Complex Mathematical Symbol Definition Structures: A Dataset and Model for Coordination Resolution in Definition Extraction

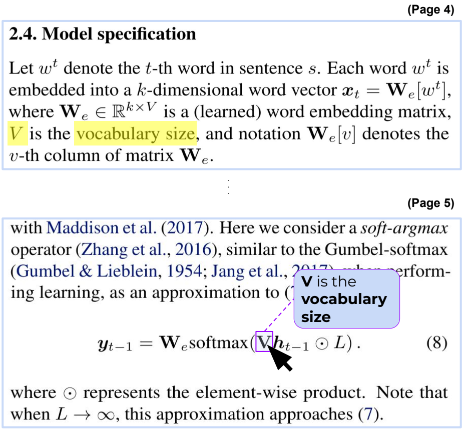
Anna Martin-Boyle[1] **Andrew Head**[2] **Kyle Lo**[3] **Risham Sidhu**[4] **Marti A. Hearst**[5] **Dongyeop Kang**[1]
[1]University of Minnesota, Minneapolis, MN [2]University of Pennsylvania, Philadelphia, PA
[3]Allen Institute for AI, Seattle, WA [4]University of Illinois Urbana-Champaign, IL
[5]University of California, Berkeley, CA
[1]{mart5877,dongyeop}@umn.edu [2]head@seas.upenn.edu
[3]kylel@allenai.org [4]rsidhu@illinois.edu [5]hearst@berkeley.edu

Abstract

Mathematical symbol definition extraction is important for improving scholarly reading interfaces and scholarly information extraction (IE). However, the task poses several challenges: math symbols are difficult to process as they are not composed of natural language morphemes; and scholarly papers often contain sentences that require resolving complex coordinate structures. We present SymDef, an English language dataset of 5,927 sentences from full-text scientific papers where each sentence is annotated with all mathematical symbols linked with their corresponding definitions. This dataset focuses specifically on complex coordination structures such as "respectively" constructions, which often contain overlapping definition spans. We also introduce a new definition extraction method that masks mathematical symbols, creates a copy of each sentence for each symbol, specifies a target symbol, and predicts its corresponding definition spans using slot filling. Our experiments show that our definition extraction model significantly outperforms RoBERTa and other strong IE baseline systems by 10.9 points with a macro F1 score of 84.82. With our dataset and model, we can detect complex definitions in scholarly documents to make scientific writing more readable.[1]

1 Introduction

As the volume of scientific publishing increases, it is becoming crucial to develop more sophisticated analysis tools and user interfaces for helping scientists make sense of this ever-growing bounty of knowledge. One particular concern is the ability to accurately extract definitions for mathematical symbols. See Figure 1 for one potential use case for mathematical symbol extraction. We find mathematical symbol definition extraction crucial enough to warrant corpora and models tailored to this specific problem.



(Page 4)

2.4. Model specification

Let w^t denote the t-th word in sentence s. Each word w^t is embedded into a k-dimensional word vector $\boldsymbol{x}_t = \mathbf{W}_e[w^t]$, where $\mathbf{W}_e \in \mathbb{R}^{k \times V}$ is a (learned) word embedding matrix, V is the vocabulary size, and notation $\mathbf{W}_e[v]$ denotes the v-th column of matrix $\mathbf{W}_e$.

⋮

(Page 5)

with Maddison et al. (2017). Here we consider a *soft-argmax* operator (Zhang et al., 2016), similar to the Gumbel-softmax (Gumbel & Lieblein, 1954; Jang et al., 2017), when performing learning, as an approximation to (7

$$\boldsymbol{y}_{t-1} = \mathbf{W}_e \text{softmax}(\mathbf{V}\boldsymbol{h}_{t-1} \odot L)\,. \qquad (8)$$

where $\odot$ represents the element-wise product. Note that when $L \to \infty$, this approximation approaches (7).

Figure 1: Reading interfaces such as ScholarPhi (Head et al., 2021) could use math symbol definition extractionto surface symbol definitions as needed. This would save the reader from having to flip between paper sections to look up the definitions of terms in mathematical expressions and algorithms, as in this example from Gu et al. (2018).

For definition recognition to be used in user-facing applications, it must achieve a high precision that has not yet been seen in work to date. This task is complicated by the fact that scientific papers often contain multiple symbols and definitions in one sentence, and their spans may be nested or overlapping. Analysis of these symbols and definitions must be coordinated such that the correct definitions are applied to each symbol. Consider for example, the following sentence fragment:

> . . . $\mathbf{A}$, $\mathbf{C}$ and $\boldsymbol{v}$ denote the within-layer adjacency, between-layer adjacency and the community label matrix, respectively.

In this case, we wish to define $\mathbf{A}$ as "within-layer adjacency matrix", $\mathbf{C}$ as "between-layer adjacency matrix", and $\boldsymbol{v}$ as "community label matrix".

For human readers, the word "respectively" immediately clarifies which definition is associated

[1] Our code and dataset are publicly available at https://github.com/minnesotanlp/taddex

with each symbol. However, even this simple "respectively" construction is not obvious to an NLP algorithm, due to the fact that the definitions for $\mathbf{A}$ and $\mathbf{C}$ are split and overlap with the definition for $\boldsymbol{v}$. Little research has been done on the "respectively" construct specifically, but other work has found resolution of coordination to be important for resolving hard NLP problems. An error analysis by Fader et al. (2011) when working on information extraction found that 52% of errors were in part due to coordination. Information extraction in biosciences (Ogren, 2010; Kolluru et al., 2020; Saha and Mausam, 2018) builds on this insight by attempting to resolve coordination relations directly. Cohen et al. (2009) showed that F-scores for recognition of protein-protein structure could be significantly increased by more accurately recognizing coordination structure (using manual rules, assuming distributed semantics, and using post-processing for specific cases). Furthermore, Systems that rely on token-wise structured prediction techniques such as IOB tagging are insufficient to capture complex coordination patterns due to their inability to accommodate overlapping entities.

In order to address the need for improved coordination resolution in scientific information extraction tasks, we developed SymDef, a corpus of scientific papers with a high frequency of complex coordination patterns. Annotations within SymDef are comprised of mathematical symbols masked as SYMBOL and their sometimes overlapping definitions. This corpus provides an interesting resource for study of complex coordination problems, not only because it contains a high frequency of coordination patterns, but also because the symbols are masked. Because the representations of each symbol are not differentiated from one another, the structure and syntax of the sentences are challenging to identify.

We achieved strong results on our SymDef dataset using a simple but effective method to find the complex mapping between multiple symbols and definitions. Specifically, we decompose the structured prediction problem into multiple passes of definition recognition, with one pass per symbol. For instance, our method would target the example sentence three times, once for each symbol in {$\mathbf{A}$, $\mathbf{C}$, $\boldsymbol{v}$}, and return the following symbol and definition pairs: <$\mathbf{A}$, "within-layer adjacency matrix">, <$\mathbf{C}$, "between-layer adjacency matrix">, and <$\boldsymbol{v}$, "community label matrix">. Since the model recognizes definitions based on a given target symbol, our model is called a *target-based* model.

Our contributions are the following:

- SymDef: a collection of 5,927 sentences from the full texts of 21 scientific papers, with symbols and definitions annotated when present. The papers included contain sentences with complex coordination patterns (such as containing more than two "and"s or "or"s, and/or the word "respectively"). In total, the dataset contains 913 sentences with complex coordination patterns.

- The development of a novel target-based approach to definition recognition that isolates one symbol at a time and finds the definition for that symbol within syntactically complex sentences. Our system outperforms two IE baselines and few-shot GPT3 inference by large margins.

2 Related Work

We discuss previous efforts towards resolving coordination problems, related work in definition recognition, and relevant definition extraction corpora.

Syntactic Structure Recognition. Coordination is well-studied in linguistics, but analysis is generally in terms of syntactic structure and logical constraints. For example, Hara et al., 2009 focus on creating tree structures or parses for coordination where determining scope is sufficient. Some notable sub-cases such as Argument-Cluster Coordination or right-node raising are often addressed in some way (Ficler and Goldberg, 2016b). There is also work determining the phrase boundaries of components in coordinate structures (Shimbo and Hara, 2007; Ficler and Goldberg, 2016a).

While previous work on the syntactic structure of linguistic coordination is useful, definition structures in our work are sometimes more flexible or varied. Furthermore, Dalrymple and Kehler (1995) found that determining the meaning of "respectively" constructions is based on semantics and pragmatics, not the syntax of coordinating conjunctions, and Ogren (2011) found that a parser-free approach works better than one based on a syntactic parse for interpretation of coordinate structures.

Teranishi et al. (2017) propose a neural model that uses similarity and substitutability to predict coordinate spans. Other work has focused on the problem of splitting sentences into two semantically equivalent ones (Ogren, 2010). However, none of the previous work on coordinated definition structures is applied towards using the resolution of coordination patterns for the extraction of term-definition pairs.

Closest to our work is that of Saha and Mausam (2018) which splits conjunctions to form multiple coherent simple sentences before extracting relation tuples. One constraint is that multiple coordination structures in a sentence must either be disjoint or completely nested, which is more restrictive than our approach.

Definition Recognition. We have found that the "respectively" construct is frequently used in the definition of mathematical terms, but its use is not discussed in the literature on definition detection. Others have noted the importance of complex conjunctions in biomedical texts: Ogren (2010) notes that there are 50% more conjunctions in biomedical scientific text than in newswire text, and Tateisi et al. (2008) also found that coordinating conjunctions occur nearly twice as often in biomedical abstracts as in newswire text. This greater frequency of complex conjunctions in scientific and biomedical texts is significant, as Saha et al. (2017) found that coordination was the leading cause of IE recall errors.

Also relevant to our work is that of Dai (2018), who summarized the state of the art in discontiguous span recognition, and Dai et al. (2020), who proposed a transition-based model with generic neural encoding for discontinuous named entity recognition, focusing on separated components and overlapping components.

Span-based information extraction models such a SciIE (Luan et al., 2018) and DyGIE++ (Wadden et al., 2019) are relevant for the task of extracting overlapping or nested entities in that span-representations are better-suited to capture overlapping tokens than traditional IOB tagging approaches; for this reason, we use SciIE and DyGIE++ as baseline models (see Section 5).

Related Corpora.

There are a few related datasets annotated for definition extraction. The word-class lattices (WCL) dataset (Navigli et al., 2010) comprises 4,564 sentences from the Wikipedia corpus, 1,717 of which have a single definition and 2,847 of which contain false definitions (patterns that resemble definitions but do not qualify as such). The W00 dataset (Jin et al., 2013) contains 2,512 sentences taken from 234 workshop papers from the ACL Anthology, 865 of which contain one or more non-overlapping definitions.

The Definition Extraction from Texts (DEFT) corpus (Spala et al., 2019) was developed with the intention to provide a more robust corpus of definition annotations with a higher incidence of complex data samples than the WCL and W00 datasets. DEFT includes 2,443 sentences from 2017 SEC filings and 21,303 sentences from open source textbooks on a number of subjects, with a total of 11,004 definition annotations. The DEFT corpus accommodates cross-sentence definitions and multiple definitions per sentence, but not overlapping or nested terms and definitions.

Observing that the extraction of definitions from math contexts requires specialized training corpora, the authors of the Wolfram Mathworld (WFM) corpus (Vanetik et al., 2020) developed a corpus of full sentence definitions. This corpus comprises 1,793 sentences taken from 2,352 articles from Wolfram Mathworld, 811 of which contain a single definition.

Most similar to our corpus is the NTCIR Math Understanding Subtask corpus (Kristianto et al., 2012). This corpus contains 10 ArXiv papers with annotated math expressions and their descriptions. Similarly to ours, the annotation scheme allows for discontinuous descriptions. The primary difference between SymDef and the NTCIR corpus is SymDef's focus on overlapping definition and respectively cases. The 21 papers in SymDef were specifically selected because they had relatively high counts of the word "respectively" and sentences with multiple "and"s, and our approach accommodates overlapping definitions (see Section 3 for details).

3 SymDef: Coordination Dataset

SymDef is annotated for the coordination of mathematical symbols and their definitions in order to provide a resource for training smart reading interfaces to recognize symbol definitions with a high level of precision. The corpus contains 5,927 English language sentences from the full texts of 21 machine learning papers published on arXiv[2]. These papers were selected by ranking arXiv publications from 2012 to 2022 by the

[2] https://arxiv.org/ ArXiv's Terms of Use states that users may "Retrieve, store, and use the content of arXiv e-prints for your own personal use, or for research purposes".

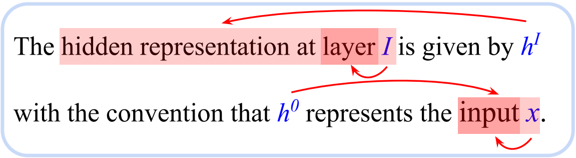

The hidden representation at layer l is given by h^l
with the convention that h^0 represents the input x.

Figure 2: An annotation example for sentences with nested symbols and definitions. I is defined as "layer", h^t is defined as "hidden representation at layer I", h^0 is defined as "input x", and x is defined as "input".

number of mathematical symbols and coordination patterns. This ranking was performed by counting qualifying coordination patterns in each paper, where higher coordination pattern counts were prioritized. These counts were determined per paper using regex pattern matching, searching for the strings "`respectively`" and "`, and`". The highest ranked papers were manually inspected and 21 papers were chosen based on prevalence of symbol-definition pairs.

The first round of annotations was performed by a subset of the authors. This round contributed to the majority of the dataset, resulting in the annotation of 5,661 sentences comprising the full texts of 20 papers.

Additional data were created to supplement the train dataset by annotating another paper containing 226 sentences. These annotations were performed by two domain experts hired through Upwork, one holding a PhD and the other a doctoral student, both in mathematics. The annotators were selected from a set of four applicants to the Upwork listing, all of whom reside in the United States. During the screening process, each of the four applicants were provided with training videos and written documentation in advance, and were trained for 10-30 minutes on 10 example sentences. Their example annotations were monitored and they were asked questions about their process. Upwork annotators were compensated during training and during the annotation task with an hourly rate of $25.00. Each annotator tracked their hours and were paid $543.75 each for their work. Upwork applicants were informed of the intended use of the data in the job description, and signed an Agreement form.

All annotations were performed using the annotation software BRAT[3].

3.1 Annotation Schema

The annotation goal for our dataset was to link symbols with the spans of text that define them at the sentence level. In our formulation, definition spans must declare the meaning of each target symbol; a detailed description of the annotation scheme appears in Appendix A. For example, definition spans may state what the symbol stands for, what a function does, or the datatype it represents. In the case that the symbol represents a collection, the definition may serve to describe the elements contained by the symbol. However, candidate phrases that merely assign a value to the symbol, describe how the symbol is used or computed, or define the symbol with an equation are not valid. Definition spans do not have to contain contiguous text, as they may be split across different parts of the sentence. Furthermore, definitions are permitted to overlap with each other and with symbols as seen in Figure 2.

3.2 Inter-Annotator Agreement

In Table 1, precision, recall, and F1 scores for exact term and definition matches were calculated to determine the inter-annotator agreement between the Upworks annotators over a subset of 266 sentences. Additionally, the mean percentage of overlapping tokens for definition spans was calculated. There was significant agreement between annotators for term identification, earning an F1 score of 0.9. Definition identification was more difficult, yielding an F1 score of 0.67 for exact span matches. However, on average 85% of each definition span overlapped between annotators, indicating that, while it is difficult to find the exact span boundaries, annotators were still in agreement on parts of the definition span.

Of the definition annotations that are not perfect matches, 26 of the annotations from one annotator are contained in the annotations from the other. 126 overlap without containment, with an average number of overlapping words of 4.8. Additionally, 7 of the annotations differ completely, without any overlap.

A review of 1,442 test samples found 76 annotator errors. 46 of these errors were missed definitions. 10 definition spans were nearly correct but contained extra words. 6 were invalid definitions. The remaining errors had to do with improperly defining enumerator variables.

As such, SymDef should continue to be used in personal or research contexts.

[3] `https://brat.nlplab.org/`, view license here

	Term	Definition	Overlap
Precision	.88 ± .08	.65 ± .12	85% ± 7%
Recall	.94 ± .05	.69 ± .11	
F1	.90 ± .06	.67 ± .11	

Table 1: IAA scores for exact term matches, exact definition matches, and mean percent of definition tokens that overlap in SymDef.

3.3 Dataset Characteristics

We measure the structural complexity of SymDef by considering how many symbols and definitions there are per sentence and how difficult they are to link, and how many sentences contain overlapping or nested symbols and definitions.

Coordination of Multiple Terms and Definitions There are a few characteristics to consider when evaluating the difficulty of coordinating multiple terms and definitions, including: the number of terms and definitions in positive sentences; whether or not every symbol is defined in the sentence (some annotated symbols do not have definitions); and how frequently the terms and definitions are collated (e.g. SYM...DEF...SYM...DEF...). The rationale is that an equal number of collated symbols and definitions could be easily coordinated using a simple rule.

The WCL and WFM corpora contain only one definition per sentence. We compare SymDef with the W00 and DEFT corpora, which sometimes contain multiple terms and definitions per sentence.

Overlapping Symbols and Definitions SymDef is uniquely focused on the problem of overlapping symbols and definitions, containing 179 sentences with overlapping spans (13% of positive sentences). Furthermore, many sentences with overlap contained multiple instances of overlapped symbols and definitions. Across all positive sentences there were 480 instances of overlapping, implying that sentences with overlapping contain 2.68 instances on average. W00 and DEFT datasets do not contain overlapping annotations.

4 TaDDEx: Coordination Resolution through Targeted Definition Extraction

Our aim is to coordinate multiple terms and definitions through targeted definition detection. This is achieved by implementing a target-based definition detection model where the target is one symbol from the sample sentence for which a definition must be recognized.

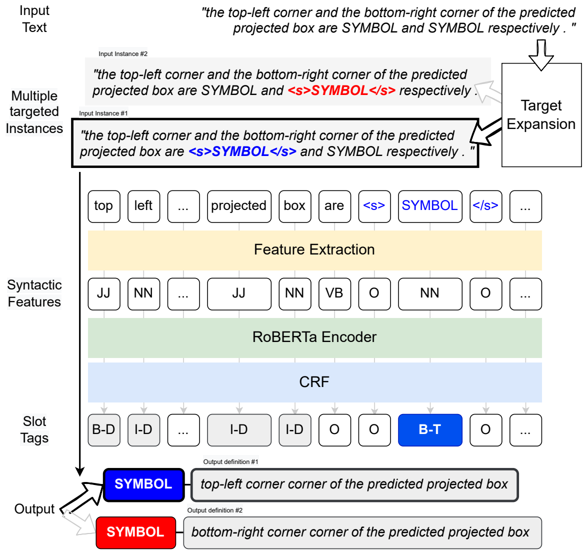

Input
Text
"the top-left corner and the bottom-right corner of the predicted projected box are SYMBOL and SYMBOL respectively . "
Target Expansion
Multiple targeted Instances
Input Instance #2
"the top-left corner and the bottom-right corner of the predicted projected box are SYMBOL and <s>SYMBOL</s> respectively ."
Input Instance #1
"the top-left corner and the bottom-right corner of the predicted projected box are <s>SYMBOL</s> and SYMBOL respectively . "
top
left
...
projected
box
are
<s>
SYMBOL
</s>
...
Feature Extraction
Syntactic Features
JJ
NN
...
JJ
NN
VB
O
NN
O
...
RoBERTa Encoder
CRF
Slot Tags
B-D
I-D
...
I-D
I-D
O
O
B-T
O
...
Output
Output definition #1
SYMBOL
top-left corner corner of the predicted projected box
Output definition #2
SYMBOL
bottom-right corner corner of the predicted projected box

Figure 3: The TaDDEx model. A sentence with n symbols is expanded into n samples. Each sample is input into the RoBERTa model individually such that a predicted definition can be recognized for each target symbol.

4.1 Targeting Individual Symbols in Complex Coordination

Mathematical symbols are masked with the term SYMBOL. Sentences with more than one symbol are duplicated once for each additional symbol. For each sample, the symbol for which a definition should be found is tagged as "</s>SYMBOL</s>". In this way, each sentence is queried once for each mathematical symbol it contains. For example, the following sentence from Zhu et al. (2019)

> *And the top-left corner and the bottom-right corner of the predicted projected box are* $(i - S\hat{o}_{t_{i,j}}, j - S\hat{o}_{l_{i,j}})$ *and* $(i + S\hat{o}_{b_{i,j}}, j + S\hat{o}_{r_{i,j}}])$ *respectively.*

would be split into the following two sentences:

> *And the top-left corner and the bottom-right corner of the predicted projected box are* ***</s>SYMBOL</s>*** *and SYMBOL respectively .*

> *And the top-left corner and the bottom-right corner of the predicted projected box are SYMBOL and* ***</s>SYMBOL</s>*** *respectively .*

dataset	# positive sentences	total terms (terms per sentence)	total defs (defs per sentence)	# equal term and def. counts	# collated terms and defs	term | def IAA
SymDef	1,403	3,290 (**2.34**)	1,713 (**1.22**)	681 (**49%**)	576 (**41%**)	**0.90 | 0.67**
W00	865	959 (1.11)	908 (1.05)	725 (84%)	699 (81%)	- | -
DEFT	**7,311**	7,847 (1.07)	7,262 (0.99)	5,220 (72%)	6,582 (90%)	0.80 | 0.54

Table 2: Column 2 shows the total number of sentences containing at least one term. Column 4 shows the total number of definitions. Columns 5 and 6 show the number of samples containing an equal number of terms and with collated terms and definitions. Column 7 shows the reported Inter-Annotator Agreement scores (DEFT was evaluated using Krippendorf's alpha). Boldface indicates the best value per column.

4.2 Definition Recognition from Target Symbol

After an individual symbol is targeted and split into separate instances, we detect a definition of the target symbol. Our model is built based on the state-of-the-art definition recognition model called Heuristically-Enhanced Deep Definition Extraction (HEDDEx) (Kang et al., 2020). HEDDEx is trained as multi-task learning with two objectives: it first performs slot-tagging using a Conditional Random Field (CRF) sequence prediction model. The model assigns each token in a sentence one of five tags: term ("B-TERM", "I-TERM"), definition ("B-DEF","I-DEF"), or other ("O"). At the same time, a binary classifier is trained to predict a label indicating if the sentence contains a definition.

In detail, after tokenizing the sentences using the ScispaCy[4] pipeline en_core_sci_md (Neumann et al., 2019), we encode input from a Transformer encoder fine-tuned on the task of definition recognition. Following Kang et al. (2020), we choose the best performing Transformer encoder, RoBERTa (Liu et al., 2019) as our main framework. We used the large version of RoBERTa from Huggingface[5] (Wolf et al., 2020). The CRF prediction model we used is torch-crf[6].

We also provide additional syntactic features as input, which are parts of speech, syntactic dependencies, abbreviations, and entities, which were extracted using ScispaCy.

5 Experiments

Datasets The dataset is split randomly into train, dev, and test splits. The full texts of papers are kept together for the test set (i.e., sentences in the test set are not members of papers in the train set). The training set contains 4,930 samples after splitting each sentence into samples according to the number of symbols. The dev and test sets contain 1,442 samples each. The data is managed using PyTorch's dataloader[7](Paszke et al., 2019).

Baselines We trained and tested two span-based information extraction models on our dataset, SciIE (Luan et al., 2018) and DyGIE++ (Wadden et al., 2019). We transformed our dataset into the SciIE format, where TERM and DEF are named entities, and DEFINITION-OF is the relation between coordinated terms and their definitions. Mathematical symbols were masked with SYMBOL, but the models were not pointed towards a targeted symbol. Instead, the models were tasked with extracting multiple TERM and DEFINITION pairs per training sample. Each model's ability to coordinate multiple terms and definitions was measured by looking at its ability to extract DEFINITION-OF relations between the named entities. Details on the setup for these experiments can be found in Appendix B.

We also calculated zero-, one-, and few-shot GPT3 baselines using text-davinci-003 in a question-answer format. For details on the experimental setup and post-processing, see Appendix C.

Training For TaDDEx, we trained RoBERTa large (Liu et al., 2019) on the tokenized samples

[4] `https://allenai.github.io/scispacy/`, Apache License 2.0

[5] `https://huggingface.co/docs/transformers/model_doc/roberta`, Apache License 2.0

[6] `https://github.com/yumoh/torchcrf`, MIT Licence

[7] `https://pytorch.org/docs/stable/data.html`, view license here

"Each word w^t is embedded into a k-dimensional word vector $x_t = W_e[w^t]$, where $W_e \in R^{kxV}$ is a (learned) word embedding matrix, V is the vocabulary size, and notation $W_e[v]$ denotes the v-th column of matrix W_e."

Term	Gold	TaDDEx	HEDDEx	SciIE	DyGIE++	GPT3
w^t	word	word	word	word	word	each word
k	-dimensional	-dimensional	-dimensional	-	-	-dimensional
$x_t = \mathbf{W_e}[w^t]$	k-dimensional word vector	k-dimensional word vector	-	-	-	word vector
$\mathbf{W}_e \in \mathbf{R}^{k \times V}$	(learned) word embedding matrix	learned) word embedding matrix	learned) word embedding matrix	-	-	learned word embedding matrix
V	vocabulary size	vocabulary size	vocabulary size	-	-	vocabulary size
$\mathbf{W_e}[v]$	notation v-th column of matrix $\mathbf{W_e}$	notation v-th column of matrix $\mathbf{W_e}$	notation	-	-	notation
v	-	column	-th column of matrix $\mathbf{W_e}$	-	-	-th column
$\mathbf{W_e}$	matrix	matrix	-	-	-	matrix

Figure 4: An example ground-truth annotation in the test of SymDef: (left) a complex sample including the terms, definitions, and relations between them. (right) Eight ground-truth and predicted term-definition pairs. Exact correct definitions are shown in blue. Nothing output shown as -. From Zhang et al. (2017).

Model		Macro	Term	Def
TaDDEx (ours)	F	**84.82**	**81.54**	**73.56**
	P	82.08	74.83	71.91
	R	**88.04**	89.56	**75.28**
HEDDEx (Kang et al., 2020)	F	64.13	64.63	36.03
	P	64.80	61.68	44.37
	R	64.26	67.87	30.33
SciIE (Luan et al., 2018)	F	63.22	53.16	37.49
	P	84.76	79.53	76.47
	R	54.85	39.92	24.83
DyGIE++ (Wadden et al., 2019)	F	73.92	65.44	57.03
	P	**98.02**	**98.41**	**97.05**
	R	63.12	49.01	40.38
GPT3 (few-shot) (Brown et al., 2020)	F	50.51	66.30	37.22
	P	43.79	50.53	25.06
	R	66.53	**96.39**	72.31

Table 3: Comparison of definition recognition systems on SymDef: F1, precision, and recall scores. The Macro scores were calculated by finding the mean of the individual scores each of the three labels "O", "I-DEF", and "I-TERM". The Term and Definition scores are a binary measure of the system's ability to classify Terms and Definitions.

and syntactic features from the training set for 50 epochs using a batch size of 12, and maximum sequence length of 100. AdamW[8] is used for optimization, with a learning rate of 2e−5 and Adam's epsilon set to 1e−6. These hyperparameter settings were based on the results of the parameter sweep performed for Kang et al. (2020). After each epoch, the model is validated on the dev set, and the model weights are updated upon improved performance. Loss is calculated using cross entropy loss[9].

Evaluation Metrics We used BOI tagging to evaluate model performance, where words in the sample sentence that are not a part of a term or definition are assigned "O", terms are assigned "B-TERM", and definition spans are indicated with the tags "B-DEF" (for the first word in the span) and "I-DEF" (for the remaining words in the span). We ultimately merged the "B-DEF" and "I-DEF" tags. The predicted labeling is compared with the ground truth by calculating the macro F1, precision, and recall scores for three classes "O", "B-TERM", and "I-DEF". We also report the F1, precision, and recall scores for "B-TERM" and "I-DEF" individually. FAll scores were calculated for all models using scikit-learn (Pedregosa et al., 2011).

[8] https://huggingface.co/transformers/v3.0.2/main_classes/optimizer_schedules.html

[9] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

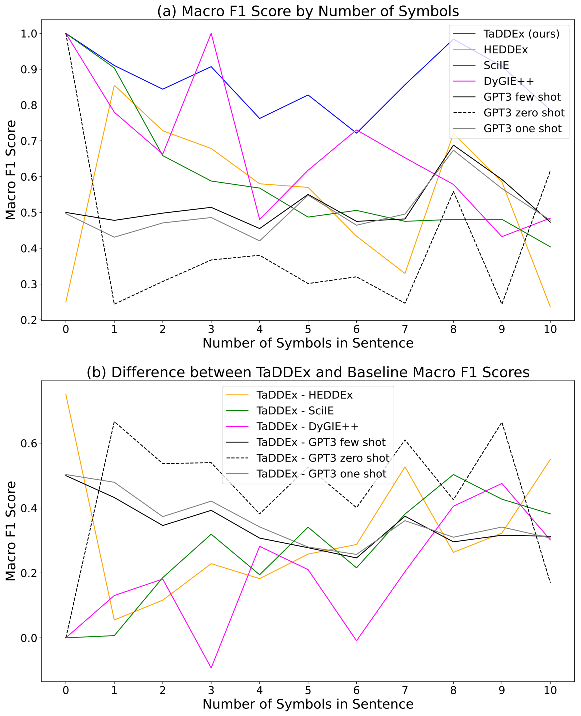


Figure 5: (a) The macro F1 score based on the number of symbols in the sample, and (b) the difference in scores calculated by subtracting baseline F1 scores from TaDDEx.

5.1 Main Results

The evaluation scores can be seen for TaDDEx and the baseline systems in Table 3. Results were generated with a single run. Both IE baseline models were able to extract the named entities TERM and DEF, as well as the relation DEFINITION-OF. See Table 3 for the resulting scores.

Figure 4 shows a sample from the test set containing a complicated coordination. This sample has 8 terms and 8 definitions, some of which are overlapping.

5.2 Error Analysis

Of the 1,442 test samples, our system made incorrect predictions for 135 samples. Of the 135 errors, 28 (20.7%) of them were false negatives, 33 (24.4%) of them were false positives, and 74 (54.8%) were labeled incorrectly. Often, the system's predicted definition overlapped with the ground truth, but added or omitted tokens. Sometimes, the system incorrectly labeled definitions in sentences without a symbol definition.

There was not a strong correlation in terms of system accuracy and total number of symbols in the sample for the TaDDEx model and GPT3 baselines, but HEDDEx, SciIE, and DyGIE++ performed much better for samples with fewer symbols (see Figure 5). All three systems performed perfectly on sentences without a symbol. TaDDEx was least accurate for sentences with six or ten symbols, but did not generally perform worse as the number of symbols increased: the mean macro F1 score for samples with between 1 and 5 symbols was 85.03 with standard deviation ±5.48, and the mean score for samples with between 6 and 10 symbols was 85.11 ± 9.15. SciIE's scores decreased as the number of symbols per sample increased from 0 to 5 symbols, remained stable from 5 to 9 symbols (scores ranging between 47.51 and 50.56), then dropped to 40.39 for ten samples. DyGIE++ assigned"O" to every token, yielding a perfect score for samples with zero symbols, and between 31.84 and 32.84 for all other samples. These results are significant, because they show that the targeted definition recognition method is better at complex term-definition coordination than traditional span-based information extraction techniques.

6 Limitations and Future Work

Having to point the model to the term targeted for definition identification requires prior knowledge of the terms in the dataset. This requires either a dataset with annotated terms such as SymDef, or an initial classification step to extract the terms for each sentence.

Within the domain of our SymDef dataset, terms are restricted to mathematical expressions, which are masked with the single token SYMBOL. One limitation of our model is that it under performs for non-symbolic terms. However, we emphasize that the problem of mathematical symbol definition extraction is important enough that it is appropriate to target an approach specifically to this problem. Furthermore, we believe the inability of information extraction systems such as DyGIE++ and SciIE to adapt to the challenges of SymDef warrants the development of approaches that work specifically for the extraction of mathematical symbol definitions.

7 Potential Risks

A system that surfaces automatically extracted definitions to readers without 100% accuracy will occasionally surface an inaccurate definition. Intelligent reading interfaces that use definition extraction run the risk of providing wrong information to the reader. Furthermore, the "illusion of clarity" that information systems provide can elicit a false sense of complete understanding in human users, which discourages the users from looking more closely (Nguyen, 2021).

8 Conclusion

In this paper we describe the creation of a dataset of 21 scientific papers containing a high concentration of complex term and definition coordinations. We also provide a novel methodology for recognizing multiple coordinated term and definition pairs by targeting one term at a time. Our results show improvement on the span-based approach to relation extraction. Particularly promising is the consistency that our model maintains as the number of symbols per sentence increases.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

K. Bretonnel Cohen, Karin Verspoor, Helen Johnson, Chris Roeder, Philip Ogren, William Baumgartner, Elizabeth White, and Lawrence Hunter. 2009. High-precision biological event extraction with a concept recognizer. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 50–58, Boulder, Colorado. Association for Computational Linguistics.

Xiang Dai. 2018. Recognizing complex entity mentions: A review and future directions. In *Proceedings of ACL 2018, Student Research Workshop*, pages 37–44, Melbourne, Australia. Association for Computational Linguistics.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2020. An effective transition-based model for discontinuous NER. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5860–5870, Online. Association for Computational Linguistics.

Mary Dalrymple and Andrew Kehler. 1995. On the constraints imposed by 'respectively'. *Linguistic Inquiry*, pages 531–536.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016a. Coordination annotation extension in the Penn Tree Bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842, Berlin, Germany. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016b. Improved parsing for argument-clusters coordination. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–76, Berlin, Germany. Association for Computational Linguistics.

Yue Gu, Kangning Yang, Shiyu Fu, Shuhong Chen, Xinyu Li, and Ivan Marsic. 2018. Multimodal affective analysis using hierarchical attention strategy with word-level alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2225–2235, Melbourne, Australia. Association for Computational Linguistics.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, Suntec, Singapore. Association for Computational Linguistics.

Andrew Head, Kyle Lo, Dongyeop Kang, Raymond Fok, Sam Skjonsberg, Daniel S. Weld, and Marti A. Hearst. 2021. Augmenting scientific papers with just-in-time, position-sensitive definitions of terms and symbols. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.

Yiping Jin, Min-Yen Kan, Jun-Ping Ng, and Xiangnan He. 2013. Mining scientific terms and their definitions: A study of the ACL Anthology. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 780–790, Seattle, Washington, USA. Association for Computational Linguistics.

Dongyeop Kang, Andrew Head, Risham Sidhu, Kyle Lo, Daniel Weld, and Marti A. Hearst. 2020. Document-level definition detection in scholarly documents: Existing models, error analyses, and future directions. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 196–206, Online. Association for Computational Linguistics.

Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Mausam, and Soumen Chakrabarti. 2020. OpenIE6: Iterative Grid Labeling and Coordination Analysis for Open Information Extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3748–3761, Online. Association for Computational Linguistics.

Giovanni Kristianto, Minh Nghiem, Nobuo Inui, Goran Topi ́ctopi ́c, and Akiko Aizawa. 2012. Annotating mathematical expression descriptions for automatic detection.

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

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

Roberto Navigli, Paola Velardi, and Juana Maria Ruiz-Martínez. 2010. An annotated dataset for extracting definitions and hypernyms from the web. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.

C. Thi Nguyen. 2021. The seductions of clarity. *Royal Institute of Philosophy Supplements*, 89:227–255.

Philip Ogren. 2010. Improving syntactic coordination resolution using language modeling. In *Proceedings of the NAACL HLT 2010 Student Research Workshop*, pages 1–6, Los Angeles, CA. Association for Computational Linguistics.

Philip Victor Ogren. 2011. *Coordination resolution in biomedical texts*. Ph.D. thesis, University of Colorado at Boulder.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 610–619, Prague, Czech Republic. Association for Computational Linguistics.

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

Yuka Tateisi, Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2008. GENIA-GR: a grammatical relation corpus for parser evaluation in the biomedical domain. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 264–272, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Natalia Vanetik, Marina Litvak, Sergey Shevchuk, and Lior Reznik. 2020. Automated discovery of mathematical definitions in text. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2086–2094, Marseille, France. European Language Resources Association.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 4006–4015. JMLR.org.

Chenchen Zhu, Yihui He, and Marios Savvides. 2019. Feature selective anchor-free module for single-shot object detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 840–849.

A Annotation Guidelines

The annotation goal was to determine which mathematical symbols in a sentence have definitions; to determine the span of the definitions; and to link the symbols with their definitions. Symbols can take a few forms, including the following:

- single letters such as x;
- composite symbols comprising multiple characters:
  - letters with subscripts or superscripts such as x_i^j;
  - function declarations like $f(x, y)$;
  - and derivative deltas and gradients (dx, δx, ΔJ)
- and longer patterns such as sequences, expressions, or formulae:
  - $(x_1, x_2, \ldots x_n)$;
  - $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2 \mathbf{I})$.

A definition is a span of text that declares the meaning of the symbol, beginning and ending on word boundaries. Definitions may provide clarity by showing what a symbol represents, the type of information the symbol represents, what a function does, the elements in a collection represented by a symbol, or what differentiates the symbol from other symbols in the sentence. To help identify definitions, the annotators were asked to mark a span of text as a definition if it answers at least one of the questions in Table 4.

The following constructs may resemble a symbol definition, but did not count as such for this annotation project.

- Equations defining a symbol: "We define x to be $x = a^2 + c$."
- Values assigned to the symbol: "We set x to 5."
- How the symbol is meant to be used: "x is then passed as an argument to function $func(x)$ to compute a score."
- How the symbol is computed: "x is derived by taking the weighted sum of input values."
- The syntactic structure of a phrase implies a meaning without explicitly stating the meaning: "…i^{th} item…" implies that i is an index, but is not explicit so the symbol i does not have a definition.

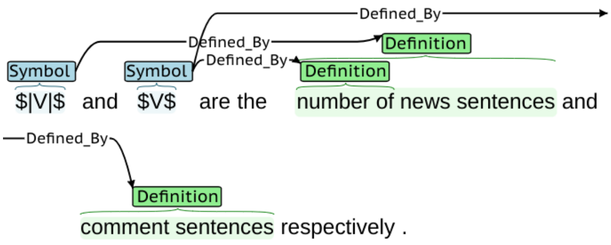

Defined_By
Defined_By
Defined_By
Symbol
Symbol
Definition
Definition
$|V|$ and V are the number of news sentences and
Defined_By
Definition
comment sentences respectively .

Figure 6: A simple example with the keyword "respectively".

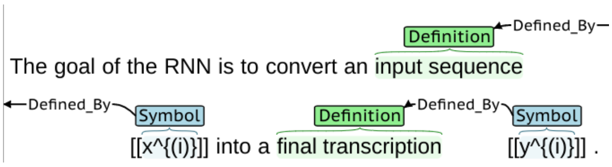

Definition
Defined_By
The goal of the RNN is to convert an input sequence
Defined_By
Symbol
Definition
Defined_By
Symbol
[[x^{(i)}]] into a final transcription [[y^{(i)}]] .

Figure 7: The symbols in this sentence are defined using the appositive structure, where the adjacent nouns "sequence" and "transcription" define them.

Additionally, symbols appearing in a label macro or in a standard math operator such as "log" or "sqrt" should not be annotated.

We provided instructions on how to determine the boundaries of definition spans. In particular, we specified what kinds of words to include in spans, what kinds of words to omit, and how to determine definition spans when the definition contains non-contiguous tokens. See Table 5 for examples.

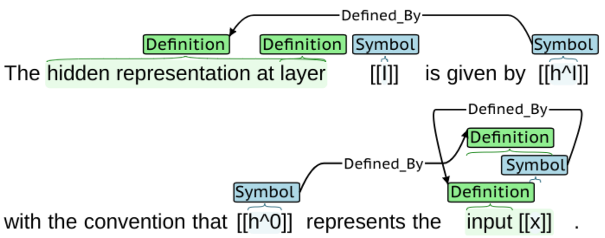

Defined_By
Definition
Definition
Symbol
Symbol
The hidden representation at layer [[l]] is given by [[h^l]]
Defined_By
Definition
Symbol
Defined_By
Symbol
Definition
with the convention that [[h^0]] represents the input [[x]] .

Figure 8: An example of overlapping definitions and symbols. "layer" defines "l", and "hidden representation at layer [[l]]" defines "h^l". "Input" is the definition for "x", and "input[[x]]" defines "h^0".

Question	Sentence	Symbol	Definition
What does this symbol stand for?	“. . . the function f . . . ”	f	“function”
What does this function do?	“$func(x)$ maps a vector to a continuous value.”	$func(x)$	“maps a vector to a continuous value”
What is the information or type of the data this symbol represents?	“. . . the vector x. . . ”	x	“vector”
	“p is a set of programs.”	p	“set of programs”
What are the elements that make up a vector or a set or other collection represented by the symbol?	“Θ contains all parameters of the model”	Θ	“contains all parameters of the model”
What differentiates this symbol from other symbols if there are other related symbols?	“This produces output embeddings E_O from input embeddings E_I.”	E_O	“output embeddings”

Table 4: Questions that help determine whether a candidate definition span is valid.

Guidelines for determining Span Boundaries	Sentence	Symbol	Correct Definition Span
Include multiple definition spans if the definition information is split on either side of the symbol.	“The function f computes an accuracy score.”	f	“function”, “computes an accuracy score”
Include multiple definition spans if there are multiple definitions offering distinct interpretations of the same symbol.	“f, the output function, is a linear model.”	f	“output function”, “linear model”
Include definitions even if they look vague.	“. . . function f. . . ”	f	“function”
Include parentheticals that appear within an otherwise contiguous definition span.	“f is a neural network (NN) for labeling inputs.”	f	“a neural network (NN) for labeling inputs”
Include citations that appear within an otherwise contiguous definition span.	“f is a spectral neural network CITATION for labeling inputs.”	f	“a spectral neural network CITATION for labeling inputs”
For composite symbols, include definitions of the subsymbols that are part of the composite symbol.	“x_i is an element at index i.”	x_i	“an element at index i”
Omit determiners (“the”, “a”, “some”, etc.)	“The function f. . . ”	f	“function”
Omit definition verbs (“is a”, “means”, “denotes”, etc.)	“f is a function.”	f	“function”
Omit information about the dimensionality or length of data	“A is a 3x3 array”	A	“array”
Split definition spans for symbols coordinated with a conjunction	“x and y are the model’s input and output.”	x	“model’s”, “input”
		y	“model’s”, “output”

Table 5: Guidelines for what to include and what to omit from definitions

B Experimental Setup

Hyperparameter	Value
epochs	50
batch size	12
sequence length	100
learning rate	$2e-5$
Adam's epsilon	$1e-6$
optimizer	AdamW
loss function	cross entropy loss

Table 6: TaDDEx and HeDDEx hyperparameters

B.1 SciIE Setup

To reproduce our SciIE experiments, follow these steps:

1. Clone or download the scierc repository.
2. Create the following directory in the scierc folder: "./data/processed_data/json/". Find our dataset in the SciERC format at `anonymous` and copy the train, dev, and test json files to this new directory.
3. Follow the steps in the README under the "Requirements", "Getting Started", and "Setting up for ELMo" headers.
4. Edit the file "experiments.conf": find the experiment called "scientific_best_relation" (at the bottom of the file). Set the coref_weight to 0 and the ner_weight to 1.
5. When running "singleton.py" and "evaluator.py", pass "scientific_best_relation" as a command line argument.
6. Proceed to follow the instructions under "Training Instructions" and "Other Quirks".
7. To compare results to TaDDEx, use the script `anonymous` to convert the output into our format.

B.2 DyGIE++ Setup

To reproduce our DyGIE++ experiments, follow these steps:

1. Clone or download the dygiepp repository.
2. Create a folder in the repository called "data/". Find our dataset in the SciERC format at `anonymous` and copy the train, dev, and test json files to this new directory.
3. Setup your environment with the requirements specified in the README.
4. Navigate to the "training_config" folder and copy "scierc.jsonnet" to a new file called "symdef.jsonnet".
5. Open "symdef.jsonnet" and update "data_paths" so that "train" is set to "data/train.json", "validation" is set to "data/dev.json", and "test" is set to "data/test.json".
6. Run "bash scripts/train.sh symdef" to train the model. To evaluate, follow the instructions in the README under the header "Evaluating a model". You will run a command such as "allennlp evaluate models/symdef/model.tar.gz data/test.json - -include-package dygie - -output-file models/symdef/metrics_test.json"
7. To compare results to TaDDEx, use the script `anonymous` to convert the output into our format.

C GPT3 Experimental Setup

We generated GPT3 baselines using text completion with text-davinci-003 in a question-answer format. We prepared the prompts by concatenating

> Question: given the following sentence,

with the sample sentence, replacing each of N symbols in the sentence with SYMBOL1, SYMBOL2, . . . , SYMBOLN, and appending one of the following based on the number of symbols to the end:

> what are the definitions, if any, of SYMBOL1, SYMBOL2, . . . and SYMBOLN? Answer:

For example, the sentence

> Each word SYMBOL is embedded into a SYMBOL -dimensional word vector SYMBOL , where SYMBOL is a (learned) word embedding matrix , SYMBOL is the vocabulary size , and notation SYMBOL denotes the SYMBOL -th column of matrix SYMBOL .

would be transformed into the following prompt:

> Question: given the following sentence, "Each word SYMBOL1 is embedded into a SYMBOL2 -dimensional word vector SYMBOL3 , where SYMBOL4 is a (learned) word embedding matrix , SYMBOL5 is the vocabulary size , and notation SYMBOL6 denotes the SYMBOL7 -th column of matrix SYMBOL8 .", what are the definitions, if any, of SYMBOL1, SYMBOL2, SYMBOL3, SYMBOL4, SYMBOL5, SYMBOL6, SYMBOL7, and SYMBOL8? Answer:

To reduce the likelihood of GPT3 completing the prompt with text outside of the sample sentence, we set the temperature to 0.0.

C.1 GPT3 One-Shot and Few-Shot Examples

For the one-shot experiments, we prepended the following example to each prompt:

> Question: given the following sentence, “It can be represented as: SYMBOL1 where SYMBOL2 is the bidirectional GRU, SYMBOL3 and SYMBOL4 denote respectively the forward and backward contextual state of the input text.”, what are definitions, if any, of SYMBOL1, SYMBOL2, SYMBOL3, and SYMBOL4?
>
> ANSWER: SYMBOL1 has no definition. SYMBOL2 is defined as bidirectional GRU. SYMBOL3 is defined as forward contextual state of the input text. SYMBOL4 is defined as backward contextual state of the input text.

For the few-shot experiments, we prepended four examples to each prompt:

> Question: given the following sentence, “It can be represented as: SYMBOL1 where SYMBOL2 is the bidirectional GRU, SYMBOL3 and SYMBOL4 denote respectively the forward and backward contextual state of the input text.”, what are definitions, if any, of SYMBOL1, SYMBOL2, SYMBOL3, and SYMBOL4?
>
> ANSWER: SYMBOL1 has no definition. SYMBOL2 is defined as bidirectional GRU. SYMBOL3 is defined as forward contextual state of the input text. SYMBOL4 is defined as backward contextual state of the input text.
>
> Question: given the following sentence, “In general, gradient descent optimization schemes may fail to converge to the equilibrium by moving along the orbit trajectory among saddle points CITATION (CITATION).”, what is the definition, if any, of SYMBOL?
>
> ANSWER: There is no definition.
>
> Question: given the following sentence, “For each target emotion (i.e., intended emotion of generated sentences) we conducted an initial MANOVA, with human ratings of affect categories the DVs(dependent variables) and the affect strength parameter SYMBOL1 the IV (independent variable).”, what is the definition, if any, of SYMBOL1?
>
> ANSWER: SYMBOL1 is defined as affect strength parameter.
>
> Question: given the following sentence, “The CSG program in our example consists of two boolean combinations: union, SYMBOL1 and subtraction SYMBOL2 and two primitives: circles SYMBOL3 and rectangles SYMBOL4, specified by position SYMBOL5, radius SYMBOL6, width and height SYMBOL7, and rotation SYMBOL8.”, what are definitions, if any, of SYMBOL1, SYMBOL2, SYMBOL3, SYMBOL4, SYMBOL5, SYMBOL6, SYMBOL7, and SYMBOL8?
>
> ANSWER: SYMBOL1 is defined as union. SYMBOL2 is defined as subtraction. SYMBOL3 is defined as circles. SYMBOL4 is defined as rectangles. SYMBOL5 is defined as position. SYMBOL6 is defined as radius. SYMBOL7 is defined as height. SYMBOL8 is defined as rotation.

C.2 GPT3 Post-Processing

In order to fairly compare GPT3 to the other models in this study, its output must be reformatted into slot labels. Our post-processing script carries out the following steps:

1. Using regex, chunk the output according to the symbols in the sentence so that there is

one snippet per symbol. The remaining steps are performed for each snippet.

2. Using regex, detect whether the symbol definition is negative ("SYMBOL1 has no definition"). If so, assign all slot labels to “O”.
3. Words in the response that are not found in the sentence are deleted.
4. Words in the response that only occur once in the sentence are automatically labeled as B-DEF (for the first word in the current response snippet) or I-DEF.
5. Responses that have words with multiple occurrences in the sentence are printed and a human indicates which slot should be selected.

Multiple challenges arise in post-processing GPT3’s output. Firstly, the responses often contain words that are not present in the sample text. This occurs when GPT3’s output contains a meta-description of its own output (for example, it might print "It is not possible to say for certain what the definition of SYMBOL1 is. However, SYMBOL1 might be defined as..."). This text can be trimmed out using regex pattern matching. Additional text also occurs when GPT3 provides external information (for example, the presence of the phrase “word embeddings” may trigger GPT3 to provide general information about word embeddings rather than a definition for a symbol in the sentence). Our post-processing script deletes words in the output that are not present in the input, which can mitigate some instances of external information. However, sometimes the GPT3 response contains words that are in the sentence. If the gold label for such words is B-DEF or I-DEF, the GPT-3 score benefits despite the answer being inappropriate. If the gold label is “O”, then the GPT-3 score suffers. Sometimes GPT3 copies a large span of text from the sentence as its answer. Sometimes this results in the GPT3 output containing a “respectively” construct, which is not useful since the purpose of our system is to resolve those constructs. An instance of this from the zero-shot experiments is as follows:

> There are no specific definitions given for any of the symbols in the sentence. However, we can infer that SYMBOL1 refers to a pixel location inside of SYMBOL2, which is likely a box or other geometric shape. SYMBOL3 represents the projected box onto SYMBOL2, and SYMBOL4 is a 4-dimensional vector that encodes the projected box. SYMBOL5, SYMBOL6, SYMBOL7, and SYMBOL8 represent the distances between the current pixel location and the top, left, bottom, and right boundaries of the projected box, respectively.

The post-processing difficulties, lack of consistency in responses, and lack of reliability in terms of truthfulness or appropriateness of responses make GPT3 inference difficult to use in this particular scientific document processing task.

D GPU Usage

This section provides an estimation of GPU usage and the model sizes for TaDDEx and the baseline systems.

Model sizes:

- TaDDEx and HEDDEx are based on RoBERTa-large, which contains 355 million parameters;
- DyGIE++ uses SciBERT (Beltagy et al., 2019) which contains 110 million parameters;
- SciIE uses ELMo (Peters et al., 2018), which contains 93.6 million parameters;
- and GPT3 contains 175 billion parameters.

Training and testing for TaDDEx, HEDDEx, DyGIE++, and SciIE was performed on a single NVIDIA RTX A6000 GPU. Using our training set as input, it takes approximately 3.5 hours to train TaDDEx, 3 hours to train HEDDEx, 6 hours to train DyGIE++, and 6 hours to train SciIE. These models were trained multiple times over the course of this study with an approximate GPU usage between 80 and 100 hours. 3,354 requests were made to GPT3’s text-davinci-003 model, resulting in a total of 741,680 input and output tokens.