denote the spans of the mention pair. After feeding the input into SciBERT, the contextualized embeddings of `[E1-op-mix-start]` and `[E2-rg-start]` are concatenated as input to a linear layer that is used to predict the entity's argument role. Arguments of an operation can be selected

from anywhere in the protocol, leading to many cross-sentence operation-argument link candidates. To accommodate cross-sentence argument roles, we use the entire document as input to SciBERT for each mention pair. However, SciBERT is limited to processing sequences of at most 512 tokens. To address this limitation, longer documents are truncated in a way that preserves surrounding context, when encoding mention pairs.[2] Only 8 of the 279 protocols in our dataset contain more than 512 tokens.

Temporal ordering. Finally, we model order of operations using the `succ` relation. These are predicted using a similar approach as argument role labeling, where special tokens are used to encode operation spans.

Jointly-Trained Model (MULTI-TASK) To explore the benefits of jointly modeling mentions and relations, we experiment with a graph-based multi-task framework based on DYGIE++ model [72]. Candidate mention spans are encoded using SciBERT, and a graph is constructed based on predicted X-WLP relations and argument roles. A message-passing neural network is then used to predict mention spans while propagating information about related spans in the graph [76, 77, 78].

This approach requires computing hidden state representations for all $O(n^4)$ pairs of spans in an input text, which for long sequences, will exhaust GPU memory. While [72] considered primarily within-sentence relations, our model must consider relations across the entire protocol, which makes this a problem of practical concern. To address this, we encode a sliding window of w adjacent sentences when the full protocol does not fit into memory, allowing smaller windows for the start and end of the protocol, and concatenate sentences within each window as inputs to the model. As a result, each sentence is involved in w windows leading to repeated, possibly contradicting predictions for both mentions and relations. To handle this, we output predictions agreed upon by at least k windows, where k is a hyperparameter tuned on a development set.

[2]Given an input document, which has more than 512 words, with n words between two mentions, we truncate the context to keep at most $(512 - n)/2$ words for each side.

Table 3.5: Statistics of X-WLP [67]. X-WLP annotates complex documents, constituting more than 13 sentences on average. X-WLP overall size is on par with other recent procedural corpora, including ProPara [79], material science (MSPTC; [80]) and chemical synthesis procedures (CSP; [81]). CSP is comprised of annotated sentences (document-level information is not provided).

	X-WLP	MSPTC	CSP	ProPara
# words	54k	56k	45k	29k
# words / sent.	14.6	26	25.8	9
# sentences	3,708	2,113	1,764	3,300
# sentences / docs.	13.29	9	N/A	6.8
# docs.	279	230	N/A	488

3.1.3 Experiments

In subsection 3.1.2, we present a pipelined approach to PEG prediction based on SciBERT and a message-passing neural network that jointly learns span and relation representations. Next, we describe the details of our experiments and present empirical results demonstrating that X-WLP supports training models that can predict PEGs from natural language instructions.

Data. X-WLP [67] is our main dataset including 279 fully annotated protocols. Statistics of X-WLP are presented in Table 3.5. Additionally, we have 344 protocols from the original WLP dataset. We use this auxiliary data only for training mention taggers in the pipeline model, and use X-WLP for all other tasks. For argument role labeling and temporal ordering, negative instances are generated by enumerating all possible mention pairs whose types appear at least once in the gold data. We use 5-fold cross validation; 2 folds (112 protocols) are used for development, and the other 3 folds (167 protocols) are used to report final results.

Model setup. The PIPELINE framework employs a separate model for each task, by default using the propagated predictions from previous tasks as input. In addition, we evaluate the model for each task with gold input denoted as PIPELINE(gold). Finally, the MULTI-TASK

Table 3.6: Mention identification test set F_1 scores for models on the WLP dataset. Top: WLP dataset with the original train/dev/test split. Bottom: excluding X-WLP protocols from the WLP training data, and using them for evaluation.

Data Split	System	F_1
original	[83]	78.0
	[72]	**79.7**
	PIPELINE	78.3
X-WLP-eval	PIPELINE	74.7

Table 3.7: Predicate grounding test set results.

System	P	R	F_1
MULTI-TASK	76.0	69.0	72.3
PIPELINE	71.8	76.3	74.0
• w/ gold mentions	79.0	80.2	79.6

framework learns all tasks together and we decompose its performance into the component subtasks.

Implementation details. We use the uncased version of SciBERT[3] for all our models due to the importance of in-domain pre-training. The models under the PIPELINE system are implemented using Huggingface Transformers [82], and we use AdamW with the learning rate 2×10^{-5} for SciBERT finetuing. For the MULTI-TASK framework, we set the widow size w to 5, the maximum value that enables the model to fit in GPU memory. For all other hyperparameters, we follow the settings of the WLP experiments in [72].

3.1.4 Results

The results of the two models on the different subtasks are presented in Tables Table 3.6-Table 3.9. We identify three main observations based on these results.

First, PIPELINE outperforms MULTI-TASK on the operation classification task in Table 3.7, as it uses all protocols from WLP as additional training data to improve mention

[3] https://github.com/allenai/scibert

Table 3.8: Operation argument role labeling (core and non-core roles, decomposed by relation) and temporal ordering test set F_1 performance.

Task	MULTI-TASK	PIPELINE	# gold
Core			
• All roles	**57.9**	53.7	2839
• All roles (gold mentions)	-	76.5	2839
• ARG0	**61.0**	57.1	2313
• ARG1	**36.1**	32.9	412
• ARG2	**69.7**	61.4	114
Non-Core			
• All roles	**55.7**	48.8	4826
• All roles (gold mentions)	-	78.1	4826
• site	**58.7**	55.4	962
• setting	**77.4**	74.7	974
• usage	**35.6**	33.0	296
• co-ref	**39.8**	36.7	1014
• measure	**63.3**	56.6	804
• modifier	**51.0**	41.8	519
• located-at	9.7	**13.3**	179
• part-of	0.5	**10.8**	78
Temporal Ordering	**61.8**	57.3	2176
Temp. Ord. (gold mentions)	-	76.3	2176

Table 3.9: Operation argument role labeling (core roles) test set F_1, decomposed based on whether the operation and the argument are triggered within the same sentence (intra-sentence) versus different sentences (inter-sentence).

Split	MULTI-TASK	PIPELINE	# gold
Intra-sentence	**63.4**	58.2	2160
Inter-sentence	**32.5**	39.1	679

tagging.

Second, MULTI-TASK performs better than the PIPELINE approach on most relation classification tasks in Table 3.8, but is worse than PIPELINE when PIPELINE uses gold mentions, demonstrating that jointly modeling mentions and relations helps in mitigating error propagation.

Third, *cross-sentence* relations are challenging for both models, as shown in Table 3.9. This explains the low performance of `co-ref`, which is comprised of 92.4% cross-sentence

relations.

In addition, there are a couple of interesting points to note. In Table 3.6, the performance of PIPELINE on the X-WLP subset is lower than its performance on the WLP test set, likely because there are fewer protocols in the training set. For the relation-decomposed performance in Table 3.8, we can see that some of the relations like "ARG2" can be correctly predicted by MULTI-TASK using only a few gold labels while some more widely used relations are harder to learn, such as "ARG0" and "site"; indeed, "ARG2" is only used in the `spin` operation (see Table 3.3), while the other roles participate in more diverse contexts.

3.2 Knowledge Distillation from In-context Learned LMs

Apart from direct fine-tuning, in-context learning emerges as an alternative strategy to customize pre-trained LMs for specific tasks, including scientific IE. This technique involves feeding the LM a sequence of relevant, annotated examples from the target task as part of its input. The LM then applies this contextual knowledge to deduce labels for new examples. A key advantage of this method is that it eliminates the need for additional fine-tuning or retraining, and it can be effective with limited training data. However, its efficacy is often constrained to larger models, hindering its broader applicability. This bottleneck can be addressed through knowledge distillation [84, 85]. In this section, we delve into how we can distill in-context learned LMs to develop efficient scientific IE systems. Our focus is on anaphora resolution, a task that is challenging and highly pertinent in the scientific domain, often compounded by complex linguistic structures and the scarcity of training data.

3.2.1 Anaphora Resolution in Scientific Protocols

Split-antecedent anaphors [86, 87, 88] are plural mentions that refer to two or more antecedents in the previous discourse. For instance in the following text: "`[Alice]`$_{\texttt{antecedent}}$ `and [Bob]`$_{\texttt{antecedent}}$ `went to the store. [They]`$_{\texttt{anaphor}}$ `bought some bread.`" the word "`[They]`" refers to two both "`[Alice]`" and "`[Bob]`".

Similar references to multiple antecedents often appear in chemical synthesis protocols, for example, "*the mixture*". These references arise naturally as the result of context change accommodation [89], and are crucial for understanding the steps needed to synthesize a molecule [90]. Resolving anaphoric references in synthetic protocols could be beneficial for automating protocols described in natural language [91, 92], in addition to automatically extracting chemical reaction databases from scientific literature [93, 80]. However, anaphora is costly to annotate [94] and scientific protocols are not easily amenable to annotation by non-expert crowd workers [95]. This motivates the need for few-shot learning methods that can resolve anaphora in procedural texts without extensive annotated resources.

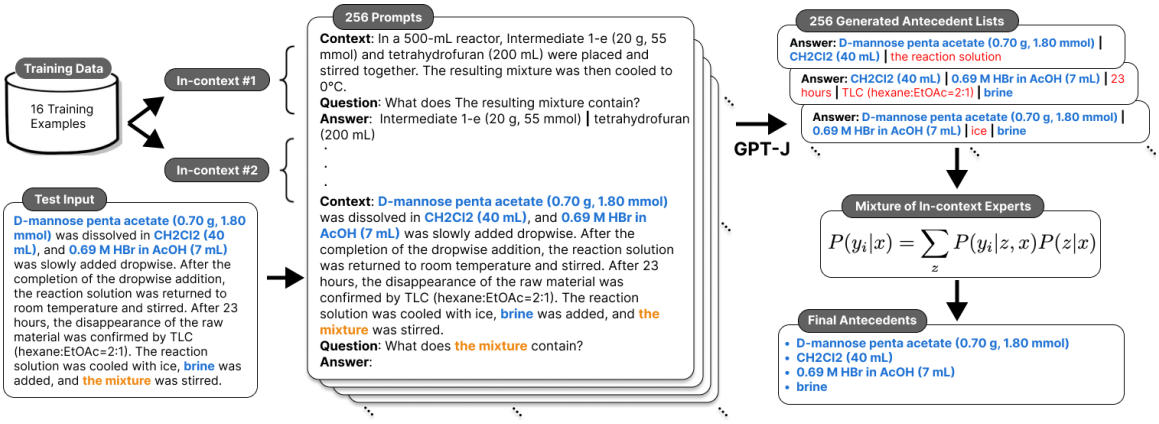


Figure 3.2: Resolving antecedents of *"the mixture"* in a chemical synthesis procedure using MICE. Given a small training set of 16 examples and a test input, we construct 256 prompts, each with two in-context demonstrations. The prompts are then fed into a pre-trained language model (e.g. GPT-J) to generate candidate antecedents. The probabilities of each candidate antecedent are computed and combined in a mixture of in-context experts using a similarity-based gating function. MICE then selects the antecedents with the highest probabilities. In the figure, orange, blue, red denote **the anaphor**, **the true antecedents**, and incorrect antecedents, respectively.

3.2.2 Distillation from In-context Learned LMs

Below, we first introduce how we formulate the task of anaphora resolution under in-context learning. We then describe the details of our teacher model MICE, followed by an in-depth explanation of the distillation process.

In-Context Learning We formulate the task of anaphora resolution in synthetic protocols as follows. The input includes a document D and a query anaphor a. Our goal is to identify a set of antecedents $\mathcal{Y} = \{y_0, y_1, ..., y_m\}$ that correspond to text spans in D. To tackle this problem via in-context learning, we frame it as a SQuAD-style extractive question answering task [96]. Specifically, as shown in Figure 3.2, each example (D, a) is formatted as the concatenation of document D and template question: "`What does` a `contain?`" An autoregressive language model then completes this sequence by generating $\mathcal{Y}$, with the antecedents separated by a special marker "`|`". Following the typical approach to in-context learning, the prompt includes a few demonstrations in the prefix and ends with the test input.

Teacher Model - MICE For a given test input $x = (D, a)$, we aim to find the antecedents y_i with the highest probabilities $P(y_i|x)$. The notation y_i denotes an antecedent from the union of antecedents generated by all prompts. MICE computes $P(y_i|x)$ using a mixture of experts [97, 98], treating the prompt, z, as a latent variable [99]:

$$P(y_i|x) = \sum_z P(y_i|z, x)P(z|x) \tag{3.1}$$

In Equation 3.1, $P(z|x)$ represents the likelihood that prompt z is constructed given x, and $P(y_i|z, x)$ represents the probability that the LM predicts antecedent y_i when prompted with z and x.

Similarity-based Gating We compute $P(z|x)$ by summing similarity scores $s(x, u_1), ..., s(x, u_d)$ between x and the in-context demonstrations $u_1, ..., u_d$ encoded in z:[4]

$$P(z|x) \propto \exp \sum_{i=1}^{d} s(x, u_i)$$

where $s(x, u_i)$ is the cosine similarity between the embeddings of x and u_i. Details of the similarity measures used in our experiments are presented in subsection 3.2.4.

[4]We also experimented with multiplying the similarity scores and observed similar results.

Estimating Antecedent Probabilities Computing probabilities $P(y_i|z,x)$ that are comparable across variable-length antecedents is not easy. Longer sequences will naturally have smaller LM probabilities, suggesting the need for length normalization, or averaging per-token probabilities, neither of which we found to work well.[5] Therefore, following [100], we estimate $P(y_i|z,x)$ using first token probabilities. Specifically, let $y_{i,0}$ denote the first token of y_i. We then have:

$$P(y_i|z,x) \approx \max_j P_j(y_{i,0}|z,x) \tag{3.2}$$

where the max is taken over all mentions of y_i found in the k^d prompts. $P_j(y_{i,0}|z,x)$ is the probability of token $y_{i,0}$ being the first token of the jth antecedent generated by the language model, when prompted with z and x.

Approximation with Sampling Approximating $P(y_i|z,x)$ using first-token probabilities in Equation 3.2 has a drawback: it disregards the length of the antecedents and treats different antecedents with the same first token as equivalent. As an alternative, we present MICE-SAMPLING: a simple Monte Carlo approximation of $P(y_i|z,x)$ that uses a binary indicator $\mathbb{1}[y_i \in \mathcal{Y}_{z,x}]$ of whether or not antecedent y_i is in $\mathcal{Y}_{z,x}$ (the set of generated antecedents given prompt z and x):

$$\begin{aligned} P(y_i|x) &= \sum_z P(y_i|z,x)P(z|x) \\ &\approx \sum_z \mathbb{1}[y_i \in \mathcal{Y}_{z,x}]P(z|x) \end{aligned}$$

In subsection 3.2.3, we show empirically that when using hundreds of prompts, MICE-SAMPLING is actually a better approximation than the LM probability of the first token [100]. In addition, MICE-SAMPLING is simpler to implement, as it does not require storing

[5] Similar to [100], we observe that, for a generated antecedent, the first token probabilities vary the most, while probabilities of subsequent tokens are highly deterministic.

output logits and computing the softmax to obtain first token probabilities.

Knowledge Distillation To produce a compact model for inference, we perform knowledge distillation [84, 85] via self-training, where the student model is trained on pseudo labels generated MICE. We experiment with three few-shot settings $k \in \{8, 32, 64\}$. In each setting, out of five training runs, we select the MICE-SAMPLING-{2} model checkpoint (2 in-context examples) that achieves the median performance on the Dev-256 set as the teacher model. To train the student model [101, 102], we randomly sample real unlabeled synthetic protocols from the chemical patent corpus collected in [103]. We then run rule-based anaphor detection to identify anaphors within the sampled documents. Subsequently, the teacher model is used to predict antecedents. Finally, we train the ProcBERT student model in a two-stage process. We first fine-tune ProcBERT on M pseudo-labeled examples ($M \in \{50, 100, ..., 2000\}$) for 50 epochs, then further fine-tune it on the k examples with gold labels for 200 epochs. For the student model, we frame antecedent resolution as a sequence-labeling problem by transferring span-level antecedent labels into token-level labels with a BIO tagging scheme. We train the student model for token-level classification, where the input anaphor is marked with two special tokens `[Ana-start]` and `[Ana-end]`.

Table 3.10: Statistics of selected CHEMU-REF splits.

	Train (full)	**Dev-64**	**Dev-256**	**Test**
# anaphors	4,766	64	256	898
# antecedents	21,673	293	996	4,016
# documents	856	11	52	166
# sentences	5,833	76	346	1,066
# tokens	984,032	10,150	45,825	140,614

3.2.3 Experimental Setup

We evaluate our approach on CHEMU-REF, a corpus of synthetic procedures from chemical patents that are annotated with coreference and anaphora [90]. The CHEMU-REF anno-

tations contain fine-grained chemistry-specific relations such as TRANSFORMED, which indicates the mixture components undergo a chemical transformation, or WORK-UP, which represents a combination of compounds to isolate or purify a reaction product. In this work, we focus on modeling the general structure of anaphora in scientific protocols and understanding which compounds are combined at each step of the procedure, which does not require making these more fine-grained distinctions. Therefore, we pre-processed the data by collapsing all one-to-many CHEMU-REF relations into a single MULTIPLE-ANTECEDENT relation. We remove anaphors comprising compounds described with IUPAC nomenclature [104], for example *"2,2,6,6-tetramethylpiperidine"*, while retaining nominal anaphors, such as *"the mixture"*, *"the solution"* and *"the reaction"*, which make up 90% of anaphors in the CHEMU-REF corpus. For train/dev/test splits, we use the original train split for training and development, and the original dev split for evaluation.[6] Following [105], for each k-shot experiment, we sample five different training sets from the full training split using different seeds and report the mean. For model development and ablation studies, we use a small development set of 64 examples (Dev-64) to simulate a true few-shot learning setting. For evaluation on held-out data, we use the full CHEMU-REF development set as test data (Test) as well as a sub-sampled version with 256 examples (Dev-256) to control for high GPT-J inference costs. Selected dataset statistics are shown in Table 3.10.

We use F_1 as our evaluation metric. In particular, we compute the micro-F_1 between the predicted and gold sets of antecedents of the evaluation data. A predicted and a gold antecedent are considered the same if they are an exact match.

3.2.4 Implementation Details

Models We use GPT-J-6B [106] as the backbone language model for MICE and in-context baselines since it was the largest publicly available autoregressive language model at the time we started this work.[7] Compared with other similar language models like GPT-2, GPT-J

[6]The official CHEMU-REF test set is hidden at `http://chemu2021.eng.unimelb.edu.au/`.

[7]Larger models, such as OPT-175B [107], have become available recently.

has a larger maximum sequence length of 2048. It is also pre-trained on the Pile [108], which covers in-domain data including chemical patents from USPTO[8] and PubMed articles. Similarly, we choose ProcBERT [103] for the student model in knowledge distillation due to its in-domain pre-training on synthetic procedures.

MICE We ensemble up to 256 prompts, with 2 or 5 in-context demonstrations in the prompt, for all few-shot settings. To calculate the similarity score $s(x, u)$ between the input x and a demonstration example u, we use SBERT model [109][9] with the `roberta-large` checkpoint since this model is widely used for measuring text similarity [110]. We also experimented with calibrating the language model to be bias-free by applying the calibration procedure described in [100]; however, we found that it did not improve performance in the context of anaphora resolution in scientific protocols.[10]

Computational Cost In addition to performance, we also measure the computational cost of the teacher and student models in knowledge distillation. Concretely, we measure floating point operations (FLOPs) of the two models for both training and inference using the FLOPs-counting code provided in [111][11]. Note that, the training of the student model requires M pseudo-labeled examples, so the training FLOPs of the student model will also include the FLOPs of generating those pseudo labels using the teacher model.

For all GPT-J-based experiments, generation is performed using greedy decoding up to a maximum of 256 tokens on 48GB A40 GPUs. When using 256 in-context experts, about three anaphors can be resolved per GPU hour.

Table 3.11: Dev-256 F_1 and training/inference FLOPs of the teacher model (MICE-SAMPLING) and the student model in knowledge distillation. The training FLOPs of the student model under different few-shot settings are almost the same. With 2000 pseudo-labeled examples, the student model (110M parameters) can match or outperform the teacher model in terms of F_1. Although inference FLOPs of the teacher model are around 1500 times the FLOPs of the student model, considering the cost of training the student model to match the teacher model's performance, the teacher model is actually more cost effective than the student model for inference when the number of inference examples is low, e.g., less than 2100 for 32-shot. For rows where the number of FLOPs varies depending on the number of training examples, we show the maximum.

Model	**F_1** 8-shot	32-shot	64-shot	**Train FLOPs**	**Infer FLOPs** (1000 exam.)
ProcBERT	28.0	41.1	56.7	3.2e15	1.2e14
Teacher	54.1	55.7	59.9	0	1.9e17
Student (# pseudo labels)					
- 50	36.0	46.2	58.4	1.4e16	1.2e14
- 100	43.5	49.9	62.2	2.4e16	1.2e14
- 200	42.1	52.3	60.4	4.5e16	1.2e14
- 500	49.1	52.2	63.3	1.1e17	1.2e14
- 1000	50.1	51.8	64.1	2.1e17	1.2e14
- 2000	**54.3**	**55.8**	**64.3**	4.1e17	1.2e14

3.2.5 Results

The impact of knowledge distillation is presented in Table 3.11. We observe that with only 50 pseudo-labeled examples, the student model outperforms the ProcBERT baseline by 8.0 F_1 for 8-shot, 5.1 F_1 for 32-shot and 1.7 F_1 for 64-shot, showing the effectiveness of distillation. The performance of the student model improves as the number of pseudo-labeled examples increases. With 2000 pseudo-labeled examples, the inference-efficient student model (110M parameters) outperforms the teacher model MICE-SAMPLING. We hypothesize that the student model outperforms the teacher because after training on pseudo-labels generated by MICE, the student model is further fine-tuned on the k available gold-labeled examples, as described in subsection 3.2.4. This approach combines the benefits of in-context learning

[8] https://www.uspto.gov/

[9] https://www.sbert.net/docs/pretrained_models.html

[10] When prompted with content-free inputs, such as "N/A", the language model already generates bias-free answers (e.g. the LM generates "N/A" given "N/A" as a test input) without calibration, suggesting there is no significant bias towards specific answers for anaphora resolution.

[11] https://github.com/google-research/electra/blob/master/flops_computation.py

with fine-tuning achieving better performance than either in isolation. In-context learning does not incur any training costs, but the number of FLOPs required by MICE-SAMPLING for inference is roughly 1,500 times the computational cost of the student model.

3.3 Conclusion

In this chapter, we validate two distinct strategies for establishing competent and streamlined scientific IE systems using pre-trained LMs across two challenging yet crucial scientific IE tasks. We illustrate the effectiveness of task-specific fine-tuning in standard supervised scenarios, showing that the end-to-end multi-task approach exhibits superior performance compared to the pipeline approach, especially in minimizing error propagation. Moreover, we establish that knowledge distillation from in-context learned LMs is conducive to developing efficient, compact models in few-shot learning scenarios.

CHAPTER 4

PRE-TRAIN OR ANNOTATE? DOMAIN ADAPTATION WITH A CONSTRAINED BUDGET

This chapter contains materials that were originally published in [103] and [112].

With the launch of BERT [69] in late 2018, a significant shift occurred. As with many NLP tasks, there was an increased focus on fine-tuning scientific information extraction models using pre-trained language models. Although having high-quality annotated data is essential for downstream results, the compatibility of the pre-trained models with the target domain has gained prominence. Acknowledging the demanding resource requirements of these elements, an important question is raised: given a fixed budget to improve a model's performance, what steps should an NLP practitioner take? On one hand, they could hire annotators to label in-domain task-specific data, while on the other, they could buy or rent GPUs or TPUs to pre-train large in-domain language models. In this chapter, we empirically study the best strategy for adapting to a new domain given a fixed budget.

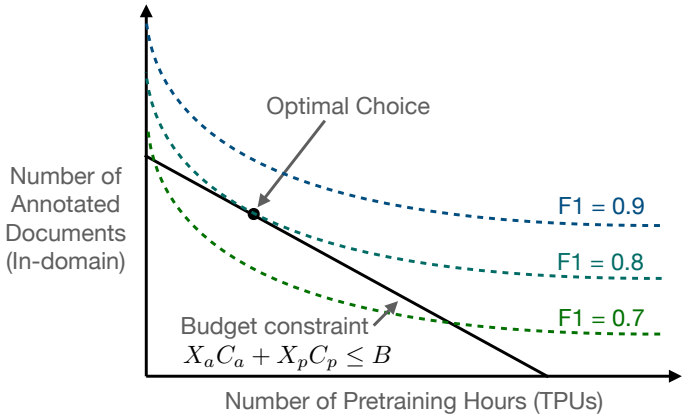


Figure 4.1: We view domain adaptation as a consumer choice problem [113, 114]. The NLP practitioner (consumer) is faced with the problem of choosing an optimal combination of annotation and pre-training under a constrained budget. This figure is purely for illustration and is not based on experimental data.

We view the NLP practitioner's dilemma of how to adapt to a new domain as a problem

of consumer choice, a classical problem in microeconomics [113, 114]. As illustrated in Figure 4.1, the NLP practitioner (consumer) can obtain X_a annotated documents (by hiring annotators) at a cost of C_a each, and X_p hours of pre-training (by renting GPUs or TPUs) at a cost of C_p per hour. Given a fixed budget B, the consumer may choose any combination that fits within the budget constraint $X_a C_a + X_p C_P \leqslant B$. The goal is to choose a combination that maximizes the utility function, $U(X_a, X_p)$, which can be defined using an appropriate performance metric, such as F_1 score, that is achieved after pre-training for X_p hours and then fine-tuning on X_a in-domain documents.

To empirically estimate the cost of annotation, we hire annotators to label domain-specific documents for supervised fine-tuning in three procedural text domains: wet-lab protocols, paragraphs describing scientific procedures in PubMed articles, and chemical synthesis procedures described in patents. We choose to target natural language understanding for scientific procedures in this study, because there is an opportunity to help automate lab protocols and support more reproducible scientific experiments, yet few annotated datasets currently exist in these domains. Furthermore, annotation of scientific procedures is not easily amenable to crowdsourcing, making this an ideal testbed for pre-training-based domain adaptation. We measure the cost of in-domain pre-training on a large collection of unlabeled procedural texts using Google's Cloud TPUs.[1] Model performance is then evaluated under varying budget constraints in six source and target domain combinations.

Our analysis suggests that given current costs of pre-training large Transformer models, such as BERT [69], and RoBERTa [115], in-domain data annotation should always be part of an economical strategy when adapting a single NLP system to a new domain. For small budgets (e.g. less than $800 USD), spending all funds on annotation is the best policy; however, as more funding becomes available, a combination of pre-training and annotation is the best choice.

[1] `https://cloud.google.com/tpu`

4.1 Scope of the Study

In this study, we focus on a typical scenario faced by an NLP practitioner to adapt a single NLP system to a single new domain, maximizing performance within a constrained budget. We consider only the direct benefit on the target task in our main analysis (section 4.4), however we do provide additional analysis of positive externalities on other related tasks that may benefit from a new pre-trained model in section 4.5.

We estimate cost based on two major expenses: annotating task-specific data (section 4.2) and pre-training domain-specific models using TPUs (section 4.3). Note that fine-tuning costs are not included in our analysis, as they are nearly equal whether budget is invested into pre-training or annotation.[2] We assume a generic BERT is the closest zero-cost model that is initially available, which is likely the case in real-world domain adaptation scenarios (especially for non-English languages). Our experiments are designed to simulate a scenario where no domain-specific model is initially available. We also assume that the NLP engineer's salary is a fixed cost; in other words, their salary will be the same whether they spend time pre-training models or managing a group of annotators.[3] Our primary concerns are about financial and environmental costs, rather than the overall time needed to obtain the adapted model. If the timeline is an important factor, the annotation process can be possibly sped up by hiring more annotators.

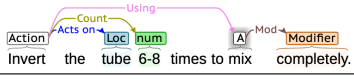
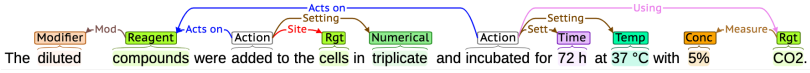
4.2 Estimating Annotation Cost (C_a)

In this section, we present our estimates of the annotation cost for three procedural text datasets from specialized scientific domains, which enable a comparison of model performance under varying budget constraints (subsection 4.4.4).

[2]These are also not a significant portion of overall costs (we estimate $1.95 based on Google Cloud rates for P100 GPUs.)

[3]Based on our experience, pre-training in-domain models (collecting training corpus is non-trivial) and managing a team of annotators are roughly comparable in terms of effort.

Table 4.1: Statistics and examples of three procedural text datasets.

Dataset	Domain	Task	#Files (train/dev/test)	#Sentences	#Cases	#Classes	Total Cost	Price/File	Price/Sent.
WLP [116]	biology	NER RE	726 (492/123/111)	17,658	185,313 124,803	20 16	$7,820	$10.8	$0.44
									
PubMedM&M (this work)	biomed	NER RE	191 (100/41/50)	1,699	12,131 8,987	24 17	$1,730	$9.1	$1.02
									
ChemSyn (this work)	chemistry	NER RE	992 (793/99/100)	8,331	53,423 46,878	24 17	$5,000	$4.7	$0.60
									

Annotated Procedural Text Datasets. We experiment with three procedural text corpora, including Wet Lab Protocols [WLP; 116] and two new datasets we created for this study, which include scientific articles and chemical patents. Statistics of the three datasets are shown in Table 4.1. The **WLP corpus** includes 726 wet lab experiment instructions collected from `protocols.io` which are annotated using an inventory of 20 entity types and 16 relation types. Following the same annotation scheme, we annotate PubMedM&M and ChemSyn. The **PubMedM&M corpus** consists of 191 double-annotated experimental paragraphs extracted from the *Materials and Methods* section of PubMed articles. The **ChemSyn corpus** consists of 992 chemical synthesis procedures described in patents, 500 of which are double-annotated. Unlike the succinct, informal language style in WLP, PubMedM&M represents an academic writing style, as it comes from published research papers (see Table 4.1). More details on data pre-processing, annotation and inter-annotator agreement scores can be found in Appendix section B.1

Annotation Cost. We recruit undergraduate students to annotate the datasets using the BRAT annotation tool.[4] Annotators are paid 13 USD / hour throughout the process, which

[4] `https://github.com/nlplab/brat`

is the standard rate for undergraduate students at our university. Estimates of the cost of annotation, C_a, per-sentence are presented in Table 4.1.[5]

4.3 Estimating Pre-training Cost (C_p)

To evaluate varied strategies for combining pretraining and annotation given a fixed budget, we need accurate estimates on the cost of annotation, C_a, and pretraining, C_p. Having estimated the cost of annotating in-domain procedural text corpora in section 4.2, we now turn to estimate the cost of in-domain pretraining. Specifically, we consider two popular approaches: 1) training an in-domain language model from scratch; 2) continued pre-training using an off-the-shelf model.

PROCEDURE Corpus Collection. To pre-train our models, we create a novel collection of procedural texts from the same domains as the annotated data in section 4.2, hereinafter referred to as the PROCEDURE corpus.

Specially trained classifiers were used to identify paragraphs describing experimental procedures. For PubMed, a classifier was used to identify paragraphs describing experimental procedures by fine-tuning SciBERT [6] on the SciSeg dataset [117], which is annotated with scientific discourse structure, to extract procedures from the *Materials and Methods* section of 680k articles. For the chemical synthesis domain, the chemical reaction extractor developed by [118] was applied to the *Description* section of 303k patents (174k U.S. and 129k European) we collected from USPTO[6] and EPO[7]. More details of our data collection process can be found in Appendix section B.2.

Cooking recipes are also an important domain for research on procedural text understanding, therefore we include the text component of the Recipe1M+ dataset [119] in the PROCEDURE pre-training corpus. In total, our **PROCEDURE** collection contains around 1.1

[5]The average time to annotate each sentence varies across datasets based on the complexity of the text, average length of sentences, etc.

[6]`https://www.uspto.gov/`

[7]`https://www.epo.org/`

billion words; more statistics are shown in Table 4.2. In addition, we create an extended version, **PROCEDURE+**, consisting of 12 billion words, where we up-sample the procedural paragraphs 6 times and combine them with the original full text of 680k PubMed articles and 303k chemical patents. This up-sampling ensures at least half of the text is procedural.

Pre-training Process and Cost. We train two procedural domain language models on the Google Cloud Platform using 8-core v3 TPUs: 1) **ProcBERT**, a BERT_{base} model pre-trained from scratch using our PROCEDURE+ corpus, and 2) **Proc-RoBERTa**, for which we continued pre-training RoBERTa_{base} on the PROCEDURE corpus following [120].

We pre-train ProcBERT using the TensorFlow codebase of BERT.[8] Following [69], we deploy a two-step regime: the model is trained with sequence length 128 and batch size 512 for 1 million steps at a rate of 4.71 steps/second. Then, it is trained for 100k more steps using sequences of length 512 and a batch size of 256 at a rate of 1.83 steps/second. The pretraining process takes about 74 hours, and the total cost is about 620 USD, which includes the price for on-demand TPU-v3s (8 USD/hour)[9] plus auxiliary costs for virtual machines and data storage.

We considered the possibility of evaluating checkpoints of partially pre-trained models, for fine-grained variation of the pre-training budget, however after some investigation we chose to only report results on fully pre-trained models, using established training protocols (learning rate, number of parameter updates, model size, sequence length, etc.) to ensure fair comparison.

In addition to pre-training from scratch, we also experiment with Domain-Adaptive Pre-training, using the codebase[10] released by AI2 to train Proc-RoBERTa. Similar to [120], we fine-tune RoBERTa on our collected PROCEDURE corpus for 12.5k steps with the averaged speed of 27.27 seconds per step, which leads to a TPU time of 95 hours.[11] Thus,

[8] `https://github.com/google-research/bert`
[9] `https://cloud.google.com/tpu/pricing`
[10] `https://github.com/allenai/tpu_pretrain`
[11] This is comparable to the number reported by the authors of [120] on GitHub.

the total cost of Proc-RoBERTa is around 800 USD after including the auxiliary expenses.

Finally, we estimate the cost of training for **SciBERT** [6], which was also trained on an 8-core TPU v3 using a two-stage training process similar to ProcBERT. The overall training of SciBERT took 7 days (5 days for the first stage and 2 days for the second stage) with an estimated cost of 1,340 USD.

Carbon Footprint. Apart from the financial cost, we also estimate the carbon footprint of each in-domain pre-trained language model for its environmental impact. We measure the energy consumption in kilowatt-hours (KWh) as in [121]:

$$Energy = H \times N \times P \times PUE/1000,$$

where H is the number of training hours, N is the number of processors used, P is the average power per processor,[12] and PUE (Power Usage Effectiveness) indicates the energy usage efficiency of a data center. In our case, the average power per TPU v3 processor is 283 watts, and we use a PUE coefficient of 1.10, which is the average trailing twelve-month PUE reported for all Google data centers in Q1 2021.[13] Once we know the energy consumption, we can estimate the CO_2 emissions (CO_2e) as follows:

$$CO_2e = (Energy \times CO_2e/KWh)/1000$$

where CO_2e/KWh measures the amount of CO_2 emission when consuming 1 KWh energy, which is 474g/KWh for our pre-training.[14] For example, ProcBERT is pre-trained on a single 8-core TPU v3 for 74 hours, resulting in CO_2 emission of $(74 \times 8 \times 283 \times 1.10/1000) \times 474/1000 = 87.4$ kg. The estimated CO_2 emissions for three in-domain language models

[12]Unlike [122] which measured GPU, CPU and DRAM's power separately, [121] measured the power of a processor together with other components including fans, network interface, host CPU, etc.

[13]`https://www.google.com/about/datacenters/efficiency/`

[14]Our models were pre-trained in the data center of Google in Netherlands: `https://cloud.google.com/sustainability/region-carbon`.

Table 4.2: Statistics of our newly created PROCEDURE and PROCEDURE+ corpora, which are used for pre-training Proc-RoBERTa and ProcBERT, respectively.

Corpus	#Tokens	Text Size	Pre-trained Model
Wiki + Books	3.3B	16GB	BERT
Web crawl	-	160GB	RoBERTa
PMC + CS	3.2B	-	SciBERT
BioMed	7.6B	47GB	BioMed-RoBERTa
PROCEDURE	1.05B	6.5GB	Proc-RoBERTa
- PubMed	0.32B	2.0GB	–
- Chem. patent	0.61B	3.9GB	–
- Cook. recipe	0.11B	0.6GB	–
PROCEDURE+	12B	77GB	ProcBERT
- PROCEDURE ($\times$ 6)	6.3B	39GB	–
- Full articles	5.7B	38GB	–

Table 4.3: Carbon footprint of three in-domain pre-trained language models. CO_2e is the number of metric tons of CO_2 emissions with the same global warming potential as one metric ton of another greenhouse gas.

Consumption	CO_2e (kg)
Air travel, one person, SF↔NY	1200
SciBERT [6]	198.3
Proc-RoBERTa	112.1
ProcBERT	87.4

are shown in Table 4.3.

4.4 Measuring Utility $U(X_a, X_p)$ under Varying Budget Constraints

Given the estimated unit cost of annotation C_a (section 4.2) and pre-training C_p (section 4.3), we now empirically evaluate the utility $U(X_a, X_p)$, of various budgets and pre-training strategies to find an optimal policy for domain adaptation that fits within the budget constraint $X_a C_a + X_p C_P \leqslant B$.

4.4.1 NLP Tasks and Models

We experiment with two NLP tasks, Named Entity Recognition (NER) and Relation Extraction (RE). For NER, we follow [69] to feed the contextualized embedding of each token into a linear classification layer. For RE, we follow [123], inserting four special tokens specifying positions and types of each entity-pair mention, which are included as input to a pre-trained sentence encoder. Gold entity mentions are used in our relation extraction experiments, to reduce variance due to entity recognition errors.

4.4.2 Budget-constrained Experimental Setup

As we have three procedural text datasets (section 4.2) annotated with entities and relations, we can experiment with six source ⇒ target adaptation settings. For each domain pair, we compare five different pre-trained language models when adapted to the procedural text domain under varying budgets.

Based on the estimations of the annotation costs C_a (section 4.2) and pre-training costs C_p (section 4.3), we conduct various budget-constrained domain adaptation experiments. For example, if we have \$1,500 and the PUBMEDM&M corpus, to build an NER model that works best for the CHEMSYN domain (PUBMEDM&M⇒CHEMSYN), we can spend all \$1,500 to annotate 2,500 in-domain sentences to fine-tune off-the-shelf BERT. Or alternatively, we could first spend \$800 to pre-train Proc-RoBERTa, then fine-tune it on 1155 sentences annotated in the CHEMSYN domain using the remaining of \$700. Under both budgeting strategies, an additional experiment is performed to choose one of two domain adaption methods that maximizes performance: 1) a model that simply uses the annotated data in the target domain for fine-tuning; or 2) a model which is fine-tuned using a variant of EasyAdapt [124] to leverage annotated data in both the source and target domains (see below for details). We select the approach that has better development set performance and report its test set result in Table 4.4 and Table 4.5 (see Appendix section B.4 for more details about hyper-parameters).

Table 4.4: Experiment results for Named Entity Recognition (NER). With higher budgets ($1500 and $2300), our in-domain pre-training of ProcBERT achieves the best results in combination with data annotation. For a smaller budget ($700), investing all funds in annotation and fine-tuning the standard BERT_{large} (considered as cost-free) will yield the best outcome. **#sent** is the number of sentences from the target domain, annotated under the given budget, used for training. † indicates results using EasyAdapt (subsection 4.4.3), where source domain data helps.

Source ⇒ Target Domain	**Budget**	**BERT_{base}** $0		**BERT_{large}** $0		**ProcBERT** $620		**Proc-RoBERTa** $800		**SciBERT** $1340	
		F_1	(#sent)	F_1	(#sent)	F_1	(#sent)	F_1	(#sent)	F_1	(#sent)
PUBMEDM&M⇒CHEMSYN	$700	$92.99_{0.2}$†(1166)		**93.32**$_{0.3}$†(1166)		$91.07_{0.4}$†(133)		N/A		N/A	
	$1500	$93.58_{0.3}$†(2500)		$94.14_{0.2}$†(2500)		**94.73**$_{0.1}$†(1466)		$93.83_{0.1}$†(1166)		$92.46_{0.2}$†(266)	
	$2300	$94.47_{0.2}$(3833)		$94.81_{0.1}$†(3833)		**95.17**$_{0.1}$†(2800)		$94.71_{0.2}$†(2500)		$94.39_{0.2}$(1600)	
WLP⇒CHEMSYN	$700	$93.31_{0.3}$†(1166)		**93.59**$_{0.3}$†(1166)		$90.83_{0.3}$†(133)		N/A		N/A	
	$1500	$94.31_{0.3}$†(2500)		$94.44_{0.2}$†(2500)		**94.88**$_{0.1}$†(1466)		$94.20_{0.2}$†(1166)		$92.39_{0.2}$†(266)	
	$2300	$94.47_{0.2}$(3833)		$95.09_{0.3}$†(3833)		**95.52**$_{0.1}$†(2800)		$94.63_{0.3}$(2500)		$94.39_{0.2}$(1600)	
WLP⇒PUBMEDM&M	$700	$74.41_{0.8}$†(686)		**75.41**$_{0.5}$†(686)		$68.30_{0.7}$†(78)		N/A		N/A	
	$1500	N/A		N/A		**76.76**$_{0.7}$†(862)		$76.02_{0.5}$(686)		$72.66_{0.4}$†(156)	
CHEMSYN⇒PUBMEDM&M	$700	$75.10_{0.5}$†(686)		**75.71**$_{0.4}$†(686)		$72.85_{0.6}$†(78)		N/A		N/A	
	$1500	N/A		N/A		**77.62**$_{0.5}$†(862)		$76.28_{0.5}$(686)		$73.93_{0.4}$†(156)	
CHEMSYN⇒WLP	$700	$72.23_{0.4}$†(1590)		**73.21**$_{0.2}$†(1590)		$72.42_{0.4}$†(181)		N/A		N/A	
	$1500	$73.30_{0.4}$†(3409)		$73.46_{0.4}$(3409)		**75.15**$_{0.4}$†(2000)		$73.90_{0.5}$†(1590)		$72.48_{0.4}$†(363)	
	$2300	$73.18_{0.4}$(5227)		$74.12_{0.2}$†(5227)		**75.88**$_{0.3}$†(3818)		$74.67_{0.4}$†(3409)		$74.68_{0.5}$†(2181)	
PUBMEDM&M⇒WLP	$700	$72.66_{0.3}$†(1590)		**73.18**$_{0.7}$†(1590)		$72.91_{0.3}$†(181)		N/A		N/A	
	$1500	$73.62_{0.4}$†(3409)		$73.46_{0.4}$(3409)		**75.25**$_{0.1}$†(2000)		$73.98_{0.2}$†(1590)		$72.58_{0.2}$†(363)	
	$2300	$73.73_{0.3}$†(5227)		$74.26_{0.2}$(5227)		**75.78**$_{0.3}$†(3818)		$74.68_{0.2}$†(3409)		$74.80_{0.2}$†(2181)	

4.4.3 EasyAdapt

In most of our experiments, we have access to a relatively large amount of labeled data from a source domain, and varying amounts of data from the target domain. Instead of simply concatenating the source and target datasets for fine-tuning, we propose a simple, yet novel variation of EasyAdapt [124] for pre-trained Transformers. More specifically, we create three copies of the model's contextualized representations: one represents the source domain, one represents the target, and the third is domain-independent. These contextualized vectors are then concatenated and fed into a linear layer that is 3 times as large as the base model's. When encoding data from a specific domain (e.g. CHEMSYN), the other domain's representations are zeroed out (1/3 of the new representations will always be 0.0). This

enables the domain-specific block of the linear layer to encode information specific to that domain, while the domain-independent parameters can learn to represent information that transfers across domains. This is similar to prior work using EasyAdapt [125] for LSTMs.

4.4.4 Experimental Results and Analysis

We present the test set NER and RE results with five annotation and pre-training combination strategies under six domain adaptation settings in Table 4.4 and Table 4.5, respectively. We report averages across five random seeds with standard deviations as subscripts. If pre-training costs go over the total budget, or available data goes under the annotation budget, we indicate the result as "NA". We now discuss a set of key questions regarding pre-training-based domain adaptation under a constrained budget, which our experiments can shed some light on.

Should we prioritize pre-training or data annotation for domain adaptation? For all six domain adaptation settings in Table 4.4, spending the entire budget on annotation and using the off-the-shelf language model BERT$_{large}$ works the best for NER when the budget is 700 USD, showing the effectiveness of data annotation in low-resource scenarios. As the budget increases, performance gains from labelling additional data diminish, and pre-training in-domain language models takes the lead. ProcBERT, which is pre-trained from scratch on the PROCEDURE+ corpus costing only 620 USD, performs best at budgets of 1500 and 2300 USD. This demonstrates that combining domain-specific pre-training with data annotation is the best strategy in high-resource settings. Similarly for RE, as shown in Table 4.5, using all funds for data annotation and working with off-the-shelf models achieves better performance at lower budgets, while domain-specific pre-training starts to excel as the budget increases past a certain point.

What is the starting budget to consider pre-training an in-domain language model? To answer this question, we plot test set NER performance for two strategies, BERT$_{large}$

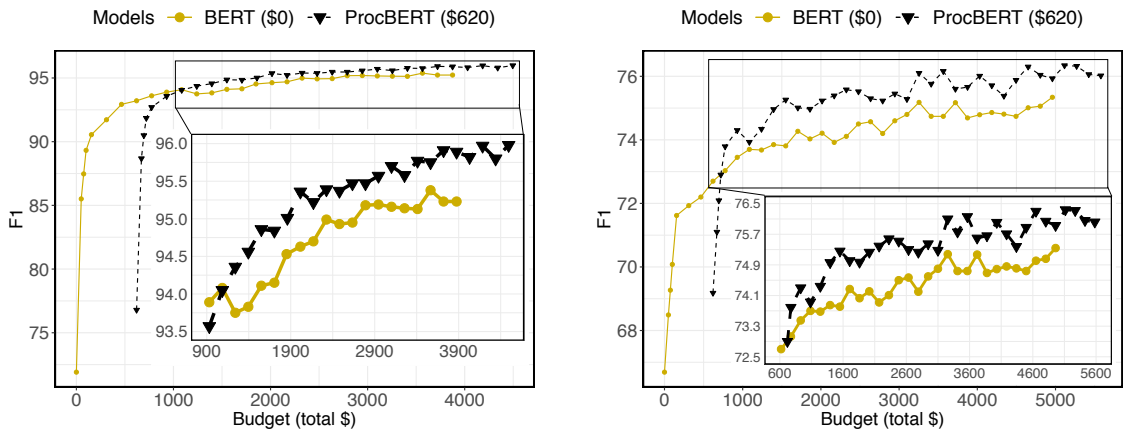
Table 4.5: Experiment results for Relation Extraction (RE). Similar to the observations in Table 4.4, regardless of a small or large budget, prioritizing data annotation in the target domain is the most beneficial. † indicates results using EasyAdapt (subsection 4.4.3), where source domain data helps.

Source ⇒ Target Domain	Budget	**BERT**$_{base}$ $0		**BERT**$_{large}$ $0		**ProcBERT** $620		**Proc-RoBERTa** $800		**SciBERT** $1340	
		F_1	(#sent)	F_1	(#sent)	F_1	(#sent)	F_1	(#sent)	F_1	(#sent)
PUBMEDM&M⇒CHEMSYN	$700	$90.12_{0.4}$†	(1166)	**90.71**$_{0.8}$†	(1166)	$88.21_{0.6}$†	(133)	N/A		N/A	
	$1500	$91.30_{0.2}$†	(2500)	$91.84_{0.5}$†	(2500)	**92.06**$_{0.4}$†	(1466)	$91.25_{0.7}$†	(1166)	$89.58_{0.5}$†	(266)
	$2300	$91.81_{0.2}$†	(3833)	**92.90**$_{0.4}$†	(3833)	$92.57_{0.2}$†	(2800)	$92.07_{0.3}$†	(2500)	$91.55_{0.2}$†	(1600)
WLP⇒CHEMSYN	$700	**90.20**$_{0.7}$†	(1166)	$90.15_{0.9}$	(1166)	$88.01_{0.6}$†	(133)	N/A		N/A	
	$1500	$91.34_{0.3}$†	(2500)	$91.61_{0.5}$†	(2500)	**92.27**$_{0.2}$†	(1466)	$91.77_{0.2}$†	(1166)	$89.16_{0.7}$†	(266)
	$2300	$92.08_{0.4}$†	(3833)	$92.73_{0.4}$†	(3833)	**92.85**$_{0.2}$†	(2800)	$92.44_{0.6}$†	(2500)	$91.42_{0.4}$†	(1600)
WLP⇒PUBMEDM&M	$700	$77.74_{0.7}$†	(686)	**79.33**$_{1.3}$†	(686)	$74.63_{0.4}$†	(78)	N/A		N/A	
	$1500	N/A		N/A		**80.10**$_{0.6}$†	(862)	$79.33_{1.3}$†	(686)	$75.70_{1.1}$†	(156)
CHEMSYN⇒PUBMEDM&M	$700	$77.02_{0.6}$†	(686)	**77.31**$_{0.4}$†	(686)	$73.92_{0.8}$†	(78)	N/A		N/A	
	$1500	N/A		N/A		**79.50**$_{0.7}$†	(862)	$77.12_{1.2}$†	(686)	$75.43_{0.6}$†	(156)
CHEMSYN⇒WLP	$700	$78.81_{0.6}$†	(1590)	**79.80**$_{0.6}$†	(1590)	$78.97_{0.7}$†	(181)	N/A		N/A	
	$1500	$79.91_{0.4}$†	(3409)	$80.15_{0.8}$†	(3409)	**80.87**$_{0.7}$†	(2000)	$79.69_{0.4}$†	(1590)	$79.44_{0.8}$†	(363)
	$2300	$80.54_{0.6}$†	(5227)	$80.97_{0.2}$†	(5227)	$81.15_{0.6}$†	(3818)	**81.33**$_{0.5}$†	(3409)	$80.16_{0.7}$†	(2181)
PUBMEDM&M⇒WLP	$700	$78.34_{0.8}$†	(1590)	**78.44**$_{1.2}$†	(1590)	$77.98_{0.8}$†	(181)	N/A		N/A	
	$1500	$79.40_{0.6}$†	(3409)	$79.78_{0.6}$†	(3409)	**80.45**$_{0.2}$†	(2000)	$78.79_{1.1}$†	(1590)	$78.76_{0.5}$†	(363)
	$2300	$79.93_{0.5}$†	(5227)	$80.04_{0.8}$†	(5227)	**80.85**$_{0.6}$†	(3818)	$79.49_{0.7}$†	(3409)	$80.33_{0.3}$†	(2181)

(investing all funds on annotation) and ProcBERT (combining annotation with pre-training), against varying budgets in Figure 4.2. Specifically, the budget of each strategy starts with the pre-training cost of its associated language model, and is increased by 155 USD increments until the total budget reaches the total cost of available data[15]. We observe a similar trend for both PUBMEDM&M⇒CHEMSYN and CHEMSYN⇒WLP: annotation alone works better in lower budgets while in-domain pre-training (ProcBERT) excels at higher budgets. However, the intersection of the curves for these two strategies occurs at different points, which are around 1085 USD and 775 USD.

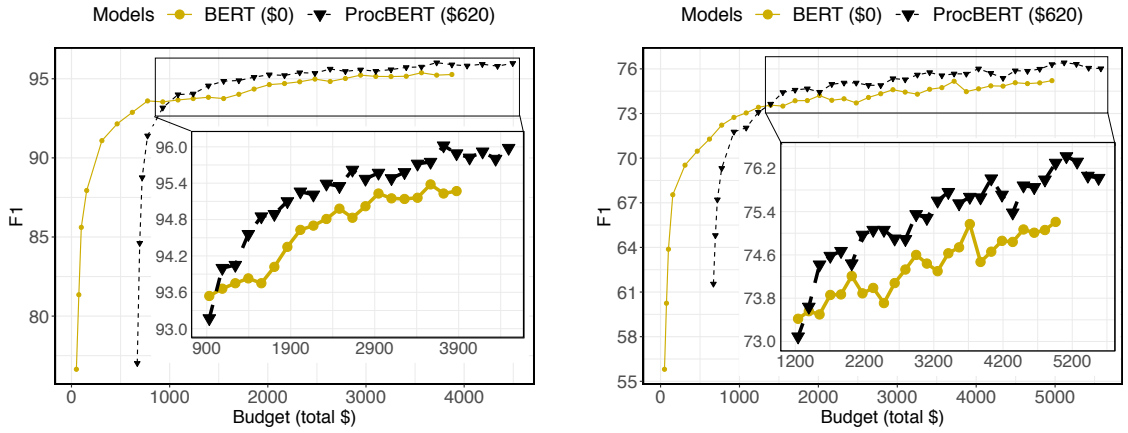
We hypothesize there are two reasons for this difference: 1) each target domain may require different amounts of labeled data to generalize well; 2) the quantity of labeled data from the source domain may also impact the need for data annotation in the target domain.

[15]We also add a few points at the start of each curve to make them smoother. For BERT$_{large}$, we add 50 USD, 75 USD and 100 USD. For ProcBERT, we add 670 USD, 695 USD and 720 UDS.



(a) PUBMEDM&M⇒CHEMSYN (b) CHEMSYN⇒WLP

Figure 4.2: Comparison of two domain adaptation strategies: 1) allocate all available funds to data annotation; 2) pre-train ProcBERT on in-domain data, then use the remaining budget for annotation. For small budgets, the former yields the best performance on NER, but as the budget increases, the later becomes the best choice.



(a) CHEMSYN (b) WLP

Figure 4.3: Comparison of spending the entire budget on data annotation ( ) and pre-training followed by in-domain annotation ( ), where models are trained on **target domain labeled data only**. The crossover point for WLP moves from 775 USD (adapted from CHEMSYN) to around 1395 USD (WLP only) demonstrating that a large source domain dataset can reduce the need for target domain annotation.

To testify our hypotheses, we evaluate the utility of annotation vs. pre-training where no source-domain data is available in Figure 4.3. This is almost identical to the setting of Figure 4.2 except models are trained on **target domain labeled data only**. Here, we observe the intersection for CHEMSYN is still around 1085 USD while the crossover point for WLP

Table 4.6: Test set F_1 on NER for entities seen and unseen in the training data for BERT_{large} and ProcBERT, when the two achieve very similar overall performance under the same budget constraints in Figure 4.3. ProcBERT performs better on the unseen entities.

Target	**Entities**	**BERT_{large}** \$0	**ProcBERT** \$620
CHEMSYN (budget \$1085)	seen by both	**97.63**	97.45
	unseen by both	84.96	**85.65**
	seen by BERT	**94.07**	93.44
	All	93.78	93.82
WLP (budget \$1395)	seen by both	**85.42**	84.78
	unseen by both	55.75	**57.19**
	seen by BERT	**76.50**	76.27
	All	73.51	73.51

moves from the original 775 USD (in Figure 4.2) to around 1395 USD. Our hypothesis is that WLP is a broader domain compared to CHEMSYN (WLP covers a more diverse range of protocols that include cell cultures, DNA sequencing, etc.), so it requires more annotated data to perform well under the setting of Figure 4.3. However, when adapted from a large source domain dataset like CHEMSYN, the need for annotated WLP corpus is reduced so that ProcBERT can outperform BERT_{large} at a lower budget.

Note that our estimated annotation cost, C_a, (per sentence) includes the annotation of both entity mentions and relations, so our analysis amortizes the cost of pre-training across both tasks. In a scenario where more tasks need to be adapted for the target domain, this could be accounted for simply by dividing the cost of pre-training among tasks, which would shift the black curves in Figure 4.2 and Figure 4.3 to the left, making pre-training an economical choice at lower budgets.[16]

When using the same budget and achieving similar F_1, how do pre-training and annotation differ? In the previous experiments, we show that in-domain pre-training is an

[16]For more discussion on our assumptions, see section 4.1.

effective domain adaptation method especially in high-budget settings. ProcBERT can work very well when trained with less labeled data. A plausible explanation is that in-domain pre-training improves generalization to new entities in the target domain, whereas additional annotation improves the performance on entities that are observed in the training corpus. To evaluate this hypothesis, we compare model predictions of the two strategies at the crossover points in Figure 4.3[17], and consider each entity in the test set as "Seen" or "Unseen" based on whether it was observed in the training set. Then, we calculate the F_1 score for each category as shown in Table 4.6. Although the models that are compared achieve nearly identical overall performance, the decomposition of performance on seen and unseen entities in Table 4.6 clearly suggests that in-domain pre-training leads to better generalization on unseen entities, whereas allocating more budget to annotation boosts performance on entities that were seen during training. This may help provide an explanation for the main finding of this study: in-domain pre-training results in better generalization on unseen mentions, leading to better marginal utility, but only after enough in-domain annotations are observed to fully cover the head of the distribution.

4.5 Positive Externalities of Pretraining on Downstream Tasks

So far, we have discussed domain adaptation as a consumer choice problem where annotation and pre-training costs are balanced to maximize performance in a target domain. However, pre-training on large quantities of natural language instructions can improve performance on additional tasks in the procedural text domain, as demonstrated in the following subsections.

4.5.1 Ancillary Procedural NLP Tasks

In addition to the procedural text datasets discussed in section 4.4, we experiment with three ancillary procedural text corpora, to explore how in-domain pretraining can benefit other tasks.

[17]We choose Figure 4.3 for this analysis instead of Figure 4.2 because we want to isolate the impact of source domain labeled data.

Table 4.7: Test set F_1 on six procedural text datasets. The best task performance is boldfaced, and the second-best performance is underlined. For the SOTA model of each dataset, we refer readers to the corresponding paper for further details: [67] for XWLP, [126] for CHEMU, [127] for RECIPE, [128] for NER on WLP, and [129] for RE on WLP.

Model	X-WLP		CHEMU		RECIPE	WLP		PUBMEDM&M		CHEMSYN	
	Core	Non-Core	NER	EE	ET	NER	RE	NER	RE	NER	RE
BERT$_{base}$	$74.79_{0.6}$	$77.57_{0.6}$	$95.14_{0.1}$	$91.93_{0.6}$	$80.62_{0.7}$	$73.87_{0.4}$	$80.25_{0.4}$	$74.80_{0.7}$	$78.73_{0.9}$	$95.09_{0.2}$	$92.63_{0.2}$
BERT$_{large}$	$\underline{75.53}_{1.7}$	$76.77_{0.5}$	$95.10_{0.2}$	$92.10_{0.9}$	$81.53_{0.5}$	$74.97_{0.3}$	$81.39_{0.5}$	$77.06_{0.3}$	$78.44_{1.3}$	$95.26_{0.1}$	$92.87_{0.5}$
RoBERTa$_{base}$	$75.04_{0.8}$	$77.24_{0.6}$	$95.05_{0.1}$	$\mathbf{92.54}_{1.2}$	$83.41_{0.1}$	$74.97_{0.5}$	$80.94_{0.5}$	$76.21_{0.3}$	$78.95_{0.7}$	$95.30_{0.2}$	$93.39_{0.3}$
RoBERTa$_{large}$	$73.77_{1.6}$	$74.37_{0.2}$	$95.16_{0.2}$	$92.10_{1.1}$	$\mathbf{84.54}_{0.8}$	$\mathbf{76.37}_{0.5}$	$79.76_{0.3}$	$\mathbf{78.70}_{0.6}$	$75.66_{0.3}$	$95.66_{0.2}$	$92.87_{0.2}$
SciBERT	$75.48_{0.7}$	$\underline{78.51}_{0.6}$	$\underline{95.63}_{0.0}$	$91.80_{0.5}$	$81.75_{0.4}$	$75.89_{0.4}$	$81.29_{0.5}$	$\underline{77.81}_{0.2}$	$79.54_{0.9}$	$95.82_{0.2}$	$93.27_{0.2}$
BioMed-RoBERTa	$74.89_{0.6}$	$76.39_{0.8}$	$95.32_{0.2}$	$\underline{92.42}_{0.6}$	$82.92_{0.3}$	$75.55_{0.2}$	$\mathbf{81.56}_{0.4}$	$77.14_{0.1}$	$78.44_{2.0}$	$95.38_{0.2}$	$93.16_{0.3}$
Proc-RoBERTa	$74.76_{0.7}$	$76.12_{0.9}$	$95.49_{0.1}$	$91.55_{0.6}$	$\underline{84.19}_{0.3}$	$75.76_{0.4}$	$80.79_{0.9}$	$76.91_{0.7}$	$79.16_{0.9}$	$95.67_{0.1}$	$93.31_{0.2}$
ProcBERT	$\mathbf{76.73}_{0.9}$	$\mathbf{78.57}_{0.8}$	$\mathbf{96.19}_{\mathbf{0.1}}$	$92.32_{0.2}$	$84.10_{0.3}$	$\underline{76.04}_{0.2}$	$\underline{81.44}_{0.4}$	$77.31_{0.5}$	$\mathbf{80.19}_{0.6}$	$\mathbf{95.97}_{0.2}$	$\mathbf{93.57}_{0.2}$
SOTA	76.5	78.1	95.70	95.36	81.96	77.99	80.46	–	–	–	–

The **CHEMU corpus** [130] contains NER and event annotations for 1500 chemical reaction snippets collected from 170 English patents. Its NER task focuses on identifying chemical compounds, and its event extraction (EE) task aims at detecting chemical reaction events including trigger detection and argument role labeling.

The **XWLP corpus** [67] provides the Process Event Graphs (PEG) of 279 wet-lab biochemistry protocols. The PEG is a document-level graph-based representation specifying the involved experimental objects.

The **RECIPE corpus** [131] includes annotation of entity states for 866 cooking recipes. It supports Entity Tracking (ET) task which predicts whether or not a specific ingredient is involved in each step of the recipe.

4.5.2 Experiments on Ancillary Tasks

For CHEMU, gold arguments are provided, so we only need to identify the event trigger and predict the role of the gold arguments. An event prediction is correct if the event trigger, associated arguments, and their roles match with the gold event mention. We tackle this task using a pipeline model similar to [123]. For XWLP, we focus on the operation argument role labeling task, where gold entities are provided as input. Following [67], we decompose

the results into "Core" and "Non-Core" roles. For the RECIPE task, we follow the data splits and fine-tuning architecture of [127]. The state of an ingredient in each cooking step is correct if it matches with the gold labels, as either present or absent.

Results. Test set results of eight pre-trained language models on six procedural text datasets are presented in Table 4.7.[18] ProcBERT, performs best in most tasks and even achieves the state-off-the-art performance on operational argument role labeling ("Core" and "Non-Core") of XWLP, showing the effectiveness of in-domain pre-training.

4.6 Positive Externalities on Semantic Search for Synthetic Procedures

Built upon our annotated datasets and pre-trained models, we present SYNKB, a semantic search engine for synthetic procedures. SYNKB has a number of advantages with respect to existing chemistry databases such as Reaxys: (1) We show that by automatically extracting information from millions of synthesis procedures described in U.S. and European patents using state-of-the-art NLP methods, we can achieve significantly higher recall than existing chemistry databases while maintaining high precision. In subsection 4.6.2, we demonstrate SYNKB's coverage is complementary to Reaxys; see Figure 4.5 for details. (2) SYNKB's novel graph search supports better coverage of reaction conditions than existing chemistry databases; this includes concentrations, reaction times, order of the addition of reagents, catalysts, etc. (3) We will make SYNKB available as open-source software on publication, in contrast, most existing chemistry databases are proprietary, with the notable exception of [132], which we compare to in subsection 4.6.2. We have built an online demo, which can be viewed at the following URL: `https://tinyurl.com/synkb`.

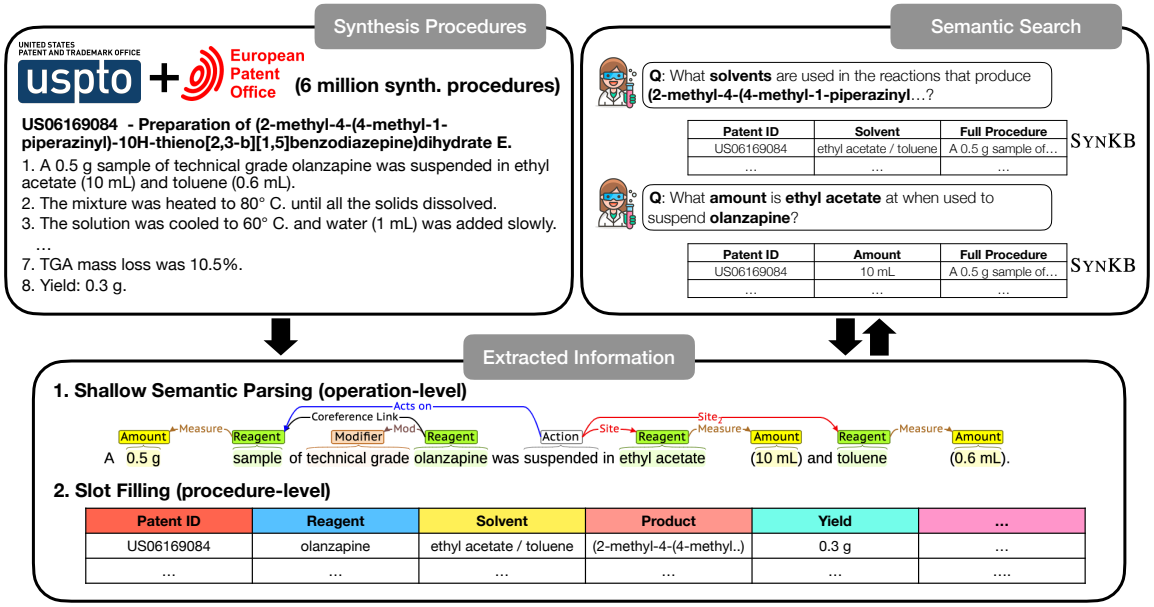


Figure 4.4: Overview of our semantic search system SYNKB, which searches over 6 million chemical synthesis procedures collected from patents. Users can enter structured queries to retrieve procedures concerning procedure-level or operation-level information.

4.6.1 SYNKB

As illustrated in Figure 4.4, SYNKB is an open-source system that allows chemists to perform structured queries across extensive collections of synthesis procedures. This system sources its procedures from the chemical patent segment of the PROCEDURE corpus, comprising six million chemical synthesis procedures derived from approximately 300k U.S. and European patents (detailed in Appendix section B.2). Below, we first describe how this corpus is automatically annotated with sentence-level action graphs, in addition to protocol-level slots relevant to chemical reactions. Following this, we demonstrate the semantic search capabilities enabled by SYNKB in paragraph 4.6.1.

Extracting Reaction Details from Synthetic Procedures To facilitate semantic search, we automatically annotate the corpus of 6 million synthetic procedures with semantic action graphs [95] in addition to chemical reaction slots [130].

[18]For CHEMU, we report the development set results because its test set is not publicly available.

Shallow Semantic Parsing. We first perform sentence-level annotation, where each step in the procedure is annotated with a semantic graph [133]. Nodes in the graph are experimental operations and their typed arguments, whereas labeled edges specify relations between the nodes (see the example shallow semantic parse in Figure 4.4). Here we use the CHEMSYN framework [103], which covers 24 types of nodes (such as *Action*, *Reagent*, *Amount*, *Equipment*, etc.) and 17 edge types (e.g. *Acts-on* and *Measure*). With these annotated semantic graphs, users can search for operation-level information, for example, the amount of DMF when used as a solvent to dissolve HATU (this will be further discussed in subsection 4.6.2). Following [133], we split semantic graph annotation into two sub-tasks, Mention Identification (MI) for node prediction and Argument Role Labeling (ARL) for edge prediction.

We use the same fine-tuning architectures as in [133]. Models are fine-tuned on the CHEMSYN corpus, which consists of 992 chemical synthesis procedures extracted from patents, and the resulting performance (averages across five random seeds) is shown in Table 4.8. We select model checkpoints via the Dev set performance out of five random seeds, and use the selected checkpoint for inference on our 6 million synthetic procedures.

Slot Filling. In the second task, we annotate procedures from a protocol perspective, i.e., identifying key entities playing certain roles in a protocol, which can be queried in a slot-based search. We use the CHEMU training corpus proposed in [130]. This dataset includes 10 pre-defined slot types concerning chemical compounds and related entities in chemical synthesis processes such as *Starting Material*, *Solvent*, and *Product*. Similar to the Mention Identification task, we treat Slot Filling as a sequence tagging problem. However, the input in Slot Filling is the entire protocol, rather than a single sentence, as in mention identification. We fine-tune models on the CHEMU dataset (see Table 4.8 for results), and then run inference on the chemical patent corpus using the learned model.

We use ProcBERT as the backbone for all fine-tuned models. The comparison between ProcBERT and other pre-trained models is presented in Table 4.8. Because ProcBERT

is pre-trained using in-domain data, we find that it outperforms both $\text{BERT}_{\text{large}}$ [69] and SciBERT [6] on all three tasks.

Table 4.8: Test set F_1 scores of fine-tuned models for the three annotation tasks. These numbers, averages across five random seeds with standard deviations as subscripts, are taken from our previous work [103]. Models using ProcBERT for contextual embeddings perform the best on all three tasks and are used for automatic annotations on six million synthesis procedures to construct SYNKB.

Annotation Task	**Dataset**	**Pre-trained Model**		
		$\text{BERT}_{\text{large}}$	SciBERT	ProcBERT
Mention Identification	CHEMSYN	$95.26_{0.1}$	$95.82_{0.2}$	$\mathbf{95.97}_{0.2}$
Argument Role Labeling		$92.87_{0.5}$	$93.27_{0.2}$	$\mathbf{93.57}_{0.2}$
Slot Filling	CHEMU	$95.10_{0.2}$	$95.63_{0.1}$	$\mathbf{96.19}_{0.1}$

Table 4.9: Comparison between our SYNKB and two performant databases. Our SYNKB provides more fine-grained annotations (more entity types and unique relation annotations) than the other two systems and covers more procedures than USPTO-Lowe, a database built using the largest open-source synthesis procedure corpus [132].

	SYNKB **(ours)**	**USPTO-Lowe**	**Reaxys**
License	Open source	Open source	Subscription
# Procedures (mill.)	6	2.4	57
# Entity Types	24	8	10
# Relation Types	17	-	-
Annotation	Automatic	Automatic	Manual

Semantic Search SYNKB offers search modalities specific to each of these two forms of annotation, i.e., semantic action graphs and chemical reaction slots, along with features designed to support practical use. The first type of query supported by SYNKB is **semantic graph search**, which allows users to search for synthesis procedures based on the semantic parse of the constituent operations. We adapt the graph query formalism proposed originally for syntactic dependencies in [134].[19] Formally, the input query $G = (V, E)$ is a labeled

[19]We refer readers to the tutorial of Odinson query language for more details of this graph query formalism.

directed graph. Each node $v_i \in V$ is specified as a set of constraints on matching entities (a single or multi-token span). For example, users can specify the node as `DMF` or `[word=DMF]`, which triggers an exact match on entity mentions containing the word "DMF". They can also constrain the entity type of the node using the expression `[entity=Type]`.[20] Moreover, nodes can be named `captures` when surrounded with `(?<name>...)`, e.g., the query `(?<solvent> DMF)` captures `DMF` as the `solvent`. As for the edge $e = (v_i, v_j, l) \in E$, we need to specify the direction and the semantic relation. Considering the query `(?<solvent> DMF) >measure (?<amount> 1 ml)`, it represents a semantic graph containing two entity nodes captured as `solvent` and `amount`, and an edge signaling the `measure` relation and its direction (from `solvent` to `amount`).

In addition, SYNKB supports **slot-based search**, which presents a structured search interface, with entries corresponding to CHEMU slots. A keyword entered into any entry restricts the retrieved set to procedures where the extracted slot contains the indicated keyword. Like the graph search, this returns a set of tuples with elements named with matching slots and containing the matching entity strings. The special token "?" can be used to match *any* slot value.

As for the implementation, the semantic graph search module is powered by Odinson [134], an open-source Lucene-based query engine. Odinson pre-indexes the annotated corpus by generating the inverted index for each procedure. Given an input query, Odinson performs a two-step matching process, where it first examines the node constraints via the inverted index; if this step works well, the semantic relations will be verified in the second step. The two-step matching process improves the speed of Odinson, and thus enables interactive querying. As for the slot-based search, it is supported by Elasticsearch[21] with the exception that, when users perform both types of search at the same time, we use the metadata search feature of Odinson for slot filters (we store slot values as metadata) to

[20] We store entity labels with the BIO tagging scheme, so users can match a single token entity with the expression `[entity=B-Type]` and a multi-token entity with the expression `[entity=B-Type][entity=I-Type]*`.

[21] `https://www.elastic.co/elasticsearch/`

improve the system's response speed.

4.6.2 Empirical Comparison

In subsection 4.6.1, we described the design and implementation of SYNKB including the underlying models, data preparation, and semantic search features. To demonstrate the utility of SYNKB for assisting chemists to search the literature for reaction details, we now evaluate its search features on ten example questions (Q1-Q10 in Table 4.10), which were collected from synthetic chemists working on the design of new synthesis protocols. In paragraph 4.6.2, we evaluate the slot-based search module of SYNKB and compare it with two existing databases which provides similar search features. In paragraph 4.6.2, we demonstrate how to use our novel semantic graph search module to answer operation-specific questions and evaluate its retrieved answers and procedures.

Slot-based Search Evaluation We benchmark the slot-based search module of SYNKB against Reaxys, one of the leading proprietary chemistry databases, and USPTO-Lowe, an automatically extracted database built using a large open-source synthesis procedure corpus [132]. Below, we first introduce these two databases briefly, and then evaluate the results of all three systems on the chemist-proposed questions.

Chemistry Databases The first database we compare with is **Reaxys**, a web-based commercial chemistry database, which contains comprehensive chemistry data, including chemical properties, compound structures, etc. What particularly interests us in Reaxys is that it contains expert-curated reaction procedures collected from extensive published literature such as chemistry-related patents and periodicals.[22] Also, key experimental entities in those reaction procedures, like participating reagents and reaction temperature, are specified. Thus, similar to our slot-based search, Reaxys allows users to search for reaction procedure information by applying text filters. Users can use its *Query Builder* module to specify multiple chemical reaction-specific filters, and then Reaxys returns all

[22] `https://www.elsevier.com/solutions/reaxys/features-and-capabilities/content`

Table 4.10: Search queries and resulting performance on 10 chemist-proposed questions for Reaxys, USPTO-Lowe, and SYNKB (ours). **# Proc.** is the number of returned procedures containing valid answers, and **# Ans.** refers to the number of distinct answer slots or captures in these procedures. The first six questions (Q1-Q6) are answerable for all three databases as they only require entity annotation while the last four questions (Q7-Q10) can only be answered by our SYNKB using our unique semantic action graph annotation. SYNKB consistently shows better recall than two compared databases while being highly accurate.

System	Input Query	# Proce.	# Ans.	Ans. Prec.
	Slot-based Search			
Q1 - What are the **solvents** used for reactions containing the reagent **triphosgene**?				
Reaxys	`{"reagent":"triphosgene"}`	35	7	**100**%
USPTO-Lowe	`{"reagent":"triphosgene", "solvent":"?"}`	3157	104	90%
SYNKB		**7184**	**127**	94%
Q2 - What are the **yields** (percent) of reactions producing **(5-Methylpyrimidin-2-yl)methanol**?				
Reaxys	`{"product":"(5-Methylpyrimidin-2-yl)methanol"}`	1	1	100%
USPTO-Lowe	`{"product":"(5-Methylpyrimidin-2-yl)methanol", "yield (percent)":"?"}`	1	1	100%
SYNKB		1	1	100%
Q3 - What are the **products** of reactions containing the reagent **trimethylsilyldiazomethane**?				
Reaxys	`{"reagent":"trimethylsilyldiazomethane"}`	438	75	**100**%
USPTO-Lowe	`{"reagent":"trimethylsilyldiazomethane", "product":"?"}`	517	335	98%
SYNKB		**1033**	**708**	96%
Q4 - What are the **products** of reactions containing the reagent **chlorosulfonic acid** and the solvent **chlorobenzene**?				
Reaxys	`{"reagent":"chlorosulfonic acid"} AND {"solvent":"chlorobenzene"}`	**148**	**65**	100%
USPTO-Lowe	`{"reagent":"chlorosulfonic acid", "solvent":"chlorobenzene", "product":"?"}`	6	2	100%
SYNKB		9	4	100%
Q5 - What are the **reaction times** for reactions using reagent **CDI (carbonyldiimidazole)**?				
Reaxys	`{"reagent":"CDI"} OR {"reagent":"carbonyldiimidazole"}`	93	24	**100**%
USPTO-Lowe	`{"reagent": "CDI OR carbonyldiimidazole", "reaction time":"?"}`	3722	339	100%
SYNKB		**6377**	**511**	94%
Q6 - What are the **reaction temperatures** for reactions containing reagent **trifluoromethanesulfonic acid**?				
Reaxys	`{"reagent":"trifluoromethanesulfonic acid"}`	104	3	**100**%
USPTO-Lowe	`{"reagent":"trifluoromethanesulfonic acid", "temperature":"?"}`	727	124	100%
SYNKB		**1937**	**243**	98%
	Semantic Graph Search			
Q7 - What are the **reagents** used to dilute **plasma**?				
SYNKB	`plasma <acts-on diluted >using (?<reagent> [entity=B-Reagent][entity=I-Reagent]*)`	24	16	100%
Q8 - What is the **pH** of a solution after being titrated with **NaOH**?				
SYNKB	`(?<ph> [entity=B-pH][entity=I-pH]+) <setting titrated >using NaOH`	39	21	95%
Q9 - What are the common **pore sizes** of **PTFE filters**?				
SYNKB	`PTFE filter >measure (?<pore_size> [entity=B-Generic-Measure][entity=I-Generic-Measure]*)`	183	39	92%
Q10 - What **molar concentration** is the reagent **HATU** at when **dissolved** in the solvent **DMF**?				
SYNKB	`HATU >measure (?<mole> [] [word=mmol|word=mol]) []{1,10} DMF >measure (?<volume> [] [word=ml|word=l])`	447	289	100%

matched reaction procedures along with identified entities in those procedures, which are available for download.

Apart from Reaxys, we also build a database using **USPTO-Lowe** [132], the largest available open-source chemical synthesis procedure corpus, for comparison. Similar to our SYNKB, this corpus includes automatic annotations of experimental entities on 2.4 million contained reaction procedures.[23] However, our SYNKB provides more fine-grained and comprehensive entity annotations (see Table 4.9 for the statistics of three experimented databases), and also annotates the relations between extracted entities, which constitute semantic graphs (paragraph 4.6.1) enabling operation-specific semantic graph search. As for the implementation, we load USPTO-Lowe's entity annotation into Elasticsearch, so this customized database can be used in the same way as the slot-based search module of our SYNKB.

Comparison with Examples We now compare three systems on six questions that were proposed by chemists (Q1-Q6) as these questions only require annotations on experimental entities and thus can be answered in all three systems. For example, Q1 ("What solvents are used in reactions involving triphosgene?") can be answered by the SYNKB query `{"reagent":"triphosgene", "solvent":"?"}`, as *reagent* and *solvent* are query-able ChEMU slots. Similarly, for Reaxys, experimental entities are specified for corresponding text filters.

We evaluate the output of each system from two perspectives: 1) recall, which is measured by the number of returned procedures containing valid answers and the number of distinct answer slots or captures in these procedures; and 2) precision, the proportion of correct answers among all predicted answers. In cases where the number of answers exceeds 50, we sample 50 answers from the full set to estimate precision.

The search queries and performance on each question for the three systems are shown in Table 4.10. We can see that, SYNKB consistently retrieves a larger number of relevant

[23] `https://www.nextmovesoftware.com/leadmine.html`

procedures and answers than Reaxys (5 out of 6 questions) while maintaining high precision. USPTO-Lowe, which uses a rule-based annotation model, shows competitive performance on precision but trails behind our SYNKB in terms of recall for all 6 questions. This comparison clearly shows the strength of our system: by leveraging state-of-the-art NLP for chemical synthesis procedures [103], we can provide chemists with abundant information, which is non-proprietary and delivered with high precision. Furthermore, we plot the Venn diagram (Figure 4.5) over the retrieved answers, which shows the percentage of unique and shared answers for each system out of all retrieved answers (we do macro-average across six questions.) Interestingly, only 18.1% of retrieved answers are shared among all three systems, and both our SYNKB and Reaxys contain a large number of unique answers, which take 31.5% and 17.4% of retrieved answers respectively. This shift in answer distribution suggests that our open-source SYNKB can be a good complement to proprietary chemistry databases like Reaxys, and it is better for users to use both of them if possible instead of choosing one over the other.

Semantic Graph Search Evaluation We evaluate our novel semantic graph search on four operation-specific questions (Q7-Q10). Unlike the six questions introduced above, these questions place constraints on the relations between mentioned entities, and thus are not answerable for Reaxys and USPTO-Lowe (due to the lack of relation annotation). For instance, to answer Q7 "What are the reagents used to dilute `plasma`?", a system needs to first locate the particular operation in a procedure where `plasma` is diluted, and then identify the reagent, which facilitates this dilution operation. This whole process can be realized in our semantic graph search module. Concretely, the graph-based query we use for Q7 is: "`plasma <acts-on diluted >using (?<reagent> [entity=B-Reagent][entity=I-Reagent]*)`", which matches procedures containing "`plasma`" and "`diluted`" connected in the same semantic graph and returns used reagents in the form of named captures. We evaluate the performance of the semantic graph search module by manually inspecting predicted answers (randomly

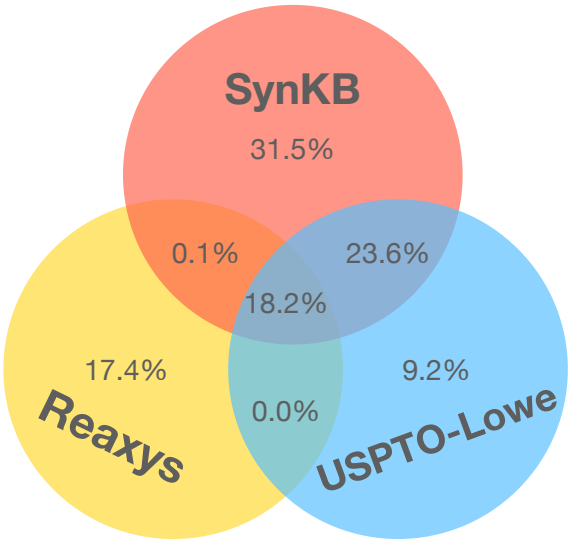


Figure 4.5: Venn diagram on the answer distribution of six slot-based search questions (macro-average) for all three databases. We can see that both our SYNKB and Reaxys cover high percentage of unique answers, suggesting that users should use them together if possible.

sampling 50 answers for Q10), and show results in Table 4.10. Similar to the findings in the slot-based search evaluation, SYNKB shows good coverage while maintaining high precision.

4.7 Conclusion

In this chapter, we address a number of questions related to the costs of adapting an NLP model to a new domain [135, 136], an important and well-studied problem in NLP. We frame domain adaptation under a constrained budget as a problem of consumer choice. Experiments are conducted using several pre-trained models in three procedural text domains to determine when it is economical to pre-train in-domain transformers [120], and when it is better to spend available resources on annotation. Our results suggest that when a small number of NLP models need to be adapted to a new domain, pre-training, by itself, is not an economical solution. Moreover, built upon our created datasets and pre-trained models, we present SYNKB, a system for large-scale extraction and querying of chemical synthesis procedures. SYNKB provides efficient searches against semantic action graphs and

chemical reaction slots derived from 6 million chemical synthesis procedures. A quantitative comparison with Reaxys, one of the leading commercial databases of reaction information, demonstrates the competence and versatility of our freely accessible system.

CHAPTER 5

SCHEMA-DRIVEN INFORMATION EXTRACTION FROM HETEROGENEOUS TABLES

Vast quantities of data are locked away in tables found in scientific literature, webpages, and more. These tables are primarily designed for visual presentation, and the underlying data is typically not available in any structured format, such as a relational or graph database. Some table collections have simple or uniform structures [137], making them easy to convert to relational data, for example Wikipedia tables [138, 139], however a lot of information is stored in tables with complex and varied layouts, such as tables of experimental data presented in scientific literature.

Prior work on extracting structured data from tables has focused on developing custom pipelines for each new table format or domain, for example extracting machine learning leaderboards from LaTeX result tables [140]. Importantly, the development of these specialized pipelines necessitates domain-specific labeled data, which not only incurs a significant cost in collection for every new extraction task but also constrains their applicability outside the originating domain.

In this chapter, we demonstrate that large language models can enable accurate domain-independent extraction of data from heterogeneous tables. To show this, we present a new formulation of the table extraction problem, which we refer to as Schema-Driven Information Extraction. In Schema-Driven IE, the only human supervision provided is a schema that describes the data model, including the attributes targeted for extraction, along with their corresponding data types. This resembles a JSON schema,[1] with a few customizations. Given an extraction schema, and a table as input, the model then outputs a sequence of JSON objects, each of which describes a single cell in the table and adheres

[1] `https://json-schema.org/`

to the user-provided schema. For example, as demonstrated in Figure 5.1, a domain expert outlines the attributes of interest related to experimental result cells in a machine learning table, and the model extracts JSON objects following this schema.

To evaluate the ability of LLMs to perform Schema-Driven IE, we introduce a new benchmark consisting of table extraction datasets in four diverse domains: machine learning papers, chemistry literature, material science journals, and webpages - each of which has a different data format (LaTeX, XML, CSV, and HTML, respectively). We curate and annotate data for the first two domains, while adapting existing datasets for the latter two.

We then use this benchmark to analyze the performance of open-source and proprietary LLMs. We find that proprietary models perform well across diverse domains and data formats, compared to open-source ones. For example, GPT-4 [141] and `code-davinci` [142], are capable of accurate table extraction (ranging from 74.2 to 96.1 F_1), given only a relevant data schema as supervision. This performance is comparable to fully supervised models, which operate at an F_1 range of about 64.1 to 96.1. We also present a number of analyses on various factors that are key to achieving good performance while minimizing inference costs, including retrieving text from outside the table, in addition to an iterative error recovery strategy. Moreover, we demonstrate the utility of Schema-Driven IE by evaluating performance on the downstream task of leaderboard extraction from machine learning papers [140].

While open-source models have yet to match the performance of their proprietary counterparts, our analysis reveals that recent models like CodeLlma-instruct-13B [143] show significant progress, e.g., on ML tables, its performance is comparable to that of GPT-3.5-turbo.

Furthermore, we demonstrate the feasibility of distilling efficient table extraction models, without sacrificing performance. We also highlight the potential of multi-modal models, such as GPT-4V, to extract data from table images, significantly expanding the set of documents this approach could be applied to. Through the introduction of a new benchmark for Schema-

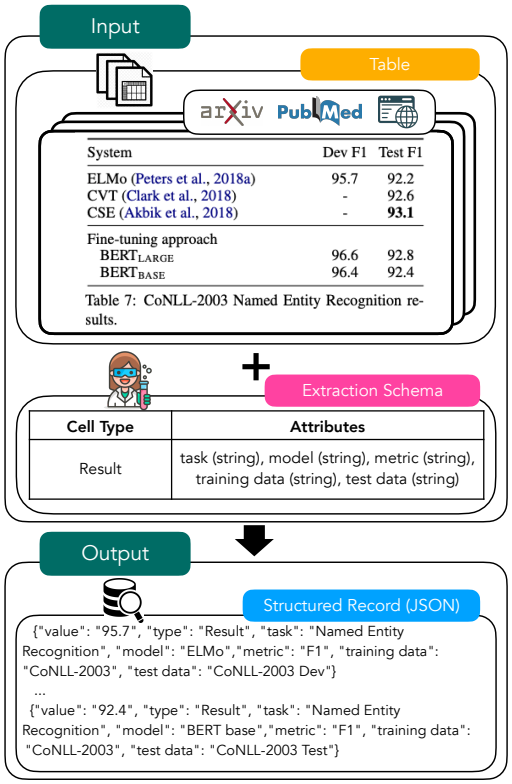


Figure 5.1: Overview of Schema-Driven Information Extraction. The input includes two elements: the source code of a table and a human-authored extraction schema, outlining the target attributes and their data types. The output consists of a sequence of JSON records that conform to the extraction schema.

Driven IE, we aim to foster the development of future open-source instruction-following models that can perform this task without relying on proprietary models.

5.1 Schema-Driven Information Extraction

We present a new task for extracting structured records from tables. As shown in Figure 5.1, the task input contains two elements: 1) a table T, comprised of n cells $\{c_1, c_2, \ldots, c_n\}$, optionally supplemented with contextual text, e.g., retrieved paragraphs from the same document; and 2) an extraction schema S that specifies attributes for k types of records, each containing m attributes $a_1, a_2, \ldots, a_m$ along with their respective data types (formatted as JSON templates in implementation). Given the input, the model generates a sequence of

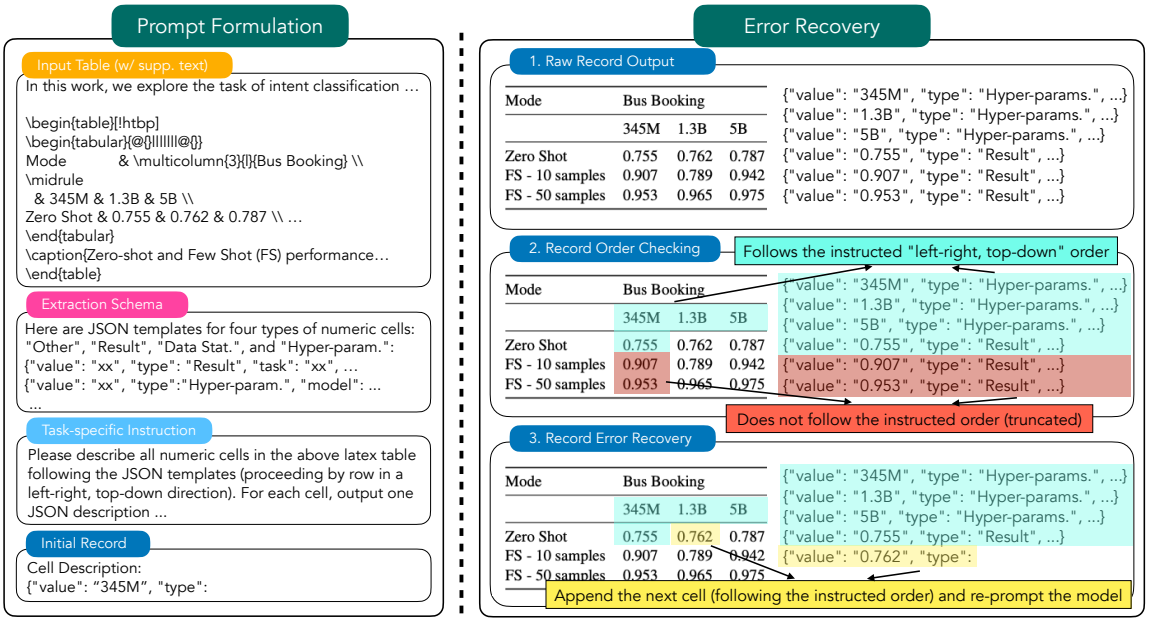


Figure 5.2: Left: Prompt formulation of our proposed method INSTRUCTE. Right: Illustration of our error-recovery strategy, which ensures the model compliance of the instructed cell traversal order and reduces inference costs.

n JSON objects, represented as $\{o_1, o_2, \ldots, o_n\}$. Each object o_i corresponds to one record type and comprises m key-value pairs $\{(a_{i,1}, v_{i,1}), \ldots, (a_{i,m}, v_{i,m})\}$, where $v_{i,j}$ is the value of attribute $a_{i,j}$ extracted for cell c_i.

Consider a table in an ML paper that displays various models' results. Our proposed task enables the extraction of result records from each cell in the table. These records include relevant attributes such as the evaluation metric, task, etc, which are structured in corresponding JSON objects and could facilitate meta-analysis of experiments or support research on reproducibility.

To demonstrate the feasibility of Schema-Driven IE on tables, we introduce **INSTRUCTE**, a baseline method to extract structured records from a broad range of semi-structured data, using only task-specific instructions. INSTRUCTE uses a template-based approach to information extraction [144, 145], where the extraction schema is represented as a series of JSON templates. The underlying LLM is instructed to select the appropriate template and populate it with extracted values for all the cells in an input table, following a specified cell

traversal order. As illustrated in Figure 5.2 (left), the prompt used by INSTRUCTE consists of four key components: an input table (optionally) supplemented with contextual text, an extraction schema, task-specific instructions, and an initial record for starting the process. For more details on the instruction formulation, please refer to Appendix section C.1.

Despite explicit instructions, we found that models often fail to generate valid JSON describing all cell descriptions in a single inference pass. Instead, models will often deviate from the instructed cell traversal order, leading to partial extraction of the input table's cells. To mitigate this, we use an iterative error recovery strategy. As shown on the right side of Figure 5.2, we detect deviations from the instructed *left-right, top-down* order by comparing predicted cell values with those from a rule-based cell detector (see Appendix section C.1). Then, we truncate the LLM's output and re-prompt the model with the truncated sequence, adding the value of the next target cell. This strategy guides the model to adhere to the instructed order, and continues iteratively until all records are generated. In subsection 5.3.4, we show that this approach achieves comparable performance to cell-by-cell prompting, while significantly reducing inference costs.

5.2 The SCHEMA-TO-JSON Benchmark

We now present the details of our benchmark, SCHEMA-TO-JSON, which is designed to assess the capabilities of LLMs to extract data from tables, adhering to a predefined schema. This benchmark contains tables from four diverse domains: machine learning papers, chemistry literature, material science journals, and webpages. Each domain features a unique textual format, namely, LaTeX, XML, CSV, and HTML, requiring models to generalize across domains and formats. For ML tables, we add relevant paragraphs from the same documents to provide additional context, testing the models' capacity to jointly understand tabular and textual data. We manually annotate datasets for the first two domains and adapt pre-existing datasets into our unified format for the latter two. Statistics of the four datasets are summarized in Table 5.1.

arXiv Machine Learning Tables We create a manually annotated dataset focused on tables from arXiv ML papers, emphasizing numeric cells that are classified into four categories: Experimental Results, Hyper-parameters, Data Statistics, or Other. Extraction attributes are pre-defined for the first three categories; for instance, Result records incorporate textual attributes such as *evaluation metric* (e.g., F_1) and *dataset* (e.g., SQuAD), as shown in Figure Figure 5.1. We collect papers from three subfields: Machine Learning, Computer Vision, and Natural Language Processing, and employ experts with backgrounds in ML research for annotation.[2] For more details of the ML tables, such as pre-defined attributes, annotation process, and inter-annotator agreement (IAA), see Appendix subsection C.2.1.

PubMed Chemistry Tables We also annotate a new dataset of PubMed tables describing physical properties of chemical compounds. The automated extraction of physical properties from such tables could provide substantial real-world benefits, for example collecting much-needed data for training ML models that can support inverse molecular design [146] and thus accelerating the drug design process [147, 148]. Here, we focus on cells concerning five important physical properties identified by chemists: IC_{50}, EC_{50}, GI_{50}, CC_{50}, and MIC. Three common attributes are manually extracted from tables for all properties: *unit*, *treatment* (experimental compound), and *target* (measured biological entity, usually a gene expression or disease organism). Similar to ML tables, domain experts annotate JSON records for relevant cells, and the high IAA score (Appendix subsection C.2.2) underscores the reliability of the dataset.

DISCOMAT [149] In addition to arXiv ML papers and PubMed chemistry publications, we incorporate DISCOMAT, an existing dataset focusing on glass composition tables from Elsevier material science journals. The task in DISCOMAT is to extract tuples comprising (*material*, *constituent*, *percentage*, *unit*) from given tables. We adapt DISCOMAT to fit our

[2]We limit the paper selection to those published between Oct. and Nov. 2022 to avoid contamination with GPT-4 (`0613`) and GPT-3.5, which were trained on data until Sep. 2021.

Table 5.1: Dataset statistics of four datasets in our SCHEMA-TO-JSON benchmark.

	ML (ours)	**Chemistry** (ours)	**DISCOMAT** (2022)	**SWDE** (2011)
Textual format	LaTeX	XML	CSV	HTML
# cell types	4	6	2	8
# attr. types	11	4	4	32
# papers (web.)	25	16	2,536	80
# tables (pages)	122	26	5,883	1,600
# anno. records	3,792	1,498	58,481	1,600
# records / table	31.1	57.6	9.9	1

Schema-Driven IE framework by grounding the *percentage* element to numeric cells in the table and considering the other elements as attributes. The model is tasked to identify numeric cells representing constituent percentages and predict the associated three attributes. We refer readers to [149] for more details of DISCOMAT.[3]

SWDE [150] Finally, we add SWDE (Structured Web Data Extraction) as a fourth dataset, aimed at extracting pre-defined attributes from HTML webpages. This dataset comprises roughly 124K pages gathered from eight distinct verticals, such as *Autos*, *Books*, and *Movies*. Each vertical includes ten unique websites (200 to 2000 pages for each website) and is associated with a set of 3 to 5 target attributes for extraction (refer to Table C.4 in the appendix for detailed statistics and vertical-specific attributes). Due to API budget limitations, we estimate InstrucTE performance on a sample of 1,600 pages (20 per website), and compute a boostrap confidence interval to ensure this provides a reasonable performance estimate.[4]

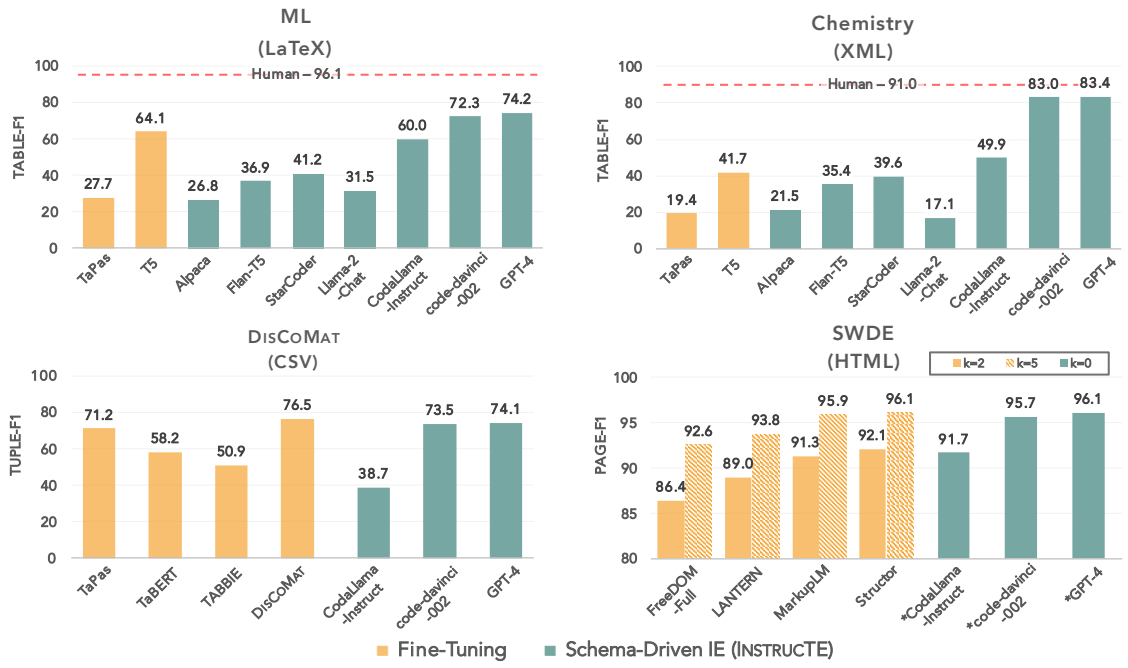


Figure 5.3: Capability of various LLMs to perform Schema-Driven IE, measured using the SCHEMA-TO-JSON benchmark. As noted in subsection 5.3.1, we employ Table-F_1 for our two newly annotated datasets and provide a measure of human performance, while Tuple-F_1 is used for DISCOMAT [149]. For SWDE [150], we report Page-F_1, and k represents the number of websites used in training from each vertical. *INSTRUCTE results on SWDE are computed on a 1,600 webpage sample due to API budget limitations - see section 5.2 and Footnote footnote 4.

5.3 Experiments

We evaluate the capability of various LLMs to perform schema-driven information extraction, in addition to full fine-tuning using our benchmark. For machine learning and chemistry tables, we use a subset of 10 and 7 randomly sampled papers separately for model development, which facilitates the training of supervised models, thereby enabling comparison with a schema-driven approach. For the two pre-existing datasets, we follow the data splits used in the original experiments.

[3]In the released corpus, tables are represented as matrices; we, therefore, transform them into CSV tables (using the pipe symbol "|" as the delimiter) prior to feeding them into LLMs.

[4]To validate that this subset is representative of the larger dataset, we estimate a 95% confidence interval for INSTRUCTE's performance using 1,000 bootstrap samples, and the resultant margin of error is 0.009, suggesting the sampling strategy is unlikely to substantially impact our performance estimate.

5.3.1 Evaluation Metrics

We introduce Table-F_1, a new reference-based evaluation metric gauging attribute prediction performance within a table. Table-F_1 represents the harmonic mean of precision and recall, with precision being the ratio of correctly predicted attributes to total predicted attributes. At the attribute level, we consider two metrics: token-level F_1 and exact match (EM). For token-level F_1, a prediction is deemed correct if the score exceeds a specific threshold, which is determined by maximizing the alignment between model predictions and human judgments on the dev set. (see Appendix section C.3). An example of Table-F_1 calculation is shown in Figure C.4 in the appendix. We report macro-averaged Table-F_1 given the wide variance in table sizes.

For DISCOMAT and SWDE, we use the metrics specified in the original papers to support comparisons with prior work. In the case of DISCOMAT, we report Tuple-F_1 ([149]), where a predicted 4-element tuple is considered correct only if it exactly matches the gold tuple. For SWDE, we report Page-F_1 [150], which measures the number of pages where the attributes are accurately predicted.[5]

5.3.2 Baselines & Implementation Details

We evaluate the capability of multiple LLMs to perform Schema-Driven IE, including API-based GPT-4 and GPT-3.5 models and open-source models, such as Llama2-Chat-13B [151], CodeLlama-instruct-13B [143], StarCoder-15.5B [152], LLaMA-7B [151], and Alpaca-7B [153]. We also frame Schema-Driven IE as a TableQA problem, applying multi-choice and extractive QA prompts for template selection and cell attribute prediction, respectively. Furthermore, we also evaluate T5-11B [70] and TaPas [154], a table-specialized LM. For implementation details of INSTRUCTE and other methods, see Appendix section C.4.

For DISCOMAT and SWDE, we compare INSTRUCTE with existing methods, which

[5]Notably, SWDE primarily focuses on identifying textual HTML nodes containing attribute values rather than exact text spans, so we use token-level F_1 to identify the most relevant HTML node for each extracted attribute.

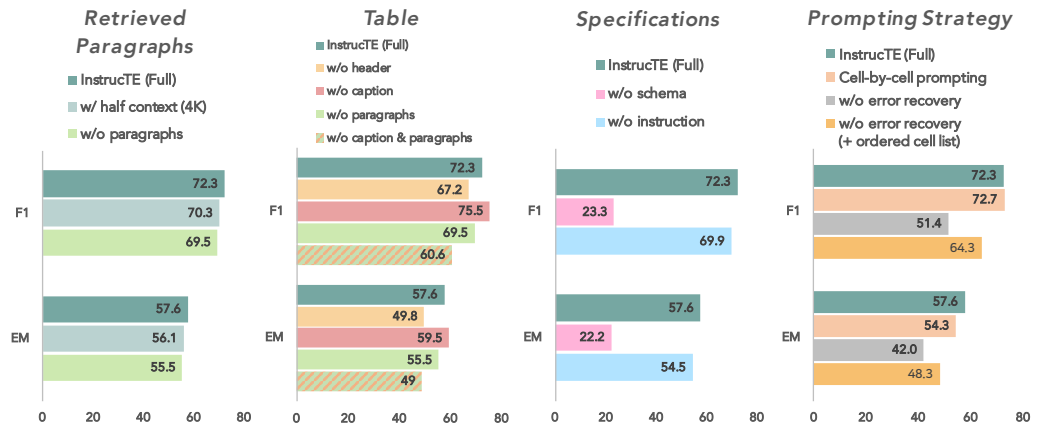


Figure 5.4: Ablation studies on various components of our INSTRUCTE (w/ `code-davinci-002`) on the ML tables. Interestingly, excluding the table caption improves performance. Our detailed analysis in Appendix section C.5 reveals that low-quality captions (e.g., lack of specificity) may confuse the model, leading to inaccurate predictions.

either design task-specific architectures, such as FreeDom [155] and LANTERN [156], or use LMs pretrained on tables or web pages, like TaPas [154], TaBERT [157], and MarkupLM [158].

5.3.3 Main Results

Figure 5.3 presents the main results from the comparison between INSTRUCTE and other methods on our SCHEMA-TO-JSON benchmark. We observe that INSTRUCTE, in conjunction with API-based models, achieves strong performance across domains and input formats, without any domain-specific labels. With GPT-4, INSTRUCTE can outperform fine-tuned models on ML and chemistry tables. However, a substantial disparity remains compared to human performance, e.g., the Table-F_1 on double-annotated examples for ML tables stands at 96.6 when applying thresholded token-level F_1 for attribute matching, which is 22.4 F_1 points higher than GPT-4.

For DISCOMAT and SWDE, GPT-4 performs on par or slightly trails behind the fully supervised state-of-the-art methods, signifying the potential of LLMs to act as flexible, powerful tools for extracting information from tables across diverse data formats and domains.

Despite a noticeable gap when compared to API-based LLMs, open-source models, like CodeLlama-instruct-13B, show promising results in ML and web domains, achieving 60.0 Table-F_1 and 91.7 Page-F_1 on ML tables and SWDE, respectively. This success is likely due to these domains being more represented in the model's pre-training corpus, suggesting that enhancing the pre-training data in less represented domains, such as chemistry and material science, may be an avenue for narrowing the gap with API-based models.

5.3.4 Analysis & Ablation Studies

We analyze the impact of different components of INSTRUCTE, including instruction formulation and error recovery, using ML tables.

LLMs & Instruction Formulation In Table 5.2, we compare different LLMs for INSTRUCTE, leading to two principal observations. First, code-oriented models have strong performance on Schema-Driven IE. This is evident from several key comparisons, such as the similarity in performance between `code-davinci-002` and GPT-4, the superior performance of `code-davinci-002` compared to its GPT-3.5 counterparts like `text-davinci-003` and `gpt-3.5-turbo`, and the fact that CodeLlama-instruct-13B significantly outperforms Llama2-chat-13B, approaching the performance of `gpt-3.5-turbo`. This superiority of code models might be attributed to the fact that Schema-Driven IE converts the source code of tables into JSON records. Second, non-code open-source models with similar sizes (for instance, those in the 6-7B range) tend to achieve comparable fine-tuning performance, though they might exhibit variations in prompting performance. Subsequently, we compare two prompt formulations: SCHEMA-TO-JSON and TableQA. From the T5-11B fine-tuning experiments, we observe that SCHEMA-TO-JSON attains better performance, demonstrating that incorporating task-specific instructions along with an extraction schema appears to be beneficial.

Table 5.2: TEST set performance on ML tables with different prompt formulations and LLMs.

Exp. Setup	Formulation	Model	Token-F_1	EM
Fine-tuning (# Train=1169)	TableQA	TaPas (large)	27.7	21.6
		T5 (11B)	61.2	46.2
	SCHEMA-TO-JSON	GPT-J (6B)	49.6	38.4
		LLaMA (7B)	51.3	38.0
		Alpaca (7B)	50.2	39.4
		T5 (11B)	64.1	50.2
Prompting (Schema only)	TableQA	Flan-T5 (11B)	36.9	27.7
	SCHEMA-TO-JSON	GPT-J (6B)	18.6	16.2
		LLaMA (7B)	13.5	11.5
		Alpaca (7B)	26.8	21.1
		Llama2-chat (13B)	31.5	23.0
		StarCoder (15.5B)	41.2	32.3
		CodeLlama-instruct (13B)	60.0	44.0
		`gpt-3.5-turbo`	64.1	47.9
		`text-davinci-003`	67.4	50.4
		`code-davinci-002`	72.3	57.6
		`gpt-4 (0613)`	**74.2**	**58.1**

Prompt Components & Error Recoverery Figure 5.4 shows INSTRUCTE's performance subject to the exclusion of varying instruction components. We use `code-davinci-002` for these experiments considering API budget limitations and its resemblance to GPT-4 in terms of performance and context length. We observe that removing supplementary text degrades performance. Table headers contribute positively as expected, while captions surprisingly do not. Further analysis on table captions is provided in Appendix section C.5, which suggests that unclear captions can sometimes mislead the model, resulting in inaccurate predictions. Notably, discarding the extraction schema, specifically JSON templates, causes a substantial performance decline, primarily due to attribute name mismatches in the evaluation. Lastly, we show that INSTRUCTE's performance drops significantly without error recovery. Compared to a cell-by-cell prompting strategy, error recovery offers similar performance at a fraction of the API cost ($100 v.s. $670 based on Azure pricing).[6]

[6]The pricing for `code-davinci-002` on Azure is $0.1 per 1,000 tokens as of June 23rd, 2023.

Table 5.3: Experimental results for knowledge distillation on the ML tables. Student models are trained on the synthetic data generated by the teacher. `GPU hours` refers to the training time ($\times$ number of GPUs) of student models for one epoch.

	Model (`GPU hours`)	**Token-Level F_1**			**EM**		
		P	**R**	**F_1**	**P**	**R**	**F_1**
Teacher	`code-davinci-002`	74.1	71.8	72.3	59.4	56.9	57.6
Student	LLaMA-7B (`50h`)	74.1	67.6	69.1	56.8	53.4	54.3
	Alpaca-7B (`50h`)	72.7	64.8	67.5	56.1	50.0	52.0
	T5-11B (`380h`)	75.8	71.4	73.2	60.3	56.7	58.1

5.3.5 Knowledge Distillation

Considering the strong performance of API-based models on Schema-Driven IE, we now show that it is possible to use knowledge distillation [68, 159] to build a cost-efficient compact model, using ML tables as a demonstration. Specifically, this process first generates synthetic data by performing inference on unlabeled tables using `code-davinci-002`, followed by fine-tuning a smaller model (e.g., 7B parameters) using the synthetic data. We compile a collection of 979 arXiv ML papers, submitted between 2008 and 2019, yielding 3,434 tables (containing a total of 100K cells). In Table 5.3, we can see that LLaMA-7B and Alpaca-7B demonstrate similar performance as seen in the fine-tuning results (Table 5.2). While fine-tuning LLaMA with LoRA presents noticeable computational and parameter efficiency, using the synthetic data to fine-tune the full parameters of T5-11B results in performance that matches that of the teacher model.[7]

5.3.6 Extraction from Image Tables

One practical challenge with INSTRUCTE is the need for table source code in a textual format. This issue is particularly pronounced in scientific domains where many articles are accessible only as PDFs or images. To address this, we propose a pipeline that first employs

[7]As we observe that the T5-11B student model slight outperforms the teacher model, we conduct a statistical significance test following [160]. With 1000 bootstrap samples, the p-value is 42.3%, suggesting that the performance gap between the T5-11B student model and the teacher model is not statistically significant.

multi-modal models to transform image-based tables into their textual counterparts, and then run INSTRUCTE on the resulting textual tables.

In a preliminary study with ML tables, we use GPT-4V to produce the LaTeX code (see Figure C.3 in Appendix), which is subsequently fed into INSTRUCTE. We find that the generated LaTeX code for all 68 test set tables can be rendered into valid PDFs, but only 51 tables accurately preserve headers and cells after the image-to-text conversion. Performance-wise, with GPT-4, our INSTRUCTE extraction pipeline achieves a Table-F_1 score of 70.2 from image inputs. This result is closely competitive to the 74.2 Table-F_1 achieved with direct textual inputs, demonstrating the potential for Schema-Driven IE directly from image inputs.

Additionally, we test `IDEFICS-80b-instruct` [161],[8] a leading open-source model on various image-text benchmarks, which unfortunately could not perform the table OCR.[9] This suggests a clear avenue for future research to focus on improving multimodal language models, enhancing their capability to accurately interpret and process image-based tables.

5.4 Extrinsic Evaluation: Extracting Leaderboards from ML Papers

In this section, we showcase the applicability of INSTRUCTE by extracting leaderboards from ML publications, a task that has been widely studied [162, 140]. This task differs from our proposed task as it involves linking results in ML tables to pre-defined leaderboard tuples (`task`, `dataset`, `metric`, `score`), whereas our task requires exhaustive extraction of table cells, in a domain-agnostic way. Comprehensive details on task definition, the leading supervised method AXCELL, and INSTRUCTE's application to the task are available in Appendix section C.6.

We compare INSTRUCTE with AXCELL on PWC LEADERBOARDS [140], the largest dataset for leaderboard extraction. For INSTRUCTE, we use `code-davinci-002` as the

[8] `https://huggingface.co/HuggingFaceM4/idefics-80b-instruct`

[9] The `IDEFICS-80b-instruct` model either produces unrelated content or simply output "I am sorry, but I cannot generate LaTeX code from the table."

Table 5.4: Leaderboard extraction results on the PWC LEADERBOARDS dataset.

Method	Micro-Average			Macro-Average		
	P	**R**	**F_1**	**P**	**R**	**F_1**
AXCELL	**25.4**	18.4	21.3	**21.5**	21.5	20.0
INSTRUCTE	20.1	20.8	20.5	20.3	23.1	19.6
INSTRUCTE+	23.9	**21.2**	**22.4**	21.2	**23.7**	**20.5**

backbone given its excellent performance on SCHEMA-TO-JSON. Table 5.4 presents the results of both methods, averaged over the selected papers. We can see that INSTRUCTE achieves competitive performance compared to the supervised AXCELL, highlighting the efficacy of our proposed approach. When we enhance INSTRUCTE with AXCELL's cell selection capabilities to create INSTRUCTE+ (more details in Appendix section C.6), it outperforms AXCELL, demonstrating the promising potential of combining these two approaches.

5.5 Related Work

Recently there have been many research efforts involving tables, particularly, table-to-text generation [163, 164, 165]. For example ToTTo [163] introduced the task of open-domain table-to-text generation. Another burgeoning area is question answering and fact verification on tables [166, 167, 168, 169]. Various approaches have emerged, such as semantic parsing for compositional question answering [170], and symbolic reasoning for fact verification [171]. In contrast, our work transforms tables into structured JSON records, where a data schema is the only supervision provided.

Furthermore, the rise of pre-trained language models [69], has stimulated interest in pre-training on semi-structured data. TaPas [154] and TaBERT [157] pre-train on linearized tables with a specialized cell index embedding. TABBIE [172] employs dual transformers for separate row and column encoding. HTLM [173] uses an HTML-specialized pre-training objective, facilitating a novel structured prompting scheme. Similarly, TabLLM

[174] utilizes linearized tables for general-purpose LLMs. In contrast to prior work on table pre-training, our work focuses on schema-driven IE rather than table classification or question answering.

Apart from traditional text-based information extraction [72, 103, 133], there is growing interest in semi-structured data [175, 176, 177, 140, 149, 178]. OpenCeres [176] and ZeroShotCeres [179], showcase open-domain information extraction from semi-structured websites, but their approach relies on seed examples from a knowledge base, or labeled webpages making these methods unsuitable for extracting data from scientific literature, etc. Our approach distinguishes itself by leveraging LLMs, to enable accurate extraction of a broad range of data formats and domains without any labels or custom extraction pipelines.

5.6 Conclusion

This chapter explores the capabilities of LLMs for extracting structured data from heterogeneous tables. We introduce a new task, Schema-Driven Information Extraction, which converts tables into structured records guided by a human-authored data schema. To facilitate this task, we present a comprehensive benchmark and develop INSTRUCTE, an innovative prompting-based extraction method. This method leverages instruction-tuned LLMs and integrates a unique error-recovery strategy for cost-efficient extraction. INSTRUCTE demonstrates impressive performance on four different datasets in our benchmark. Our work also elucidates the potential of building more compact models through distillation to reduce dependency on APIs, as well as extraction from image tables by using multi-modal models for image-to-text conversion. These contributions lay a solid foundation for the continued advancement of efficient table data extraction.

CHAPTER 6
CONCLUSION

In this thesis, we explore several aspects of label-efficient scientific information extraction, aligning our investigation with the progression of the NLP field. Our first study, presented in Chapter 2, introduces DNMAR, an approach for distantly supervised relation extraction. The supervision of this method comes from knowledge bases, which was a prominent label-efficient learning paradigm before 2018. The primary challenge in this paradigm lies in addressing the noise inherent to distant supervision.

The introduction of BERT [69] in late 2018 signaled a pivotal transition. Like many other NLP tasks, scientific information extraction began emphasizing task-specific fine-tuning built upon pre-trained language models. In Chapter 3, we investigate the application of these pre-trained models in both fully supervised and few-shot learning settings. Our study compares various fine-tuning architectures for scientific IE in a supervised context and evaluates the effectiveness of knowledge distillation from in-context learned models for developing compact yet robust methods in a few-shot setting.

Acknowledging the critical role of high-quality labeled data in fine-tuning paradigms, we also recognize the importance of the domain-specificity of pre-trained models. Chapter 4 delves into the dilemma of resource allocation between these two crucial aspects, a common quandary for NLP practitioners. Through experiments across three scientific procedural text domains, our findings consistently demonstrate that, within limited budgets, directing all resources towards annotation yields the most optimal performance. However, as the budget expands, a blend of data annotation and in-domain pre-training emerges as the more effective strategy. Building on this, we introduce SYNKB, an open-source knowledge base for chemical synthesis, which competes favorably with established chemistry databases like Reaxsys.

From 2022 onwards, the rise of instruction-following language models such as InstructGPT [180] and GPT-4 [141] brought about a new learning paradigm. In this framework, simple instructions can guide a model in performing an NLP task. Within this context, Chapter 5 presents SCHEMA-TO-JSON, a benchmark designed to assess LLMs' proficiency in schema-driven information extraction from tabular data. This benchmark encapsulates tables from diverse scientific arenas, machine learning, chemistry, and material science, as well as webpages from 80 distinct sites spanning 8 verticals. Alongside the benchmark, we present INSTRUCTE, an extraction method based on instruction-tuned LLMs. Remarkably, INSTRUCTE demonstrates competitive performance without task-specific labels, achieving F_1 scores ranging from 74.2 to 96.1. Moreover, we validate the feasibility of distilling compact models from INSTRUCTE to minimize extraction costs and reduce API reliance.

6.1 Future Directions

Our latest research into information extraction from tabular data highlights the immense potential of Large Language Models (LLMs) in facilitating label-efficient information extraction, especially within specialized arenas like scientific domains. Below, we outline prospective avenues for addressing new challenges arising from employing LLMs in label-efficient information extraction.

Human-AI Collaborated Annotation Even though our proposed INSTRUCTE (in conjunction with GPT-4) achieves competitive performance in table extraction, a noticeable gap remains compared to human performance. For instance, the Table-F_1 on double-annotated examples for the ML tables stands at 96.6, which is 22.4 F_1 points higher than INSTRUCTE. This suggests an enduring need for task-specific data annotation, primarily to establish a more precise task boundary. Given that LLMs often exhibit strong performance, future research could focus on pinpointing the more demanding cases. By directing human expertise to these specific situations, we can bolster annotation efficiency, leading to significant

savings in resources and time.

Robust Extraction with LLMs While there's notable success in extracting information from tabular data, previous studies [181, 182] have indicated the limitations of LLMs in processing human language text. One significant challenge arises because information extraction necessitates that the extracted content strictly originate from the input. Given the inherent generative propensity of LLMs, any deviance from this, such as producing hallucinated output, compromises the quality of results. Hence, there exists a compelling imperative to formulate a more resilient information extraction method leveraging LLMs. Such a method should harness the flexibility and adaptability of LLMs while addressing their propensity to generate extraneous content.

Appendices

APPENDIX A
STRUCTURED MINIMALLY SUPERVISED LEARNING FOR NEURAL RELATION EXTRACTION

A.1 MIRA

Prior work on minimally supervised structured learning has made use of sparse feature representations in combination with perceptron-style parameter updates. We found these updates result in poor performance on held-out development data, however, when using fixed, pre-trained continuous sentence representations. Perhaps this is not surprising because, intuitively, the margin of the dataset is likely to be smaller when using lower dimensional, continuous representations, leading to a larger mistake bound for convergence of the perceptron. To address this, we applied the the **M**argin **I**nfused **R**elaxation **A**lgorithm [27], as described below. In Section 3.1, we show empirically that MIRA is crucial for achieving good performance when using continuous representations, and consistently improves performance when using sparse features as well.

As discussed above, we have $\hat{\mathbf{z}}^{\text{KB}}$ the most likely sentence extractions conditioned on the KB and $\hat{\mathbf{z}}$, the MAP assignment to $\mathbf{z}$, ignoring the KB. MIRA updates parameters of the PCNN factors as follows:

$$\theta_j \quad = \quad \theta_j + \tau \cdot \big(F_j(x_i, \hat{z}_i^{\text{KB}}) - F_j(x_i, \hat{z}_i)\big)$$

here τ is an adaptive learning rate that scales the update to the smallest step size that achieves 0 loss on each mention-level classification:

$$\tau \quad = \quad \min\left(C, \frac{1 - \theta \cdot \left(F(x_i, \hat{z}_i^{\text{KB}}) - F(x_i, \hat{z}_i)\right)}{2||x_i||^2}\right)$$

θ is the concatenation of parameters θ_j across relations j, and similarly $F(\cdot)$ is the concatenation of PCNN features across relations. C is a hyper-parameter that truncates large steps and helps to prevent overfitting.

A.2 Differing Versions of the NYT-Freebase Corpus Used in Prior Work

We evaluate our models on the NYT-Freebase dataset [13] which was created by aligning relational facts from Freebase with the New York Times corpus, and has been used in a broad range of prior work on minimally supervised relation extraction. Originally, Riedel et. al. created two separate datasets for their HELDOUT and MANUAL evaluations. In the HELDOUT dataset, Freebase entity pairs are divided into two parts, one for training and one for testing. Training dyads are aligned to the 2005-2006 portion of the NYT corpus while testing dyads are aligned to the year 2007. In the MANUAL evaluation data, **all** Freebase entity pairs are matched against the 2005-2006 articles and used as training instances. Testing data in the Riedel et. al. MANUAL evaluation consists of dyads found within sentences in the 2007 NYT articles, for which at least one entity does not appear in Freebase; their models' predictions on this data were annotated manually. The Riedel et. al. data splits ensure it is not possible to have overlapping train/test entity pairs in either the HELDOUT or MANUAL evaluation.

As neural models with many parameters typically benefit significantly from larger quantities of training data, Lin et. al. [20] added training data from the Riedel et. al. MANUAL-TRAIN dataset into their training dataset. This modification of the training data leads to overlap in the entity pairs in the Lin et. al. training/test split. We found 11,424 entity pairs appearing in both training and test sets, however no sentences appear in both the

training and test sets, as the matched NYT articles came from different time periods.[1] In all our evaluations we remove these overlapping entity pairs from the training set, to ensure the models are not simply memorizing KB facts that appear in the training data. Figure A.1 shows that after removing these shared entity pairs from the training data, performance of the Lin et. al. PCNN+ATT model does not change very much when evaluating against held out facts from Freebase.

We name two versions of the NYT-Freebase dataset according to the number of training entity pairs they include. Table A.1 shows that NYTFB-280K training set has around 4 times the number of sentences and entity pairs as NYTFB-68K, and the proportions of multi-sentence entity pairs in NYTFB-280K is higher. In Table A.2, we can see that the distribution of relations in the two datasets are comparable, but NYTFB-280K has much more entity pairs for each relation. Also, Figure A.2 tells us that NYTFB-280K has a wider bag-size range and more large training bags.

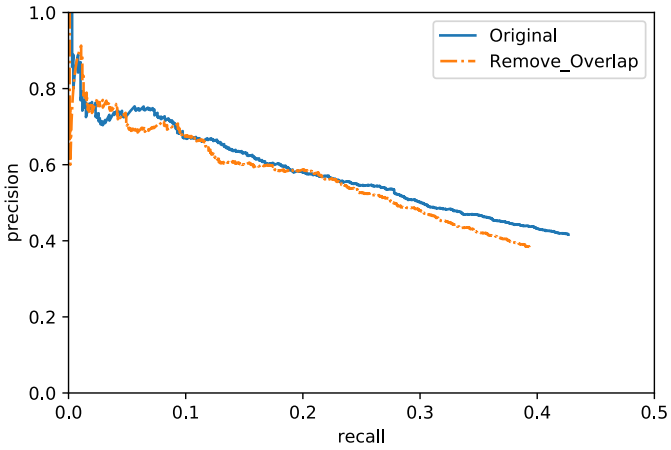


Figure A.1: Held-out evaluation precision / recall curves for PCNN+ATT model on original NYTFB-280K and its shared-entity-pairs-removed version.

[1]We downloaded the Lin et. al. [20] dataset from the associated Github repository (`https://github.com/thunlp/NRE`) on June, 2017. The repository was updated in March and May 2018, addressing the overlapping-entity-pairs issue using the same approach described in our paper.

Table A.1: Number of entity pairs and sentences in the training portion of Riedel's HELDOUT dataset (NYTFB-68K) and Lin's dataset (NYTFB-280K).

Dataset	NYTFB-68K (Riedel et. al. 2010)	NYTFB-280K (Lin et. al. 2016)
Entity pairs	67,946	280,275
Sentences	126,184	523,312
Distinct sent.	96,340	340,970
Relations	52	53

Table A.2: Distribution of the most frequent relations in the training set of NYTFB-68K and NYTFB-280K.

Relation	NYTFB-68K		NYTFB-280K	
	# EPs	percent	# EPs	percent
NA	63596	93.12	263372	93.52
/location/contains	2147	3.14	7760	2.76
/person/place_lived	581	0.85	2300	0.86
/person/nationality	436	0.64	2553	0.87
/person/place_of_birth	370	0.54	1400	0.49
/person/company	357	0.52	1417	0.50

Table A.3: AUC of sentential evaluation precision / recall curves for PCNNNMAR with three loss functions trained on NYTFB-68K. Mention-level hamming loss has some advantages over other two loss functions.

Method		DEV	TEST
0/1 loss	normal	82.6	82.8
	weighted	83.9	81.3
relation-level	normal	83.9	83.1
	weighted	84.6	81.1
mention-level	normal	82.4	83.9
	weighted	**85.4**	**86.0**

A.3 Variations on Structured Hinge Loss

Since we use the hinge loss as the loss function in our proposed PCNNNMAR model, the way that the hamming loss is calculated decides how we solve the argmax problem in loss-augmented search. In our experiments, we explore three ways to compute the loss: 0/1 loss, relation-level hamming loss and mention-level hamming loss. Table A.3 shows that

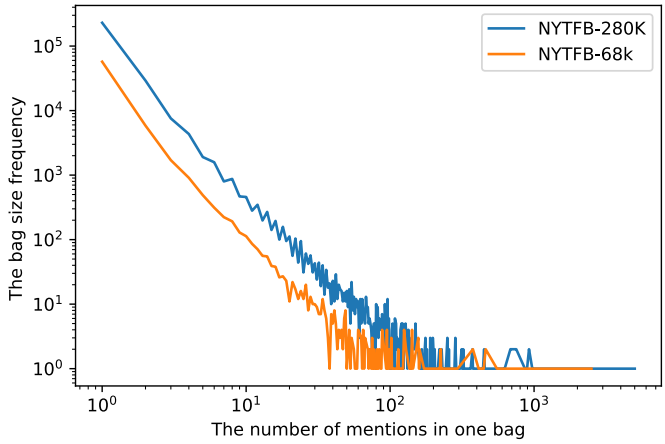


Figure A.2: Distribution of bag size in the training set of the NYTFB-68K and NYTFB-280K.

mention-level hamming loss has obvious advantage on AUC performance over other two methods. Although theoretically relation-level hamming loss should be better, it is really hard to find the exact argmax solution in loss-augmented inference with local search while we can easily get it with mention-level hamming loss.

APPENDIX B

PRE-TRAIN OR ANNOTATE? DOMAIN ADAPTATION WITH A CONSTRAINED BUDGET

B.1 Data Annotation

We annotate two datasets PUBMEDM&M and CHEMSYN in the domain of scientific articles and chemical patents mainly following the annotation scheme of the Wet Lab Protocols [WLP; 116]. On top of 20 entity types and 16 relation types in WLP, we supplement four entity types (COMPANY, SOFTWARE, DATA-COLLECTION and INFO-TYPE) and one relation type (BELONG-TO) due to two key features of our corpus: 1) scientific articles usually specify the provenance of reagents for better reproducibility; 2) it covers a broader range of procedures such as computer simulation and data analysis.

We recruit four undergraduate students to annotate the datasets using the BRAT annotation tool.[1] We double-annotate all files in PUBMEDM&M and half of the files in CHEMSYN. For those double-annotated files, the coordinator will discuss the annotation with each annotator making sure their annotation follows the guideline and dissolve the disagreement. As for the inter-annotator agreement (IAA) score, we treat the annotation from one of the two annotator as the gold label, and the other annotation as the predicted label, and then use the F1 scores of Entity(Action) and Relation evaluations as the final inter-annotator agreement scores, which are shown in Table B.1. We can see that CHEMSYN has higher IAA scores, and there are two potential reasons: 1) we annotate PUBMEDM&M first, so the annotators might be more experienced when they annotate CHEMSYN; 2) PUBMEDM&M contains more diverse content like wet lab experiments or computer simulation procedures while CHEMSYN is mainly about chemical synthesis.

[1] `https://github.com/nlplab/brat`

Table B.1: Inter-Annotator Agreement (F_1 scores on Entity/Action Identification and Relation Extraction).

Dataset	Entities/Actions	Relations
PUBMEDM&M	70.54	51.94
CHEMSYN	79.87	87.20

B.2 Procedural Corpus Collection

PubMed Articles. The first source of our procedural corpus is PubMed articles because they contain a large number of freely accessible experimental procedures. Specifically, we extract procedural paragraphs from the *Materials and Methods* section of articles within the Open Access Subset of PubMed. XML files containing full text of articles are downloaded from NCBI[2] and then processed to obtain all the paragraphs within the *Materials and Methods* section.

To improve the quality of our collected corpus, we develop a procedural paragraph extractor by fine-tuning SciBERT [6] on the SciSeg dataset [117], which includes discourse labels (`{Goal, Fact, Result, Hypothesis, Method, Problem, Implication}`) for PubMed articles. This extractor achieves an average F1 score of 72.65% in a five-fold cross validation, and we run it on all acquired paragraphs. We consider a paragraph as a valid procedure if at least 40% of clauses are labeled as `Method`. This threshold is obtained by manual inspection of the randomly sampled subset of the data.

In total, the PubMed Open Access Subset contains 2,542,736 articles, of which about 680k contain a *Materials and Methods* section. After running our trained procedural paragraph extractor, we retain a set of 1,785,923 procedural paragraphs. Based on a manual inspection of the extracted paragraphs, we estimate that 92% consist of instructions for carrying out experimental procedures.

[2] `https://www.ncbi.nlm.nih.gov/`

Chemical Patents. The second source of our corpus is the patent data because chemical patents usually include detailed procedures of chemical synthesis. We download U.S. patent data (1976-2016) from USPTO[3] and European data (1978-2020) from EPO[4] as XML files. Then we apply the reaction extractor developed by [118], a trained Naive Bayes classifier, to the *Description* section of our collected patents. Note that the U.S. patent data has two subsets, "Grant" (1976-2016) and "Application" (2001-2016). The "Application" subset covers the "Grant" subset from the same year, so for those overlapping years (2001-2016), we only use the U.S. patents from the "Application" subset. As a result, we get 2,435,999 paragraphs from 174,554 U.S. patents and 4,039,6065 paragraphs from 129,035 European patents. Lastly, we use the language identification tool **langid**[5] to build an English-only corpus, which includes 6,107,481 paragraphs.

B.3 Pre-training Details

We pre-train ProcBERT using the TensorFlow codebase of BERT [69].[6] We use the Adam optimizer [183] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $L2$ weight decay of 0.01. Following [69], we deploy the two-step regime. In the first step, we pre-train the model with sequence length 128 and batch size 512 for 1 million steps. The learning rate is warmed up over the first 100k steps to a peak value of 1e-4, then linearly decayed. In the second step, we train 100k more steps of sequence length 512 and batch size 256 to learn the positional embeddings with peak learning rate 2e-5. We use the original sub-word mask as the masking strategy, and we mask 15% of tokens in the sequence for both training steps.

For Proc-RoBERTa, we use the codebase from AI2,[7] which enables language model pre-training on TPUs with PyTorch. Similar to [120], we train RoBERTa on our collected procedural text corpus for 12.5k steps with a learning rate of 3e-5 and an effective batch size

[3] https://www.uspto.gov/learning-and-resources/bulk-data-products
[4] https://www.epo.org/searching-for-patents/data/bulk-data-sets.html
[5] https://github.com/saffsd/langid.py
[6] https://github.com/google-research/bert
[7] https://github.com/allenai/tpu_pretrain

Table B.2: Hyperparameters for models with BERT$_{base}$ or RoBERTa$_{base}$ architecture on budget-constrained domain adaptation experiments (denoted as "BUDGET") (section 4.4) and ancillary tasks (section 4.5).

Hyperparam.	BUDGET		X-WLP	CHEMU		RECIPE	WLP		PUBMEDM&M		CHEMSYN	
	NER	RE	ARL	NER	EE	ET	NER	RE	NER	RE	NER	RE
Learning Rate	1e-5	1e-5	1e-5	1e-5	1e-5	2e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
Batch Size	16	48	16	16	16	16	16	128	16	64	16	48
Max Seq. Length	512	256	512	512	512	512	256	128	512	256	512	256
Max Epoch	20	5	5	20	5	20	20	5	60	5	20	5

2048, which is achieved by accumulating the gradient of 128 steps with a basic batch size of 16. The input sequence length is 512 throughout the whole process, and 15% of words are masked for prediction.

B.4 Hyper-parameters for Downstream Tasks

We use the same five random seeds as [184] for all our experiments in section 4.4 and section 4.5.[8] We select the best hyperparameter values based on the average development set performances over five random seeds by grid search. For models with BERT$_{base}$ or RoBERTa$_{base}$ architecture, the search range includes learning rate (1e-5, 2e-5), batch size (16, 48, 64, 128), max sequence length (128, 256, 512) and epoch number (5, 20, 60), and the used hyperparameter values on budget-constrained domain adaptation experiments (denoted as "BUDGET") (section 4.4) and ancillary tasks (section 4.5) are shown in Table B.2. For BERT$_{large}$ and RoBERTa$_{large}$, the search range is different in learning rate (5e-6, 1e-5), batch size (4, 8, 12, 24, 64) and epoch number (3, 5, 10, 20), and the used values are shown in Table B.3.

[8] https://github.com/allenai/kb/blob/master/bin/run_hyperparameter_seeds.sh

Table B.3: Hyperparameters for BERT$_{large}$ and RoBERTa$_{large}$ on budget-constrained domain adaptation experiments (section 4.4) and ancillary tasks (section 4.5).

Hyperparam.	**BUDGET**		**X-WLP**	**CHEMU**		**RECIPE**	**WLP**		**PUBMEDM&M**		**CHEMSYN**	
	NER	RE	ARL	NER	EE	ET	NER	RE	NER	RE	NER	RE
Learning Rate	1e-5	5e-6	5e-6	1e-5	5e-6	5e-6	1e-5	5e-6	1e-5	5e-6	1e-5	5e-6
Batch Size	4	12	4	4	4	4	4	64	4	24	4	12
Max Seq. Length	512	256	512	512	512	512	256	128	512	256	512	256
Max Epoch	10	3	3	10	3	5	10	3	20	5	10	3

APPENDIX C
SCHEMA-DRIVEN INFORMATION EXTRACTION FROM HETEROGENEOUS TABLES

C.1 INSTRUCTE

Prompt Formulation Our proposed prompt consists of four components: 1) "Input Table (w/ supp. text)" includes the source code of the input table paired with supplementary text from the document; 2) "Extraction Schema" defines the JSON formats for extracted records, encompassing the record type, attribute names, and associated data types; 3) "Task-specific Instructions" outline the task execution process, addressing both the extraction process from individual cells and the model's traversal strategy across cells, such as *"left-right, top-down"*; 4) "Initial Record" is used to jump-start the prompting process, including the partial JSON record of the first cell.

For "Input Table (w/ supp. text)", we employ the BM25 algorithm to retrieve the most relevant paragraphs for each table. For "Extraction Schema", we propose two guidelines for schema design: 1) Attribute names should be specific, which decreases the probability of the model generating incorrect attributes, or hallucinations. For instance, when extracting relevant attributes about a movie from a movie webpage, it's advisable to use specific terms such as "`movie name`" or "`director name`", rather than the generic "`name`"; 2) Attributes should be strategically ordered, placing simpler attributes ahead of more complex ones as errors in preceding attributes can adversely affect the prediction of subsequent ones due to the autoaggressive nature of LMs. The exact INSTRUCTE prompts used in our experiments are shown in Table C.2 and Table C.3.

Cell Detector We develop a rule-based method to identify numeric cells for both the ML and chemistry tables. Specifically, for the ML tables, we use the row separator "\\" and the

column separator "&" to divide the table into cells. We then loop over each cell, checking for numeric values after stripping away any stylized text. In cases, where a cell contains multiple numeric values, such as "0 ± 0", we consistently choose the first numeric value. For the chemistry tables, the parsing process is more straightforward, owing to the structured XML format of the table. Here, we iterate over each cell, verifying if it contains a numeric value once stylized text has been removed. The performance of our rule-based cell detector on two datasets is presented in Table C.1. In the case of DISCOMAT, we use the cell detector provided by the original paper [149].

Table C.1: Results of (numeric) cell detection on ML and chemistry tables.

Dataset	**Split**	**P**	**R**	**F_1**
ML Tables	Dev	100.0	97.0	98.0
	Test	99.9	99.6	99.7
Chem. Tables	Dev	100.0	100.0	100.0
	Test	100.0	98.3	99.2

C.2 The SCHEMA-TO-JSON Benchmark

C.2.1 arXiv Machine Learning Tables

Extraction Attributes We design a set of extraction attributes for each of the three primary types of numeric cells in ML tables: "Result", "Hyper-parameter", and "Data Statistics". These attributes are outlined in detail below.

- "Result" includes seven attributes: `training data`, `test data`, `task`, `metric`, `model`, `model settings` and `experimental settings`. The first five attributes are fixed, with answers being text spans in the paper. The last two attributes, `model settings` and `experimental settings`, are free-form attributes, with answers being JSON objects. For example, the `experimental settings` attribute may be *{"number of*

training examples": "0"} for a zero-shot setting. This scheme is more detailed than previous approaches [162, 140] and can accommodate a broader range of ML paradigms and provide more granular information.

- "Hyper-parameter" includes optimization parameters like learning rate and batch size, as well as numeric descriptions of model architectures such as layer count. The three fixed attributes for this category are: `model`, `parameter/architecture`, and `dataset`.

- "Data Stat." covers four attributes: `dataset`, `dataset attribute`, `sub-set/group`, and `dataset features`. The `sub-set/group` specifies a dataset subset (e.g., "train" or "test"), while `dataset features`, a free-form attribute, captures various dataset characteristics like language or domain.

Annotation Process We sample 10 papers from each of three pertinent arXiv fields: Machine Learning, Computer Vision, and Computation and Language. After removing papers without LaTeX source code or any tables, a total of 25 papers are covered in our dataset. To optimize the annotation budget and the dataset diversity, we cap the number of annotated tables to five per paper. Recognizing the domain-specific expertise needed, we employ expert annotators with backgrounds in ML research, who are provided with tables in both LaTeX and PDF formats and encouraged to thoroughly read the paper before annotation. The annotation process comprises two steps: 1) identifying the numeric cells and their record types, and 2) filling in the slots of pre-determined attributes, forming a JSON record with keys as attribute names and values as extracted content, in a text editor. Consequently, the dataset contains 122 tables, with 3,792 cells and 21K attributes annotated.

Inter-annotator Agreement Score For the inter-annotator agreement (IAA) score, we treat double annotations of 13 tables as the gold and predicted labels separately, and calculate the mircro-averaged Table-F_1 (detailed in subsection 5.3.1) as the final IAA score. By

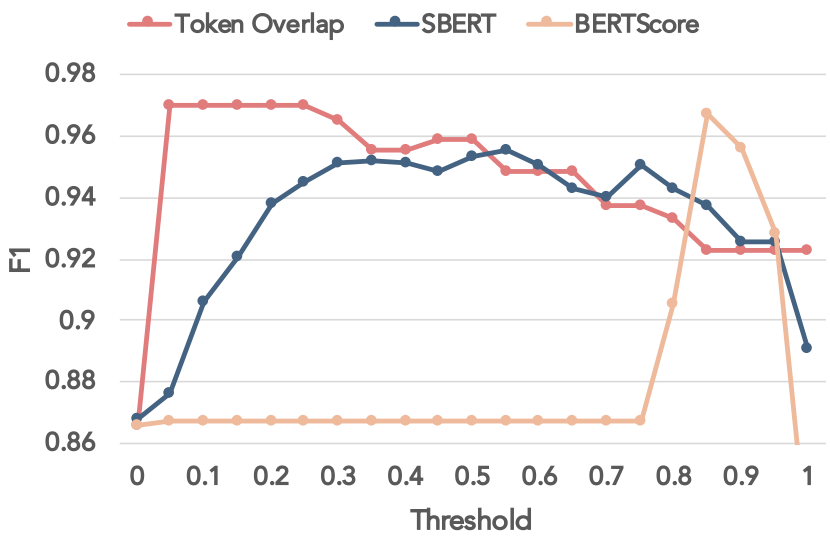


Figure C.1: Results of comparing various metrics, including token-level F_1, SBERT, and BERTScore, to human judgment over different thresholds. Numbers are computed over 677 sampled attributes that are paired with respective gold references.

applying thresholded token-level F_1 for attribute matching, we reach a Table-F_1 of 96.6. When we require attributes to be perfectly matched, the Table-F_1 drops to 76.6.

C.2.2 PubMed Chemistry Tables

Inter-annotator Agreement Score Similar to the ML tables, we calculate the Table-F_1 using the double-annotated 10 tables for the IAA score. For the chemistry tables, we achieve a Table-F_1 of 91.0 and 83.9 when applying thresholded token-level F_1 and exact match for attribute-level evaluation respectively.

C.3 Evaluation Metrics

Comparing an LLM-predicted JSON object with a gold JSON object is a non-trivial task, as those generative LLMs may produce text spans that do not exactly exist in the input table. Consequently, we devote substantial effort to examining various metrics to determine the one best suited for our task. Here, we consider three metrics: the standard token-level F_1 to capture the level of lexical overlap between the predicted and gold attributes, and two

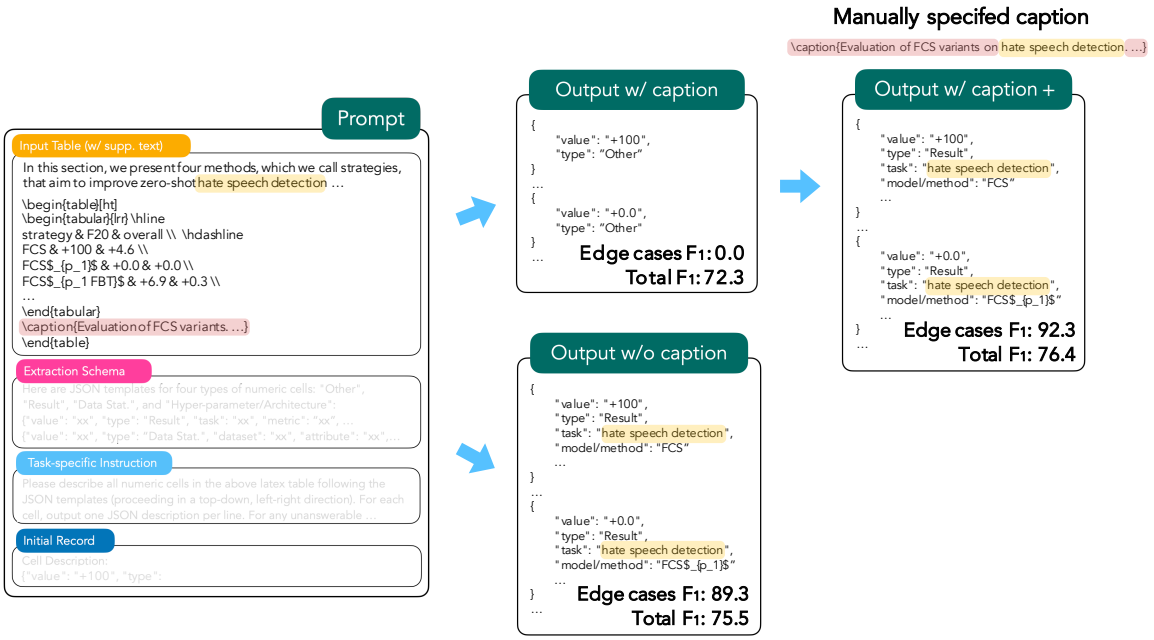


Figure C.2: An error analysis of edge cases in which the predictions made by INSTRUCTE with captions default to "Other" (resulting in an 0 F_1). Our hypothesis that this issue may stem from the caption's lack of specificity is tested by manually expanding the caption (displayed on the right). This amendment significantly improves the performance on these edge cases, increasing the F_1 score to 92.3.

semantic similarity metrics, SBERT [109] and BERTScore [185], to identify semantically similar expressions (e.g., # params vs. the number of parameters).

Meta Evaluation To assess how accurate each metric is compared to human evaluation, we manually annotated predicted-gold attribute pairs as to whether or not each pair matches. We consider a given pair to "match" if they are semantically equivalent, meaning they can be used interchangeably. For attributes that encapsulated multiple sub-attributes, we consider a pair to match if at least half of the sub-attributes are matched (i.e., F_1 score $\geqslant 0.5$), with the decision for each sub-attribute being based on the same as in the text-span attributes. For the set of pairs to annotate and use as a test set, we sample a total of 100 cell pairs (i.e., 677 attribute pairs) according to the following process: 1) we first uniformly sample a table from the development set (containing 10 papers); and 2) we then sample a random cell from the table, ensuring there were no duplicate cells. For each pair of predicted-gold attributes, each

metric's decision (1 or 0) is made using a specific threshold. For example, if the token-level F_1's score for paired attributes is 0.4 and the threshold is 0.5, then the decision would be 0, indicating no match. The decisions over the test set containing 677 attribute pairs are then compared to human evaluation. In this binary classification problem, F_1 is used to evaluate the performance of the metrics.

In Table C.5, we present the performances of each metric with the optimal threshold for each. Surprisingly, we find that the token-level F_1 (with a threshold of 0.25) decision aligns nearly perfectly with human judgment, and performs the best among all metrics for our task. This might suggest that discerning subtle differences is more crucial than identifying different phrases with the same meaning for this task. Based on these empirical findings, we opt for the token-level F_1 for automatic evaluation at the attribute level. This choice is highly desirable not only because of its high accuracy but also due to its simplicity.

C.4 Implementation Details

Considering the lengthy source code for tables, we employ different strategies to encode the input table and perform Schema-Driven IE, based on the context length of the chosen LLM. For LLMs with a larger context length, such as GPT-4, `code-davinci-002`, and CodeLlama, we input the full table and conduct the proposed error recovery process. For LLMs with a more limited context length, such as LLaMA, Alpaca, and T5-11B, we query each target cell individually. The input table is condensed by rows, retaining only the first two rows, typically containing headers, and the row with the query cell, with a special token `<select>` pinpointing the position of the query cell. We use greedy decoding to maximize the reproducibility of our results.

For the TableQA setting, we divide the problem into two steps: selecting the record type and predicting the relevant attributes. For T5 and Flan-T5, the first step is modeled as a multi-choice QA problem, where the model chooses the type of the query cell from a list of provided options. The second step is modeled as an extractive QA task, asking the model to

pinpoint the answer spans for the attributes associated with the selected type. For TaPas, the initial step is treated as a classification problem, whereas the latter one is handled as a cell selection problem. The hyper-parameters used for fine-tuning T5 and TaPas are presented in Table C.6.

C.5 Error Analysis of Caption

In subsection 5.3.4, we observe an unexpected finding that table captions do not enhance performance, but rather seem to detract from it, which is counterintuitive. To delve deeper into this observation, we conduct an error analysis. This involves comparing the performances of our INSTRUCTE system with and without captions at the table level. This analysis uncovers a few outliers (3 out of 68) where including a caption leads to a 0 F_1 score, whereas the score is near perfect when the caption is excluded. For instance, as depicted in Figure C.2, the predictions all fall into the "Other" category when a caption is included, leading to a 0 F_1 score in these outlier instances. Conversely, removing the caption results in an F_1 score of 89.3. This high score is due to the fact that retrieved paragraphs provide ample contextual information (e.g., "hate speech detection") even without the presence of a caption.

We hypothesize that the model's inclination to predict "Other" in the presence of a caption may be a consequence of the captions' lack of specificity with respect to the attributes relevant to the table cells (for example, "hate speech detection"). This lack of explicit, relevant details could create confusion in associating the caption with the retrieved paragraphs, thereby misleading the model. To test our hypothesis, we manually adjust the captions to include more specific attributes, such as "hate speech detection" and "T5-Base." As a result, we observe an improvement in the model's performance with the revised caption, with the total F_1 score even exceeding that achieved without a caption. This outcome partially supports our hypothesis and suggests that carefully crafted captions could indeed be beneficial, aligning with our initial expectations. However, this investigation also points to the fact that the model currently lacks robustness in handling these outlier scenarios.

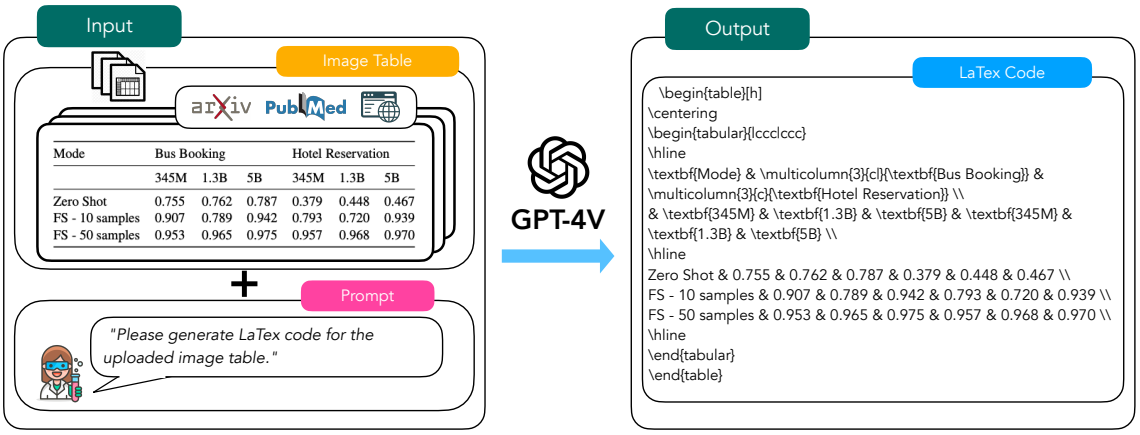


Figure C.3: Generate LaTeX code for image tables using GPT-4V.

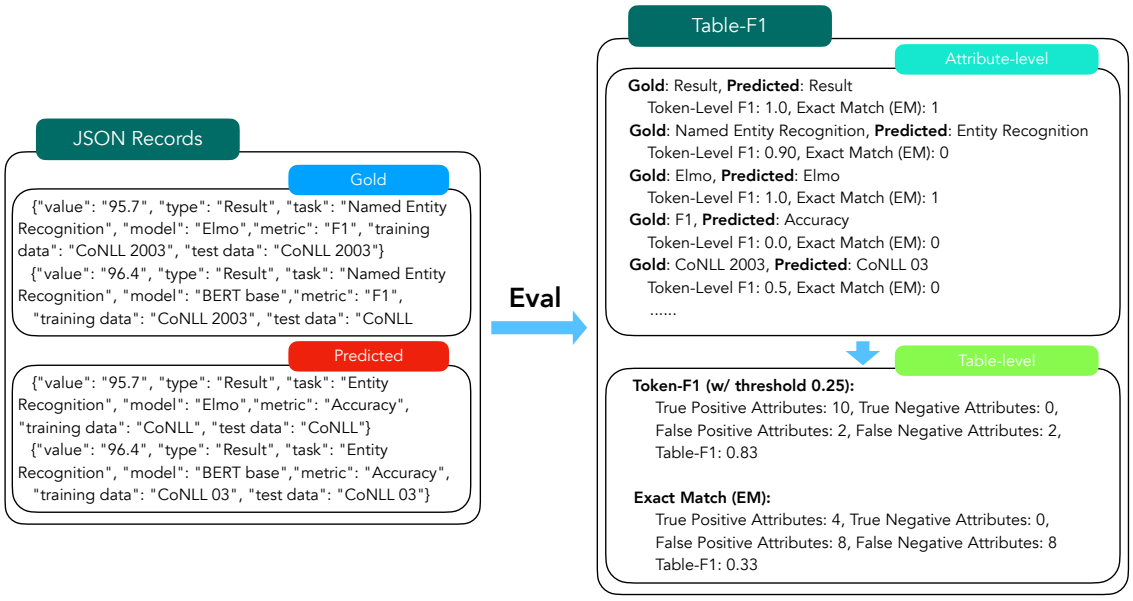


Figure C.4: An example of Table-F_1 calculation, where two predicted records are compared against the two gold records.

C.6 Leaderboard Extraction from ML Papers

C.6.1 Task Definition and State-of-the-Art

The task of leaderboard extraction entails extracting leaderboard tuples (`task`, `dataset`, `metric`, `score`) from tables in ML papers. Unlike our proposed Schema-Driven extraction, which requires open-domain span identification, leaderboard extraction presumes prior knowledge of all leaderboards, represented as pre-defined (`task`, `dataset`, `metric`) tuples,

and centers on linking numeric cells to these leaderboards.

The state-of-the-art leaderboard extraction method, AXCELL [140], is a comprehensive pipeline system comprising four components: Table Type Classification, Table Segmentation, Cell Linking, and Filtering. For each component, except the last one, AXCELL employs a supervised model. It starts with table type classification to identify result-related tables, which are then passed to the table segmenter responsible for annotating the header cells of the table. Following this step, a retrieval model links numeric cells in the table to pre-defined leaderboards using human-engineered features. Lastly, AXCELL filters and selects the best record based on the leaderboard taxonomy criteria, such as retaining higher values for "Accuracy" and lower ones for "error rate".

C.6.2 Leaderboard Extraction using INSTRUCTE

To extract leaderboards from an ML paper, we consider all tables that contain numeric cells, instead of selecting tables via a trained classifier as in AXCELL. For each table, we run INSTRUCTE using a customized leaderboard extraction JSON template. This template resembles the ML-table template with two additional fixed attributes: `eval split` and `eval class` in the "Result" cell template. We add the `eval split` attribute because the evaluated split is essential information for this task; for instance, *"dev F_1"* and *"test F_1"* are treated as different metrics in the leaderboard taxonomy. The `eval class` attribute is used to exclude sub-set or sub-class results that are typically present in analysis tables. After generating all predicted cell descriptions, we filter them based on three criteria: 1) the `type` attribute must be *"Result"*; 2) the `eval class` attribute must be *"all"* or *"Null"* as observed on the development set; and 3) the cell must be bolded in the table, as this usually indicates its superior performance and possible relevance to the leaderboard. For papers without any bolded cells, we experiment with two strategies: 1) include all the remaining cells in the table that meet the first two criteria; 2) use cells selected by AXCELL, as its engineered features for cell selection may be useful. This hybrid system is referred to as INSTRUCTE+.

We then use the predicted `task`, `dataset`, and `metric` attributes in each JSON record to match with the pre-defined leaderboards using token-level F_1, and we select the leaderboard with the highest average score over three attributes. Finally, following AXCELL, we choose the best record based on the leaderboard taxonomy criteria, e.g., retaining higher values for "Accuracy" and lower ones for "error rate".

Table C.2: INSTRUCTE prompts used for ML and chemistry tables.

Dataset	Full Prompt
ML Tables	[Retrieve paragraphs] [Input table] Here are JSON templates for four types of numeric cells: “Other”, “Result”, “Data Stat.”, and “Hyper-parameter/Architecture”: {“value”: “xx”, “type”: “Other”} {“value”: “xx”, “type”: “Result”, “task”: “xx”, “metric”: “xx”, “training data/set”: “xx”, “test data/set”: “xx”, “model/method”: “xx”, “model/method settings”: {“xx”: “yy”}, “experimental settings”: {“xx”: “yy”}} {“value”: “xx”, “type”: “Data Stat.”, “dataset”: “xx”, “attribute name”: “xx”, “sub-set/group name”: “xx”, “dataset features”: {“xx”: “yy”}} {“value”: “xx”, “type”: “Hyper-parameter/Architecture”, “model”: “xx”, “parameter/architecture name”: “xx”, “dataset”: “xx”} Please describe all numeric cells in the above latex table following the JSON templates (proceeding by row in a left-right, top-down direction). For each cell, output one JSON description per line. For any unanswerable attributes in the templates, set their value to the placeholder “xx” if it is of string type and {“xx”: “yy”} if it is of dictionary type. Cell Description: {“value”: “[Query cell]”, “type”:
Chem. Tables	[Input table] Here are JSON templates for six types of numeric cells: “Other”, “IC50”, “EC50”, “CC50”, “MIC”, and “GI50”: {“value”: “xx”, “type”: “Other”} {“value”: “xx”, “type”: “IC50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”} {“value”: “xx”, “type”: “EC50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”} {“value”: “xx”, “type”: “CC50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”} {“value”: “xx”, “type”: “MIC”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”} {“value”: “xx”, “type”: “GI50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”} Please describe all numeric cells in the above XML table following the JSON templates (proceeding by row in a left-right, top-down direction). For each cell, output one JSON description per line. For any unanswerable attributes in the templates, set their value to the placeholder “xx”. Cell Description: {“value”: “[Query cell]”, “type”:

Table C.3: INSTRUCTE prompts used for DISCOMAT and SWDE. For SWDE, we use the "Auto" vertical as an illustrative example, and the prompts for other verticals differ only in attribute names (refer to Table C.4 for the attributes of each vertical).

Dataset	Full Prompt
DISCOMAT	[Input table] Here are JSON templates for two types of numeric cells: "Other" and "Glass_Compound_Amount": {"value": "xx", "type": "Other"} {"value": "xx", "type": "Glass_Compound_Amount", "constituent compound name": "xx", "unit": "xx", "glass material/sample name/id/code": "xx"} Please describe all numeric cells in the above table following the JSON templates (proceeding by row in a left-right, top-down direction). For each cell, output one JSON description per line. For any unanswerable attributes in the templates, set their value to the placeholder "xx". Cell Description: {"value": "[Query cell]", "type":
SWDE-auto	[Input webpage] Here is the JSON template for automobile attribute extraction: {"webpage title": "xx", "automobile model (year)": "xx", "price": "xx", "engine type": "xx", "fuel economy": "xx"} Please extract the automobile' attributes from the HTML code above following the JSON template. For any unanswerable attributes in the template, set their value to the placeholder "<NULL>". {"webpage title": "[webpage title]", "automobile model (year)":

Table C.4: SWDE statistics.

Vertical	# Sites	# Pages	Attributes
Auto	10	17,923	`model, price, engine, fuel-economy`
Book	10	20,000	`title, author, ISBN-13, publisher, publish-date`
Camera	10	5,258	`model, price, manufacturer`
Job	10	20,000	`title, company, location, date`
Movie	10	20,000	`title, director, genre, rating`
NBA Player	10	4,405	`name, team, height, weight`
Restaurant	10	20,000	`name, address, phone, cuisine`
University	10	16,705	`name, phone, website, type`

Table C.5: Results of comparing various metrics, including token-level F_1, SBERT, and BERTScore, to human judgment. Numbers are computed over 677 sampled attributes that are paired with gold references. The highest achieved F_1 scores are displayed alongside the thresholds. A complete illustration of results, sorted by thresholds, can be found in Figure C.1 in Appendix.

	token-level F_1	**SBERT**	**BERTScore**
Meta Eval. F_1	97.0	95.6	96.7
Threshold	0.25	0.55	0.85

Table C.6: Hyper-parameters used for fine-tuning T5 and TaPas.

	T5 (11B)	**TaPas**
learning rate	1e-4	5e-5
batch size	8	32
# epoches	5	10

REFERENCES

[1] P. O. Larsen and M. von Ins, "The rate of growth in scientific publication and the decline in coverage provided by science citation index," *Scientometrics*, vol. 84, pp. 575–603, 2010.

[2] L. Bornmann, R. Mutz, and R. Haunschild, "Growth rates of modern science: A latent piecewise growth curve approach to model publication numbers from established and new literature databases," *Humanities and Social Sciences Communications*, vol. 8, pp. 1–15, 2020.

[3] L. Baldini Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski, "Matching the blanks: Distributional similarity for relation learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2895–2905.

[4] Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi, "A general framework for information extraction using dynamic span graphs," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3036–3046.

[5] S. Jain, M. van Zuylen, H. Hajishirzi, and I. Beltagy, "SciREX: A challenge dataset for document-level information extraction," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7506–7516.

[6] I. Beltagy, K. Lo, and A. 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)*, Hong Kong, China, Nov. 2019, pp. 3615–3620.

[7] Y. Gu *et al.*, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, pp. 1–23, 2020.

[8] F. Bai and A. Ritter, "Structured Minimally Supervised Learning for Neural Relation Extraction," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3057–3069.

[9] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel, "The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation," *LREC*, 2004.

[10] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, Association for Computational Linguistics, 2009, pp. 1003–1011.

[11] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1499–1509.

[12] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piecewise convolutional neural networks," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1753–1762.

[13] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2010.

[14] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 541–550.

[15] A. Ritter, L. Zettlemoyer, Mausam, and O. Etzioni, "Modeling missing data in distant supervision for information extraction," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 1, pp. 367–378, 2013.

[16] W. Xu, R. Hoffmann, L. Zhao, and R. Grishman, "Filling knowledge base gaps for distant supervision of relation extraction," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, 2013.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[18] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *Advances in neural information processing systems*, 2004, pp. 25–32.

[19] D. Belanger and A. McCallum, "Structured prediction energy networks," in *International Conference on Machine Learning*, 2016, pp. 983–992.

[20] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2124–2133.

[21] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

[22] J. Eisner and R. W. Tromble, "Local search with very large-scale neighborhoods for optimal permutations in machine translation," in *Proceedings of the HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, 2006.

[23] P. Liang, M. I. Jordan, and D. Klein, "Type-based mcmc," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 573–581.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[25] P. Verga, D. Belanger, E. Strubell, B. Roth, and A. McCallum, "Multilingual relation extraction using compositional universal schema," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 886–896.

[26] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, Association for Computational Linguistics, 2002.

[27] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 951–991, 2003.

[28] H. Ji, R. Grishman, H. T. Dang, K. Griffitt, and J. Ellis, "Overview of the tac 2010 knowledge base population track," in *Third Text Analysis Conference (TAC 2010)*, 2010.

[29] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[30] T. Berg-Kirkpatrick, D. Burkett, and D. Klein, "An empirical investigation of statistical significance in nlp," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, 2012.

[31] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, ser. COLING '92, Nantes, France: Association for Computational Linguistics, 1992, pp. 539–545.

[32] S. Brin, "Extracting patterns and relations from the world wide web," in *International Workshop on The World Wide Web and Databases*, Springer, 1998, pp. 172–183.

[33] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proceedings of the fifth ACM conference on Digital libraries*, ACM, 2000, pp. 85–94.

[34] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning.," in *AAAI*, Atlanta, vol. 5, 2010, p. 3.

[35] S. Gupta and C. Manning, "Improved pattern learning for bootstrapped entity extraction," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, 2014, pp. 98–108.

[36] M. Qu, X. Ren, Y. Zhang, and J. Han, "Weakly-supervised relation extraction by pattern-enhanced embedding learning," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2018.

[37] M. Craven, J. Kumlien, *et al.*, "Constructing biological knowledge bases by extracting information from text sources.," in *ISMB*, vol. 1999, 1999, pp. 77–86.

[38] B. Snyder and R. Barzilay, "Database-text alignment via structured multilabel classification.," in *IJCAI*, 2007, pp. 1713–1718.

[39] F. Wu and D. S. Weld, "Autonomously semantifying wikipedia," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ACM, 2007, pp. 41–50.

[40] M. Surdeanu, "Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling.," in *TAC*, 2013.

[41] M. Surdeanu *et al.*, "A simple distant supervision approach for the tac-kbp slot filling task.," in *TAC*, 2010.

[42] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning, "Multi-instance multi-label learning for relation extraction," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, Association for Computational Linguistics, 2012, pp. 455–465.

[43] L. Wang, Z. Cao, G. de Melo, and Z. Liu, "Relation classification via multi-level attention cnns," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Aug. 2016.

[44] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 35–45.

[45] M. Yu, M. R. Gormley, and M. Dredze, "Combining word embeddings and feature embeddings for fine-grained relation extraction," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[46] G. Angeli, J. Tibshirani, J. Wu, and C. D. Manning, "Combining distant and partial supervision for relation extraction," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[47] I. Beltagy, K. Lo, and W. Ammar, "Improving distant supervision with maxpooled attention and sentence-level supervision," *arXiv preprint arXiv:1810.12956*, 2018.

[48] Y. Wu, D. Bamman, and S. Russell, "Adversarial training for relation extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1778–1783.

[49] Y. Yaghoobzadeh, H. Adel, and H. Schütze, "Noise mitigation for neural entity typing and relation extraction," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, vol. 1, 2017, pp. 1183–1194.

[50] P. Qin, W. XU, and W. Y. Wang, "Dsgan: Generative adversarial training for distant supervision relation extraction," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

[51] L. Yao, A. Haghighi, S. Riedel, and A. McCallum, "Structured relation discovery using generative models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

[52] A. Ritter, Mausam, O. Etzioni, and S. Clark, "Open domain event extraction from twitter," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12, 2012.

[53] G. Stanovsky, I. Dagan, *et al.*, "Open ie as an intermediate structure for semantic tasks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015.

[54] L. Huang *et al.*, "Liberal event extraction and event schema induction," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[55] N. Weber, N. Balasubramanian, and N. Chambers, "Event representations with tensor-based compositions," *AAAI*, 2017.

[56] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "Relation extraction with matrix factorization and universal schemas," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 74–84.

[57] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *International Conference on Machine Learning (ICML)*, 2011, pp. 809–816.

[58] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[59] K. Chang, W. Yih, B. Yang, and C. Meek, "Typed tensor decomposition of knowledge bases for relation extraction," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[60] D. Weiss, C. Alberti, M. Collins, and S. Petrov, "Structured training for neural network transition-based parsing," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 323–333.

[61] G. Durrett and D. Klein, "Neural crf parsing," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 302–312.

[62] D. Andor *et al.*, "Globally normalized transition-based neural networks," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[63] C. Cherry and H. Guo, "The unreasonable effectiveness of word representations for twitter named entity recognition," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 735–745.

[64] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 1064–1074.

[65] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of NAACL-HLT*, 2016, pp. 260–270.

[66] C. Li, A. Porco, and D. Goldwasser, "Structured representation learning for online debate stance prediction," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018.

[67] R. Tamari, F. Bai, A. Ritter, and G. Stanovsky, "Process-Level Representation of Scientific Protocols with Interactive Annotation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, Online, Apr. 2021, pp. 2190–2202.

[68] N. T. Le, F. Bai, and A. Ritter, "Few-shot anaphora resolution in scientific protocols via mixtures of in-context experts," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2693–2706.

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[70] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, 2020.

[71] S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan, and L. Cronin, "A universal system for digitization and automatic execution of the chemical synthesis literature," *Science*, vol. 370, no. 6512, pp. 101–108, 2020. eprint: `https://science.sciencemag.org/content/370/6512/101.full.pdf`.

[72] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, relation, and event extraction with contextualized span representations," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5784–5789.

[73] P. L. Lee and B. N. Miles, "Autoprotocol driven robotic cloud lab enables systematic machine learning approaches to designing, optimizing, and discovering novel biological synthesis pathways," in *SIMB Annual Meeting 2018*, SIMB, 2018.

[74] P. Kingsbury and M. Palmer, "Propbank: The next level of treebank," 2003.

[75] I. Beltagy, K. Lo, and A. 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620.

[76] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning*, 2016, pp. 2702–2711.

[77] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017.

[78] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *International Conference on Machine Learning*, 2018.

[79] B. Dalvi, L. Huang, N. Tandon, W.-t. Yih, and P. Clark, "Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1595–1604.

[80] S. Mysore *et al.*, "The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures," in *Proceedings of the 13th Linguistic Annotation Workshop*, 2019, pp. 56–64.

[81] A. C. Vaucher, F. Zipoli, J. Geluykens, V. H. Nair, P. Schwaller, and T. Laino, "Automated extraction of chemical synthesis actions from experimental procedures," *Nature Communications*, vol. 11, no. 1, pp. 1–11, 2020.

[82] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[83] C. Kulkarni, W. Xu, A. Ritter, and R. Machiraju, "An annotated corpus for machine reading of instructions in wet lab protocols," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 97–106.

[84] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, 2019. arXiv: 1910.01108.

[85] X. Jiao *et al.*, "TinyBERT: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online: Association for Computational Linguistics, Nov. 2020, pp. 4163–4174.

[86] H. Vala, A. Piper, and D. Ruths, "The more antecedents, the merrier: Resolving multi-antecedent anaphors," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2287–2296.

[87] J. Yu, N. S. Moosavi, S. Paun, and M. Poesio, "Free the plural: Unrestricted split-antecedent anaphora resolution," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6113–6125.

[88] S. Paun, J. Yu, N. S. Moosavi, and M. Poesio, "Scoring coreference chains with split-antecedent anaphors," *arXiv preprint arXiv:2205.12323*, 2022.

[89] B. Webber and B. Baldwin, "Accommodating context change," in *30th Annual Meeting of the Association for Computational Linguistics*, 1992.

[90] B. Fang, C. Druckenbrodt, S. A. Akhondi, J. He, T. Baldwin, and K. Verspoor, "ChEMU-ref: A corpus for modeling anaphora resolution in the chemical domain," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online: Association for Computational Linguistics, Apr. 2021, pp. 1362–1375.

[91] K. Sanderson, "Automation: Chemistry shoots for the moon," *Nature*, 2019.

[92] A. C. Vaucher, P. Schwaller, J. Geluykens, V. H. Nair, A. Iuliano, and T. Laino, "Inferring experimental procedures from text-based representations of chemical reactions," *Nature communications*, 2021.

[93] A. J. Lawson, J. Swienty-Busch, T. Géoui, and D. Evans, "The making of reaxys—towards unobstructed access to relevant chemistry information," in *The Future of the History of Chemical Information*, ACS Publications, 2014.

[94] M. Yuan, P. Xia, C. May, B. Van Durme, and J. Boyd-Graber, "Adapting coreference resolution models through active learning," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[95] C. Kulkarni, W. Xu, A. Ritter, and R. Machiraju, "An annotated corpus for machine reading of instructions in wet lab protocols," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 97–106.

[96] W. Wu, F. Wang, A. Yuan, F. Wu, and J. Li, "CorefQA: Coreference resolution as query-based span prediction," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 6953–6963.

[97] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, 1991.

[98] J. Cho, M. Seo, and H. Hajishirzi, "Mixture content selection for diverse sequence generation," 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)*, 2019.

[99] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International Conference on Machine Learning*, 2020.

[100] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *ICML*, 2021.

[101] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 2339–2352.

[102] H. Lang, M. Agrawal, Y. Kim, and D. A. Sontag, "Co-training improves prompt-based learning for large language models," *CoRR*, 2022. arXiv: 2202.00828.

[103] F. Bai, A. Ritter, and W. Xu, "Pre-train or annotate? domain adaptation with a constrained budget," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5002–5015.

[104] S. Skonieczny, "The IUPAC rules for naming organic molecules," *Journal of chemical education*, 2006.

[105] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 3816–3830.

[106] B. Wang and A. Komatsuzaki, *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*, https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[107] S. Zhang *et al.*, *OPT: open pre-trained transformer language models*, 2022.

[108] L. Gao *et al.*, "The Pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[109] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Nov. 2019.

[110] L. L. Wang *et al.*, "CORD-19: The COVID-19 open research dataset," in *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, 2020.

[111] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," in *International Conference on Learning Representations*, 2020.

[112] F. Bai, A. Ritter, P. Madrid, D. Freitag, and J. Niekrasz, "SynKB: Semantic search for synthetic procedures," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Abu Dhabi, UAE: Association for Computational Linguistics, Dec. 2022, pp. 311–318.

[113] G. S. Becker, "A Theory of the Allocation of Time," *The economic journal*, no. 299, pp. 493–517, 1965.

[114] K. J. Lancaster, "A New Approach to Consumer Theory," *Journal of political economy*, no. 2, pp. 132–157, 1966.

[115] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *ArXiv*, 2019.

[116] J. Tabassum, W. Xu, and A. Ritter, "WNUT-2020 Task 1 Overview: Extracting Entities and Relations from Wet Lab Protocols," in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, Online, Nov. 2020, pp. 260–267.

[117] P. Dasigi, G. Burns, E. Hovy, and A. D. Waard, "Experiment Segmentation in Scientific Discourse as Clause-level Structured Prediction using Recurrent Neural Networks," *ArXiv*, 2017.

[118] D. M. Lowe, "Extraction of Chemical Structures and Reactions from the Literature," 2012.

[119] J. Marín *et al.*, "Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 187–203, 2021.

[120] S. Gururangan *et al.*, "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, Jul. 2020, pp. 8342–8360.

[121] D. Patterson *et al.*, "Carbon Emissions and Large Neural Network Training," *arXiv*, 2021.

[122] E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Jul. 2019, pp. 3645–3650.

[123] Z. Zhong and D. Chen, "A Frustratingly Easy Approach for Joint Entity and Relation Extraction," *ArXiv*, 2020.

[124] H. Daumé III, "Frustratingly Easy Domain Adaptation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, Jun. 2007, pp. 256–263.

[125] Y.-B. Kim, K. Stratos, and R. Sarikaya, "Frustratingly Easy Neural Domain Adaptation," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, 2016.

[126] J. Wang, Y. Ren, Z. Zhang, and Y. Zhang, "Melaxtech: A Report for CLEF 2020 - ChEMU Task of Chemical Reaction Extraction from Patent," in *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, L. Cappellato, C. Eickhoff, N. Ferro, and A. Névéol, Eds., ser. CEUR Workshop Proceedings, vol. 2696, CEUR-WS.org, 2020.

[127] A. Gupta and G. Durrett, "Effective use of transformer networks for entity tracking," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 759–769.

[128] J. Knafou, N. Naderi, J. Copara, D. Teodoro, and P. Ruch, "BiTeM at WNUT 2020 shared task-1: Named entity recognition over wet lab protocols using an ensemble of contextual language models," in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 305–313.

[129] M. G. Sohrab, A.-K. Duong Nguyen, M. Miwa, and H. Takamura, "Mgsohrab at WNUT 2020 Shared Task-1: Neural Exhaustive Approach for Entity and Relation Recognition Over Wet Lab Protocols," in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, Online, Nov. 2020, pp. 290–298.

[130] D. Q. Nguyen *et al.*, "ChEMU: Named Entity Recognition and Event Extraction of Chemical Reactions from Patents," *Advances in Information Retrieval*, pp. 572–579, 2020.

[131] C. Kiddon, L. Zettlemoyer, and Y. Choi, "Globally Coherent Text Generation with Neural Checklist Models," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, Nov. 2016, pp. 329–339.

[132] D. Lowe, "Chemical reactions from US patents (1976-2016)," 2017.

[133] R. Tamari, F. Bai, A. Ritter, and G. Stanovsky, "Process-level representation of scientific protocols with interactive annotation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online: Association for Computational Linguistics, Apr. 2021, pp. 2190–2202.

[134] M. A. Valenzuela-Escárcega, G. Hahn-Powell, and D. Bell, "Odinson: A fast rule-based information extraction framework," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, May 2020, pp. 2183–2191, ISBN: 979-10-95546-34-4.

[135] J. Blitzer, R. McDonald, and F. Pereira, "Domain Adaptation with Structural Correspondence Learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*, 2006, pp. 120–128.

[136] X. Han and J. Eisenstein, "Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling," 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)*, Hong Kong, China, Nov. 2019, pp. 4238–4248.

[137] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 538–549, 2008.

[138] R. Lebret, D. Grangier, and M. Auli, "Neural text generation from structured data with application to the biography domain," *arXiv preprint arXiv:1603.07771*, 2016.

[139] M. Iyyer, W.-t. Yih, and M.-W. Chang, "Search-based neural structured learning for sequential question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1821–1831.

[140] M. Kardas *et al.*, "AxCell: Automatic extraction of results from machine learning papers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 8580–8594.

[141] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023.

[142] M. Chen *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[143] B. Rozière *et al.*, *Code llama: Open foundation models for code*, 2023. arXiv: 2308.12950 [cs.CL].

[144] N. Chambers and D. Jurafsky, "Template-based information extraction without the templates," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 976–986.

[145] Y. Chen, W. Gantt, T. Chen, A. S. White, and B. Van Durme, "A unified view of evaluation metrics for structured prediction," *arXiv preprint arXiv:2310.13793*, 2023.

[146] K. Kim *et al.*, "Deep-learning-based inverse design model for intelligent discovery of organic molecules," *npj Computational Materials*, 2018.

[147] G. B. Fields, "The rebirth of matrix metalloproteinase inhibitors: Moving beyond the dogma," *Cells*, vol. 8, no. 9, p. 984, 2019.

[148] J. M. Stokes *et al.*, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, 688–702.e13, 2020.

[149] T. Gupta, M. Zaki, N. M. A. Krishnan, and Mausam, *Discomat: Distantly supervised composition extraction from tables in materials science articles*, 2022. arXiv: 2207.01079 [cs.CL].

[150] Q. Hao, R. Cai, Y. Pang, and L. 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*, ser. SI-

GIR '11, Beijing, China: Association for Computing Machinery, 2011, pp. 775–784, ISBN: 9781450307574.

[151] H. Touvron *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[152] R. Li *et al.*, "Starcoder: May the source be with you!" *arXiv preprint arXiv:2305.06161*, 2023.

[153] R. Taori *et al.*, *Stanford alpaca: An instruction-following llama model*, `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[154] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos, "TaPas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 4320–4333.

[155] B. Y. Lin, Y. Sheng, N. Vo, and S. Tata, "Freedom: A transferable neural architecture for structured information extraction on web documents," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 1092–1102, ISBN: 9781450379984.

[156] Y. Zhou, Y. Sheng, N. Vo, N. Edmonds, and S. Tata, "Learning transferable node representations for attribute extraction from web documents," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '22, Virtual Event, AZ, USA: Association for Computing Machinery, 2022, pp. 1479–1487, ISBN: 9781450391320.

[157] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8413–8426.

[158] J. Li, Y. Xu, L. Cui, and F. Wei, "MarkupLM: Pre-training of text and markup language for visually rich document understanding," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6078–6087.

[159] J. Kang, W. Xu, and A. Ritter, "Distill or annotate? cost-efficient fine-tuning of compact models," *Proceedings of ACL*, 2023.

[160] T. Berg-Kirkpatrick, D. Burkett, and D. Klein, "An empirical investigation of statistical significance in NLP," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language*

Learning, Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 995–1005.

[161] H. Laurençon *et al.*, *Obelics: An open web-scale filtered dataset of interleaved image-text documents*, 2023. arXiv: 2306.16527 [cs.IR].

[162] Y. Hou, C. Jochim, M. Gleize, F. Bonin, and D. Ganguly, "Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5203–5213.

[163] A. Parikh *et al.*, "Totto: A controlled table-to-text generation dataset," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1173–1186.

[164] F. Wang, Z. Xu, P. Szekely, and M. Chen, "Robust (controlled) table-to-text generation with structure-aware equivariance learning," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 5037–5048.

[165] H. Hu, Y. Liu, Z. Yu, and L. Perez-Beltrachini, "Improving user controlled table-to-text generation robustness," in *Findings of the Association for Computational Linguistics: EACL 2023*, Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 2317–2324.

[166] S. K. Jauhar, P. Turney, and E. Hovy, "Tables as semi-structured knowledge for question answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 474–483.

[167] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[168] T. Yu *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3911–3921.

[169] M. S. Schlichtkrull, V. Karpukhin, B. Oguz, M. Lewis, W.-t. Yih, and S. Riedel, "Joint verification and reranking for open fact checking over tables," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume*

1: Long Papers), Online: Association for Computational Linguistics, Aug. 2021, pp. 6787–6799.

[170] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1470–1480.

[171] W. Chen *et al.*, "Tabfact: A large-scale dataset for table-based fact verification," in *International Conference on Learning Representations*, 2020.

[172] H. Iida, D. Thai, V. Manjunatha, and M. Iyyer, "Tabbie: Pretrained representations of tabular data," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 3446–3456.

[173] A. Aghajanyan *et al.*, "Htlm: Hyper-text pre-training and prompting of language models," in *International Conference on Learning Representations*, 2022.

[174] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag, "Tabllm: Few-shot classification of tabular data with large language models," in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, F. Ruiz, J. Dy, and J.-W. van de Meent, Eds., ser. Proceedings of Machine Learning Research, vol. 206, PMLR, 25–27 Apr 2023, pp. 5549–5581.

[175] A. Carlson and C. Schafer, "Bootstrapping information extraction from semi-structured web pages," in *ECML/PKDD*, 2008, p. 16.

[176] C. Lockard, P. Shiralkar, and X. L. Dong, "Openceres: When open information extraction meets the semi-structured web," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3047–3056.

[177] X. L. Dong, H. Hajishirzi, C. Lockard, and P. Shiralkar, "Multi-modal information extraction from text, semi-structured, and tabular data on the web," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, Online: Association for Computational Linguistics, Jul. 2020, pp. 23–26.

[178] Y. Lou, B. Kuehl, E. Bransom, S. Feldman, A. Naik, and D. Downey, *S2abel: A dataset for entity linking from scientific tables*, 2023. arXiv: 2305.00366 [cs.CL].

[179] C. Lockard, P. Shiralkar, X. L. Dong, and H. Hajishirzi, "ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages," in *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, Online: Association for Computational Linguistics, Jul. 2020, pp. 8105–8117.

[180] L. Ouyang *et al.*, *Training language models to follow instructions with human feedback*, 2022. arXiv: 2203.02155 [cs.CL].

[181] X. Wei *et al.*, *Zero-shot information extraction via chatting with chatgpt*, 2023. arXiv: 2302.10205 [cs.CL].

[182] R. Han, T. Peng, C. Yang, B. Wang, L. Liu, and X. Wan, *Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors*, 2023. arXiv: 2305.14450 [cs.CL].

[183] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[184] M. E. Peters *et al.*, "Knowledge enhanced contextual word representations," 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)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 43–54.

[185] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020.