Information Extraction for Low-Resource Schemas

by

Justin Xu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2026

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis combines two manuscripts[89, 90] written for publication, but which have not yet been accepted by reviewers.

Chapters 1 and 5 These chapters combine the Introduction and Conclusion sections of both works[89, 90]. Professor Pascal Poupart provided vital guidance when writing these sections in the original manuscripts.

Chapter 2 This chapter combines the Background section of [89] and the Related Works section of [90] into a single coherent discussion of previous works. The Related Works section of [89] includes works found by co-author Prashanth Arun during a literature review, which are incorporated here.

Chapter 3 This research was conducted at the University of Waterloo and the Vector Institute by Justin Xu and Prashanth Arun, under the supervision of Prof. Pascal Poupart, supported by funding from the Natural Sciences and Engineering Research Council of Canada (NSERC). Justin Xu is the primary author of this work, which originates from ideas initially developed when he was a contract research assistant at the University of Waterloo with Prof. Poupart from June to August 2023, before the official start of his Masters program. The majority of the work was conducted during Justin's time as a Masters student, including all experiments. Besides providing support with the ideation and literature review, Prashanth Arun developed early prototypes of the extension of the system to large language models. Prashanth Arun also developed some visualizations and diagrams for the manuscript, of which revised versions are included in this thesis.

Chapter 4 This research was conducted at the University of Waterloo and the Vector Institute by Justin Xu under the supervision of Prof. Pascal Poupart and Jason Sun, supported by funding from AdaNomad via the Mitacs Accelerate program. Justin Xu performed all of the data collection, coding, and experiments. A small user study was conducted during the course of this work, which has ultimately been excluded from the manuscript and this thesis. However, the authors would like to thank William Loh, Hossein Mohebbi, Prashanth Arun, and Mohammed Abdulrahman for the hours they have dedicated as participants of that user study.

As the lead author of all chapters, Justin Xu was responsible for most of the research ideation and direction, coding, experiment design, analysis, and drafting and submitting manuscripts.

Abstract

Information Extraction (IE) is a set of important tasks in the study of creating structured data such as knowledge graphs from unstructured data such as text. The past paradigm of IE focused on models with specialized neural network architectures, usually based on transformer encoders. These models typically focus on a single subtask of IE, following a single schema of entity and relation types, and are trained via supervised learning on large datasets of annotated texts. Meanwhile, the current paradigm of IE, called Universal IE (UIE), involves large language models which can generalize across IE subtasks and to completely unseen schemas, but which lack other abilities such as entity grounding and calibration.

We first discuss structural consistency, a new measure of robustness in information extraction based on compositionality. We present structural consistency post-training (SCPT) as a data augmentation method to boost structural consistency for a wide range of model architectures. Besides greatly improving robustness, SCPT significantly reduces the amount of labelled data needed to achieve the same level of performance when training specialized IE models.

Second, we use reasoning-based data augmentation techniques to gather AdaIE, a very large collection of human-annotated information extraction schemas. We diverge from UIE and align the dataset with a new task we call Guided Information Extraction (GIE). GIE emphasizes the tight grounding and schema-following requirements which have been largely neglected in UIE. Evaluations of state-of-the-art UIE models reveal that state of the art UIE methods can be surpassed by recent commercial large language models (LLMs). Although those LLMs achieve below human performance on AdaIE, they are rapidly advancing.

Overall, we hope that both works presented will steer the IE research community towards unifying the strengths of the old and new IE paradigms, while casting light on their weaknesses.

Acknowledgements

First, I would like to thank my supervisor, Professor Pascal Poupart. We first met at his excellent Introduction to Machine Learning lectures when I was finishing my undergraduate studies, with only a slight inkling that I wanted to do a Masters degree. It was Pascal's belief in me and my research ideas that led me to this program, and his valuable advice that shaped those ideas into something that actually works.

I would also like to thank my committee members, Professor Freda Shi and Professor Yuntian Deng, for taking the time to read and provide insights regarding this thesis.

The work in this thesis has been funded by the Vector Institute, the Mitacs Accelerate program in collaboration with AdaNomad, the NSERC Alexander Graham Bell Canada Graduate Scholarship, and of course the University of Waterloo. In particular, Jason Sun from AdaNomad has been a great professional mentor and friend.

Many friends – Prashanth, Hossein, Michelle, Rikin, Clara, Kevin, Scott, and Adam, just to name a few – have supported me during my journey, whether by providing emotional and moral support, discussing research, or being a timely distraction. But most of all, I owe this thesis to my parents and grandparents, who have patiently and insistently taken good care of me for my whole life, especially the last three years. In any given week I bounce between up to four different cities, but no matter where I end up it is thanks to them that I find the comforts of home even on the hardest of days.

Dedication

This work is dedicated to my parents and grandparents.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Knowledge Graphs

A knowledge graph (KG) is an arrangement of data which uses some kind of graph to store information about the world; each vertex (or entity) represents a concept (an individual human, species, drug, academic paper, sport, etc.), and each edge (or relation) represents a relationship between concepts [28]. Beyond this, there is no widely accepted formal definition of KGs [28], though modern conceptions trace back to the model commonly attributed to Google in 2012 [70] (which also does not provide any formal definition).

One particular application of knowledge graphs, and the original impetus of the author in pursuing this line of research, is to model personal memories and facts: financial records, biographical details about friends, life events, hobbies, and so on. Such a personal knowledge graph (PKG) faces a number of challenges, including the question of *How can PKGs be automatically populated and reliably maintained?* [8] Any such system for automatic PKG population ought to be held to high standards beyond just accuracy.

1. **Low-Resource**: The construction or configuration of the system must be feasible for a single person to do in a small amount of time.

2. **Schema-Following**: The user can supply an arbitrarily specific set of entity and relation types that the system must follow.

3. **Grounding**: Entities and relations are linked back to the source data they are extracted from.

4. **Calibration**: Confidence scores accurately indicate the model's confidence in its extractions, allowing the user to prioritize validating low-confidence extractions.

5. **Multi-Modality**: Integrate all sorts of everyday data including emails, voice conversations, chat messages, and images.

1.2 Information Extraction

Information extraction (IE) is a task within natural language processing (NLP) that addresses the creation of knowledge graphs. Its primary subtasks are named entity recognition (NER), relation extraction (RE), and event extraction (EE) [88] – all different types of structures that can be found in a knowledge graph. In the past, IE has been performed by specialized models capable of grounding and calibration, but which were trained on large amounts of labelled data fixed to a single schema, putting them out of the reach of PKG construction. Meanwhile, the zero-shot and few-shot capabilities of emerging large language models (LLMs) led to a paradigm shift towards generative methods for IE, under a subtask called Universal IE (UIE) where low-resource schema-following is now somewhat possible [88]. However, we find that leading LLM-based techniques neglect grounding and calibration, meaning that neither paradigm in its current state adequately addresses all of the above desiderata for PKGs.

1.3 Contributions

This thesis contributes two works that aim to address gaps in current research.

The first work focuses on a measure of consistency for IE models with respect to structural data transformations such as concatenation and entity replacement. We demonstrate how to train for this consistency on a variety of specialized IE model architectures, in a self-supervised manner that does not require any additional labelled data. For ease of integration, we frame this as a post-training technique, using multi-objective optimization and hyperparameter search to prevent the model from losing its performance at the original IE task. In low-data regimes, we are also able to produce performance gains at the original IE tasks.

The second work rigorously defines the task of guided information extraction (GIE) with explicit use of schema guidelines, noting that existing tasks often end up being underdetermined even when they claim to incorporate annotation guidelines. It then presents

a dataset collected explicitly for GIE, extended from gold-standard annotated data using a reasoning-based data augmentation technique. This dataset represents the first known standardized attempt to evaluate the performance of existing UIE methods at extracting information following low-resource schemas. We evaluate some compatible leading generative IE methods and find that they are dominated by commercial LLMs, but all models still perform far worse than the inter-annotator agreement of humans on the same source data.

Chapter 2

Background

2.1 Knowledge Graphs

2.1.1 Entities

An entity is any concept of interest in a knowledge graph; what counts or does not count as an entity depends on the purpose of the graph. Below, we discuss entities from the perspective of information extraction, using published datasets as examples.

Entity Types

A knowledge graph schema defines the set of possible entity types, like person (often abbreviated `PER`) that may be included in the graph. Generally, entities in IE datasets are labelled with a single type, but one might immediately see an issue with this practice. For example, a province such as British Columbia is both a geopolitical entity (`GPE`) as well as a location (`LOC`), two common types of entities found in many datasets like WikiEvents. Under a single-label approach, we must forego the `LOC` type to annotate the more specific `GPE` type. To somewhat overcome this, a knowledge graph may define a taxonomy in which one entity type is explicitly the subtype of another parent entity type, automatically implying that any instance of the child is an instance of the parent without needing the label. An alternative solution, which we argue is better, is to treat entity typing as a multi-label classification. Fine-grained entity types, like those present in DWIE, treat entity types as independent predicates, with multiple types such as `type::person`,

`type::education_student`, `gender::male`, `type::researcher` applying to a single entity (the author of this thesis) [96].

Entity Mentions

An entity mention is a specific occurrence of an entity in a supporting document. While not necessary for a functioning knowledge graph, an annotated entity mention gives users a link between an entity and a document about that entity. In text documents, a mention is almost always represented as a single span – a contiguous sequence of characters or tokens in the text. Technically, it would be more precise for a single mention to be represented by a set of tokens which is not necessarily contiguous, such as "reinforcement [...] learning" in the text "we compare reinforcement and imitation learning." The brat rapid annotation tool (brat)[1] [73], frequently used for data annotation on information extraction tasks, includes functionality for annotating disconnected spans forming parts of the same single mention. However, disconnected spans are uncommon in practice; in this work, we encounter just one dataset, DEFT [72], which annotates disconnected spans for definitions.

While a single entity mention is usually enough to invoke a particular entity in the mind of a human reader, *the mention is not the entity*. It is generally not valid to assume that every identical string refers to the same entity, nor is it valid to assume that an entity has only one string representation. Entities called "Turkey", "information extraction", and "Obama" in one sentence may be referred to as "Türkiye", "it", and "Michelle" in the next sentence. Even when an entity (like Canada) may seem to have a single canonical representation, that string can appear in contexts where it does not meaningfully refer to that entity (like in "Canada goose").

Properties

Property graphs extend knowledge graphs by annotating additional key-value properties on entities. They have become quite popular compared to conventional entity-relation knowledge graphs, as modern information extraction methods are able to output them readily. In this thesis we ignore properties, noting that a property p and its value v on an entity e is equivalent to a relation (see Subsection 2.1.2) of type p between e and v, so long as v is treated as an entity. Indeed, many older information extraction datasets such as WebNLG do exactly this [12]. Inside them, one might be surprised to find numbers to be

[1] http://brat.nlplab.org

considered valid entities, for the purpose of creating relations such as (Al-Shifa Hospital, number of beds, 700).

2.1.2 Relations

Relational Triples

The simplest interpretation of a relation is a relational triple, a structured triple (h, r, t) (or sometimes ordered differently like (h, t, r)) where h is the head entity, t is the tail entity, and r is a relation type. The triple augments the directed edge $h \rightarrow t$ with an additional label for the relation type. h is often also called the subject entity, and t is often also called the object entity [39], though in this thesis we prefer to call them head and tail. "Subject" and "object" imply that each entity plays a certain grammatical role in the relationship that does not actually hold up in practice; for example, the relation (Google, parent company, Alphabet) can be naturally stated as "Alphabet is the parent company of Google" in which the head ends up being the grammatical object and the tail ends up being the grammatical subject. Even the head-tail terminology is problematic when dealing with symmetric relations, as it becomes impossible to choose one entity over the other to be the head as they can be interchanged.

Relation Types

There is an IE subtask called open information extraction (OIE) that addresses the construction of KGs with arbitrary relationships between entities [19]. This thesis focuses on IE subtasks where the relation types are a predefined set determined by the schema of the knowledge graph. Datasets studied in this work have anywhere from a single relation type to the 216 relation types found in WebNLG [12]. Almost all relation types of practical interest have restrictions on the entity types that the head and tail entities must follow. Even if not stated in any explicit schema, we find in our own analysis of datasets that most relation types predominantly end up being between at most a few head entity types, and another few tail entity types.

Schemas can also mix directed and symmetric relation types. In a symmetric relation type, the triples (h, r, t) and (t, r, h) are identical. This is a challenge during evaluation, as exact matching might miss (t, r, h) as a true positive for the label (h, r, t). Some datasets like DWIE enforce symmetry through inference rules to ensure that each relation implies its inverse [96].

Higher-Order Relations

Hyper-relations extend a relational triple by attaching additional qualifiers to the triple as a whole, such as the time, location, or quantity [13]. They provide an increase in expressivity; for example, an employment relation between a person and a company has a job title, but there is no way to decompose that into a set of purely binary relations [13]. Besides qualifiers, some relation types are simply inherently n-ary. The SciREX dataset features the 4-ary relation of (Dataset, Metric, Task, Method) to represent scientific benchmark evaluations [32].

2.1.3 Events

Similar to other terms within knowledge graphs, events are not well defined, but it is vaguely understood to mean "a specific occurrence involving participants" [47]. Events are usually distinguished from relations, which do not really occur at a specific point in time. In practice, however, we note that the distinction is blurred in a number of ways. First, any event type can be expressed as a relation type "something occurred between `{x}` and `{y}`", and we have already seen that qualifiers can add temporal or location context to a relation. Second, events of sufficient notoriety can also be considered named entities in datasets such as CrossRE, allowing the event to be related to entities via binary relations like `part-of` [10]. Third, even without occurring at a specific point in time, many relations are still dynamic, and can instead be modelled as an event indicating the moment at which the relation became true (or false). For example, the relation (Mark Carney, leader of, Canada) may just as well be modelled by the event in which his party successfully formed government.

Event Triggers

The main remaining distinction between events and (higher-order) relations is an event trigger, a span of text analogous to an entity mention. Event triggers locate an event inside of a supporting text document, and are usually verbs like "threw" or "killed" [47].

Event Types and Arguments

As with entities and relations, events follow a schema. However, we find event schemas in practice to be comparatively much more detailed than entity or relation schemas. Datasets

such as ACE and WikiEvents feature entire tree-shaped ontologies of different categories of events like `Life` and `Conflict` under which finer-grained events like `Marry` and `Divorce` are defined [47].

For each event type, a set of roles are defined that participating entities may fill as arguments. In datasets like WikiEvents, each role can take arbitrary numbers of entities (or none at all) [43]. We see this as having two strengths. First, this ensures that events can be extracted even under partial information (e.g., an `Attack` occurred but the `Attacker` is not known). Second, this reflects real-life in which many entities can play the same role in a single event, (e.g., an `Attack` occurred with multiple `Attacker`s). Because of this, events are analogous to higher-order relations, but with fluid schemas that allow different instances of the same event type to have wildly different numbers of participating entities.

2.2 Information Extraction

Information extraction (IE) is a family of tasks relating to the construction of knowledge graphs. We describe the tasks relevant to this thesis, though this is in no way exhaustive. A common theme throughout IE is that the same term is often used to mean slightly different things; this applies even to the task definitions in IE. It is generally safest to analyze tasks through the datasets that pose them, since downstream methods usually adhere to those same definitions when evaluating on those datasets.

2.2.1 Named Entity Recognition

Named entity recognition (NER) is the task of identifying entity spans in a text, and labelling the entity type represented by each span. It is represented by datasets such as CoNLL-2003 [76]. Formulated as a token tagging problem, NER in CoNLL-2003 was restricted to only a single entity per span of text; in the case of overlap, only the outermost entity is taken [76]. For better or worse, that convention persists to this day, such as in brat [73] and all datasets annotated using brat.

Many datasets share various definitions of entity types corresponding to proper nouns like "person", "organization", and "geopolitical entity". Despite being called *named* entity recognition, entity types can also include common nouns like "decision tree" or "hydrogen", as is common in domain-specific datasets [48]. The same annotation format used for named entities can also be used for things that would usually not be considered entities at all, such as numbers or dates.

2.2.2 Coreference Resolution

Coreference resolution is the task of merging entity mentions (like pronouns and abbreviations) in a document into a single entity. For short sentences, coreference resolution is sometimes treated as optional, as it is likely that an entity only appears a few times at most. However, on longer documents, coreference resolution becomes essential. Coreferences are annotated explicitly in some datasets like DWIE [96], DocRED [93], or SciERC [52] which provide distinctions between entities and entity mentions. Alternatively, other datasets like CrossRE [10], SoMeSci [68], and WebNLG [12] instead provide relation types such as `same-as`, `alternate-name`, and `abbreviation` which can be understood as equivalence relations that link spans of the same entity.

Constrained Entity-Alignment F-Measure

Coreference resolution presents a challenge for IE evaluation metrics, as treating an entity as a set of arbitrary spans rather than a single span makes it possible and likely for an extraction to only partially match the ground truth. Exact matching would be too punishing on near misses, so this leads to soft metrics such as Constrained Entity-Alignment F-Measure (CEAF) [54] for comparing sets of sets. Let $\mathcal{G}$ denote the set of ground-truth extractions, and $\mathcal{P}$ denote the set of predicted extractions. We provide a score function $\phi(G, P)$ for any ground-truth extraction G and predicted extraction P. The higher the score, the better the match; therefore $\phi(x, x)$ (matching an extraction to itself) should be at least as large as $\phi(x, y)$ for any $y \neq x$.

An alignment M is a set of pairs between $\mathcal{G}$ and $\mathcal{P}$, such that each extraction is part of at most one pair. Define

$$\Phi(M) = \sum_{(G,P)\in M} \phi(G, P) \tag{2.1}$$

to be the total score obtained by M, and

$$M^* = \operatorname{argmax}_M \Phi(M) \tag{2.2}$$

is the alignment attaining the maximum score $\Phi(M^*)$. Determining M^* is an instance of the well-known maximum bipartite matching problem, and can be solved using the Kuhn-Munkres ("Hungarian") algorithm [37], which is implemented in libraries such as SciPy [77].

$\Phi(M^*)$ is bounded below by $\Phi(\emptyset) = 0$, and should be bounded above by both

$$\Phi(\mathcal{G}) = \sum_{G \in \mathcal{G}} \phi(G, G) \tag{2.3}$$

$$\Phi(\mathcal{P}) = \sum_{P \in \mathcal{P}} \phi(P, P) \tag{2.4}$$

which correspond to matching $\mathcal{G}$ and $\mathcal{P}$ with themselves, respectively.

We therefore compute recall r, precision p, and F1-score F via

$$r = \frac{\Phi(M^*)}{\Phi(\mathcal{G})} \quad p = \frac{\Phi(M^*)}{\Phi(\mathcal{P})} \quad F = \frac{2rp}{r+p} \tag{2.5}$$

2.2.3 Relation Classification

Given a head entity and a tail entity, relation classification (RC) is the task of finding the relation type that best describes the relationship from the head to the tail. This is equivalent to extracting a single relational triple, where the head and tail entities are already known. Over a single fixed schema, RC is almost never studied today. Rather, research focuses on generalization to lower-resource schemas, such as the few-shot relation classification task popularized by the FewRel dataset [26]. RC can also still be found in research on multi-task IE, like InstructUIE which calls the task "entity pair relationship identification" (EPR) [78].

An unspoken rule in many relation classification datasets is that the relation types are mutually exclusive, such that any given pair of entities may only have one relation between them. This interpretation allows one to reduce the task to single-label classification per entity pair, where the possible classes include each relation type, or no relation at all. However, this restriction leads to unexpected nonsense such as an employee being an `employee-of` a company but not a `member-of` or `part-of` it, prioritizing the most specific or descriptive type rather than evaluating the truthfulness of each type.

2.2.4 Relation Extraction

Classically, given a set of already-labelled entities in a text, relation extraction (RE) is the task of determining all pairs of related entities and identifying the relation type of each pair. A standard convention of RE is the "reasonable reader rule" from ACE, wherein a

relation should only be annotated when there is no reasonable interpretation of the text in which it does not hold [55].

Relation extraction is often confused with joint entity-relation extraction (JERE) in which entities are *not* provided. JERE is roughly equivalent to extracting entities, followed by relations between those extracted entities; however, unlike a pipelined approach of NER followed by RE, there is often no need to extract entity types. Many generative IE techniques such as InstructUIE [78] or GoLLIE [66] perform JERE by going straight to relational triples, where strings inside the generated relations stand in for the participating entities, despite the issues with this assumption described in Section 2.1.1. Thus, when "relation extraction" is said today, the term typically refers to JERE rather than true RE.

Across both tasks, the typical evaluation metrics used to measure similarity between sets of relations are based on the confusion matrix. On a text x with ground-truth extractions y and predictions $\hat{y}$, the F_1 score, also known as the Dice-Sørensen coefficient (DSC), is defined as

$$F_1 = \frac{2\,|y \cap \hat{y}|}{|y| + |\hat{y}|} \tag{2.6}$$

2.2.5 Event Extraction

A common breakdown of event extraction from text involves performing event detection (ED) followed by event argument extraction (EAE) [47]. ED is similar to NER, and involves locating and classifying triggers for each event in the text. Next, EAE focuses on a single event (identified by its trigger and type), and involves locating participating entities in the text and then annotating their roles in the event.

Modern generative approaches to event extraction such as KnowCoder [45] do not necessarily break things down this way, as they can perform both steps at once by generating a structure that contains both the trigger and arguments.

2.2.6 Universal Information Extraction

Universal information extraction (UIE) is a vague task encompassing most major information extraction tasks – entity extraction, relation extraction, and event extraction [88]. Notably, coreference resolution is generally forgotten in UIE; it is not acknowledged in any of the prior UIE datasets in Table 2.1 other than AdaIE (which is our dataset). UIE models can be conditioned at inference time on a potentially unseen schema of entity, relation, or event types describing what needs to be extracted. The degree of conditioning is

not formalized in UIE; some techniques [40, 51, 78] use the names of the types, but others [98, 79, 66, 60, 23] use full guidelines for each type.

Large language models (LLMs) form the backbone for most UIE methods. Methods can be differentiated by extraction formats, use of in-context examples, and use of further training [88]. Most methods, including the eponymous UIE [51], use some form of structured text output, however object-oriented (e.g., Python) code has emerged as an interesting extraction format, used in models such as CODE4STRUCT [79], Code4UIE [23], GoLLIE [66], and KnowCoder [45]. In such models, each entity, relation, or event type is expressed as a class definition or function, and extractions are expressed as calls to the class constructor or function. One landmark finding in UIE was that coding models doing extraction via code generation, can outperform models that operate in natural language [42], suggesting that IE and coding may be complementary tasks. Ironically, one strength of code-based formats is their natural ability to include lengthy free-form text descriptions, via documentation comments inside each class definition. Thus, we see strong overlap between coding-based UIE models and models which fully leverage guidelines (Section 2.2.7).

Composite Datasets for UIE

Due to its universal nature, training and evaluation in universal IE tends to involve large numbers of IE datasets. Generally, the evaluation practice in UIE involves training (if at all) on some datasets, then evaluating on held-out datasets to demonstrate generalization. This has led to some attempts to publish composite datasets, where multiple sources have been normalized into a single format. IE INSTRUCTIONS [78], IEPILE [22], and `IEInstruct` [60] are recent instruction tuning datasets constructed by transforming dozens of datasets for various IE sub-tasks into LLM prompt-response pairs. Similarly, B^{2}NERD [92] is a recent dataset specifically constructed for entity extraction, with greater emphasis on data quality, such as by normalizing type names. Most of these datasets only use the names of the types. We summarize these datasets and their sizes in Table 2.1.

2.2.7 Guidelines

[31] demonstrate empirically that IE datasets contain a task definition bias, causing the same type name to have different interpretations in different datasets. Indeed, most human-annotated datasets provide annotation guidelines to thoroughly interpret the types, their interactions, special cases, and exceptions. The annotation guidelines can be comparable in length to the papers describing the datasets themselves (see [56] versus [62]). Beyond

Dataset	Task	$\lvert D_E \rvert$	$\lvert D_R \rvert$	$\lvert S_E \rvert$	$\lvert S_R \rvert$	Guidelines
IE INSTRUCTIONS [78]	UIE	34	12	204	126	no
from GoLLIE [66]	UIE	20	5	227	208	yes
from [7]	RE	3	3	–	423	used but not available
IEPile [22]	UIE	46	25	209	746	no
`IEInstruct` [60]	UIE	19	10	100	508	on ∼20% of the data
B^{2}NERD [92]	NER	50	–	486	–	no
AdaIE (ours)	GIE	15	15	497	747	yes

Table 2.1: A comparison of the numbers of constituent datasets supporting entity extraction ($|D_E|$) and relation extraction ($|D_R|$), entity types ($|S_E|$), and relation & event types ($|S_R|$) across various composite datasets related to UIE. Counts are aggregated across all splits. To make a fair comparison we only count CrossNER as a single dataset. Values for `IEInstruct` are estimated based on de-duplicated upper bounds from each of the constituent datasets used, excluding Open IE and On-Demand IE.

this, datasets with multiple human annotators can require multiple rounds of annotation to reconcile different interpretations of the same schema [24]. Thus, a schema described by single words and short phrases should not be sufficient for a model, let alone a human, to follow accurately.

The authors of B^{2}NERD identify inconsistent and vague type definitions as a problem when combining multiple datasets [92]. Besides inconsistent definitions of types with the same name, many datasets are annotated in a way such that entity or relation types are *mutually exclusive*, e.g., a `country` may not be labelled as a `location`. Other inconsistencies identified include whether to extract pronouns, and the exact boundaries on spans. [92] respond to this by adjusting each dataset's entity type names to include these differences, e.g., `location` could become `location (without country)`. However, in reviewing their relabelled types, we still find it insufficient for an annotator to accurately reproduce the ground truth annotations, therefore motivating even more detailed schemas. A proper description of a type needs at least a sentence.

Communicating Guidelines A number of datasets and models aim to provide and leverage complete natural language type descriptions. Event extraction typically sees single-sentence templates for each event type. The template includes a placeholder for each argument of the event. The WikiEvents dataset [43] provides templates with placeholders of the form `<arg`*i*`>`, while DEGREE [29] uses placeholders starting with "some"

Dataset	Name	Verbalization
CrossRE (music)	`organisation`	`{organisation}` is a name (abbreviated or full) of an organisation, excluding musical bands.
DialogRE	`organization`	`{organization}` is the name of a corporation, agency, or other group with an established structure.
WikiEvents	`organization`	`{organization}` refers to (by name or pronoun), a corporation, agency, or other group of people defined by an established organizational structure, which can change members without changing identity.

Table 2.2: Examples of the "same" organisation/organization entity type from various IE datasets, described using verbalizations. Verbalizations vary due to different levels of detail in annotation guidelines, inclusion or exclusion of pronouns, and exclusions of other entity types (e.g., musical bands).

(e.g., "someone", "some place"). Regardless of the format, a template can be used by a generative language model such as BART-Gen (from WikiEvents) or DEGREE, as input alongside the text. Extraction is performed by regenerating the template, but with the placeholders replaced by tokens from the source text.

Meanwhile, in relation extraction, [67] pioneered the use of templates with placeholders (e.g., `{obj}` is president of `{subj}`), the format we use in this work. This format continued to be used in works such as QA4RE [98]. By filling in entity names in placeholder locations, relation extraction can be reframed as an entailment [67] or summarization [50] task, by asking whether the resulting statement accurately follows from the text. Other works, such as GoLLIE [66] do not always use explicit placeholders, and use the guidelines as direct input to a model.

Relation or event types define their arguments as fields of the class – this naturally allows higher-order relations. Based on the text and class definitions, a coding model constructs instances of the classes, which are interpreted as extractions.

Dataset	**Argument Types**		
Relation Name	Argument	Entity Types	**Verbalization**
HyperRED	`person`	`per`	`{person}` works or worked for `{employer}` since `{start}` and until `{end}`, in the position of `{position}`.
`employer`	`employer`	`org`, `per`	
	`start`	`time`	
	`end`	`time`	
	`position`	`string`	
WikiEvents	`employees`	`per`	`{employees}` started working in `{position}` position at `{employer}` organization, and `{place}` was the most specific location of their work.
`start_position`	`position`	`title`	
	`employer`	`GPE`, `org`	
	`place`	`GPE`, `loc`	

Table 2.3: Example employment relation types from various IE datasets.

Deriving New Types

There is limited prior exploration into the use of reasoning as a data augmentation in information extraction. LogiRE aims to automatically learn logical reasoning rules for relation extraction, but remains confined within the existing relation types of a dataset [65]. ReLA explores generating alternative relation names for an existing dataset during training, demonstrating that training with diverse types improves overall extraction performance [40]. Likewise, for guidelines, GoLLIE uses a language model (Vicuna 33B v1.3) to generate many alternate paraphrases for the guidelines of each type, though this does not actually change the meaning of the type [66]. However, no prior works investigate creating types which are logically different from the original types.

2.3 Data Augmentation for Text

Data augmentations extend a limited amount of labelled data to span a much larger space, generating new units by transforming existing units. Data augmentations in NLP domains include

- back-translation [69]
- positional encoding shifts [35]

- replacing a word with a random word or a suitable synonym [83]
- inserting or deleting a random word [83]
- negating the text by the addition of "not" [27]
- introducing character-level typos [20]
- paraphrasing using generative models [14]
- generating similar texts using generative models [30, 94]

For an augmentation to be useful, it must have a predictable effect on the corresponding labels with high probability. This depends on the NLP task. For example, negation changes textual entailment labels in a way that can be logically inferred from the original label [27], but its effect would be far less reliable in question answering. The safest and simplest transformations are intended to be label-preserving, in that they (with high probability) do not change the label of a text. However, the high-dimensional labels in IE are hard to preserve, since they depend on precise span annotations to locate each entity. Data augmentation in IE tends to focuses on entity replacement, where the spans of one entity are replaced with the spans of another, such as in [59], but the same relations are expected to hold after replacement.

2.4 Consistency Training

[86] propose unsupervised consistency training under *label-preserving* data augmentations. That is, they consider augmentations of the form $\hat{x} = q(x, \epsilon)$, where x is the original input, ϵ is a source of random noise, and the ground-truth labels of $\hat{x}$ and x must be the same. They evaluate this method on a simple text classification task, where data augmentations such as back-translation and random token replacement easily satisfy the label-preserving condition.

Besides transformations to the data, one can also modulate the output by modifying the model itself, as [41] do by resampling the dropout nodes multiple times for the same input.

Prior works establish a precedent [57, 86] to detach the gradients from the model outputs on the original (unaugmented) inputs from the dataset. Empirically, the stop-gradient not only reduces the tendency to collapse toward consistent but incorrect solutions (e.g.

classifying everything under the same class) during consistency training, but it also leads to better-performing models even in comparisons where collapse does not occur [85].

Chapter 3

Structural Consistency for Information Extraction

While text is unstructured, grammatical meaning still follows many compositional rules. In this chapter, we present self-consistency as a framework for measuring an IE model's ability to follow the rules of compositionality. A model is self-consistent when its outputs are equivariant to transformations that change the meaning of the text. By analyzing how a model's output changes on texts which have been joined together (concatenation), or after entities have been replaced (substitution), we identify two types of errors (cross-association and memorization) indicative of a model's failure to understand context. Moreover, we devise metrics to quantify the model's self-consistency with respect to those two transformations. They are important to measure, as an IE model cannot be correct unless it is self-consistent.

It is also possible to compute a differentiable consistency loss, enabling self-supervised training for structural consistency on potentially unlabelled data. In theory, this pulls the model toward parameterizations that are more consistent, thus increasing the likelihood of successfully reaching weights that are also optimal for correctness. However, in practice, there is a trade-off; models degrade in performance on the original task if trained solely for consistency. Instead, as shown in Figure 3.1, we propose structural consistency post-training (SCPT), which uses multi-objective optimization as a means to increase a model's consistency while continuing to train the model on its original training objective. The effect is a significant increase in both concatenation and substitution consistency, caused by a decrease in cross-association and memorization errors. This comes with little to no loss in performance at the original IE task. Rather, when only a small part of the data is labelled, SCPT leverages the unlabelled portion to significantly increase IE performance.

3.1 Transformations

Consider a supervised learning task with an input space X and output space Y. For relational triple extraction, we call X "text-space" and Y "triple-space". A **transformation** T is a pair of actions (T_X, T_Y), where T_X applies to the input space, and T_Y applies to the output space.

3.1.1 Concatenation

In text-space, the concatenation operation joins together a pair of *semantically unrelated* texts, making appropriate whitespace, punctuation, or capitalization adjustments if needed. In triple-space, the corresponding operation takes two sets of relational triples, and outputs their set-union.

$$T_X(x_1, x_2) = x_1 x_2 \tag{3.1}$$

$$T_Y(y_1, y_2) = y_1 \cup y_2 \tag{3.2}$$

Higher-order concatenations are possible, but we limit our study to binary concatenation.

It is important that the sentences are *semantically unrelated*, meaning that the second sentence does not in any way make reference to anything from the first text, including via pronouns. For the sentence-level datasets used in this work, a random sample of two sentences from the dataset is almost always semantically unrelated.

A dataset of size N admits N^2 unique pairs of concatenated texts. It is not practical to iterate over all possible pairs. For evaluation, we do a single pass over the entire shuffled dataset, using fixed seeding to ensure a consistent ordering, and concatenate each adjacent pair of samples. During training, we similarly take each adjacent pair of inputs within a single batch. Reshuffling the training dataset before each epoch ensures that new pairs are seen each epoch.

Cross-Association Errors

We observe a failure mode in some IE models, where entities which occur in unrelated clauses of the same text are incorrectly extracted as being related. In the text *"Ottawa is the capital of Canada, and Paris is the capital of France,"* it is obvious that

(Canada, capital, Ottawa) and (France, capital, Paris) are the correct extractions, whereas (Canada, capital, Paris) and (France, capital, Ottawa) would be false positives. We call this type of false positive a **cross-association error**. This is distinct from the misalignment errors addressed by [100] and [15].

Within a single unaugmented text, the only way to identify likely cross-associations is by parsing the text, leading to a chicken-and-egg problem of needing a good NLP module to diagnose NLP errors. However, when concatenation joins two unrelated texts, it is never correct to extract a triple that has one entity from each text, and we can confidently identify any prediction crossing a concatenation boundary as a cross-association error.

3.1.2 Substitution

In text-space, the substitution (also known as entity replacement) operation takes a single text as input, and replaces some of the entities in that text with *counterfactual* entities randomly sampled from an external collection of entities. In triple-space, substitution makes the same entity replacements to the head and tail entities inside the relational triples. In this work, we always use entities from other texts in the same dataset. For this transformation to result in reasonable sentences, each entity must be replaced by an entity of the same type. If the dataset lacks entity type annotations, a naïve method to achieve this is to use a named entity recognition (NER) module, such as spaCy, to label the entity types of the entire dataset, and then replace entities with any entity of the same type.

However, on many datasets it is possible to do better by inferring entity types from the relational triples themselves. For example, if an entity e is the head in a relation of type `lives-in` and is the tail in a relation of type `CEO-of`, then it can be replaced by any other entity that also participates in those same two relation types. This is more likely to retrieve similar entities, (e.g., Henry Ford $\leftrightarrow$ Steve Jobs), as opposed to using coarse-grained types (e.g. `PERSON`). The more similar the entity is, the more realistic the resulting text will be.

Memorization Errors

Especially from models trained on distantly-aligned data, we observe a tendency for the model to ignore contextual evidence (or lack thereof) in the text, in favour of regurgitating what the model has *memorized* about the real world. This observation has been validated in studies like [59]. Previous works address this issue by generating entity-preserving counterfactual examples [80, 99] and introducing entity-substitution methods for evaluation [81]. While regurgitating memorized relational triples can allow models to achieve high

recall on benchmarks based on public information, the model's true lower performance is revealed when extracting information about unseen entities. False positives can occur when a model outputs a relation between two entities which happens to be true in the real world, but is not supported by the text. False negatives may occur when models refuse to predict facts that contradict their memorized knowledge. This can pose problems when using IE models to update knowledge graphs containing dynamic information (e.g., the current leader of a country).

Substitution specifically allows us to automatically detect memorization errors, in the form of errors which disappear upon counterfactual entity replacement. We characterize "memorization errors" as false positive or false negative relational triples (s, r, o), where the error is no longer made after s, o, or both are replaced with counterfactual entities.

3.1.3 Composing Transformations

Transformations can be composed just as any other functions. In this work, when substitution and concatenation are used together, we compose them into a single transformation by substituting entities in each text before concatenating them.

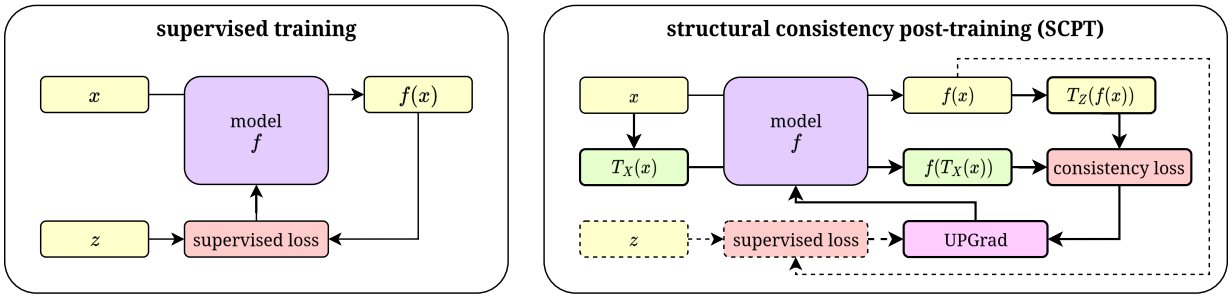

Figure 3.1: Comparison of supervised IE versus SCPT. **(Left)** IE models are typically trained in supervised fashion on labelled data to minimize loss between predictions $f(x)$ and the ground-truth label probabilities z. **(Right)** SCPT generates transformed inputs and outputs (using data that is not necessarily labelled), to compute a consistency loss for further training.

3.2 Method

To keep the notation in this section simple, we write the actions $T_X : X \mapsto X$ and $T_Y : Y \mapsto Y$ as if they are unary operations, although transformations like concatenation have higher arity.

3.2.1 Evaluating for Consistency

The *consistency* of a model $f : X \mapsto Y$, with respect to a transformation T, is measured by the similarity between $f(T_X(x))$ and $T_Y(f(x))$. This comparison takes place in the output space, so one can measure consistency using the same metric function $d_Y : Y \times Y \mapsto \mathbb{R}$ that is typically used to evaluate the model on the task itself (lower is better).

$$C_T = d_Y(f(T_X(x)), T_Y(f(x))) \tag{3.3}$$

Just as with d_Y, the consistency metric may be averaged across an entire dataset to obtain a single score for the model with respect to the transformation T. In information extraction contexts, d_Y is primarily the F1 score, but we can also evaluate consistency precision, consistency recall, and so on.

During evaluation, it is useful to perform each data augmentation (concatenation, substitution) separately to study their effects in isolation.

3.2.2 Training for Consistency

In tasks such as information extraction, coercing a model's numerical output (probabilities) to the output space (relational triples) is not a differentiable operation, meaning that C_T is not a differentiable function either.

To formalize training with consistency as an objective, we introduce a third space Z, the numerical output space of the model. We call Z "probability-space", because in information extraction, the members of Z are usually tensors of probabilities or logits, with each entry corresponding to a different candidate relational triple. However, unlike X and Y which are fixed by the task, Z can vary from model to model.

Then, we break the model down into the parametric function $f_{X \to Z} : X \mapsto Z$, and the usually-deterministic post-processing step $f_{Z \to Y} : Z \mapsto Y$.

$$f(x) = f_{Z \to Y}(f_{X \to Z}(x)) \tag{3.4}$$

Next, we define $T_Z : Z \mapsto Z$, the action of the transformation T on this space, and a metric function $d_Z : Z \times Z \mapsto \mathbb{R}$. This finally leads to the differentiable **consistency loss** for the transformation T:

$$L_{C,T} = d_Z(f_{X \to Z}(T_X(x)), T_Z(f_{X \to Z}(x))) \tag{3.5}$$

Consistency loss does not require the label y, allowing it to be used outside supervised learning contexts.

If $f_{X \to Z}$ can be batched, it is possible to combine the inputs $T_X(x)$ and x into a single batch for parallel processing.

Dice-Sørensen Consistency Loss

A potential choice for d_Z is to use the same loss function (often cross-entropy) that one would use to train the model in regular supervised learning. Cross-entropy is also the original loss function proposed by [86]. However, in information extraction settings, doing so creates a mismatch between the divergence-based losses typically used in training, versus confusion matrix-based evaluation metrics like F1 score. Viewing information extraction as a binary classification task for each candidate triple, the vast majority of labels are negative. In practice, we find that training for consistency using divergence-based losses increases the consistency precision on augmented data at the cost of a decrease in consistency recall, cancelling out to little overall improvement (if any) in consistency F1.

To better align training with evaluation, we motivate a different distance metric d_Z based on the expectation of the confusion matrix under sampling, which is differentiable. Variations of DSC have been explored as a supervised training loss in NLP, with [44] finding significant improvements on tasks with F1-based metrics such as named entity recognition. For this study, we use a simple variation of the DSC formula which is symmetric with respect to its inputs, has no hyperparameters, and has an intuitive interpretation. Our formula extends the cF1 score from [91], comparing two[1] probability vectors p and q:

$$d_Z(p, q) = 1 - \frac{2 \sum_i p_i q_i + 1}{\sum_i p_i + \sum_i q_i + 1} \tag{3.6}$$

This formula has a natural interpretation: at a single index i, consider sampling predictions from Bernoulli(p_i) and Bernoulli(q_i) independently. Then, $p_i q_i$ is the probability

[1] This "product-sum-ratio" formula admits a natural extension to 3 or more probability vectors, which may be useful for future explorations into multi-way consistency.

of both p and q agreeing on a true extraction at index i. Across an entire tensor of probabilities (masking away invalid elements), $\sum_i p_i = \mathrm{E}[\mathrm{P}_p]$ is the expected number of positive predictions made by p, $\sum_i q_i = \mathrm{E}[\mathrm{P}_q]$ is the expected number of positive predictions by q, and $\sum_i p_i q_i = \mathrm{E}[\mathrm{TP}]$ is the expected number of true positives. Thus, d_Z can be rewritten as a smoothed F1 score, computed using the expectation of the confusion matrix under sampling:

$$d_Z(p, q) = 1 - \frac{2\mathrm{E}[\mathrm{TP}] + 1}{\mathrm{E}[\mathrm{P}_p] + \mathrm{E}[\mathrm{P}_q] + 1} \tag{3.7}$$

Multi-Objective Optimization

Training solely for consistency does not guarantee model performance, as it is possible for a model to be incorrect but consistent. To regulate this, we always mix the consistency loss with the model's original training objective in a multi-objective optimization problem. We use the UPGrad algorithm [61] through the `torchjd`[2] Python library, which is theoretically guaranteed to produce gradient steps that do not conflict with the gradient directions of either component objective.

UPGrad is sensitive to the relative scales of the component objectives. The scaling factor applied to the consistency loss in this mixture is a hyperparameter. As the scaling factor increases, the model eventually enters a regime where the effect of the original loss becomes negligible, and its IE performance collapses. We tune it by performing a grid-search over powers of ten (. . ., 0.01, 0.1, 1.0, 10.0, 100.0, . . .), selecting the best value based on validation set performance on the original IE task.

Handling Multiple Transformations

For computational efficiency, we compose both concatenation and substitution into a single transformation during training. In this mixed transformation, substitution is performed randomly with an independent 50% probability for each entity to remain unchanged.

3.2.3 Consistency Post-Training

In this work we focus on structural consistency training as a *post-training* step to be performed after the main training of an information extraction model. We call this technique

[2] https://torchjd.org

structural consistency post-training (SCPT), consisting of further training epochs using the same data that was used during the training phase, potentially along with some additional data. This keeps the method easy to apply to existing information extraction models, as the main training loop can be kept without modification.

In all experiments, we start from a trained model. As a baseline, we continue to train the model for the equivalent of 10 epochs on the full training set of the original model, but at one tenth of the learning rate.

Then, for structural consistency post-training, we define the concatenation, substitution, and mixed transformations for the three spaces (text-space, probability-space, and triple-space).

3.2.4 Using Unlabelled Data

When using an unlabelled dataset, it is no longer possible to compute a model's supervised learning loss for multi-objective optimization. To keep the model from falling into the consistent-but-incorrect regime, we continue to use the model's original labelled training dataset alongside the unlabelled data. On each gradient step, two batches are sampled; one from the labelled dataset and the other from the unlabelled dataset. On the labelled batch, both original loss and consistency loss are computed. The unlabelled batch provides only consistency loss, which is added to the consistency loss from the labelled batch.

3.3 Experiments

To demonstrate the generality of SCPT, we apply it to three models with diverse architectures, all of which were state-of-the-art in the past.

3.3.1 Datasets

Table 3.1 summarizes the datasets used in our experiments:

- **Relation Classification**: We use variations of SciERC [52] and ChemProt [36] as preprocessed by [11]. Each element of the preprocessed dataset is a sentence, with tags of the form `[[`, `]]`, `<<`, and `>>` around the head and tail respectively.

- **Relation Extraction**: We use variations of the WebNLG and NYT datasets as preprocessed by [97]. While they differ substantially from the original datasets, these variations have become standard in relation extraction research.

Dataset	# Train	# Validation	# Test	# Relation Types
ChemProt	4,169	2,427	3,469	13
NYT	56,196	5,000	5,000	24
SciERC	3,219	455	974	7
WebNLG	5,019	500	703	216

Table 3.1: The train, validation, and test sizes of the four datasets used in this work, as well as the number of relation types in each.

3.3.2 Models

In our experiments, we reproduce the following models and modify their training loops for consistency training, while preserving the other training tricks used by the original model, such as the optimizer, learning rate schedule, gradient clipping, and batch size.

SciBERT

SciBERT [11] is a BERT model variant specifically trained on scientific text. By attaching a multi-class classification head to the model, and doing some very light finetuning, its authors created a simple relation classification (RC) model which nevertheless achieved state-of-the-art performance on the SciERC and ChemProt datasets at the time. In our experiments we recreate the RC model by reproducing the finetuning steps on the published SciBERT model weights.

SciBERT is by far the simplest model considered in this work. Its probability-space is a class probability vector with one element per candidate relation type (and no null class). Every input to SciBERT is a sentence with a single head entity and a single tail entity, marked using delimiters inserted directly in the text. Concatenating two sentences x_1 and x_2 creates two sentence pairs – one where the entities from x_1 are marked, and another where the entities from x_2 are marked. As demonstrated in Figure 3.2, a consistent model must preserve the label corresponding to the marked entities, while ignoring the unmarked entities from the other sentence. Meanwhile, substitution transformations do not affect the class probability vector.

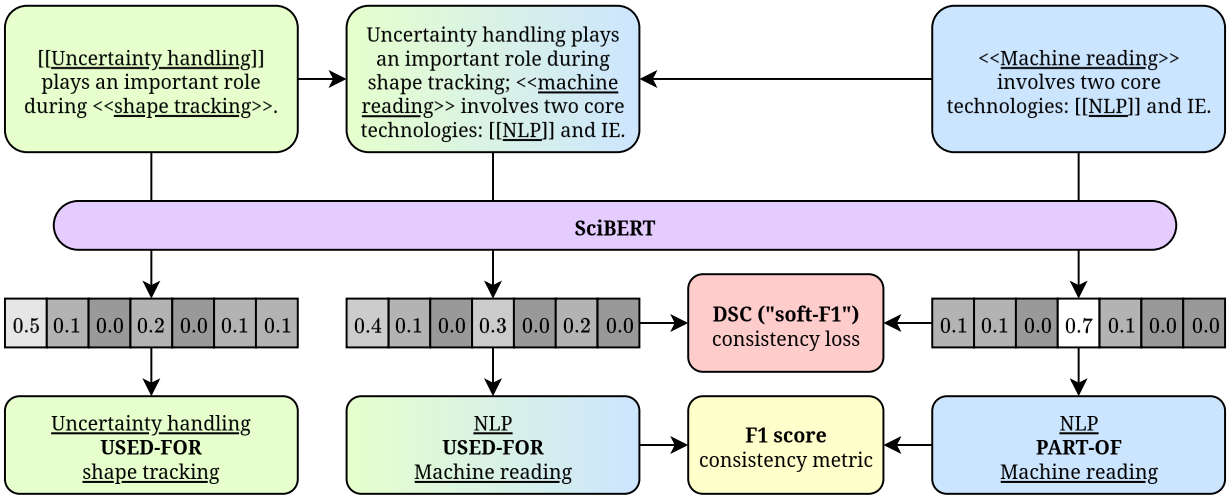

Figure 3.2: Concatenation consistency for SciBERT. The calculation of the concatenation consistency loss and metric for SciBERT. On the particular input shown here, the consistency metric would be 0, since the output relation type does not match.

EmRel

EmRel [87] performs relation extraction (RE), a task that involves identifying which directed pairs of entities in a given text are related, as well as classifying the relation types of each relation. The model does not extract its own entities. Instead, the input x includes not just the text t of T tokens, but also a ground-truth entity-token mask M of dimensions $E \times T$, indicating which tokens correspond to each of the E entities $\{e_1, \ldots, e_E\}$ in the text. When substituting the tokens of an entity, the mask must change to reflect the positions of the new tokens, but the overall position of the entity within the list of entities will stay the same. During concatenation, if M_1 is the $E_1 \times T_1$ mask of the input x_1 and M_2 is the $E_2 \times T_2$ mask of the input x_2, then the concatenated text $x_1 x_2$ has $T_1 + T_2$ tokens mapped to $E_1 + E_2$ entities via the block matrix

$$M_{ij} = \begin{bmatrix} M_1 & 0_{E_1 \times T_2} \\ 0_{E_2 \times T_1} & M_2 \end{bmatrix} \tag{3.8}$$

Indices $0 \leq i < E_1$ in the entity axis correspond to the entity e_{1k} from text x_1, while indices $E_1 \leq i < E_1 + E_2$ correspond to the entity $e_{2(i-E_1)}$ from text x_2.

The probability-space Z of EmRel is an adjacency tensor of dimensions $E \times E \times R$, where E is the number of entities in the text and R is the number of relation types. Each

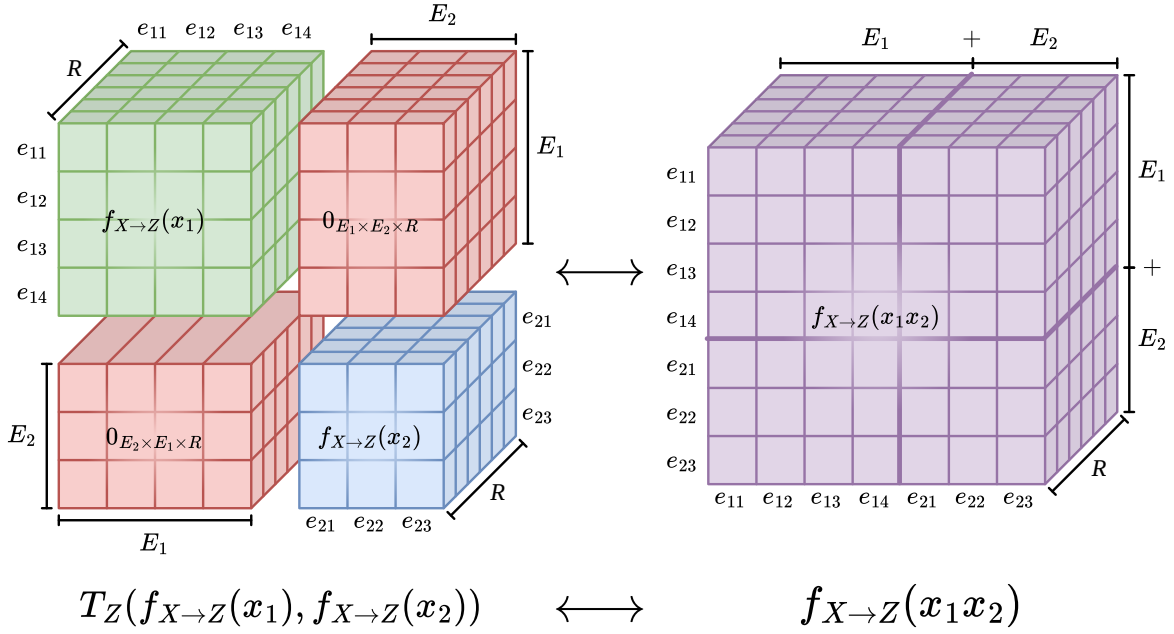

Figure 3.3: Concatenation consistency loss is computed by comparing block tensors in EmRel's probability-space. In this example, text x_i has 4 entities $\{e_{i1}, e_{i2}, e_{i3}, e_{i4}\}$, and text x_j has 3 entities $\{e_{j1}, e_{j2}, e_{j3}\}$.

entity e always corresponds to a $1 \times E \times R$ and an $E \times 1 \times R$ slice of this tensor, regardless of how many spans and tokens make up the entity. Since entity positions are invariant under substitution, this adjacency tensor is completely unaffected by substitution. Meanwhile, for concatenation, let $z_1 = f_{X\to Z}(x_1)$ be the $E_1 \times E_1 \times R$ adjacency tensor of the first text x_1, and let $z_2 = f_{X\to Z}(x_2)$ be the $E_2 \times E_2 \times R$ adjacency tensor of the second text x_2. Then, the concatenated input $x_1x_2 = T_X(x_1, x_2)$ has a $(E_1 + E_2) \times (E_1 + E_2) \times R$ adjacency tensor $z_{12} = T_Z(f_{X\to Z}(x_1x_2))$. Similar to the entity-token mask, indices $0 \le i < E_1$ in each entity axis correspond to the entity e_{1i} from input x_1, while indices $E_1 \le i < E_1 + E_2$ correspond to the entity $e_{2(i-E_1)}$ from input x_2. At positions in z_{12} corresponding to one entity from x_1 and another entity from x_2, the adjacency tensor should be 0, since there should be no evidence of relation between two entities from semantically different texts. Thus, similar to concatenation for entity-token masks, the concatenation operation in EmRel's probability

space is a block tensor operation:

$$z_{12} = T_Z(z_1, z_2) = \begin{bmatrix} z_1 & 0_{E_1 \times E_2 \times R} \\ 0_{E_2 \times E_1 \times R} & z_2 \end{bmatrix} \tag{3.9}$$

This formula is illustrated in Figure 3.3.

SOIRP

SOIRP [15] performs joint entity and relation extraction (JERE) following a table-filling approach. Its probability-space consists of two tensors whose dimensions are determined by the number of tokens T in the text. First, an entity-span tensor of dimensions $T \times T \times 2$ contains the class probabilities of a binary classification performed for each token pair. For tokens i and j with $i \leq j$, the model predicts that an entity exists beginning at i and ending at j if the entry at position $ij0$ of this tensor is less than the entry at position $ij1$. This could have been modelled more simply using a single probability per pair, but it was not. Second, a subject-object tensor of dimensions $T \times T \times R \times 6$ contains the class probabilities of a 6-way classification performed for each pair of tokens, for each of the R relation types. For each position ijr, the tag corresponding to the greatest probability is predicted. One tag is `NA`, indicating no relation. The other five tags all indicate that tokens i and j are part of two entities, and that these two entities form the head and tail respectively of a relation of type r. The interpretation differs depending on which part of the entity each token comes from:

- `H`: token i is the *first* token of an entity that consists of more than 1 tokens, token j is the *first* token of an entity that consists of more than 1 tokens
- `T`: token i is the *last* token of an entity that consists of more than 1 tokens, token j is the *last* token of an entity that consists of more than 1 tokens
- `S`: token i is the *only* token of an entity that consists of exactly 1 token, token j is the *only* token of an entity that consists of exactly 1 token
- `SH`: between tokens i and j, one of them is the *only* token of an entity that consists of exactly 1 token, and the other is the *first* token of an entity that consists of more than 1 tokens
- `ST`: between tokens i and j, one of them is the *only* token of an entity that consists of exactly 1 token, and the other is the *last* token of an entity that consists of more than 1 tokens

Combining the two tables using an algorithm created by the authors of [15], there is enough information to simultaneously extract entity spans and relations in a single pass, without needing to classify every single possible combination of head/tail start/end tokens and relation types, which would require an impractically-large $T \times T \times T \times T \times R$ adjacency tensor. However, this decomposition means that every relational triple requires the combination of 3 or 4 different classifications, corresponding to 10 to 16 different individual probabilities across the two tensors. We train for consistency using the entity-span and subject-object tensors, rather than the large 5-dimensional adjacency tensor.

To preserve tag semantics during substitution, we must handle single-token and multi-token entities differently. A single-token entity may only be replaced by another single-token entity, and a multi-token entity may only be replaced by another multi-token entity. Entities which overlap with other entities may never be substituted. When the length of the entity changes, we add or remove positions in the middle of the entity, ensuring that the start and end tokens of the entity are aligned before and after the transformation.

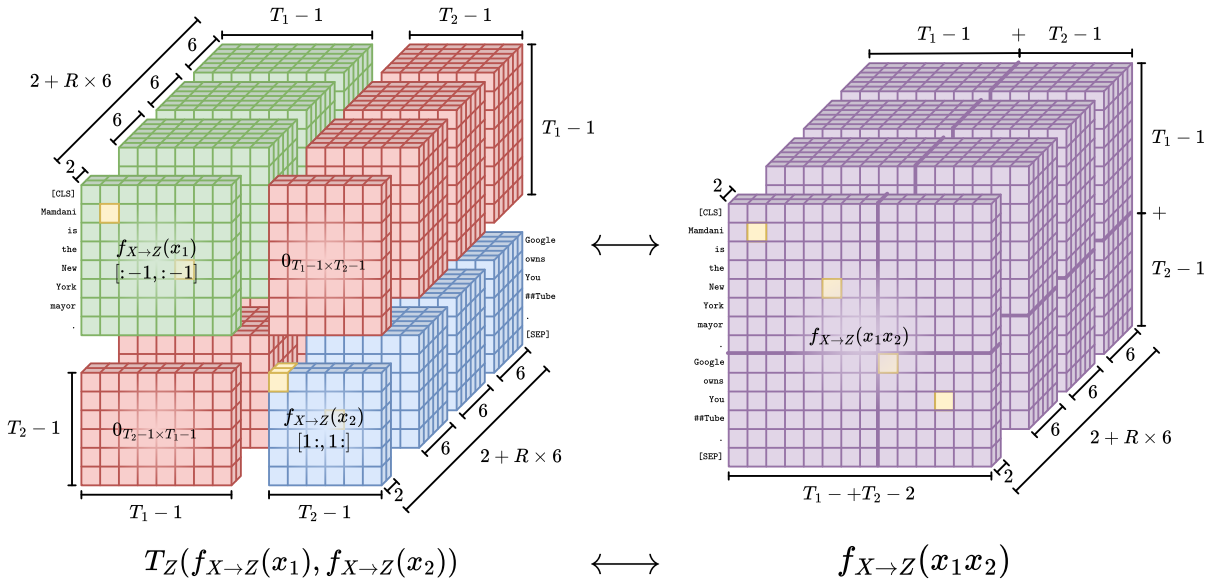

Figure 3.4: Concatenation consistency loss is computed by comparing multiple block tensors in SOIRP's probability space.

Meanwhile, we follow a similar block tensor construction method to that of EmRel to perform the equivalent of concatenation in probability-space, as shown in Figure 3.4.

In this case, we concatenate along the token axes instead of entity axes. Not all tokens are kept. For the concatenated tokens to continue representing a valid string, the trailing `[SEP]` token of the first text is stripped, and the leading `[CLS]` token of the second text is stripped. To simplify the notation, we concatenate and squeeze the entity-span and subject-object tensors along the class and relation axes to create a single $T \times T \times L$ tensor $z = f_{X \to Z}(x)$, where $L = 2 + 6R$. Then, we can express concatenation as a block tensor operation

$$z_{12} = T_Z(z_1, z_2) = \begin{bmatrix} z_1[\,:-1,\ :-1, \ldots] & 0_{(T_1-1)\times(T_2-1)\times L} \\ 0_{(T_2-1)\times(T_1-1)\times L} & z_2[1\colon, 1\colon, \ldots] \end{bmatrix} \tag{3.10}$$

where $[\,:-1]$ indicates taking all positions along an axis except for the last, and $[1\colon]$ indicates taking all positions along an axis except for the first.

When measuring consistency loss for the purpose of training, we compute the Dice-Sørensen losses separately for the entity-span and subject-object tensors and add them together. Otherwise, if the loss were computed in a single component, then the subject-object tensors would completely dominate the loss calculation due to having $3R$ times as many elements as the entity-span tensor.

3.3.3 Statistical Significance

All reported results are based on an evaluation with 5 random seeds. For each seed, we prepare a fully-trained version of the original model, giving 5 copies of the model that we then perform SCPT on. In Tables 3.4 and 3.5, an asterisk on a metric value indicates that the value attained by SCPT surpasses the value attained by the other method with a significance of $p < 0.05$, when tested using the one-sided Wilcoxon signed-rank test [84], implemented in SciPy [77], across the 5 seeds. For a comparison on 5 seeds, this means that all 5 comparisons must favour our method.

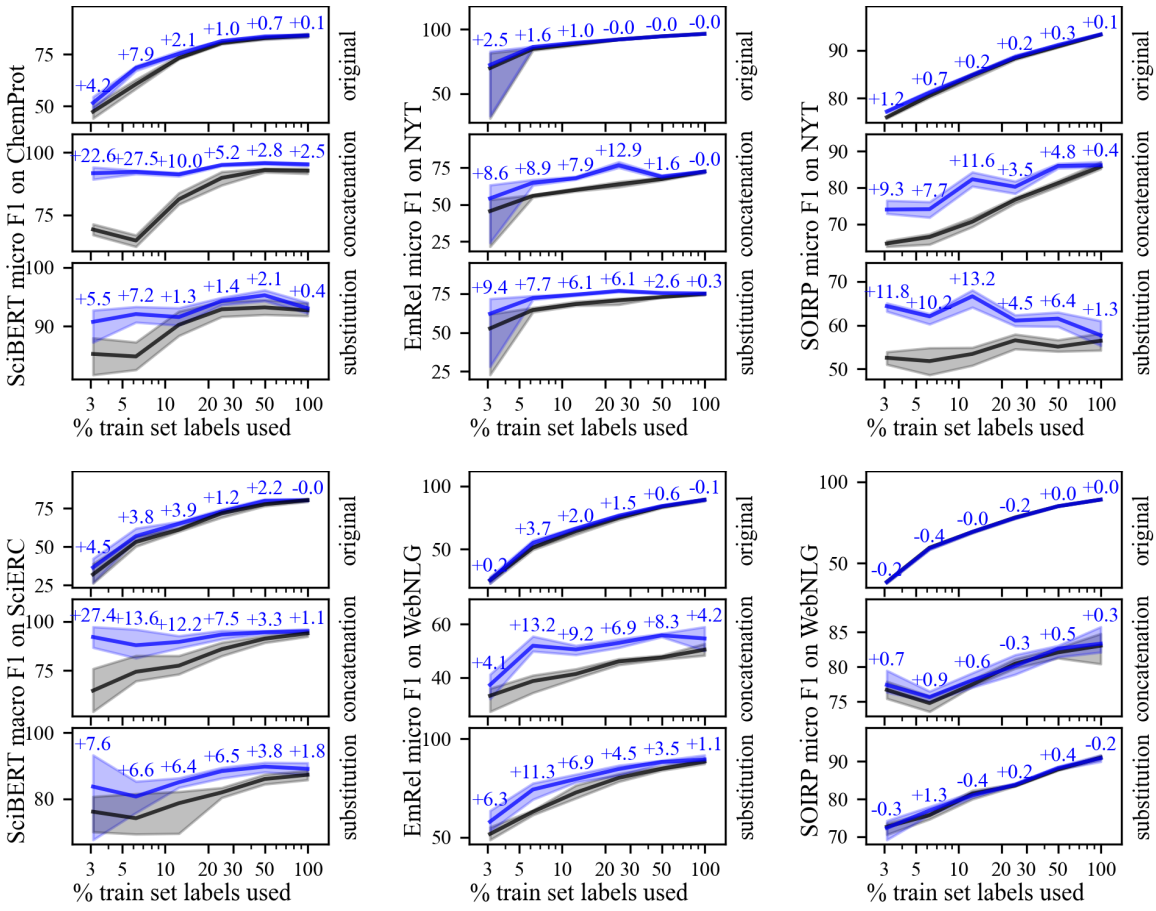

Figure 3.5: Three line plots for each combination of model and dataset, showing the model's mean performance before (black) and after SCPT (blue), as a function of the fraction of labelled training data used both for training the base model and during SCPT. Performance gains (ours minus original) are indicated in blue text. Shaded areas indicate the minimum to maximum range of the metric across the 5 evaluation seeds.

3.3.4 Low-Data Settings

To simulate low-data settings, we randomly partition the training sets of the datasets listed previously into a labelled portion, containing a $\frac{1}{\alpha}$ fraction of the data, and an unlabelled portion containing the remaining $1-\frac{1}{\alpha}$ proportion. The downsampled data has roughly the same relation type distribution as the original dataset, although we force the sampling to include at least one instance of each relation type. This matches practical settings where a large text corpus has been collected by automated means, but only part of it (1 in α) is manually labelled.

To produce a base model, we repeat a model training process on the labelled portion only, for α times as many epochs as the original model. This keeps the overall number of gradient steps the same as if using the full dataset. Nevertheless, the model performance degrades as expected, due to the smaller quantity of data.

Next, during SCPT, we continue to use the labelled portion, while also iterating over the unlabelled portion following Section 3.2.4. Here, we also do α times as many epochs over the labelled portion to ensure the same number of gradient steps.

Figure 3.5 shows the performance gains from SCPT compared to the base model, using powers of 2 for α, up to 32. In general, both methods' performance at the original task follow steady upward trends as the amount of labelled training data increases. The SCPT model has greater performance in most settings (except for SOIRP on WebNLG), but the gap closes as the fraction of labelled training data approaches 100%, showing that SCPT, like other data augmentation methods, is most effective in lower data regimes. Interestingly, the upward trend of original performance is not necessarily mirrored in the consistency metrics, where we see in examples such as SOIRP on NYT that it is possible to have roughly the same substitution consistency at low and high amounts of data. This demonstrates that it is indeed possible to have a model that extracts consistently but incorrectly.

3.3.5 Error Analysis

Averaged across 5 seeds, we count the different types of consistency-related errors made by the EmRel and SOIRP models across the full test sets of NYT and WebNLG, when using $\frac{1}{16}$ of the labelled data. Absolute error counts are always greater for NYT, whose test set is about 7 times larger than that of WebNLG.

method, dataset	IN↓	IP↓	XA↓
EmRel, NYT			
original	2171.4	728.8	7679.8
SCPT (*ours*)	<u>1579.4</u>	<u>376.2</u>	<u>5392.8</u>
EmRel, WebNLG			
original	819.2	247.4	722.0
SCPT (*ours*)	<u>638.6</u>	<u>212.4</u>	<u>645.6</u>
SOIRP, NYT			
original	3260.6	258.8	1880.2
SCPT (*ours*)	<u>2553.6</u>	<u>252.6</u>	<u>1329.4</u>
SOIRP, WebNLG			
original	<u>243.8</u>	62.0	462.6
SCPT (*ours*)	282.0	<u>44.6</u>	<u>330.4</u>

Table 3.2: Counts of each concatenation-related error type before and after SCPT. The best value in each comparison is underlined.

Concatenation There are 4 types of extractions relevant to concatenation consistency. Extractions which match before and after concatenation are **consistent positives**, while 3 types of errors include

- **Inconsistent Negatives** (IN) extracted before concatenation, but not after.
- **Inconsistent Positives** (IP) between entities from the same text, extracted after concatenation, but not before.
- **Cross-Association Errors** (XA) between entities from different texts.

Table 3.2 shows that SCPT almost always reduces all three error types. The total number of concatenation-related errors decreases by an average of 27.0% on NYT and 15.4% on WebNLG, explaining the increases observed in concatenation consistency.

Substitution Memorization errors can be split into **memorized false positives** (MFP) and **memorized false negatives** (MFN). Table 3.3 shows that SCPT almost always reduces both error types. The total number of memorization errors decreases by an average of 22.3% on NYT and 8.0% on WebNLG, explaining the increases observed in substitution consistency.

method, dataset	MFN↓	MFP↓	total↓
EmRel, NYT			
original	490.4	774.2	1264.6
SCPT (*ours*)	<u>457.2</u>	<u>439.4</u>	<u>896.6</u>
EmRel, WebNLG			
original	110.6	258.0	368.6
SCPT (*ours*)	<u>103.4</u>	<u>223.6</u>	<u>327.0</u>
SOIRP, NYT			
original	<u>320.8</u>	1634.0	1954.8
SCPT (*ours*)	345.4	<u>1306.0</u>	<u>1651.4</u>
SOIRP, WebNLG			
original	92.8	167.2	260.0
SCPT (*ours*)	<u>90.4</u>	<u>157.6</u>	<u>248.0</u>

Table 3.3: Counts of each memorization-related error type before and after SCPT. The best value in each comparison is underlined.

3.3.6 Ablation Studies

Averaged across 5 seeds, we evaluate the importance of the key parts of our method, when using $\frac{1}{16}$ of the labelled data. For fairness, we repeat the full grid search process of selecting an optimal consistency loss scaling factor for each variation of the technique, if applicable. The F1 scores of the model with respect to the original IE task, concatenation consistency, and substitution consistency, are summarized in Tables 3.4 and 3.5.

Supervised Augmentation To verify that consistency training is meaningfully different from directly supervising on the labels of the augmented texts, we investigate what happens when consistency loss $L_{C,T}$ is replaced with Equation 3.11, which replaces $f_{X \to Z}(x)$ with the ground-truth label z.

$$L_{A,T} = d_Z(f_{X \to Z}(T_X(x)), T_Z(z)) \tag{3.11}$$

The major downside of supervision is that only the $\frac{1}{16}$ portion of labelled data is accessible. Indeed, except for SOIRP on WebNLG, the resulting model fails to improve very much on the original model without SCPT, especially with respect to the consistency metrics.

	ChemProt			SciBERT		
method	orig. ↑	cat. ↑	sub. ↑	orig. ↑	cat. ↑	sub. ↑
SciBERT						
original	*60.6	*64.7	*84.9	*53.1	74.3	74.1
SCPT (*ours*)	68.5	92.3	92.1	57.0	87.9	80.7
supervised augmentation	67.0	*87.7	*87.3	55.6	*80.6	80.4
concatenation only	68.2	92.2	*89.1	61.1	87.9	83.7
substitution only	*62.5	*66.6	90.5	58.5	79.5	88.2
JS divergence	*66.9	*91.2	90.8	56.4	86.7	80.4

Table 3.4: Ablation study results of SCPT on ChemProt and SciERC. The best value in each comparison is underlined. Values beaten by SCPT (our) method with a significance of $p < 0.05$ (in all 5 pairwise comparisons) are indicated with a leading asterisk.

	NYT			WebNLG		
method	orig. ↑	cat. ↑	sub. ↑	orig. ↑	cat. ↑	sub. ↑
EmRel						
original	*84.8	*56.0	*64.6	*51.2	*38.7	*63.0
SCPT (*ours*)	86.3	64.9	72.3	54.9	52.0	74.3
supervised augmentation	*84.6	*57.4	*61.7	*51.2	*39.6	*64.6
concatenation only	85.9	66.1	*64.0	53.9	52.5	*66.2
substitution only	86.2	*56.5	76.0	55.7	*42.8	77.6
JS divergence	*84.9	*57.3	*64.5	*51.2	*39.2	*62.5
SOIRP						
original	*80.5	*66.4	*51.8	59.7	74.8	*75.7
SCPT (*ours*)	81.2	74.1	62.0	59.3	75.7	77.0
supervised augmentation	*73.6	78.0	70.7	58.3	83.1	81.0
concatenation only	*81.0	*68.2	*52.2	59.4	76.3	*76.2
substitution only	81.1	*67.2	55.2	59.3	75.8	80.7
JS divergence	81.3	74.7	62.8	59.6	75.2	76.1

Table 3.5: Ablation study results of SCPT on NYT and WebNLG. The best value in each comparison is underlined. Values beaten by SCPT (our) method with a significance of $p < 0.05$ (in all 5 pairwise comparisons) are indicated with a leading asterisk.

Removing Transformations Instead of composing substitution followed by concatenation, we investigate the effect of removing either of the transformations. Relative to our base SCPT method, using only one transformation usually results in an equal or greater gain with respect to that transformation's consistency metric. However, this almost always comes at the cost of decreasing the model's consistency with respect to the transformation that was removed, losing most of the advantage that SCPT had over the original model there. The most optimal way to increase both concatenation and substitution consistency together remains SCPT with both transformations.

Divergence-Based Consistency Loss Instead of our Dice-Sørensen consistency loss, we substitute the Jensen-Shannon (JS) divergence [46], which is symmetric and bounded like the DSC. However, it is based on a sum of Kullback-Leibler (KL) divergence values, similar to the loss functions more typically used in consistency training [86] and supervised information extraction. JS divergence does slightly worse than SCPT for SciBERT, where relation classification is a multi-class classification problem relatively-balanced labels. For EmRel, JS divergence fails to improve on the original model at all.

3.4 Summary

Concatenation and substitution are structural transformations of text which do not preserve labels in the high-dimensional NLP tasks of information extraction (IE). We develop evaluation metrics and differentiable training losses for measuring how consistent an IE model is with respect to such transformations. We then present structural consistency post-training (SCPT), a post-training regime to optimize models for consistency after their main training. This greatly reduces the rates of cross-association and memorization errors made by the models, leading to increased concatenation and substitution consistency. SCPT often also increases the model's performance at the original IE task. The method can incorporate unlabelled data in a self-supervised fashion, leading to its strength in settings with large text datasets that are only partially labelled. We envision that consistency training and evaluation will drive models toward greater robustness, and that SCPT will enable more efficient use of labels in low-resource settings. Future works may consider applying SCPT, especially concatenation, as a transformation for other NLP tasks, potentially including generative tasks involving large language models.

Chapter 4

Guided Information Extraction

As discussed in Section 2.2.7, the trend in the state of the art in universal information extraction (UIE) is toward methods that utilize annotation guidelines to prompt a language model with exact definitions of what to extract. This is a major improvement over methods which simply state the name of the types to be extracted, which suffer from variations in how the same names are interpreted across different datasets, as demonstrated in [92, 66] and Table 2.2. However, we notice a cycle in which each state-of-the-art UIE method comes with a new dataset of schemas that it was trained on, with promises that the dataset can be used to develop further UIE methods. That dataset then sees little reuse in subsequent works. To name just a few instances, as seen in Table 2.1: OneKE leverages IEPile [22], GoLLIE collects an unnamed dataset [66], and B^{2}NER uses B^{2}NERD [92].

The reason for this pattern is that the subtasks of UIE – including named entity recognition (NER), relation extraction (RE), and event extraction (EE) – are all classically defined at the level of a single dataset with a single schema, and the fair evaluation of that one dataset is completely unconcerned with foreign schemas from other datasets that the method may be fitted on. This leaves no limits on the schema resources that can be potentially used. Every new paper is incentivized to collect more schemas, find the optimal way to fit a model to those schemas, and then publish the method and dataset together. The dataset is already thoroughly exploited by its own creators before it is even published, so any researcher pursuing an improvement over that model should rather seek a new dataset or an extension to an existing one.

Not only can this cycle create duplicate work, the lack of constraints on schema usage makes it difficult to answer important research questions such as: *For a given dataset of schemas and labelled texts, what is the best method that can be developed to do information*

extraction for arbitrary schemas? That is the real question of value, and it is complementary to the collection of increasingly large schema datasets, since this best method can be applied to any arbitrary schema dataset. We are interested in answering this question for both *unseen* schemas outside of that given dataset, and *seen* schemas that may be present in the given dataset with some amount of labelled texts annotated with extractions. In practice, not every paper has provided evaluations for both methods, and even when they do, the setup varies wildly. For instance, InstructUIE and GoLLIE are only evaluated in 0-shot settings, though data leakage means that some of the types in those schemas are actually seen to GoLLIE [78, 66]. Other methods such as Code4UIE are only evaluated in few-shot settings [23], excluding them from the evaluation of unseen schemas entirely unless we consider the degenerate case of formatting a few-shot prompt template with 0 in-context examples. While few-shot methods only incorporate a small number of examples at a time, they retrieve their examples from large collections derived from entire training sets; this does not realistically represent truly low-resource schemas, which may only have a few labelled texts in total. KnowCoder is a good example of a model whose evaluation clearly attempts to answer the question for both unseen (0-shot) and seen (5-shot, supervised finetuning) settings [45]. In supervised finetuning, the authors provide 1%, 5%, or 10% of the training set of a whole dataset for finetuning before evaluating the fine-tuned model on that dataset and its schema.

Producing a consistent and quantitatively comparable answer to these questions is impossible on the current trajectory of UIE. In this chapter, we present and define Guided Information Extraction (GIE), a task which formalizes the trend towards the use of guidelines in modern universal IE. We further increase the rigour in GIE by making qualities such as entity grounding and coreference resolution non-negotiable, and establish strict evaluation regimes. Then, using this task definition, we build a new challenge dataset we call AdaIE. It contains 15 gold datasets worth of schemas and corresponding guidelines, collected from across information extraction, further expanded using reasoning-based data augmentations to create new entity and relation types. By partitioning AdaIE into evaluation schemes, we obtain fair and comparable measurements that answer the question of extraction performance on both seen and unseen schemas.

We specifically provide this dataset without training any new IE methods on it ourselves. Instead, we focus our efforts on evaluating existing models, and invite other researchers to exploit AdaIE for developing future information extraction techniques for low-resource schemas.

Concept	Example
symmetric relation	$\left\{ \begin{array}{l} (\texttt{spouse}, \text{Bonnie}) , \\ (\texttt{spouse}, \text{Clyde}) \end{array} \right\}$
directional relation	$\left\{ \begin{array}{l} (\texttt{wife}, \text{Bonnie}) , \\ (\texttt{husband}, \text{Clyde}) \end{array} \right\}$
event arguments; hyper-relations	$\left\{ \begin{array}{l} (\texttt{bride}, \text{Bonnie}) , \\ (\texttt{groom}, \text{Clyde}) , \\ (\texttt{date}, \text{1926-09-25}) \end{array} \right\}$

Table 4.1: GIE relations unify relation extraction, hyper-relational extraction, and event argument extraction.

4.1 Guided Information Extraction

This section formally defines Guided Information Extraction (GIE) and its elements.

4.1.1 Entities

Let $\mathcal{S}$ be the set of all strings, and a text document is any string $d \in \mathcal{S}$. An **entity** e is a concept **grounded** by a nonempty set of **mentions** $\{[i_1, j_1) , [i_2, j_2) , \ldots\}$. Each mention is a character span $[i, j)$, such that the substring of d from indices i inclusive to j exclusive refers to the entity e.

An **entity type** E models an underlying unary predicate $p_E : \mathcal{E} \to \{0, 1\}$ on the universe $\mathcal{E}$ of all entities, which is true for exactly those entities that *evidently* belong to type E according to the document d. E has a short name s_E (not necessarily unique to E), and a detailed sentence-length natural language template v_E that **verbalizes** the type. The predicate p_E is never directly observed. In classical entity extraction, one might use a dataset of positive examples to learn an estimate $\hat{p}_E$. Instead, in GIE, one is expected to leverage v_E. See Table 2.2 for examples.

An entity type E may also indicate any number of **entity supertypes** E', meaning that $p_E(e) \Rightarrow p_{E'}(e)$ for any entity e. We then say that E is an **entity subtype** of E'.

4.1.2 Relations

A **relation** is a set of **arguments**, where each argument is a string-entity pair. The string indicates a semantic role (e.g., `date`, `owner`, `nationality`) of its corresponding entity. Strings may be repeated to assign multiple entities to the same role, and the same entity may be assigned to multiple roles. As shown in Table 4.1, this covers both relation and event extraction.

A **relation type** R models a predicate $p_R : \mathcal{R} \to \{0, 1\}$ on the universe $\mathcal{R} = 2^{\mathcal{S} \times \mathcal{E}}$ of relations, which is true for exactly those relations that are *evident* according to the document d. R has a short name s_R (not necessarily unique to R), a detailed sentence-length natural language template v_R that verbalizes the type, and a set of **argument types** $\mathcal{A}_R$. Each argument type $A \in \mathcal{A}_R$ describes a valid role for the relation type. The argument type consists of a unique name s_A, a minimum cardinality $m_A \geq 0$, a maximum cardinality $n_A \geq m_A$, and a set $\mathcal{T}_A$ of entity types. For a relation r to satisfy the relation type R, it must contain at least m_A and no more than n_A entities associated with each argument type A, and all of those entities must satisfy at least one entity type in $\mathcal{T}_A$. A minimum cardinality of 0 means that the argument is optional. Again, the predicate p_R is never directly observed, and the goal is to leverage v_R to estimate $\hat{p}_R$. See Table 2.3 for examples.

4.1.3 Mutually Independent Verbalizations

We follow [67] and use templates with placeholders as GIE verbalizations, such as the examples in Table 2.2 and Table 2.3. While it is not always grammatically precise to substitute an entity for its placeholder, the placeholder naturally indicates the association between the name of an argument and its semantic role in a relation. Verbalizations do not always use placeholders (e.g., [66]). However, placeholders are motivated by models such as BART-Gen [43].

Unlike single-label classification interpretations of IE, we allow multiple entity types to apply to a single entity, and we allow multiple relation types to be satisfied by the same group of entities. All entity and relation types must be **mutually independent**, meaning that the type predicate is unaffected by the other entity and relation types to be extracted – including its own supertypes and subtypes. The benefit of this framing is to make the extraction task **partition-equivariant** by definition; any mutually independent set of types can be partitioned into chunks, and a separate extraction for each chunk can then be unified after the fact, enabling the extraction of a large number of types even when

model context is limited. This is unlike many existing datasets, where an entity or relation may only be annotated with the most specific type that best describes it (e.g., a `{doctor}` might not be annotated as a `{person}`). To achieve this, each verbalization must be a self-contained description of its type. If a type is intended to be mutually exclusive with another type, its GIE verbalization must indicate this (e.g., by adding clauses such as "... but `{person}` is not a doctor", or "`{person}` is a person, excluding doctors").

4.1.4 Subtasks

A **schema** $S = (S_E, S_R)$ consists of a set of entity types S_E, and a set of relation types S_R. Given a document d and schema S, the **guided entity extraction (GEE)** subtask of GIE is very similar to the combination of entity recognition and coreference resolution: to output a set of entities, and to identify all applicable entity types from S_E for each entity. Given a document d, schema S, and a set of entities, the **guided relation extraction (GRE)** subtask is very similar to the combination of relation extraction and event argument extraction: to output a set of all relations between the entities that satisfy each relation type in S_R. To evaluate GRE on its own, we use the ground-truth entities as input. Finally, given a document d and schema S, **guided information extraction (GIE)** is the end-to-end task of outputting both the entities and the relations between those entities. It may be performed in a pipelined fashion by doing GEE first, then passing the extracted entities as input to the next GRE stage, but this is not the only approach.

Unlike entities, we do not require grounding annotations for relations, such as event triggers in event extraction datasets, relation triggers in DialogRE [95], and sentence-level supporting evidence annotations in DocRED [93]. Very few existing datasets in IE include such annotations, and the concept of evidence is not treated consistently, making it impossible to collect a composite dataset with relation grounding.

4.1.5 Evaluation Metrics

For each subtask of GIE, we evaluate precision, recall, and F1 score separately for each entity or relation type. However, predicted extractions can be *partially correct* without exactly matching the ground truth, and they can also partially match multiple ground-truth extractions. We take inspiration from Constrained Entity-Alignment F-Measure (CEAF) [54], a metric developed for coreference resolution which can generally be applied to any task where the output is a set of extractions with inexact matching, as described in Section 2.2.2. Given a score function $\phi(x, y)$ computing the similarity between two

individual extractions (entities or relations, in our case), CEAF computes a matching that maximizes the pairwise similarity between extractions. Using this matching, CEAF then computes recall r, precision p, and F1-score F.

Guided Entity Extraction Before evaluating GEE, for each entity e, we resolve supertype-subtype relationships by recursively adding the supertypes of any entity type found on e. Then, to evaluate GEE on a single given entity type E, we filter the entities across the entire dataset down to only those which belong to E. While [54] originally treats an entity to be a set of text spans, we soften this further to allow inexact span boundaries, by treating an entity instead as a set of characters. The natural score function between two sets of characters e_G and e_P is F1 score:

$$\phi(e_G, e_P) = \frac{2\,|e_G \cap e_P|}{|e_G| + |e_P|} \tag{4.1}$$

Guided Relation Extraction To evaluate GRE on a single given relation type R, we filter an entire dataset's relations down to only those of type R. Assume for now that the prediction uses the ground-truth entities as input. Treating each relation as a set of arguments (string-entity pairs), F1 score is once again a natural score function between two sets of arguments r_G and r_P:

$$\phi(r_G, r_P) = \frac{2\,|r_G \cap r_P|}{|r_G| + |r_P|} \tag{4.2}$$

Guided Information Extraction To evaluate GIE on a single given relation type R, we perform the same calculation as the GRE metric, but with an additional preprocessing step to handle the fact that the ground-truth and predicted sets of entities are different. We resolve this by projecting each predicted entity to the corresponding ground-truth entity with maximum character-level F1 score (Equation 4.1), allowing multiple predicted entities to be projected to the same ground-truth entity (this smooths out cases where a single ground-truth entity is extracted as multiple predictions). Predicted entities which do not overlap with any ground-truth entity cannot be projected, and are removed.

4.2 Data

A composite dataset for GIE, such as AdaIE, consists of multiple constituent datasets. Each constituent dataset contains a schema, a train split, and a test split. The documents

in both splits are fully annotated under the schema. A schema may contain redundant types (such as those produced via Section 4.2.3), but partition equivariance allows us to take any subset of the schema, filtering out unwanted types.

4.2.1 Constituent Datasets

Dataset	$\lvert S_E \rvert$	$\lvert S_R \rvert$	Notable Features
BioRED [53]	$6 \to 6$	$8 \to 87$	relation novelty metadata
CrossRE [10]	$39 \to 55$	$38 \to 52$	cross-domain
DEFT [72]	$11 \to 2$	$6 \to 2$	term-definition extraction
DialogRE [95]	$6 \to 14$	$37 \to 37$	multi-party spoken dialogue
DWIE [96]	$184 \to 158$	$65 \to 61$	multi-typed entities
FIRE [25]	$13 \to 15$	$18 \to 24$	financial domain
HyperRED [13]	$1 \to 49$	$62 \to 76$	hyper-relational qualifiers
KnowledgeNet [55]	$1 \to 12$	$15 \to 16$	
ie4wills [38]	$30 \to 30$	$38 \to 23$	legal wills domain
Re-DocRED [74]	$5 \to 32$	$96 \to 95$	
SciERC [52]	$6 \to 4$	$7 \to 6$	
SciREX [32]	$4 \to 7$	$1 \to 5$	4-way relations
SoMeSci [68]	$22 \to 24$	$11 \to 10$	
WebNLG [12]	$1 \to 72$	$215 \to 153$	special identifiers (e.g., IATA code)
WikiEvents [43]	$17 \to 17$	$42 \to 100$	verbalized event types
AdaIE (ours)	$346 \to 497$	$659 \to 747$	schema augmentation

Table 4.2: The 15 constituent datasets of AdaIE. Entity ($|S_E|$) and relation ($|S_R|$) type counts in the form $x \to y$ indicate that the dataset has x initial types before performing filtering and schema augmentation, and y types after. Datasets with no explicit entity types are counted as having 1 initial entity type.

We curate only high-quality datasets for relation and event extraction with the following requirements:

Gold-standard Annotations must be produced by humans, as opposed to distant alignment techniques such as knowledge base alignment. If the annotation process involves automatically proposing annotations, then there must be a manual review step in which

missing annotations can be added. Human verification helps ensure that annotated relations are actually entailed by the text.

Entity grounding The dataset must contain exact span annotations for entities, or it must be possible to determine the spans programmatically. Furthermore, the dataset must include coreference annotations or an equivalence relation that can be used to merge multiple text spans that refer to the same entity.

Relation annotations Every constituent dataset in AdaIE must be "end-to-end", including both entity and relation annotations.

In total, we gathered 15 datasets suitable for GIE, listed in Table 4.2. A single annotator (the author) manually wrote and reviewed verbalizations for all types, using the annotation guidelines if possible, as well as up to 25 examples sampled for each type. Types with unspecific or poorly-annotated examples, such as `miscellaneous` or `not_related` were considered invalid and were removed from the schemas; in a few cases this caused a net decrease in the dataset's number of types.

4.2.2 Closed- and Open-World Assumptions

In AdaIE, entity and relation types are annotated with a boolean flag indicating whether the type is open or closed. Closed types follow the closed-world assumption; that is, the lack of an annotation implies that there is insufficient evidence in the text to conclude that the predicate is true. Open types follow the open-world assumption, which has no such guarantee.

Accurate evaluation of IE must acknowledge which assumption is being made. A false positive extraction of a closed type is always incorrect, whereas a false positive extraction of an open type may not necessarily be incorrect, since it may simply be a missing label. To avoid punishing false positives on open types, we prefer recall (r) as the evaluation metric for open types. F1 score (F) remains the standard evaluation metric for closed types. We use q_T^τ to denote the metric achieved by a model for the entity or relation type T and task

$\tau \in \{\text{GEE}, \text{GRE}, \text{GIE}\}$:

$$q_E^{\text{GEE}} = \begin{cases} r_E^{\text{GEE}} & \text{if } E \text{ open} \\ F_E^{\text{GEE}} & \text{if } E \text{ closed} \end{cases} \tag{4.3}$$

$$q_R^{\text{GRE}} = \begin{cases} r_R^{\text{GRE}} & \text{if } R \text{ open} \\ F_R^{\text{GRE}} & \text{if } R \text{ closed} \end{cases} \tag{4.4}$$

$$q_R^{\text{GIE}} = \begin{cases} r_R^{\text{GIE}} & \text{if } R \text{ open} \\ F_R^{\text{GIE}} & \text{if } R \text{ closed} \end{cases} \tag{4.5}$$

Recall metrics could be abused by predicting all possible extractions. This is mitigated in AdaIE by the presence of both open and closed types in every schema. We require that any method evaluated on AdaIE must not be aware of which types are open or closed. Mixing open and closed types inside a single schema means that predicting large amounts of low-quality extractions will still be punished severely on closed types.

4.2.3 Schema Augmentation

Schema augmentations are logical implications that can be used to infer new entities and relations from existing ones. The new entities and relations may belong to new types not yet found in the schema. In AdaIE, a single annotator manually implemented a large number of schema augmentations as Python functions and applied them to the annotated schema-document pairs. While the implementations were initially expected to be ad hoc, we quickly found a number of common forms for most schema augmentations, summarized in Table 4.3.

Supertypes & Subtypes A **supertype-subtype** augmentation indicates that an existing entity type should be a supertype of another existing type, and adds the supertype to the annotations of every entity belonging to the subtype. **New-subtype** augmentations infer a new entity subtype of an existing entity supertype based on some relational criteria. For example, all people who have composed a song may be considered musicians; this introduces a new entity type `musician` as a subtype of `person`, and it is applied to any person who participates in a `composed` relationship (among others). Conversely, **new-supertype** augmentations create a new entity supertype out of the union of one or more existing subtypes.

Category	Examples
supertype-subtype	`{u}` is a university ⇒ `{u}` is an organization
new-subtype	`{p}` is a person `{p}` is the mother of `{c}` ⇒ `{p}` is a woman
new-supertype	`{v}` is an automobile ∨ `{v}` is a bicycle ∨ ... ⇒ `{v}` is a vehicle
tightening	`{p}` uses `{o}` ∧ `{p}` is a musician ∧ `{o}` is a musical instrument ⇒ `{p}` plays `{o}`
aggregation	`{o}` was founded by `{f}` ∧ `{o}` was founded in `{y}` ⇒ `{f}` founded `{o}` in `{y}`
decomposition	`{a}` attacked `{v}` with `{w}` ⇒ `{a}` attacked `{v}` ∧ `{a}` used `{w}` as a weapon
paraphrasing	`{p}` is someone who does research ⇔ `{p}` is a researcher
	`{p}` is a parent of `{c}` ⇔ `{c}` is a child of `{p}`

Table 4.3: Categories and simplified examples of the most common schema augmentations.

Tightening Tightening augmentations infer a more specific version of a relation type based on the types of the participating entities. To quickly but accurately collect a large number of tightening augmentations, we produce a spreadsheet with one row for each combination of relation and entity types, along with a random sample of up to 10 examples. Prioritizing the most common combinations, the annotator can quickly make judgements (at a rate of approximately 1 per minute) about possible tightenings to apply to each type combination. Meanwhile, many infrequently-occurring combinations of entity types are

identified as misannotations and removed from the dataset.

Aggregation & Decomposition Aggregation augmentations merge multiple related relations involving the same entity, which acts as a pivot. For example, in a relation extraction dataset consisting only of binary relations, similar relations such as `founder` and `location_founded` can be used to assemble a single `founding` event. In the opposite direction, decomposition augmentations split a higher-order relation into smaller relations by removing non-pivot arguments. There are exponentially many decompositions possible; to avoid too many highly-similar types we proceed by iteratively removing the least significant argument.

Paraphrasing Paraphrasing creates alternative verbalizations of the same type without changing the underlying predicate. It is the only schema augmentation found in prior works; [66] use paraphrasing extensively as a regularization technique. However, in this work we avoid paraphrasing in favour of other augmentations that do modify the predicate. Nevertheless, paraphrasing is a useful technique to apply in conjunction with other augmentations to ensure that new types do not sound too similar to the types they are derived from. For example, when decomposing "`{a}` attacked `{v}` with `{w}`" into "`{v}` was attacked using `{w}`" the result may be inverted as "`{w}` was used to attack `{v}`".

If an implication is bidirectional ($\Leftrightarrow$ in Table 4.3), and all of the conditions of the rule come from closed-world types, then the resulting type is also closed. Paraphrasing is always bidirectional. New-supertype augmentations can be bidirectional if the subtypes exhaustively partition the supertype.

4.2.4 Inducing Difficulty Levels by Multiple Partitioning

Train and Test Splits We separate AdaIE into train and test splits only. Train data can be used for any form of conditioning an extraction method, including training, in-context examples, prompt engineering, or hyperparameter optimization. If a development set is needed, such as for hyperparameter tuning, the train split may be further split however the user sees fit, but the test split must not be used. In general, data from the development and test splits of each constituent dataset is used to form the AdaIE test split, while data from the train splits are used to form the AdaIE train split. The exception to this rule is in datasets where the train set does not have gold labels; in that case we use the gold-labelled development split as training data.

For statistically rigorous evaluation, we propose evaluation across 6 different "schemes" of partitioning the dataset, as described in the following paragraphs. The evaluation schemes all use the entire AdaIE test split, but they hold out different subsets of the training data to induce various levels of difficulty for different parts of the test split. Table 4.4 shows what parts of the test data will be EASY, MEDIUM, or HARD to predict in each evaluation scheme.

Partitioning of a Single Constituent Dataset Consider a constituent dataset for which the samples have already been separated into train and test splits. After all schema augmentations are applied, this dataset has a single augmented schema $S = (S_E, S_R)$. We randomly partition the relation types S_R into two halves: S_R^α and S_R^β. The constituent dataset is then split into two train sub-splits and two test sub-splits, which we call train-α, train-β, test-α, and test-β. The -α sub-splits only contain relation types from S_R^α, and the -β sub-splits only contain relation types from S_R^β. All sub-splits (both -α and -β alike) have all entity types from S_E.

If a model is not trained on either train split, then the entire schema S will be unseen by the model, inducing a greater level of difficulty of predicting extractions under the schema S or any subset of S. The entity types in S_E and relation types in S_R will all be HARD to extract. This way of evaluating on held-out datasets is standard in UIE evaluations.

If a model is trained on the train-α split, then the entity types from S_E will be seen by the model, as will the relation types from S_R^α. That is, S_E and S_R^α will be EASY to extract. However, the relation types from S_R^β will be unseen to this model, but those relation types will be based on entity types in S_E that have been seen. This induces an intermediate level of difficulty; we say that the relation types in S_R^β have MEDIUM difficulty.

Dataset Groups The 15 constituent datasets are also partitioned into 3 groups (D^a, D^b, and D^c) of 5 datasets each. In each evaluation scheme we train with either the train-α or train-β sub-splits of 2 dataset groups (10 datasets), while holding out the remaining dataset group (5 datasets) completely unseen. From the 10 datasets selected for training, we either select all train-α sub-splits or all train-β sub-splits, holding out the other sub-splits. We name the evaluation scheme after the held-out dataset group and the held-out sub-splits (e.g., $a\alpha$), and we make bold the corresponding row and column name in Table 4.4.

During evaluation, we always evaluate on both the test-α and test-β splits of all 15 datasets, no matter the evaluation scheme. As described previously, different parts of

the AdaIE test data are induced different levels of difficulty depending on the held-out training data. For example, in the evaluation scheme $a\alpha$ (top-left), the datasets in D^a are completely held out, making their entity types S_E and relation types $S_R = S_R^\alpha \cup S_R^\beta$ all HARD. The datasets in D^b and D^c contribute their train-β splits, making their entity types S_E all EASY, and their seen relation types in S_R^β EASY as well. However, their unseen relation types in S_R^α will be MEDIUM difficulty.

$a\alpha$	S_E	S_R^α	S_R^β
D^a	HARD	HARD	HARD
D^b	EASY	MEDIUM	EASY
D^c	EASY	MEDIUM	EASY

$a\beta$	S_E	S_R^α	S_R^β
D^a	HARD	HARD	HARD
D^b	EASY	EASY	MEDIUM
D^c	EASY	EASY	MEDIUM

$b\alpha$	S_E	S_R^α	S_R^β
D^a	EASY	MEDIUM	EASY
D^b	HARD	HARD	HARD
D^c	EASY	MEDIUM	EASY

$b\beta$	S_E	S_R^α	S_R^β
D^a	EASY	EASY	MEDIUM
D^b	HARD	HARD	HARD
D^c	EASY	EASY	MEDIUM

$c\alpha$	S_E	S_R^α	S_R^β
D^a	EASY	MEDIUM	EASY
D^b	EASY	MEDIUM	EASY
D^c	HARD	HARD	HARD

$c\beta$	S_E	S_R^α	S_R^β
D^a	EASY	EASY	MEDIUM
D^b	EASY	EASY	MEDIUM
D^c	HARD	HARD	HARD

Table 4.4: The 6 evaluation schemes of AdaIE, indicating the held-out dataset group and relation type group in bold. The table shows the difficulty (EASY, MEDIUM, or HARD) classifications of the entity and relation types.

4.2.5 Aggregating Metrics

This section describes the aggregation of GEE, GRE, and GIE metrics specifically over the AdaIE dataset, leveraging the multiple levels of evaluation induced by the procedure in Section 4.2.4. Other datasets invented for GIE in the future may opt for alternative methods of aggregation.

Let $\mathcal{I} = \{a\alpha, a\beta, b\alpha, b\beta, c\alpha, c\beta\}$ denote the set of the 6 evaluation schemes. For an evaluation scheme $i \in \mathcal{I}$, let $q_T^\tau(i)$ denote the evaluation metric of a model on an entity or

relation type T with regard to the task $\tau \in \{\text{GEE}, \text{GRE}, \text{GIE}\}$, when that model is trained on the training data for that partition. Across $\mathcal{I}$, every entity type is considered easy 4 times, and hard 2 times. This leads to two difficulty-tiered Guided Entity Extraction metrics for an entity type E:

$$q_E^{\text{GEE-EASY}} = \frac{1}{4} \sum_{i \in \mathcal{I} \wedge E \text{ is EASY in } i} q_E^{\text{GEE}}(i) \tag{4.6}$$

$$q_E^{\text{GEE-HARD}} = \frac{1}{2} \sum_{i \in \mathcal{I} \wedge E \text{ is HARD in } i} q_E^{\text{GEE}}(i) \tag{4.7}$$

Meanwhile, every relation type is considered easy, medium, or hard 2 times each. This leads to three difficulty-tiered metrics for a relation type R and task $\tau \in \{\text{GRE}, \text{GIE}\}$:

$$q_R^{\tau\text{-EASY}} = \frac{1}{2} \sum_{i \in \mathcal{I} \wedge R \text{ is EASY in } i} q_R^{\tau}(i) \tag{4.8}$$

$$q_R^{\tau\text{-MEDIUM}} = \frac{1}{2} \sum_{i \in \mathcal{I} \wedge R \text{ is MEDIUM in } i} q_R^{\tau}(i) \tag{4.9}$$

$$q_R^{\tau\text{-HARD}} = \frac{1}{2} \sum_{i \in \mathcal{I} \wedge R \text{ is HARD in } i} q_R^{\tau}(i) \tag{4.10}$$

To emphasize the need to achieve high extraction quality on all types (open or closed), we report overall performance as the macro-average across all types in AdaIE. All 8 aggregated AdaIE evaluation metrics are stated in Equations 4.11 through 4.18. They are divided into 2 aggregated metrics for Guided Entity Extraction:

$$q^{\text{GEE-EASY}} = \frac{\sum_{S \in D} \sum_{E \in S_E} q_E^{\text{GEE-EASY}}}{\sum_{S \in D} |S_E|} \tag{4.11}$$

$$q^{\text{GEE-HARD}} = \frac{\sum_{S \in D} \sum_{E \in S_E} q_E^{\text{GEE-HARD}}}{\sum_{S \in D} |S_E|} \tag{4.12}$$

3 aggregated metrics for Guided Relation Extraction:

$$q^{\text{GRE-EASY}} = \frac{\sum_{S \in D} \sum_{R \in S_R} q_R^{\text{GRE-EASY}}}{\sum_{S \in D} |S_R|} \tag{4.13}$$

$$q^{\text{GRE-MEDIUM}} = \frac{\sum_{S \in D} \sum_{R \in S_R} q_R^{\text{GRE-MEDIUM}}}{\sum_{S \in D} |S_R|} \tag{4.14}$$

$$q^{\text{GRE-HARD}} = \frac{\sum_{S \in D} \sum_{R \in S_R} q_R^{\text{GRE-HARD}}}{\sum_{S \in D} |S_R|} \tag{4.15}$$

and 3 aggregated metrics for Guided Information Extraction:

$$q^{\text{GIE-EASY}} = \frac{\sum_{S \in D} \sum_{R \in S_R} q_R^{\text{GIE-EASY}}}{\sum_{S \in D} |S_R|} \tag{4.16}$$

$$q^{\text{GIE-MEDIUM}} = \frac{\sum_{S \in D} \sum_{R \in S_R} q_R^{\text{GIE-MEDIUM}}}{\sum_{S \in D} |S_R|} \tag{4.17}$$

$$q^{\text{GIE-HARD}} = \frac{\sum_{S \in D} \sum_{R \in S_R} q_R^{\text{GIE-HARD}}}{\sum_{S \in D} |S_R|} \tag{4.18}$$

For 0-shot methods, all 6 AdaIE evaluation schemes would be identical due to the model not using the training data. In that case, the EASY, MEDIUM, and HARD aggregated metric values would have the same values, since they would correspond to results from the exact same evaluation on the same AdaIE test set.

4.2.6 Data Balancing

We follow the ideas of [92] and perform heavy data pruning to balance the domain and label distribution. To ensure consistent data sizes in the multi-partition evaluation, we aim for balanced contribution from each dataset rather than balanced contribution from each type. Each constituent dataset contributes 1000 units to each train sub-split and 100 units to each test sub-split. Each unit consists of a text and a different schema. To get multiple different schemas from a constituent dataset with a single schema, we sample random subsets of that schema. To choose the texts, we repeatedly sample uniformly without replacement, until the text limit is reached. If the constituent dataset runs out of texts before reaching the target number of samples, we iterate through the dataset again, as the same text paired with a different schema is still a valid different unit.

For a given constituent dataset, we select 100 random subsets of up to 20 entity types. The test-α and test-β sub-splits of the constituent dataset each contain 1 unit for each random subset of entity types. This is chosen after the relation types are partitioned into S_R^α and S_R^β as described in Section 4.2.4; that way the samples for the test-α sub-split use relation types selected only from S_R^α, and the samples for the test-β sub-split use relation types selected only from S_R^β. We also shuffle the schemas to implement the class order shuffling regularization technique proposed by [66]. The result of all of this is that schemas are almost never repeated in the test data; instead there is mostly only partial overlap.

Next, the train samples mirror the schemas from the test samples. For each sample in the sub-split test-α, we randomly select 5 train samples to be filtered down to the

same schema as the test sample and place them in train-α. We repeat this with test-β to generate train-β. When that training sub-split is included, a model can leverage those 5 train samples as in-context examples to understand the schema used by the test sample.

Subset	Texts	Entities	Relations
Raw Data	38,869	288,740	243,429
train	24,655	185,919	164,081
test	14,214	102,821	79,348
Sampled	13,000	200,061.7	50,890.7
train	10,000	143,526.7	35,069.7
test	3,000	56,534	15,821

Table 4.5: Dataset statistics for various subsets of AdaIE before and after downsampling. Sampled train subset statistics are averaged across the training sets of all 6 evaluation schemes.

4.2.7 Inter-Annotator Agreement

To estimate human performance at extraction on AdaIE, we estimate inter-annotator agreement (IAA) values for GEE, GRE, and GIE for each constituent dataset and take the mean. This is imprecise, as each dataset measures IAA differently, on different tasks using different metrics. Nevertheless, we obtain a unitless IAA of 82.1% for GEE, 76.9% for GRE, and 65.5% for GIE.

4.3 Experiments

We evaluate a variety of baselines on GEE and GIE. GRE is evaluated for models which can make use of provided ground-truth entities, otherwise its evaluation is skipped. Prompt templates for all baselines can be found in Appendix B.1.

4.3.1 Chunking

For all approaches, we follow the same chunking scheme to split documents into roughly paragraph-sized chunks. After using spaCy to perform sentence segmentation, we create

chunks of up to 2048 characters with up to 256 characters of overlap with the previous chunk. We break our chunks at sentence boundaries whenever possible, using sentence bounds determined by SpaCy.

For staged extraction approaches that perform GEE followed by GRE, unification is performed after each stage. Entities are unified by merging entities for which at least one mention is exactly the same; upon merging, the types are combined. This aggregates the entity information from the entire document before passing the entities as input to the following GRE stage. Then, extracted relations are unified by simply taking the de-duplicated set union of all relations across chunks.

4.3.2 Retrieval

In this work, we restrict our investigation to cases where all of the in-context examples must follow the current schema. Without this assumption, each in-context example must be paired with its own schema, leading to model confusion and a substantial increase in token count and evaluation cost. To enforce the assumption, when retrieving in-context examples, we do so from the pool of candidates that have been labelled for a superset of the current schema; this matches how it is done in Code4UIE [23].

To truly simulate low-resource schema following, AdaIE is designed to make retrieval challenging by making this pool of candidates extremely small. Sometimes there are fewer candidates than the targeted number of in-context examples; in those cases the missing examples are simply not displayed. This occurs frequently in hard evaluation settings, where an unseen type does not exist anywhere in the labels, thus it cannot fetch a single example and falls back to 0-shot prompting.

To unify the retrieval setup, we follow Code4UIE and index all texts from the training data using MPNet mean-pooled embeddings [23, 71]. On top of this, we incorporate chunking by splitting the training data into chunks and indexing the chunks separately. To retrieve, we embed the query text via the same method and perform a nearest-neighbour search, allowing multiple chunks of the same training unit to be retrieved. The retrieved examples are finally presented in decreasing order of relevance.

4.3.3 JSON-Based Prompting

Prompting an LLM to perform 0-shot extraction is the fastest way to set up IE, therefore it raises the question of whether this achieves satisfactory performance at GIE. Support for

structured output across all major LLM API providers (e.g., Anthropic, Google, OpenAI) as well as LLM serving engines (vLLM via `lm-format-enforcer`[1]) makes it natural to use a JSON-based output format. We set this up as a default baseline, and call it "JSON-based prompting".

This baseline prompts the LLM in two stages: GEE and GRE, with long texts split into chunks and unified after each stage. The first prompt (see Appendix B.1.1) provides the instructions for GEE, an optional description of the output format (as a JSON schema), the entity types, and the text. Using spaCy, we separate the text into a sequence of sentences and present that sequence in JSON format. The prompts are formatted in this order to maximize the benefit of prefix caching, as repeated prompts using the same schema can share the same prefix up to but not including the text. The LLM then generates a JSON-formatted list of entity objects, each one consisting of one or more entity mentions. The output format requires the mentions to include a list of sentence indices grounding the mention to the specific sentences where it occurs. Although this approach still uses string matching, the sentence indices can narrow down the search to only the sentence identified by the model, making it less likely to extract other spurious occurrences of the same string. The second prompt (see Appendix B.1.1) provides the instructions for relation extraction, an optional description of the output format (as a JSON schema), the entity types, the relation types, the text, and the entities (in JSON form). Again, the order is optimized to maximize prefix caching. The LLM then generates a JSON-formatted list of relation objects, which refer to the entities using the entities' string identifiers. Altogether, JSON-based prompting is capable of all three tasks: GEE is performed using the first prompt only, GRE is performed by skipping the first prompt and injecting the ground-truth entities directly into the second prompt, and GIE is performed by using the entities from the first prompt as the input to the second prompt.

We find an interesting failure case, even in frontier models, in which the same entity or relation is sometimes repeatedly generated until the model reaches the end of its token limit, at which point the partial malformed output usually cannot be parsed – invalidating the extractions for that text. A practical limit of 4096 output tokens is imposed to limit the cost of such an overrun.

For each commercial LLM provider we evaluate two models – an older model, and a more recent model, usually from the same (relatively small) series. From Anthropic, we choose Claude Haiku 3.5 [3] and Claude Haiku 4.5 [4]. From Google, we choose Gemini 2.0 Flash [5] and Gemini 2.5 Flash [6]. From OpenAI, we choose GPT-4.1 [1] and GPT-5 mini [2]. This allows us to analyze whether LLMs are improving at this prompting approach.

[1] https://github.com/noamgat/lm-format-enforcer

Meanwhile, to study the effects of model size, we use the 8B and 70B variants of Llama 3.1 Instruct. In total, this accounts for 8 models run with JSON-based prompting.

4.3.4 InstructUIE

InstructUIE [78] is a checkpoint of Flan-T5-XXL (11B parameters) with multi-task finetuning for a variety of information extraction tasks including named entity recognition (NER) and event extraction (EE). We translate GEE to InstructUIE's NER task format (see Appendix B.1.2), and translate GIE to InstructUIE's EE task format with slight adjustments to acknowledge the "relation" terminology as opposed to events (see Appendix B.1.2). EE cannot be conditioned on existing entities, so GRE is not evaluated; only GIE can be evaluated. In both tasks, entities are treated as strings, so coreference resolution is not possible. Moreover, this means that during relation extraction, the extracted entities do not have types; we handle this by assuming that the entity simultaneously belongs to every entity type.

We evaluate the model both as "InstructUIE pretrained" without finetuning, and as "InstructUIE finetuned" produced by continued finetuning on AdaIE. The finetuning is performed six times – once for each of AdaIE's evaluation schemes – producing six model checkpoints. For each evaluation scheme, the finetuning data consists of all training data in that evaluation scheme, chunked, and formatted in the InstructUIE formats for GEE and GIE. Finetuning involves a single epoch through the training data, using the same hyperparameters as the original InstructUIE finetuning run wherever possible. Finally, after finetuning, we evaluate each checkpoint on both GEE and GIE on the entire AdaIE test set.

Compared to the other approaches evaluated, InstructUIE is extremely compact in its prompt format, which is necessary given its relatively small context window. However, besides relying only on type names, a fundamental weakness of the prompt format is that it combines all argument names into a single list of strings that can be applied to all relation types. Because of this, the model does not have the necessary information to extract the correct arguments of a relation, leading to frequent failures to follow the schemas.

4.3.5 Code Generation

All three of the following baselines – Code4UIE [23], GoLLIE [66], and KnowCoder [45] – are UIE methods that frame IE as a Python code generation task. These methods start their prompts by defining or importing base `Entity`, `Relation`, or `Event` classes depending

on the IE subtask they are performing. Each entity, relation, or event type of interest is then defined (or imported) as a subtype of these classes. Relations and events define their arguments as fields of the class – this naturally allows for higher-order relations by adding more fields. Finally, comments are used to supply the text to be extracted and other instructions. The model then completes the code by generating more hopefully-valid Python in which the classes are instantiated.

To adapt these methods for AdaIE, we alter the `Relation` class to reflect the GIE features of higher-order relations and specific entity types per argument. Generally, this is done by merging in the characteristics of the `Event` type. The original works for these baselines do not specify how to condition relation extraction on existing entities. Although it would be easy to imagine ways to define existing entities in Python code, we treat these models as only being able to perform GIE in a single shot, so we do not evaluate GRE. Likewise, the works assume an entity format where the entity matches a specific string, making it impossible to do coreference resolution in these baselines, even though it is easy to imagine extensions where entities may have multiple mentions.

Meanwhile, the practical question of how to actually interpret the code is generally left unanswered by these prior works. For standard evaluation, we devise a technique of evaluating the code directly in Python and converting the instantiated objects into entities and relations. This reveals a disappointing observation – the models frequently fail to complete the prompts with code that could actually run without raising exceptions. To handle some common failures with leniency, we produce two versions of the prompt. The first version follows the template of the original method as faithfully as possible; it is supplied to the model, but is never executed. The second version expands the definitions of imported classes and defines modified function signatures that sanitize and accept various malformed inputs; it is prepended to the model's output before executing the code. Modified signatures can allow methods to accept constructor calls with too few or too many arguments, or incorrect types such as passing a single entity when a list of entities is expected. Even if the resulting entity or relation instance is invalid, the call is skipped without crashing the rest of the code. However, if the model completely fails to generate syntactically valid Python, then no amount of prefixing can rescue the generation, and the entire extraction is considered lost. One example of an unrecoverable error is placing keyword arguments ("kwargs") before positional arguments ("args").

When there is an entity or event type hierarchy, methods such as Code4UIE define a class hierarchy to mirror this structure. The methods do not handle any data where a single type may be the subtype of multiple non-overlapping supertypes, but we assume that if the methods were to be applied to such data, then this would be done through multiple inheritance of Python classes. We set up our execution context to automatically

resolve instances of entities to all of their supertypes.

Code4UIE

Code4UIE [23] is a retrieval-based code generation method that uses MPNet to retrieve up to 10 in-context examples to form a code completion prompt for an LLM. We reproduce Code4UIE in spirit, as there are many practical challenges against using it in its original form. Foremost, the original formulation uses OpenAI's `gpt-3.5-turbo-16k` model as its base LLM, which is far from the optimal base model to be used in 2026; we upgrade to GPT-4.1 and GPT-5 mini in our evaluations – the same two OpenAI models used to evaluate JSON-based prompting. Moreover, there is no prompt used by Code4UIE that exactly matches the GIE task; thus our higher-order relations must be modelled as events without triggers.

Code4UIE performs entity extraction in a single prompt. Meanwhile, the closest task to GIE performed by Code4UIE is event extraction (EE), which it handles in a pair of prompts. The first prompt shows the definitions of all event types and asks the model to identify which event types appear in the text; the model emits import statements for the relevant classes. The second prompt leaves only the definitions of the identified classes, while extracting the events as instances of those event types. Interestingly, both prompts use the same retrieved examples, regardless of the first prompt's result. Thus, it is very common to end up with a prompt in which the examples deal with undefined classes. This issue is one major impetus for using different Python code for the prompt versus the evaluation context.

GoLLIE

GoLLIE is a finetuned version of Code Llama focused on the use of guidelines presented as Python code comments and docstrings [66]. It is published with multiple sizes (7B, 13B, and 34B), all fine-tuned on a collection of 9 IE datasets, mostly for entity extraction. There is no overlap between those 9 datasets and AdaIE, so the risk of data leakage is low. Although the 7B variant is officially the one referred to as "GoLLIE" by name, in these experiments we exclusively use the 34B model which has the best reported performance; we call that checkpoint "GoLLIE pretrained". We also produce "GoLLIE finetuned" by continuing instruction tuning on AdaIE, using prompts and responses formatted according to GoLLIE's prompt style with minor adaptations to convert events to relations by removing the trigger (see Appendix B.1.4). Like InstructUIE, this training setup mixes GEE- and GIE-formatted texts.

Also like InstructUIE, GoLLIE's relation extraction format constructs objects where the arguments are strings rather than `Entity` objects. This makes it impossible for GoLLIE to jointly extract entity types and relation types at the same time; similar to InstructUIE, we handle this by treating any type-less string as an entity simultaneously belonging to all entity types.

KnowCoder

Finally, KnowCoder is a checkpoint of Llama 2 7B finetuned on a large silver-standard (thus not overlapping with AdaIE) dataset automatically constructed from WikiData, followed by a "refinement" stage on a large collection of human-annotated data with various schemas [45]. We call this refined checkpoint "KnowCoder pretrained". Using roughly the same finetuning setup as the refinement process, we continue instruction tuning the model on AdaIE, formatted according to KnowCoder's prompt style with minor adaptations (see Appendix B.1.5), to produce "KnowCoder finetuned". Like InstructUIE and GoLLIE, this training setup mixes GEE- and GIE-formatted texts.

We then evaluate the pretrained and finetuned models in two ways: 0-shot and 5-shot. The intended evaluation setting of KnowCoder is 0-shot, but [45] also contains a 5-shot evaluation performed on the model prior to refinement, which we aim to replicate on the refined and finetuned models in our experiments. Our finetuning process matches the refinement process, which means that the instruction tuning samples have no in-context examples. The presence of few-shot inputs during training is important to the performance of the tuned model at handling few-shot prompts during inference time [49], so this 5-shot evaluation is not anticipated to perform well.

4.3.6 Results

We separate the results into Tables 4.6 and 4.7. Table 4.6 shows evaluations at the hard difficulty, which can be measured for every method. Table 4.7 shows difficulty-tiered results for methods which actually make use of training data.

Regarding the question of whether a quick guided information extraction implementation using frontier LLMs can achieve satisfactory performance with no special tuning, we find a surprising answer. Across both tables, the highest performance is achieved by the JSON-based prompting baselines, outperforming all other baselines even with fine-tuning or in-context examples. JSON-based prompting is most successful with leading commercial LLMs such as GPT-5 mini (54.5% hard GEE) and Gemini 2.5 Flash (51.0% hard GRE,

Method	GEE-HARD ↑	GRE-HARD ↑	GIE-HARD ↑
JSON-based prompting			
Claude Haiku 3.5	40.5	39.8	24.4
Claude Haiku 4.5	47.6	45.6	31.9
Gemini 2.0 Flash	44.4	34.9	24.9
Gemini 2.5 Flash	53.9	**51.0**	**34.0**
GPT-4.1	38.4	37.3	21.5
GPT-5 mini	**54.5**	50.3	33.5
Llama 3.1 8B Instruct	26.8	18.9	8.3
Llama 3.1 70B Instruct	36.2	28.6	15.8
UIE without guidelines			
InstructUIE pretrained	26.6	–	0.0
InstructUIE finetuned	30.9	–	3.2
GPT-5 mini + InstructUIE prompt	31.9	–	0.0
UIE with guidelines			
Code4UIE (GPT-4.1), 10-shot	41.8	–	17.1
Code4UIE (GPT-5 mini), 10-shot	51.4	–	22.3
GoLLIE 34B pretrained	35.8	–	18.7
GoLLIE 34B finetuned	35.7	–	17.1
GPT-5 mini + GoLLIE prompt	51.0	–	22.3
KnowCoder pretrained, 0-shot	28.6	–	5.3
KnowCoder pretrained, 5-shot	22.6	–	5.3
KnowCoder finetuned, 0-shot	28.6	–	5.3
KnowCoder finetuned, 5-shot	22.6	–	5.3
GPT-5 mini + KnowCoder prompt, 0-shot	51.0	–	22.2
GPT-5 mini + KnowCoder prompt, 5-shot	50.9	–	22.2
inter-annotator agreement	*82.1*	*76.9*	*65.5*

Table 4.6: Evaluation scores of all baselines in the study at the hard difficulty (i.e., entity and relation types where there is no available data for fine-tuning or in-context examples).

34.0% hard GIE), indicated in bold in Table 4.6. Those scores exceed any other method's performance even on easy-difficulty types with finetuning or in-context examples.

subtask	**GEE↑**		**GIE↑**		
difficulty	EASY	HARD	EASY	MEDIUM	HARD
UIE without guidelines					
InstructUIE pretrained	*26.6*			*0.0*	
InstructUIE finetuned	38.9	30.9	5.2	5.1	3.2
UIE with guidelines					
Code4UIE (GPT-4.1) 10-shot	45.9	41.8	18.2	16.6	17.1
Code4UIE (GPT-5 mini) 10-shot	53.4	51.4	23.1	22.1	22.3
GoLLIE 34B pretrained	*35.8*			*18.7*	
GoLLIE 34B finetuned	39.0	35.7	22.2	20.3	17.1
KnowCoder pretrained, 0-shot	*28.6*			*5.3*	
KnowCoder pretrained, 5-shot	22.6	28.6	5.2	4.9	5.3
KnowCoder finetuned, 0-shot	28.6	28.6	5.3	4.9	5.3
KnowCoder finetuned, 5-shot	22.6	28.6	5.2	4.9	5.3
GPT-5 mini + KnowCoder prompt, 0-shot	*51.0*			*22.2*	
GPT-5 mini + KnowCoder prompt, 5-shot	50.9	51.0	25.4	22.1	22.2
inter-annotator agreement	*82.1*			*65.5*	

Table 4.7: Evaluation scores of the trained or few-shot UIE baselines in the study, broken down by difficulty. If available, base (pretrained, 0-shot) methods without fitting are provided for reference in italics.

Beyond JSON-based prompting, the next most successful method is Code4UIE, which also makes use of commercial LLMs. Interestingly, despite incorporating up to 10 in-context examples, Code4UIE is usually no better than 0-shot JSON-based prompting of the same model. Using GPT-4.1, Code4UIE achieves easy and hard GEE scores surpassing JSON-based prompting, but it is still worse at GIE. With GPT-5 mini, Code4UIE is worse than JSON-based prompting at both tasks at all difficulty levels.

With non-commercial LLMs, the next most successful method is GoLLIE. The pre-trained GoLLIE model with no further fine-tuning achieves 35.8% hard GEE score and 18.7% hard GIE score. After fine-tuning, its performance increases on easy-difficulty and medium-difficulty types, at the cost of a slight decrease in performance on hard-difficulty types. Indeed, finetuning generally has mixed effectiveness. For InstructUIE, finetuning is essential. The method's entity extraction prompt is fully compatible with AdaIE, giving 26.6% GEE score with no finetuning, but finetuning on AdaIE yields +12.3% GEE score for entity types seen during training and +4.3% score for entity types unseen during training. On the other hand, InstructUIE's slightly adjusted relation extraction prompt is impossible to follow without finetuning. Both the base InstructUIE model and GPT-5 mini are unable to follow the output format, resulting in 0.0% GIE scores. Even after finetuning, relation extraction lacks guidelines and is still underspecified, with all argument names provided in a single list with no distinction of which arguments belong to which relation types. Thus, InstructUIE can gain no more than 5.2% GIE performance even with finetuning.

Meanwhile, KnowCoder has generally poor performance which is not helped at all by conditioning on training data. It achieves a similar performance to finetuned InstructUIE on GIE, regardless of finetuning or the number of in-context examples. Rather, in 5-shot evaluation, the GEE performance of KnowCoder suffers a −6.0% decline on entity types for which examples are available. We hypothesize that this occurs due to the relatively small 4,096 token context window limit of the model, which is rapidly saturated by in-context examples. In both the original training setup of KnowCoder as well as our finetuning setup, prompts assume zero in-context examples, causing longer example-heavy prompts at inference time to be out of distribution.

Much more effective than finetuning is to simply replace the model with a more powerful commercial LLM. For GoLLIE and KnowCoder, running the exact same prompt through GPT-5 mini yields significantly stronger results, including GEE score improvements of +15.2% and +22.3% respectively, and GIE score improvements of +6.5% and +16.9% respectively. For both methods, and Code4UIE too, this results in a very similar score around 51% hard GEE and 22% hard GIE – indicating that variations in code generation and prompt are not important for coding-based information extraction. Meanwhile, JSON-

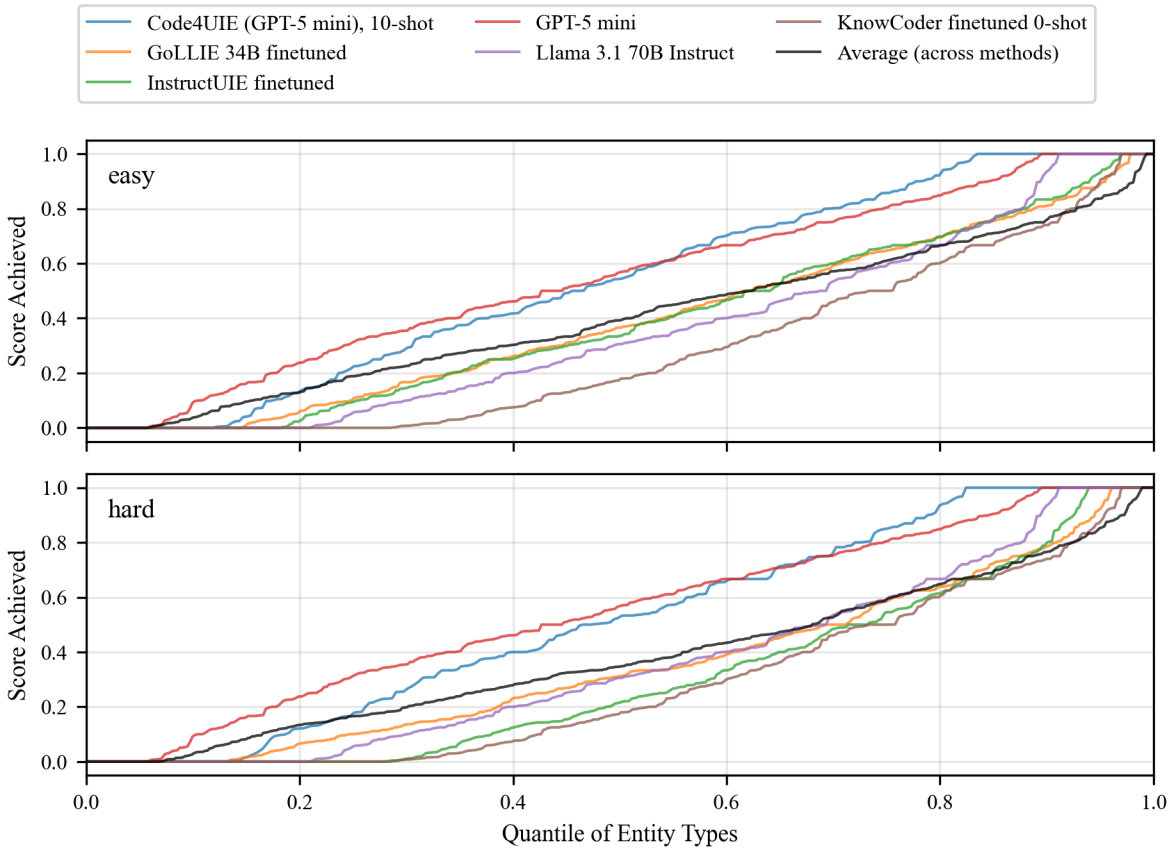

Figure 4.1: The entity types arranged in order of increasing GEE score, versus the GEE score achieved by each model at that quantile.

based prompting surpasses the performance of all of these coding-based prompts, especially in GIE (+11.2%), challenging the narrative that coding is complementary to information extraction.

Breaking down the results per entity and relation type, we plot cumulative distribution curves to visualize the performance of each model across types, in Figures 4.1, 4.2, and 4.3. For each method, we order the types by increasing score, such that any given point (x, y) along a curve means that a proportion x of types achieve a score of at most y. The area below each curve is the method's macro-averaged metric for that task and difficulty level. The black average line considers the performance of all methods averaged per type; it reveals that there exist some types for which the average performance is 0%. On those types (about 10% of entity types and 20% of relation types), no method makes a single

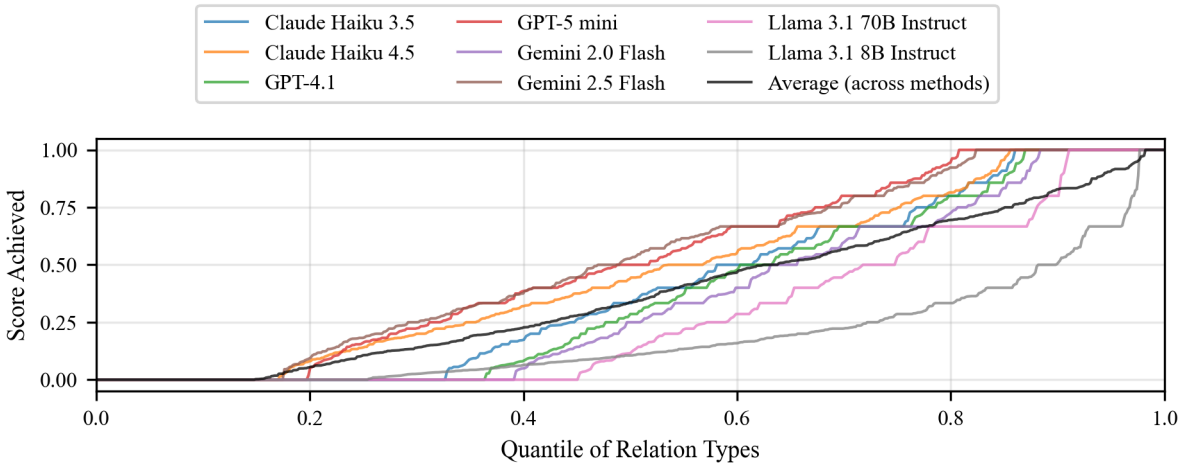

Figure 4.2: The relation types arranged in order of increasing GRE score, versus the GRE score achieved by each model at that quantile.

correct extraction. At the other extreme, most methods are able to achieve perfect scores on at least some types. Generally, the strongest methods achieve perfection on the most types and zero on the fewest types, with their quantile curves shifted to the left compared to other methods. In between the zero and perfect extremes, most methods follow a uniform distribution of scores across the entity types, characterized by the linear slopes in Figure 4.1. Meanwhile, the distribution of GIE scores across relation types is skewed right – that is, the majority of scores between zero and perfect are less than 0.5, characterized by the upward curvature in Figure 4.3. We show some examples of the easiest and hardest types in Tables 4.8 and 4.9. Notably, all of the best performing relation types are directed binary relations – that is, they have two arguments which both expect exactly one entity each.

We also aggregate the scores across each of the 15 constituent datasets of AdaIE in Table 4.10, which reveals wide variations in performance between datasets. This roughly indicates the relative difficulty of each dataset. SciREX stands out with the lowest entity extraction scores, less than 2%. This is not too surprising, as SciREX is expected to be challenging with full-length academic papers requiring context windows much longer than the chunk size used in our experiments. This also makes it one of the most difficult datasets for GIE, with about the same difficulty as BioRED. BioRED involves shorter texts, but our augmentations create a high number of almost-identical relation types differentiated by "novelty" and the types of entities involved, making accurate extraction difficult. Worse than both of these datasets, WikiEvents is the most difficult GIE dataset with around 3.2% score. This stems from the dataset's more complex event types as opposed to most

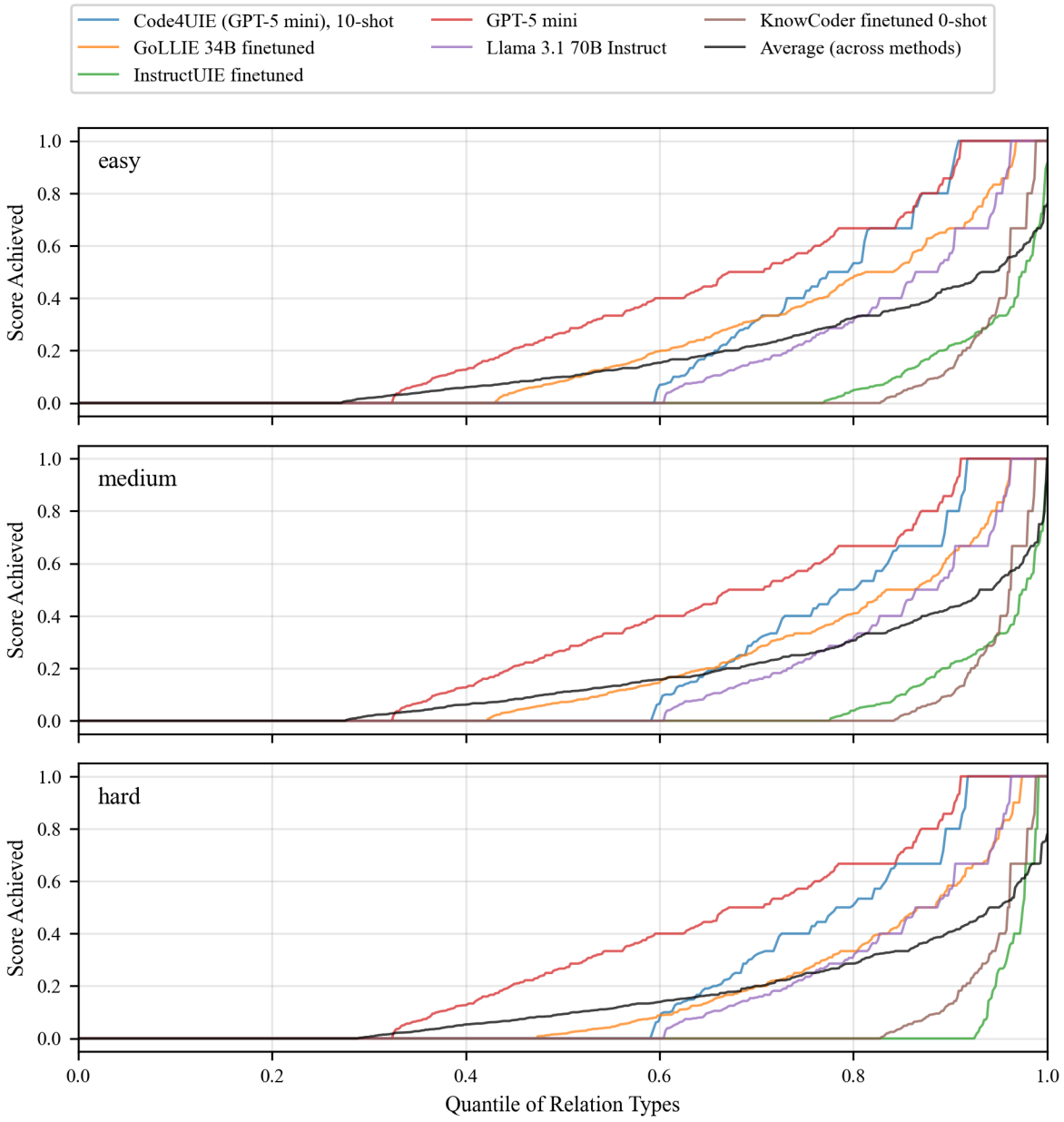

Figure 4.3: The relation types arranged in order of increasing GIE score, versus the GIE score achieved by each model at that quantile.

Score	Dataset/Type	Verbalization
Best performing entity types		
100.0	WebNLG/`broadcaster`	`{broadcaster}` is a television channel or an organisation that is involved in television broadcasting, referred to by full or abbreviated proper noun name.
97.7	HyperRED/`electoral_district`	`{electoral_district}` is an electoral district, referred to by proper noun name, including qualifiers like "district" or "constituency".
83.4	CrossRE/`programming_language`	`{programming_language}` is a programming language, such as Java and Python.
81.8	ie4wills/`county`	`{county}` is a county in the United States of America, referred to by name, including qualifiers like "County" if present, but excluding the state.
80.0	FIRE/`money`	`{money}` is a quantity and unit of money, including currency symbols like $.
Worst performing entity types		
0.4	SciREX/`method`	`{method}` is a computational method or group of methods (e.g., algorithms, architectures, models) that is actually evaluated in the paper, referred to by generic or specific name, including abbreviations and section headers.
0.0	DWIE/`war`	`{war}` is an armed conflict between nations (including conventional wars as well as the Cold War), referred to by common or official name (full or abbreviated).
0.0	Re-DocRED/`sports_player`	`{sports_player}` is or was a sports player, referred to by a full or partial name.
0.0	WikiEvents/`information`	`{information}` is an informational artifact, including digital or physical media, computer programs, intellectual property, fields of study, ideas, thoughts, opinions, beliefs, facts, data.
0.0	ie4wills/`trustee`	`{trustee}` is a person who manages a trust, referred to by name, placeholder (e.g., "[Person-2]") or pronoun, including possessive pronouns.

Table 4.8: Selected examples of the best and worst entity types in AdaIE based on average HARD-difficulty GEE scores across all methods. Open types use recall as their score, while closed types use F_1 as their score.

Score	Dataset/Type	Verbalization
Best performing relation types		
70.8	WebNLG/`hub_airport`	`{airline}` operates out of the hub airport `{airport}`.
51.3	CROSSRE/`developed`	`{researcher}` developed or co-developed `{idea}` (an algorithm, scientific theory, or product).
45.8	DWIE/`directed_by`	`{film}` was directed by `{director}`.
45.2	ie4wills/`witness_will`	`{witness}` observes the signing of `{will}` and attests to its proper execution.
44.3	FIRE/`subsidiary_of`	`{subsidiary}` is a current subsidiary or brand of `{parent}`, the parent company that owns it (wholly or majority), excluding subsidiaries that are currently being sold by `{parent}`.
Worst performing relation types		
0.0	CROSSRE/`record_label`	The individual musical artist `{artist}` has works published under the record label `{label}`.
0.0	HyperRED/`connecting_line`	`{station}` is or was a station on the `{line}` railway line, since `{start_time}` and until `{end_time}`, next to the station(s) `{adjacent_station}`.
0.0	KnowledgeNet/`ceo_of`	`{ceo}` is or was a Chief Executive Officer (CEO) of `{organization}`.
0.0	Re-DocRED/`sister_city`	`{city}` and `{city}` are twin towns, sister cities, twinned municipalities, or any other pair of localities that have a partnership or cooperative agreement, either legally or informally acknowledged by their governments.
0.0	WIKIEVENTS/`prevent_passage`	`{preventer}` prevents `{transporter}` from entering `{destination}` place from `{origin}` place to transport `{occupant}` using `{vehicle}` vehicle.

Table 4.9: Selected examples of the best and worst relation types in ADAIE based on average HARD-difficulty GIE scores across all methods. Open types use recall as their score, while closed types use F_1 as their score.

Dataset	GEE		GIE		
	EASY	HARD	EASY	MEDIUM	HARD
BioRED	45.9	40.7	5.8	5.9	5.6
CrossRE	55.8	52.3	19.0	22.3	20.4
DEFT	42.5	30.7	35.0	32.7	25.3
DWIE	21.9	21.9	11.6	11.0	10.4
DialogRE	48.2	47.4	7.2	6.9	6.1
FIRE	59.1	51.0	30.4	25.2	22.6
HyperRED	55.8	58.3	20.1	17.9	18.4
ie4wills	44.9	40.3	24.1	21.8	19.5
KnowledgeNet	38.9	39.5	16.3	14.8	13.6
Re-DocRED	39.7	40.6	14.1	15.2	14.7
SciERC	43.9	39.0	8.6	7.9	7.7
SciREX	1.9	1.6	5.9	7.1	6.8
SoMeSci	15.6	13.4	9.2	10.0	9.0
WebNLG	64.8	67.1	35.7	35.1	35.3
WikiEvents	9.1	8.3	3.2	3.3	3.2
AdaIE	39.1	38.3	15.1	14.7	14.5

Table 4.10: Mean metrics for the entity and relation types of each dataset, aggregated across all baselines evaluated in this study.

other datasets which are derived from mostly binary relations. Meanwhile, the easiest dataset for both GEE and GRE is WebNLG, which has terse knowledge graph verbalizations as its texts, which rarely contain non-essential information. Across the datasets, we see varying effects of fitting on available data. On most datasets the gains are minimal, or sometimes slightly negative. Meanwhile, for DEFT and FIRE, the presence of in-context examples or finetuning data creates large gains in the model's performance at the easy difficulty compared to hard difficulties. Those datasets cover entity and relation types that are outside the standard domains of information extraction (DEFT is for extracting relations between terms and definitions, and FIRE extracts financial relations which may include monetary amounts), suggesting that these datasets have the greatest opportunity to make performance gains, and that future expansions to AdaIE should consider collecting more datasets of this kind.

provider	time interval	GEE-HARD	GRE-HARD	GIE-HARD
Anthropic	358 days[3, 4]	+7.1	+5.8	+7.5
Google	132 days[5, 6]	+9.5	+16.1	+9.1
OpenAI	115 days[1, 2]	+16.1	+13.0	+12.0

Table 4.11: The improvement in performance of the JSON-based prompting baseline using different models released by each commercial provider, compared to the difference in time between the two models.

4.3.7 Discussion

The differences in scores between the two versions of each commercial LLM evaluated in these experiments are summarized in Table 4.11. The rapid increase in general reasoning, coding, and generation abilities result in improvements in AdaIE performance averaging to double-digit percentage points per year. Meanwhile, the jump in performance from the 8 billion parameter to the 70 billion parameter instruct variants of Llama 3.1 is also almost +10%. Similarly, GoLLIE and KnowCoder are functionally extremely similar, yet the former (34 billion parameters) achieves much greater performance than the latter (7 billion parameters). Thus, reasoning ability and scale appear to be the main drivers of performance at information extraction on AdaIE.

In comparison, fine-tuning and in-context examples have a much smaller effect – even possibly negative – when employed carelessly as in the case of KnowCoder. While there are modest gains on GoLLIE from finetuning, they are completely overshadowed by the much larger gain from simply replacing the specialized model with a commercial LLM. The only case where finetuning definitely matters is in guideline-free methods like InstructUIE where there is insufficient information in the prompt to specify the task.

If one were to rank the UIE methods from worst to best performance on AdaIE, the list would be KnowCoder, InstructUIE, GoLLIE, and finally Code4UIE, with the caveat that Code4UIE makes use of more powerful commercial LLMs than were available when it was first published. However, if all models are allowed to use those same LLMs, then a fairer comparison renders the ranking to be InstructUIE as the worst, and then Code4UIE, GoLLIE, and KnowCoder together with roughly equal performance, confirming the importance of proper schema guidelines.

There remains a performance gap of over 25% between the best GEE, GRE, and GIE metrics and the "human" performance levels estimated through inter-annotator agreement (IAA). There is no reason to expect perfectly linear progress, as the apparently consistent

progress is partly due to discrete qualitative improvements such as the addition of reasoning in GPT-5. Nevertheless, if the trend of model improvement does continue linearly until saturation, then one may expect to reach IAA within a few years. During those years, it should be extremely difficult to maintain state-of-the-art performance on AdaIE without either directly applying tremendous computational resources, or indirectly incorporating them via extremely large foundation models. These results emphasize that, if being state-of-the-art is the goal, then one should focus on methods that can be agnostic to the underlying foundation model, rather than pursuing the creation of specialized models whose extraction abilities may be soon subsumed.

In practice, however, achieving state-of-the-art metrics is not the only goal. Larger models carry far higher computational costs than their specialized counterparts, and reliance on commercial models raises concerns about privacy, data ownership, accessibility, and reproducibility. In this work, experiments based on GPT-5 mini are not fully reproducible, as the models must be fixed at a temperature of 1.0, and the model will likely eventually be deprecated like many of its predecessors. Even in this work, cost constraints implicitly limit baseline design. Besides avoiding larger or premium versions of commercial models (e.g., Claude Opus 4.5, Gemini 2.5, GPT-5), we also avoid more factorized extraction methods that might break the schema down into single types to be extracted in parallel. Even assuming batching and prefix caching, the current cost structure of commercial LLM providers still discourages the repetition of tokens in favour of using fewer prompts. Since the primary focus of this chapter is on evaluation rather than chasing the state-of-the-art, we leave the further optimization of GIE techniques to future researchers.

Overall, these experimental results challenge many conventional narratives in information extraction. First, a frequent justification of end-to-end methods is that pipelined approaches, where entity extraction and relation extraction are separate steps, are vulnerable to error accumulation [34]. Yet, a pipelined approach (JSON-based prompting) surpasses every end-to-end method in these experiments. Second, the belief that code understanding is more complementary to information extraction than natural language [42] needs re-examination under mixed regimes where natural language is combined with serial data formats such as JSON. This is especially true as we observe custom coding-based UIE models sometimes failing to emit valid Python code, either through hallucinations or outright syntax errors. Third, by putting a number of past code generation baselines (Code4UIE, GoLLIE, KnowCoder) on equal footing using the same base model (GPT-5 mini), which form a progression of state-of-the-arts in UIE literature, we reveal that their prompts are actually roughly equivalent, and that their apparent performance differences stem only from the strength of their underlying models. Yet, a general purpose base model (GPT-5 mini) proves itself far more capable at handling specialized information extraction

prompts 0-shot, than the models explicitly trained to handle those prompts.

4.4 Summary

We define guided information extraction (GIE), a refinement of universal information extraction (UIE) that is conditioned on specific sentence-level descriptions of the entities and relations to extract, required for specifying custom information needs. To standardize the evaluation of techniques leveraging guidelines, and to encourage their development, we build AdaIE, the first GIE dataset, by compiling documents, entity types, and relation types from 15 datasets. We manually annotate a large number of reasoning-based schema augmentations to increase the number of types in our schema, building the largest collection of gold-standard schemas. Next, we use the dataset to conduct evaluations to answer the research question posed at the beginning of the chapter: *For a given dataset of schemas and labelled texts, what is the best method that can be developed to do information extraction for arbitrary schemas?* The evaluations cover a number of UIE baselines, alongside a pipelined JSON-based 0-shot prompting baseline. Between the UIE baselines, the proper use of guidelines produces higher performance, but this performance gain is overshadowed by the fact that the JSON-based prompting baseline is, by far, the state-of-the-art technique on AdaIE by all measures, surpassing the specialized UIE baselines even with finetuning or in-context examples. Still, there remains a double-digit performance gap between this baseline and an estimate of human performance derived from inter-annotator agreement. The main drivers of improvement on AdaIE appear to be model capacity and reasoning ability, suggesting that much of the performance gap can be closed by further progress in foundation models.

Chapter 5

Conclusion

We opened with the question of how to achieve good automated personal knowledge graph construction, laying out 5 desiderata: **Low-Resource**, **Schema-Following**, **Grounding**, **Calibration**, **Multi-Modality**. We then divided information extraction (IE) research into two paradigms, the old paradigm where models are easily grounded and calibrated, and a new LLM-powered paradigm that accomplishes schema-following with low resources. The strengths of both paradigms are demanded by a good PKG construction system.

To achieve this, this thesis presents two works. Structural consistency, a robustness measure for information extraction, is shown to also have the effect of reducing the resources needed for training supervised information extraction models. Meanwhile, AdaIE is a large dataset for a newly defined Guided Information Extraction task in which grounding and schema-following are highly important.

5.1 Future Directions

We address the limitations of this work as a series of promising future directions.

Multi-Modality A key desideratum of a PKG system, multi-modality, remains unaddressed in this work. Nevertheless, AdaIE, a text-only benchmark, can be extended in the future to include more modalities such as images and videos, which are represented by tasks such as scene graph extraction and video event extraction. After all, as long as there are notions equivalent to entity mentions (i.e., bounding boxes), and well-defined schemas, a dataset is suited to undergo schema augmentation and thus can be added to AdaIE.

Consistency for Generative IE Most of the work in Chapter 3 was conducted before the rise of the LLM paradigm in information extraction, and thus focuses on transformations of text inputs. However, since the input space of GIE includes both schemas and text, one may consider evaluating and training new transformations:

- Permute the types in the representation of the schema given to the model, as explored in [66] but not in the context of consistency.
- Replace entity type names, relation type names, or argument names with synonyms.
- Paraphrase or translate the verbalizations of entity and relation types, as explored in [66] but not in the context of consistency.
- Split a schema into subsets. This is especially of interest due to the independence between types in AdaIE, making AdaIE one of the only datasets where it is actually well-founded to split apart the schema, extract each subset of types independently, then reunify the results.

Beyond this, it also remains possible to do concatenation and substitution, but AdaIE makes this difficult by having a different extraction schema per document.

Grounding for Relations A true fully-grounded information extraction dataset would have grounding for not just entities, but also relations. A grounded relation would indicate a span or set of spans in the text that form the minimal evidence needed by a "reasonable reader" to conclude that the relation is true. There is no good reason for leaving it out of AdaIE besides a limitation of resources. Among the relation extraction datasets sourced for AdaIE, only one – Re-DocRED [74] – contains evidence annotations for relations, as a set of sentences. For relations based on events, we could treat the event trigger as a weak form of grounding, however the link between an event's argument and the trigger could span multiple sentences [43], so this is not adequate. How to overcome this scarcity and source a sufficient set of diverse but consistent evidence annotations remains a question to be answered in future works.

Improving Performance on AdaIE This work does not aim to develop a state-of-the-art technique on AdaIE, although the JSON-based prompting baseline happens to achieve that. Its performance, although rapidly improving with stronger base models, is not yet at human level. Besides waiting for new models, future works should be able to directly improve further on that baseline through three avenues without swapping models. First,

as a zero-shot technique, the baseline does not at all tap into the available AdaIE training data. Few-shot prompting with retrieved in-context examples should produce a small improvement in extraction quality on types when the schema guidelines remain unclear to the model, though this is ineffective for types with zero annotations. Beyond this, one can consider finetuning on AdaIE training data, either using large recently-released open weights models, or through tuning APIs offered by commercial counterparts. Besides an improvement in performance, this can produce improvements in cost, as a large portion of the JSON-based prompt is used to explain the input and output formats at a level unnecessary for a finetuned model. Finally, different algorithms for chunking may yield better results; the chunk size is chosen to fit all baselines in the study (limited by the 4096 token windows of models like KnowCoder) and is much smaller than the context windows of more recent models. Larger chunks may be necessary to infer relations that require reasoning on long text passages.

References

[1] GPT-4.1 Model.

[2] GPT-5 mini Model.

[3] Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku, October 2024.

[4] Introducing Claude Haiku 4.5, October 2025.

[5] Gemini 2.0 Flash | Generative AI on Vertex AI, February 2026.

[6] Gemini 2.5 Flash | Generative AI on Vertex AI, February 2026.

[7] Fahmida Alam, Md Asiful Islam, Robert Vacareanu, and Mihai Surdeanu. Towards Realistic Few-Shot Relation Extraction: A New Meta Dataset and Evaluation. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16592–16606, Torino, Italia, May 2024. ELRA and ICCL.

[8] Krisztian Balog and Tom Kenter. Personal Knowledge Graphs: A Research Agenda. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, pages 217–220, New York, NY, USA, September 2019. Association for Computing Machinery.

[9] Elisa Bassignana, Viggo Unmack Gascou, Frida Nøhr Laustsen, Gustav Kristensen, Marie Haahr Petersen, Rob van der Goot, and Barbara Plank. How to Encode Domain Information in Relation Classification. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors,

Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 8301–8306, Torino, Italia, May 2024. ELRA and ICCL.

[10] Elisa Bassignana and Barbara Plank. CrossRE: A Cross-Domain Dataset for Relation Extraction. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3592–3604, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[11] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3613–3618, Hong Kong, China, 2019. Association for Computational Linguistics.

[12] Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task: Overview and Evaluation Results (WebNLG+ 2020). In Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina, editors, *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual), December 2020. Association for Computational Linguistics.

[13] Yew Ken Chia, Lidong Bing, Sharifah Mahani Aljunied, Luo Si, and Soujanya Poria. A Dataset for Hyper-Relational Extraction and a Cube-Filling Approach. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10114–10133, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[14] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Fang Zeng, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. AugGPT: Leveraging ChatGPT for Text Data Augmentation. *IEEE Transactions on Big Data*, 11(03):907–918, June 2025.

[15] Qicai Dai, Wenzhong Yang, Liejun Wang, Fuyuan Wei, and Meimei Tuo. SOIRP: Subject-Object Interaction and Reasoning Path based joint relational triple extraction by table filling. *Neurocomputing*, 580:127492, May 2024.

[16] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. In Maria Teresa Lino, Maria Francisca Xavier, Fátima Ferreira, Rute Costa, and Raquel Silva, editors, *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

[17] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization, February 2025. arXiv:2404.16130 [cs].

[18] Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. *Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16)*, 1695:13–16, September 2016. Place: Aachen.

[19] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, December 2008.

[20] Steven Y. Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. GenAug: Data Augmentation for Finetuning Text Generators. In Eneko Agirre, Marianna Apidianaki, and Ivan Vulić, editors, *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42, Online, November 2020. Association for Computational Linguistics.

[21] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating Training Corpora for NLG Micro-Planners. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[22] Honghao Gui, Lin Yuan, Hongbin Ye, Ningyu Zhang, Mengshu Sun, Lei Liang, and Huajun Chen. IEPile: Unearthing Large Scale Schema-Conditioned Information Extraction Corpus. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 127–146, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[23] Yucan Guo, Zixuan Li, Xiaolong Jin, Yantao Liu, Yutao Zeng, Wenxuan Liu, Xiang Li, Pan Yang, Long Bai, Jiafeng Guo, and Xueqi Cheng. Retrieval-Augmented Code Generation for Universal Information Extraction. In Derek F. Wong, Zhongyu Wei, and Muyun Yang, editors, *Natural Language Processing and Chinese Computing*, pages 30–42, Singapore, 2025. Springer Nature.

[24] Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, 45(5):885–892, October 2012.

[25] Hassan Hamad, Abhinav Kumar Thakur, Nijil Kolleri, Sujith Pulikodan, and Keith Chugg. FIRE: A Dataset for Financial Relation Extraction. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3628–3642, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[26] Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[27] Chadi Helwe, Simon Coumes, Chloé Clavel, and Fabian Suchanek. TINA: Textual Inference with Negation Augmentation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4086–4099, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[28] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid,

Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *ACM Computing Surveys*, 54(4):1–37, May 2022. arXiv:2003.02320 [cs].

[29] I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. DEGREE: A Data-Efficient Generation-Based Event Extraction Model. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States, July 2022. Association for Computational Linguistics.

[30] Xuming Hu, Aiwei Liu, Zeqi Tan, Xin Zhang, Chenwei Zhang, Irwin King, and Philip S. Yu. GDA: Generative Data Augmentation Techniques for Relation Extraction Tasks. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10221–10234, Toronto, Canada, July 2023. Association for Computational Linguistics.

[31] Wenhao Huang, Qianyu He, Zhixu Li, Jiaqing Liang, and Yanghua Xiao. Is There a One-Model-Fits-All Approach to Information Extraction? Revisiting Task Definition Biases. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10274–10287, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[32] Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. SciREX: A Challenge Dataset for Document-Level Information Extraction. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online, July 2020. Association for Computational Linguistics.

[33] Yizhu Jiao, Ming Zhong, Sha Li, Ruining Zhao, Siru Ouyang, Heng Ji, and Jiawei Han. Instruct and Extract: Instruction Tuning for On-Demand Information Extraction. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10030–10051, Singapore, December 2023. Association for Computational Linguistics.

[34] Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. GenIE: Generative Information Extraction. In Marine Carpuat, Marie-Catherine

de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, Seattle, United States, July 2022. Association for Computational Linguistics.

[35] Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. SHAPE: Shifted Absolute Position Embedding for Transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3309–3321, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[36] Martin Krallinger, Obdulia Rabal, Saber Ahmad Akhondi, Martín Pérez Pérez, Jesus Santamaría, Gael Pérez Rodríguez, Georgios Tsatsaronis, Ander Intxaurrondo, José Antonio Baso López, Umesh K. Nandal, Erin M. van Buel, Ambika Chandrasekhar, Marleen Rodenburg, Astrid Lægreid, Marius A. Doornenbal, Julen Oyarzábal, Anália Lourenço, and Alfonso Valencia. Overview of the BioCreative VI chemical-protein interaction track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, 2017.

[37] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109.

[38] Alice Kwak, Cheonkam Jeong, Gaetano Forte, Derek Bambauer, Clayton Morrison, and Mihai Surdeanu. Information Extraction from Legal Wills: How Well Does GPT-4 Do? In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4336–4353, Singapore, December 2023. Association for Computational Linguistics.

[39] Markus Lanthaler, Richard Cyganiak, and David Wood. RDF 1.1 concepts and abstract syntax. W3C Recommendation, W3C, February 2014.

[40] Bo Li, Dingyao Yu, Wei Ye, Jinglei Zhang, and Shikun Zhang. Sequence Generation with Label Augmentation for Relation Extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13043–13050, June 2023.

[41] Juntao Li, Xiaobo Liang, Lijun Wu, Yue Wang, Qi Meng, Tao Qin, Min Zhang, and Tie-Yan Liu. Randomness Regularization With Simple Consistency Training for

Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5763–5778, August 2024.

[42] Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. CodeIE: Large Code Generation Models are Better Few-Shot Information Extractors. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Toronto, Canada, July 2023. Association for Computational Linguistics.

[43] Sha Li, Heng Ji, and Jiawei Han. Document-Level Event Argument Extraction by Conditional Generation. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online, June 2021. Association for Computational Linguistics.

[44] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice Loss for Data-imbalanced NLP Tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476, Online, July 2020. Association for Computational Linguistics.

[45] Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Lixiang Lixiang, Zhilei Hu, Long Bai, Wei Li, Yidan Liu, Pan Yang, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. KnowCoder: Coding Structured Knowledge into LLMs for Universal Information Extraction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8758–8779, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[46] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, January 1991.

[47] Kang Liu, Yubo Chen, Jian Liu, Xinyu Zuo, and Jun Zhao. Extracting Events and Their Relations from Texts: A Survey on Recent Research Progress and Challenges. *AI Open*, 1:22–39, January 2020.

[48] Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. CrossNER: Evaluating Cross-Domain Named Entity Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13452–13460, May 2021.

[49] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 22631–22648. PMLR, July 2023.

[50] Keming Lu, I-Hung Hsu, Wenxuan Zhou, Mingyu Derek Ma, and Muhao Chen. Summarization as Indirect Supervision for Relation Extraction. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6575–6594, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[51] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified Structure Generation for Universal Information Extraction. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[52] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[53] Ling Luo, Po-Ting Lai, Chih-Hsuan Wei, Cecilia N Arighi, and Zhiyong Lu. BioRED: a rich biomedical relation extraction dataset. *Briefings in Bioinformatics*, 23(5):bbac282, September 2022.

[54] Xiaoqiang Luo. On Coreference Resolution Performance Metrics. In Raymond Mooney, Chris Brew, Lee-Feng Chien, and Katrin Kirchhoff, editors, *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[55] Filipe Mesquita, Matteo Cannaviccio, Jordan Schmidek, Paramita Mirza, and Denilson Barbosa. KnowledgeNet: A Benchmark Dataset for Knowledge Base Population. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 749–758, Hong Kong, China, November 2019. Association for Computational Linguistics.

[56] Antonio Miranda-Escalada, Farrokh Mehryary, Jouni Luoma, Darryl Estrada-Zavala, Luis Gasco, Sampo Pyysalo, Alfonso Valencia, and Martin Krallinger. Overview of DrugProt task at BioCreative VII: data and methods for large-scale text mining and knowledge graph generation of heterogenous chemical–protein relations. *Database*, 2023:baad080, January 2023.

[57] Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, August 2019.

[58] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, July 2019.

[59] Hao Peng, Tianyu Gao, Xu Han, Yankai Lin, Peng Li, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Learning from Context or Names? An Empirical Study on Neural Relation Extraction. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3661–3672, Online, November 2020. Association for Computational Linguistics.

[60] Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. ADELIE: Aligning Large Language Models on Information Extraction. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7371–7387, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[61] Pierre Quinton and Valérian Rey. Jacobian Descent for Multi-Objective Optimization, February 2025. arXiv:2406.16232 [cs].

[62] Obdulia Rabal, Jose Antonio López, Astrid Lagreid, and Martin Krallinger. DrugProt corpus relation annotation guidelines [ChemProt - Biocreative VI], June 2021.

[63] Shuhuai Ren, Jinchao Zhang, Lei Li, Xu Sun, and Jie Zhou. Text AutoAugment: Learning Compositional Augmentation Policy for Text Classification. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9029–9043, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[64] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling Relations and Their Mentions without Labeled Text. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg, 2010. Springer.

[65] Dongyu Ru, Changzhi Sun, Jiangtao Feng, Lin Qiu, Hao Zhou, Weinan Zhang, Yong Yu, and Lei Li. Learning Logic Rules for Document-Level Relation Extraction. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1239–1250, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[66] Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. GoLLIE: Annotation Guidelines improve Zero-Shot Information-Extraction, March 2024. arXiv:2310.03668 [cs].

[67] Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, and Eneko Agirre. Label Verbalization and Entailment for Effective Zero and Few-Shot Relation Extraction. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1199–1212, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[68] David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. SoMeSci- A 5 Star Open Data Gold Standard Knowledge Graph of Software Mentions in Scientific Articles. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pages 4574–4583, New York, NY, USA, October 2021. Association for Computing Machinery.

[69] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics.

[70] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[71] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-training for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc., 2020.

[72] Sasha Spala, Nicholas A. Miller, Yiming Yang, Franck Dernoncourt, and Carl Dockhorn. DEFT: A corpus for definition extraction in free- and semi-structured text. In Annemarie Friedrich, Deniz Zeyrek, and Jet Hoek, editors, *Proceedings of the 13th Linguistic Annotation Workshop*, pages 124–131, Florence, Italy, August 2019. Association for Computational Linguistics.

[73] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a Web-based Tool for NLP-Assisted Text Annotation. In Frédérique Segond, editor, *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April 2012. Association for Computational Linguistics.

[74] Qingyu Tan, Lu Xu, Lidong Bing, Hwee Tou Ng, and Sharifah Mahani Aljunied. Revisiting DocRED - Addressing the False Negative Problem in Relation Extraction. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8472–8487, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[75] Komal Teru. Semi-supervised Relation Extraction via Data Augmentation and Consistency-training. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1112–1124, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[76] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[77] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020.

[78] Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. InstructUIE: Multi-task Instruction Tuning for Unified Information Extraction, April 2023. arXiv:2304.08085 [cs].

[79] Xingyao Wang, Sha Li, and Heng Ji. Code4Struct: Code Generation for Few-Shot Event Structure Prediction. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3640–3663, Toronto, Canada, July 2023. Association for Computational Linguistics.

[80] Yiwei Wang, Muhao Chen, Wenxuan Zhou, Yujun Cai, Yuxuan Liang, Dayiheng Liu, Baosong Yang, Juncheng Liu, and Bryan Hooi. Should We Rely on Entity Mentions for Relation Extraction? Debiasing Relation Extraction with Counterfactual Analysis. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3071–3081, Seattle, United States, July 2022. Association for Computational Linguistics.

[81] Yiwei Wang, Bryan Hooi, Fei Wang, Yujun Cai, Yuxuan Liang, Wenxuan Zhou, Jing Tang, Manjuan Duan, and Muhao Chen. How Fragile is Relation Extraction under Entity Replacements? In Jing Jiang, David Reitter, and Shumin Deng, editors, *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 414–423, Singapore, December 2023. Association for Computational Linguistics.

[82] Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. TPLinker: Single-stage Joint Extraction of Entities and Relations Through

Token Pair Linking. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1572–1582, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[83] Jason Wei and Kai Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics.

[84] Frank Wilcoxon. Individual Comparisons by Ranking Methods. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer, New York, NY, 1992.

[85] Jianhan Wu, Shijing Si, Jianzong Wang, and Jing Xiao. Augmentation-induced Consistency Regularization for Classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, July 2022.

[86] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised Data Augmentation for Consistency Training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc., 2020.

[87] Benfeng Xu, Quan Wang, Yajuan Lyu, Yabing Shi, Yong Zhu, Jie Gao, and Zhendong Mao. EmRel: Joint Representation of Entities and Embedded Relations for Multi-triple Extraction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 659–665, Seattle, United States, July 2022. Association for Computational Linguistics.

[88] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: a survey. *Frontiers of Computer Science*, 18(6):186357, November 2024.

[89] Justin Xu, Prashanth Arun, and Pascal Poupart. Structural Consistency for Information Extraction. 2025.

[90] Justin Xu, Jason Sun, and Pascal Poupart. AdaIE: Towards Accessible Guidelines for Information Extraction. 2025.

[91] Reda Yacouby and Dustin Axman. Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models. In Steffen Eger, Yang Gao, Maxime Peyrard, Wei Zhao, and Eduard Hovy, editors, *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 79–91, Online, November 2020. Association for Computational Linguistics.

[92] Yuming Yang, Wantong Zhao, Caishuang Huang, Junjie Ye, Xiao Wang, Huiyuan Zheng, Yang Nan, Yuran Wang, Xueying Xu, Kaixin Huang, Yunke Zhang, Tao Gui, Qi Zhang, and Xuanjing Huang. Beyond Boundaries: Learning a Universal Entity Taxonomy across Datasets and Languages for Open Named Entity Recognition. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10902–10923, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

[93] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy, July 2019. Association for Computational Linguistics.

[94] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. GPT3Mix: Leveraging Large-scale Language Models for Text Augmentation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[95] Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. Dialogue-Based Relation Extraction. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4927–4940, Online, July 2020. Association for Computational Linguistics.

[96] Klim Zaporojets, Johannes Deleu, Chris Develder, and Thomas Demeester. DWIE: An entity-centric dataset for multi-task document-level information extraction. *Information Processing & Management*, 58(4):102563, July 2021.

[97] Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[98] Kai Zhang, Bernal Jimenez Gutierrez, and Yu Su. Aligning Instruction Tasks Unlocks Large Language Models as Zero-Shot Relation Extractors. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 794–812, Toronto, Canada, July 2023. Association for Computational Linguistics.

[99] Mi Zhang, Tieyun Qian, Ting Zhang, and Xin Miao. Towards Model Robustness: Generating Contextual Counterfactuals for Entities in Relation Extraction. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pages 1832–1842, New York, NY, USA, April 2023. Association for Computing Machinery.

[100] Zhi Zhang, Junan Yang, Hui Liu, Pengjiang Hu, Zhi Zhang, Junan Yang, Hui Liu, and Pengjiang Hu. BTDM: A Bi-Directional Translating Decoding Model-Based Relational Triple Extraction. *Applied Sciences*, 13(7), March 2023. Company: Multidisciplinary Digital Publishing Institute Distributor: Multidisciplinary Digital Publishing Institute Institution: Multidisciplinary Digital Publishing Institute Label: Multidisciplinary Digital Publishing Institute.

APPENDICES

Appendix A

Supplementary for Chapter 3

A.1 Unusual Dynamics of Dice-Sørensen Loss

As defined in Section 3.2.2 the definition of Dice-Sørensen consistency loss between two probability vectors p and q is given in Equation 3.6, repeated here:

$$d_Z(p,q) = 1 - \frac{2\sum_i p_i q_i + 1}{\sum_i p_i + \sum_i q_i + 1}$$

We would like to analyze the effect of minimizing the DS loss between p and q. Since the formula is symmetric for p and q, and all indices are treated equally, without loss of generality we can differentiate with respect to p_1. To make this analysis easy, define

$$A = 2\sum_{i\neq 1} p_i q_i + 1 \tag{A.1}$$

$$B = \sum_{i\neq 1} p_i + \sum_{i\neq 1} q_i + 1 \tag{A.2}$$

Notably, since $p_i q_i \leq \min(p_i, q_i)$, we have $B \geq A$. Then, we can rewrite

$$d_Z(p,q) = 1 - \frac{p_1 q_1 + A}{p_1 + q_1 + B} \tag{A.3}$$

$$\frac{\partial}{\partial p_1} d_Z(p,q) = \frac{-2q_1^2 - 2Bq_1 + A}{(p_1 + q_1 + B)^2} \tag{A.4}$$

The influence of p_1 on the gradient is minuscule, as it only appears in the denominator which is dominated by B, especially in high-dimensional vectors such as those seen in information extraction. Intuitively, this means that the gradient is near constant, similar to an L1 loss.

Meanwhile, the numerator is a downward quadratic function of q_1 with a positive root at

$$r = \frac{\sqrt{B^2 + 2A} - B}{2} < \frac{\sqrt{B^2 + 2B + 1} - B}{2} = \frac{1}{2} \tag{A.5}$$

For $q_1 < r$, the gradient is positive with respect to p_1; DS loss is minimized by decreasing p_1 even if $p_1 < q_1$. Intuitively, this is similar to an L1 loss where the target is 0. For $q_1 > r$, the gradient is negative with respect to q_1; DS loss is minimized by increasing p_1 even if $p_1 > q_1$. Intuitively, this is similar to an L1 loss where the target is 1.

Moreover, the influence is asymmetric, since $0 < r < \frac{1}{2}$. That is, for the majority of values of q_1, the tendency during training is to increase p_1 towards 1. This is an opposite influence than what is observed for divergence-based losses in consistency training experiments, in which most predictions tend toward 0. We hypothesize that this unusual dynamic is the reason for the success of DS as a consistency loss.

However, this analysis also points out a potential weakness of DS loss; optimizing for DS loss results in predictions that are close to either 0 or 1. We hypothesize that this hurts the calibration of the model by leading to overconfident predictions.

Appendix B

Supplementary for Chapter 4

B.1 Prompt Examples

B.1.1 JSON-Based Prompting

Guided Entity Extraction

```
## Task

You are a highly accurate entity extraction system that annotates entities in a
↪  passage of source text provided below.
Extract all entities in the passage provided below, carefully following the provided
↪  schema of entity types.

"Entity" does not refer to the typical single-label entity classification data model.
Depending on the entity type, a valid entity may include pronouns, entire phrases,
↪  numbers, dates, or symbols.
Entity types are not mutually exclusive; an entity may satisfy multiple
↪  possibly-overlapping entity types, all of which should be extracted.
The entity types are independent; whether an entity satisfies an entity type does not
↪  depend on any other entity type in this schema.

---

## Entity Types

Each entity type is a predicate defined by a verbalization; an entity satisfies the
↪  type if the verbalization is entailed by the passage when its text is substituted
↪  into the position of the placeholder.
```

```
Entailment is extremely important; do not mark an entity as belonging to an entity type unless it is supported by common sense or textual evidence, even if it is true.
Entity types are not mutually exclusive; an entity may satisfy multiple entity types.
The entity types are independent; whether an entity satisfies an entity type does not depend on any other entity type in this schema.

```json
[
  {
    "name": "institution",
    "verbalization": "{institution} is a university, academic department, or research institute."
  },
  {
    "name": "organization",
    "verbalization": "{organization} is a company, research lab, or other organized group."
  },
  {
    "name": "person",
    "verbalization": "{person} is a researcher, student, or academic, referred to by name or pronoun."
  }
]
```

Passage

The passage may be a chunk of a larger text, or the entire text.
The passage has been split into numbered sentences and escaped before formatting each sentence as JSON:

```json
[
  {
    "index": 0,
    "sentence": "Yoshua Bengio is a professor at the Universite de Montreal and scientific director of MILA, the Montreal Institute for Learning Algorithms, where he has spent much of his career. Among his doctoral students was Ian Goodfellow, creator of generative adversarial networks; Goodfellow completed his thesis under Bengio at the Universite de Montreal. Bengio remains affiliated with both the university and MILA."
  }
```
```

````
]
```
````

Guided Relation Extraction

````
## Task

You are a highly accurate relation extraction and event extraction system proficient
↪  in handling higher-order relations.
Extract all relations between the extracted entities in the passage provided below,
↪  carefully following the provided schema of relation types.

"Relation" does not refer to the typical relational triple data model.
Rather, relations here are more similar to event argument extraction or hyper-relation
↪  extraction.
Study the descriptions carefully to understand the difference.
Each relation is an association between at least 2 entities, but may involve an
↪  arbitrary number of entities.
Each relation has arguments indicating semantic roles of the entities involved.
The same entity may be involved in multiple arguments of a single relation.
The same entity may be involved in multiple relations of the same relation type, with
↪  different other entities.
Relation types are not mutually exclusive; the same combination of entities and
↪  arguments may belong to multiple possibly-overlapping relation types, all of which
↪  should be extracted.
Relation types are independent; whether a relation belongs to a relation type does not
↪  depend on any other relation type in this schema.

---

## Entity Types

Entities have been extracted according to the following schema of entity types:
```json
[
  {
    "name": "institution",
    "verbalization": "{institution} is a university, academic department, or research
    ↪  institute."
  },
  {
    "name": "organization",
    "verbalization": "{organization} is a company, research lab, or other organized
    ↪  group."
````

````
  },
  {
    "name": "person",
    "verbalization": "{person} is a researcher, student, or academic, referred to by
    ↪  name or pronoun."
  }
]
```

---

## Relation Types

Each relation type is defined by a verbalization; a relation belongs to the type if
↪  the verbalization is entailed by the passage when the participating entities are
↪  substituted into the position of each argument's placeholders.
Entailment is extremely important; do not extract relations unless they are supported
↪  by common sense or textual evidence, even if they are true.

Minimum and maximum argument cardinality indicates the number of entities that must be
↪  present in the relation under that argument type.
An argument type with a `minimum_cardinality` of 0 may be omitted.
An argument type with a `maximum_cardinality` of `null` indicates that it may involve
↪  an unlimited number of entities.
If provided, `entity_types` determines exactly which entity types are allowed for the
↪  argument type (i.e., any extracted entity must satisfy at least one of the entity
↪  types listed in the `entity_types` field).

The relation types are listed below with their argument types.
```json
[
  {
    "name": "affiliation",
    "verbalization": "{person} has a currently active affiliation with
    ↪  {organization}.",
    "arguments": [
      {
        "name": "organization",
        "minimum_cardinality": 1,
        "maximum_cardinality": 1,
        "entity_types": [
          "institution",
          "organization"
        ]
      },
      {
````

```
        "name": "person",
        "minimum_cardinality": 1,
        "maximum_cardinality": 1,
        "entity_types": [
          "person"
        ]
      }
    ]
  },
  {
    "name": "supervised",
    "verbalization": "{supervisor} supervised {supervisee} at {institution}.",
    "arguments": [
      {
        "name": "institution",
        "minimum_cardinality": 0,
        "maximum_cardinality": null,
        "entity_types": [
          "institution"
        ]
      },
      {
        "name": "supervisee",
        "minimum_cardinality": 1,
        "maximum_cardinality": 1,
        "entity_types": [
          "person"
        ]
      },
      {
        "name": "supervisor",
        "minimum_cardinality": 1,
        "maximum_cardinality": null,
        "entity_types": [
          "person"
        ]
      }
    ]
  }
]
```

Passage

````
The passage may be a chunk of a larger text, or the entire text.
The passage has been split into numbered sentences and has been escaped before
↪  formatting each sentence as JSON:
```json
[
  {
    "index": 0,
    "sentence": "Yann LeCun joined Bell Labs in 1988 and later became a professor at
    ↪  NYU, where he remains affiliated. At Bell Labs he supervised Leon Bottou, and
    ↪  he continued to advise Bottou during the latter's doctoral work at NYU. Since
    ↪  2013 LeCun has served as chief AI scientist at Meta."
  }
]
```

---

## Entities

The following entities were previously extracted from the passage according to the
↪  entity types listed above.
Do not attempt to add new entities or correct the extractions.
Entities are identified by their `id` number.
Entities can take multiple forms (appearing multiple times in the passage). All forms
↪  are listed in the `text` field.
Each entity satisfies at least one of the above entity types, listed in the `types`
↪  field.
```json
[
  {
    "id": 0,
    "text": [
      "Yann LeCun",
      "LeCun"
    ],
    "types": [
      "person"
    ]
  },
  {
    "id": 1,
    "text": [
      "Bell Labs"
    ],
    "types": [
      "institution"
````
````

```
    ]
  },
  {
    "id": 2,
    "text": [
      "NYU"
    ],
    "types": [
      "institution"
    ]
  },
  {
    "id": 3,
    "text": [
      "Bottou",
      "Leon Bottou"
    ],
    "types": [
      "person"
    ]
  },
  {
    "id": 4,
    "text": [
      "Meta"
    ],
    "types": [
      "organization"
    ]
  }
]
```

B.1.2 InstructUIE

Guided Entity Extraction

```
Please list all entity words in the text that fit the category.Output format is
↪  "type1: word1; type2: word2".
Option: person, organization, institution
Text: Yoshua Bengio is a professor at the Universite de Montreal and scientific
↪  director of MILA, the Montreal Institute for Learning Algorithms, where he has
↪  spent much of his career. Among his doctoral students was Ian Goodfellow, creator
↪  of generative adversarial networks; Goodfellow completed his thesis under Bengio
↪  at the Universite de Montreal. Bengio remains affiliated with both the university
↪  and MILA.
Answer:
```

Guided Information Extraction

```
Locate the argument in the text that participated in the relation based on the
↪  relation type and return it in the relation list.
 Option: Relation type: supervised, affiliation, Arguments type: person, organization,
  ↪  institution.
Text: Yoshua Bengio is a professor at the Universite de Montreal and scientific
↪  director of MILA, the Montreal Institute for Learning Algorithms, where he has
↪  spent much of his career. Among his doctoral students was Ian Goodfellow, creator
↪  of generative adversarial networks; Goodfellow completed his thesis under Bengio
↪  at the Universite de Montreal. Bengio remains affiliated with both the university
↪  and MILA.
Answer:
```

B.1.3 Code4UIE

Guided Entity Extraction

```
from typing import List

class Entity:
    def __init__(self, name: str):
        self.name = name

class Person(Entity):
    """{person} is a researcher, student, or academic, referred to by name or
    ↪  pronoun."""

    def __init__(self, name: str):
        super().__init__(name=name)

class Organization(Entity):
    """{organization} is a company, research lab, or other organized group."""

    def __init__(self, name: str):
        super().__init__(name=name)

class Institution(Organization):
    """{institution} is a university, academic department, or research institute."""

    def __init__(self, name: str):
        super().__init__(name=name)

"""
List all the Entity words in the following text as instances of corresponding
↪  subclasses of class Entity. If there do not exist any Entity words that belong to
↪  the Entity subclasses we defined, print "None".
"Yann LeCun joined Bell Labs in 1988 and later became a professor at NYU, where he
↪  remains affiliated. At Bell Labs he supervised Leon Bottou, and he continued to
↪  advise Bottou during the latter's doctoral work at NYU. Since 2013 LeCun has
↪  served as chief AI scientist at Meta."
"""
person_entity1 = Person(
    name = 'Yann LeCun',
)
institution_entity1 = Institution(
    name = 'Bell Labs',
)
institution_entity2 = Institution(
    name = 'NYU',
```

```
)
person_entity2 = Person(
    name = 'Leon Bottou',
)
organization_entity1 = Organization(
    name = 'Meta',
)


"""
List all the Entity words in the following text as instances of corresponding
↪  subclasses of class Entity. If there do not exist any Entity words that belong to
↪  the Entity subclasses we defined, print "None".
"Yoshua Bengio is a professor at the Universite de Montreal and scientific director of
↪  MILA, the Montreal Institute for Learning Algorithms, where he has spent much of
↪  his career. Among his doctoral students was Ian Goodfellow, creator of generative
↪  adversarial networks; Goodfellow completed his thesis under Bengio at the
↪  Universite de Montreal. Bengio remains affiliated with both the university and
↪  MILA."
"""
```

Guided Information Extraction

```
from typing import List

class Entity:
    def __init__(self, name: str):
        self.name = name

class Person(Entity):
    """{person} is a researcher, student, or academic, referred to by name or
    ↪  pronoun."""

    def __init__(self, name: str):
        super().__init__(name=name)

class Organization(Entity):
    """{organization} is a company, research lab, or other organized group."""

    def __init__(self, name: str):
        super().__init__(name=name)

class Institution(Organization):
    """{institution} is a university, academic department, or research institute."""
```

```
    def __init__(self, name: str):
        super().__init__(name=name)

class Relation:
    def __init__(self, name: str):
        self.name = name

class Supervised(Relation):
    """{supervisor} supervised {supervisee} at {institution}."""
    def __init__(
        self,
        institution: List[Institution] = [],
        supervisee: List[Person] = [],
        supervisor: List[Person] = [],
    ):
        self.institution = institution
        self.supervisee = supervisee
        self.supervisor = supervisor

class Affiliation(Relation):
    """{person} has a currently active affiliation with {organization}."""
    def __init__(
        self,
        organization: List[Institution | Organization] = [],
        person: List[Person] = [],
    ):
        self.organization = organization
        self.person = person

"""
List all the relations in the following text that belong to the aformentioned Relation
↪  subclass as instances of them.
"Yann LeCun joined Bell Labs in 1988 and later became a professor at NYU, where he
↪  remains affiliated. At Bell Labs he supervised Leon Bottou, and he continued to
↪  advise Bottou during the latter's doctoral work at NYU. Since 2013 LeCun has
↪  served as chief AI scientist at Meta."
"""
affiliation_event1 = Affiliation(
    organization = [
        Organization("Meta"),
    ],
    person = [
        Person("Yann LeCun"),
    ],
)
```

```
affiliation_event2 = Affiliation(
    organization = [
        Institution("NYU"),
    ],
    person = [
        Person("Yann LeCun"),
    ],
)
supervised_event1 = Supervised(
    institution = [
        Institution("Bell Labs"),
        Institution("NYU"),
    ],
    supervisee = [
        Person("Leon Bottou"),
    ],
    supervisor = [
        Person("Yann LeCun"),
    ],
)

"""
List all the relations in the following text that belong to the aformentioned Relation
↪  subclass as instances of them.
"Yoshua Bengio is a professor at the Universite de Montreal and scientific director of
↪  MILA, the Montreal Institute for Learning Algorithms, where he has spent much of
↪  his career. Among his doctoral students was Ian Goodfellow, creator of generative
↪  adversarial networks; Goodfellow completed his thesis under Bengio at the
↪  Universite de Montreal. Bengio remains affiliated with both the university and
↪  MILA."
"""
```

B.1.4 GoLLIE

Guided Entity Extraction

```
# The following lines describe the task definition
@dataclass
class Person(Entity):
    """{person} is a researcher, student, or academic, referred to by name or
    ↪  pronoun."""

    span: str
```

```python
@dataclass
class Organization(Entity):
    """{organization} is a company, research lab, or other organized group."""

    span: str


@dataclass
class Institution(Organization):
    """{institution} is a university, academic department, or research institute."""

    span: str


# This is the text to analyze
text = """Yoshua Bengio is a professor at the Universite de Montreal and scientific
↪  director of MILA, the Montreal Institute for Learning Algorithms, where he has
↪  spent much of his career. Among his doctoral students was Ian Goodfellow, creator
↪  of generative adversarial networks; Goodfellow completed his thesis under Bengio
↪  at the Universite de Montreal. Bengio remains affiliated with both the university
↪  and MILA."""
# The annotation instances that take place in the text above are listed here
result = [
```

Guided Information Extraction

```python
# The following lines describe the task definition
@dataclass
class Supervised(Relation):
    """{supervisor} supervised {supervisee} at {institution}."""

    institution: List[str]  # types: {Institution}
    supervisee: str  # at least 1; at most 1; types: {Person}
    supervisor: List[str]  # at least 1; types: {Person}


@dataclass
class Affiliation(Relation):
    """{person} has a currently active affiliation with {organization}."""

    organization: str  # at least 1; at most 1; types: {Institution, Organization}
    person: str  # at least 1; at most 1; types: {Person}
```

```
# This is the text to analyze
text = """Yoshua Bengio is a professor at the Universite de Montreal and scientific
↪  director of MILA, the Montreal Institute for Learning Algorithms, where he has
↪  spent much of his career. Among his doctoral students was Ian Goodfellow, creator
↪  of generative adversarial networks; Goodfellow completed his thesis under Bengio
↪  at the Universite de Montreal. Bengio remains affiliated with both the university
↪  and MILA."""
# The annotation instances that take place in the text above are listed here
result = [
```

B.1.5 KnowCoder

Guided Entity Extraction

```
class Entity:
    """
    The base class for all entities.
    """
    def __init__(self, name: str):
        self.name = name

class Person(Entity):
    """
    Description: {person} is a researcher, student, or academic, referred to by name
    ↪  or pronoun.
    """
    pass

class Organization(Entity):
    """
    Description: {organization} is a company, research lab, or other organized group.
    """
    pass

class Institution(Organization):
    """
    Description: {institution} is a university, academic department, or research
    ↪  institute.
    """
    pass

"""
This is an object-oriented programming task: some Entity Classes are defined above.
↪  Please instantiate all the corresponding Entity Objects in each following text.
"""
text = "Yann LeCun joined Bell Labs in 1988 and later became a professor at NYU, where
↪  he remains affiliated. At Bell Labs he supervised Leon Bottou, and he continued to
↪  advise Bottou during the latter's doctoral work at NYU. Since 2013 LeCun has
↪  served as chief AI scientist at Meta."
results = [
    Person("Yann LeCun"),
    Institution("Bell Labs"),
    Institution("NYU"),
    Person("Leon Bottou"),
    Organization("Meta")
]
```

```
text = "Yoshua Bengio is a professor at the Universite de Montreal and scientific
↪  director of MILA, the Montreal Institute for Learning Algorithms, where he has
↪  spent much of his career. Among his doctoral students was Ian Goodfellow, creator
↪  of generative adversarial networks; Goodfellow completed his thesis under Bengio
↪  at the Universite de Montreal. Bengio remains affiliated with both the university
↪  and MILA."
results = [
```

Guided Information Extraction

```
from Entities import Person, Organization, Institution

class Entity:
    """
    The base class for all entities.
    """
    def __init__(self, name: str):
        self.name = name

class Relation:
    """
    The base class for all relations.
    """
    pass

class Supervised(Relation):
    """
    Description: {supervisor} supervised {supervisee} at {institution}.
    """
    def __init__(
        self,
        institution: List[Institution],
        supervisee: Person,
        supervisor: List[Person],
    ):
        self.institution = institution
        self.supervisee = supervisee
        self.supervisor = supervisor

class Affiliation(Relation):
    """
    Description: {person} has a currently active affiliation with {organization}.
```

```
    """
    def __init__(
        self,
        organization: Institution | Organization,
        person: Person,
    ):
        self.organization = organization
        self.person = person

"""
This is an object-oriented programming task: some Relation Classes and related Entity
↪  Classes are defined above. Please instantiate all the corresponding Relation
↪  Objects in each following text.
"""
text = "Yann LeCun joined Bell Labs in 1988 and later became a professor at NYU, where
↪  he remains affiliated. At Bell Labs he supervised Leon Bottou, and he continued to
↪  advise Bottou during the latter's doctoral work at NYU. Since 2013 LeCun has
↪  served as chief AI scientist at Meta."
results = [
    Affiliation(
        organization=Organization("Meta"),
        person=Person("Yann LeCun"),
    ),
    Affiliation(
        organization=Institution("NYU"),
        person=Person("Yann LeCun"),
    ),
    Supervised(
        institution=[Institution("Bell Labs"), Institution("NYU")],
        supervisee=Person("Leon Bottou"),
        supervisor=[Person("Yann LeCun")],
    )
]


text = "Yoshua Bengio is a professor at the Universite de Montreal and scientific
↪  director of MILA, the Montreal Institute for Learning Algorithms, where he has
↪  spent much of his career. Among his doctoral students was Ian Goodfellow, creator
↪  of generative adversarial networks; Goodfellow completed his thesis under Bengio
↪  at the Universite de Montreal. Bengio remains affiliated with both the university
↪  and MILA."
results = [
```