Automatic Error Analysis for Document-level Information Extraction

Aliva Das*, **Xinya Du***, **Barry Wang***,
Kejian Shi, Jiayuan Gu, Thomas Porter, Claire Cardie
Department of Computer Science, Cornell University
{ad677, xd75, zw545, ks2325, jg844, tjp78, ctc9}@cornell.edu

Abstract

Document-level information extraction (IE) tasks have recently begun to be revisited in earnest using the end-to-end neural network techniques that have been successful on their sentence-level IE counterparts. Evaluation of the approaches, however, has been limited in a number of dimensions. In particular, the precision/recall/F1 scores typically reported provide few insights on the range of errors the models make. We build on the work of Kummerfeld and Klein (2013) to propose a transformation-based framework for automating error analysis in document-level event and (N-ary) relation extraction. We employ our framework to compare two state-of-the-art document-level template-filling approaches on datasets from three domains; and then, to gauge progress in IE since its inception 30 years ago, vs. four systems from the MUC-4 (1992) evaluation.[1]

1 Introduction

Although information extraction (IE) research has almost uniformly focused on *sentence-level* relation and event extraction (Grishman, 2019), the earliest research in the area formulated the task at the *document level*. Consider, for example, the first large-scale (for the time) evaluations of IE systems — e.g. MUC-3 (1991) and MUC-4 (1992). Each involved a complex document-level event extraction task: there were 24 types of events, over a dozen event arguments (or *roles*) to be identified for each event; documents could contain zero to tens of events, and extracting argument entities (or *role fillers*) required noun phrase coreference resolution to ensure interpretability for the end-user (e.g. to ensure that multiple distinct mentions of the same entity in the output were not misinterpreted as references to distinct entities).

The task was challenging: information relevant for a single event could be scattered across the document or repeated in multiple places; relevant information might need to be shared across multiple events; information regarding different events could be intermingled. In Figure 1, for example, the DISEASE "Newcastle" is mentioned well before its associated event is mentioned (via the triggering phrase "the disease has killed"); that same mention of "Newcastle" must again be recognized as the DISEASE in a second event; and the COUNTRY of the first event ("Honduras") appears only in the sentence describing the second event.

In fact, the problem of document-level information extraction has only recently begun to be revisited (Quirk and Poon, 2017; Jain et al., 2020; Du et al., 2021b,a; Li et al., 2021; Du, 2021; Yang et al., 2021) in part in an attempt to test the power of end-to-end neural network techniques that have been so successful on their sentence-level counterparts.[2] Evaluation, however, has been limited in a number of ways.

First, despite the relative complexity of the task, approaches are only evaluated with respect to their overall performance scores (e.g. precision, recall, and F1). Even though scores at the role level are sometimes included, no systematic analysis or characterization of the types of errors that occur is typically done. The latter is needed to determine strategies to improve performance, to obtain more informative cross-system and cross-genre comparisons, and to identify and track broader advances in the field as the underlying approaches evolve. To date, for example, there has been no attempt to directly compare the error landscape and distribution of

*These authors contributed equally to this work.

[1] Our code for the error analysis tool and its output on different model predictions are available at https://github.com/IceJinx33/auto-err-template-fill/.

[2] See, for example, Zhang et al. (2019), Du and Cardie (2020) and Lin et al. (2020) for within-sentence event extraction; Akbik et al. (2018), and Akbik et al. (2019) for named entity recognition (NER); and Zhang et al. (2018) and Luan et al. (2019) for sentence-level relation extraction.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 3960 - 3975

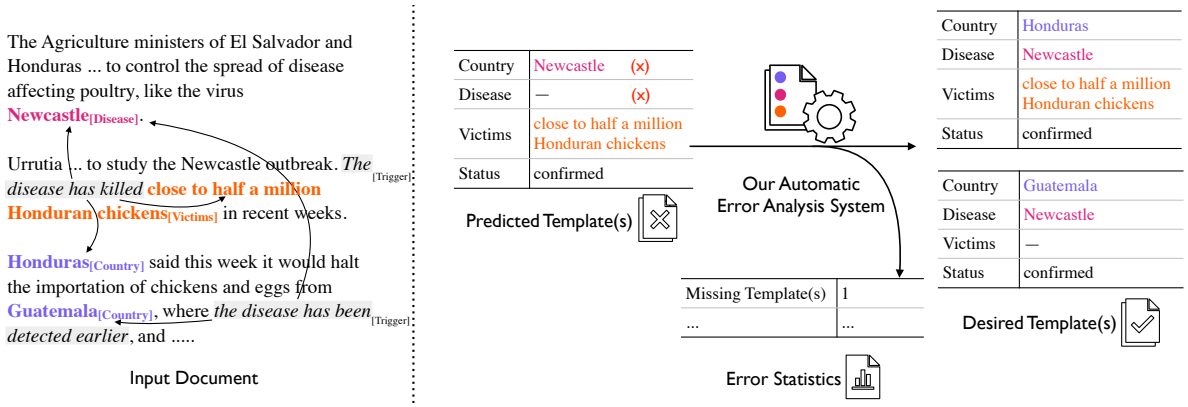

Figure 1: The document-level extraction task from the ProMED dataset on disease outbreaks (left) and the automatic error analysis process (right). Our system performs a set of transformations on the predicted templates to convert them into the corresponding gold standard templates. Transformation steps are mapped to corresponding error types to produce informative error statistics.

newly developed neural IE methods with that of the largely hand-crafted systems of the 1990s.

In this work, we first introduce a framework for automating error analysis for document-level event and relation extraction, casting both as instances of a general role-filling, or *template-filling task* (Jurafsky and Martin, 2021). Our approach converts predicted system outputs into their gold standard counterparts through a series of template-level transformations (Figure 2) and then maps combinations of transformations into a collection of IE-based error types. Examples of errors include duplicates, missing and spurious role fillers, missing and spurious templates, and incorrect role and template assignments for fillers. (See Figure 3 for the full set).

Next, we employ the error analysis framework in a comparison of two state-of-the-art document-level neural template-filling approaches, DyGIE++ (Wadden et al., 2019) and GTT (Du et al., 2021b), across three template-filling datasets (SciREX, ProMED (Patwardhan and Riloff, 2009)[3], and MUC-4).

Finally, in an attempt to gauge progress in the information extraction field over the past 30 years, we employ the framework to compare the performance of four of the original MUC-4 systems with the two newer deep-learning approaches to document-level IE.[4] We find that (1) the best of the early IE models — which strikes a better balance between precision and recall — outperforms modern models that exhibit much higher precision and much lower recall; (2) the modern neural models make more mistakes on scientific vs. news-oriented texts, and missing role fillers is universally the largest source of errors; and (3) modern models have clear advantages over the early IE systems in terms of accurate span extraction, while the early systems make fewer mistakes assigning role fillers to their roles.

2 Related Work

Aside from the original MUC-4 evaluation scoring reports (Chinchor, 1991), which included counts of missing and spurious role filler errors, there have been very few attempts at understanding the types of errors made by IE systems and grounding those errors linguistically. Valls-Vargas et al. (2017) proposed a framework for studying how different errors propagate through an IE system; however, the framework can only be used for pipelined systems, not end-to-end ones.

On the other hand, automated error analysis with linguistically motivated error types has been used in other sub-fields of NLP such as machine-translation (Vilar et al., 2006; Zhou et al., 2008; Farrús et al., 2010; Kholy and Habash, 2011; Zeman et al., 2011; Popović and Ney, 2011), coreference resolution (Uryupina, 2008; Kummerfeld and Klein, 2013; Martschat and Strube, 2014; Martschat et al., 2015) and parsing (Kummerfeld et al., 2012). Recently, generalized automated error analysis frameworks involving human-in-the-loop testing like Errudite (Wu et al., 2019), CHECK-

[3] http://www.promedmail.org

[4] The 1992 model outputs are available in the MUC-4 dataset released by NIST, available at https://www-nlpir.nist.gov/related_projects/muc/muc_data/muc_data_index.html.

LIST (Ribeiro et al., 2020), CrossCheck (Arendt et al., 2021), and AllenNLP Interpret (Wallace et al., 2019) have successfully been applied to tasks like machine comprehension and relation extraction (Alt et al., 2020). Closest to our work are Kummerfeld et al. (2012) and Kummerfeld and Klein (2013), which use model-agnostic transformation-based mapping approaches to automatically obtain error information in the predicted structured output.

3 Template-Filling Task Specification and Evaluation

As in Jurafsky and Martin (2021), we will refer to document-level information extraction tasks as *template-filling tasks* and use the term going forward to refer to both event extraction and document-level relation extraction tasks.

Given a document, D, and an IE template specification consisting of a predetermined list of roles $R_1, R_2, ..., R_i$ associated with each type of relevant event for the task of interest, the goal for template filling is to extract from D, one output template, T for every relevant event/relation $e_1, e_2, \ldots, e_n$ present in the document. Notably, in the general case, $n \geq 0$ and is not specified as part of the input. In each output template, its roles are filled with the corresponding role filler(s), which can be inferred or extracted from the document depending on the predetermined role types. We consider two role types here:[5]

Set-fill roles, which must be filled with exactly one role filler from a finite set supplied in the template specification. An example of a set-fill role in Figure 1 is STATUS, which can be `confirmed`, `possible`, or `suspected`.

String-fill roles, whose role filler(s) are spans extracted from the document, or left empty if no corresponding role filler is found in the document. VICTIMS, DISEASE and COUNTRY are string-fill roles in Figure 1. Some string-fill roles allow multiple fillers; for example, there might be more than one VICTIMS. Importantly, for document-level template filling, exactly one string should be included for each role filler entity (typically a canonical mention of the entity), i.e. coreferent mentions of the same entity are not permitted.

Evaluation. We use the standard (exact-match) F1 score (Chinchor, 1991) to evaluate the output produced by a template-filling system:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4 Methodology: Automatic Transformations for Error Analysis

Similar to the work of Kummerfeld and Klein (2013), our error analysis approach is system-agnostic, i.e. it only uses system output and does not consider intermediate system decisions. This allows for error analysis and comparison across different kinds of systems — end-to-end or pipeline; neural or pattern-based.

Given inputs consisting of the system-predicted templates and gold standard templates (i.e. desired output) for every document in the target dataset, our error analysis tool operates in three steps. For each document,

1. Perform an *optimized mapping* of the associated predicted templates and gold templates.
2. Apply a pre-defined set of *transformations* to convert each system-predicted template into the desired gold template, keeping track of the transformations applied.
3. Map the changes made in the conversion process to an IE-based set of *error types*.

We describe each step in detail in the subsections below.

4.1 Optimized Matching

The first stage of the error analysis tool involves matching each system-predicted template to the best-matching gold template for each document in the dataset. In particular, the overall F1 score for a given document can vary based on how a predicted template is individually matched with a gold template (or left unmatched).

Specifically, for each document, we recursively generate all possible *template matchings* — where each predicted template is matched (if possible) to a gold template. In particular, for a document with P predicted templates and G gold templates, the total number of possible template matchings is:

$$1 + \binom{P}{1}G + \binom{P}{2}G(G-1) + ... + \frac{G!}{(G-P)!}, \text{if } G - P \geq 0$$

$$1 + \binom{P}{1}G + \binom{P}{2}G(G-1) + ... + \binom{P}{G}G!, \text{if } G - P < 0$$

$$= \sum_{i=0}^{min(P,G)} \binom{P}{i} \frac{G!}{(G-i)!}$$

[5] There are potentially more role types depending on the dataset (e.g. normalized dates, times, locations); we will not consider those here.

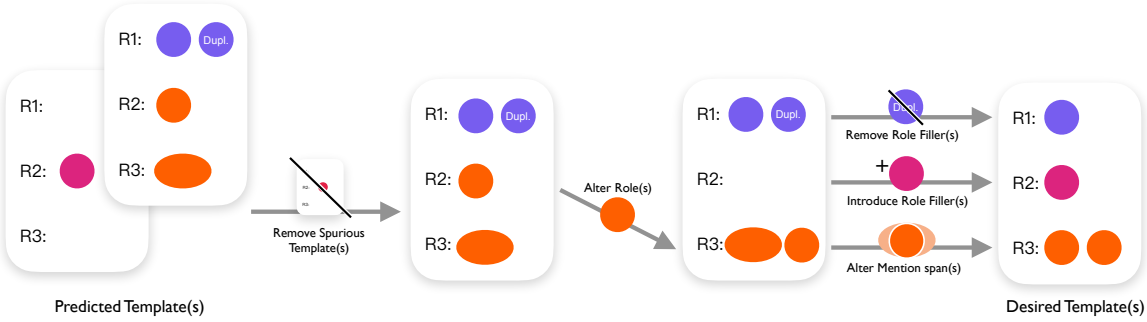

Figure 2: Automatic transformations to convert predicted templates (on the left) to gold templates (on the right). Arrows represent transformations. Colored circles represent role filler entity mentions. *Dupl.* stands for duplicate.

Note that template matching can result in unmatched predicted templates (*Spurious Templates*), as well as unmatched gold templates (*Missing Templates*).

Next, for each predicted-gold pair in a template matching, we iterate through all its roles and recursively generate all possible *mention matchings*, in each of which a predicted role filler is matched (if possible) to a set of coreferent gold role fillers. Similar to template matching, the process of mention matching can also result in unmatched predicted role fillers (*Spurious Role Fillers*) and unmatched coreferent sets of gold role fillers (*Missing Role Fillers*).

Through the process, each predicted role filler increases the denominator of the total precision by 1, and each set of coreferent gold role fillers increases the denominator of total recall by 1. Whenever there is a matched mention pair in which the predicted role filler has an exact match to an element of the set of coreferent gold role fillers, this adds 1 to the numerator of both precision and recall. These counts are calculated for each template matching.

Using precision and recall, the total F1 score across all the slots/roles is calculated and the template matching with the highest total F1 score is chosen. If there are ties, the template matching with the fewest errors is chosen (see Section 4.3).

4.2 Transformations

The second part of the error analysis tool involves changing the predicted templates to the desired gold templates with the help of a fixed set of transformations as detailed below.

- **Alter Span** transforms a role filler into the gold role filler with the lowest *span comparison score* (SCS). The tool provides two options for computing the SCS between two spans, and each depends only on the starting and ending indices of the spans.[6] SCS can be interpreted as distance and is 0 between two identical spans, and 1 for non-overlapping spans. The two modes are given as follows:

  a) *absolute*: This mode captures the (positive) distance between the starting (and ending) character offsets of spans x and y in the document, and scales that value by the sum of the lengths of x and y, capping it at a maximum of 1.

$$SCS = max\left(1, \frac{|x_{start} - y_{start}| + |x_{end} - y_{end}|}{\text{length}(x) + \text{length}(y)}\right)$$

  b) *geometric mean*:

  This mode captures the degree of disjointedness between spans x and y by dividing the length of the overlap between the two spans with respect to each of their lengths, multiplying those two fractions, and subtracting the final result from 1.

  If si is the length of the intersection of x and y, and neither x nor y have length 0, SCS is calculated as shown below; otherwise, SCS is 1.

$$overlap = min(x_{end}, y_{end}) - max(x_{start}, y_{start})$$
$$si = max\,(0, overlap)$$
$$SCS = 1 - \left(\frac{si^2}{\text{length}(x) * \text{length}(y)}\right)$$

  Thus, if the predicted role filler is an exact match for the gold role filler, the SCS is 0. If there is some overlap between the spans, the

[6] This deviates from Kummerfeld and Klein (2013), in which incorrect spans are altered to gold mentions that have the same head token, requiring the use of a syntactic parser.

	Error Type	Error Component		Error Name	Transformations(s)	Predicted	Gold
		Mis-placement	Span Error				
i)	Within Template		■	Span Error	Alter Span	PerInd: [members]	PerpInd: [members of the maoist terrorist organization shining path]
ii)				Duplicate Role Filler	Remove Role Filler	Target: [electrical appliance store], [store]	Target: [electrical appliance store, store]
iii)			■	Duplicate Partially Matched Role Filler	Alter Span + Remove Role Filler	Target: [store], [electrical]	Target: [store, electrical appliance store]
iv)		■ Spurious		Spurious Role Filler	Remove Role Filler	PerpOrg: [fmln] Victim: [rosa imelda gonzalez medrano]	PerpOrg: — Victim: [rosa imelda gonzalez medrano]
v)		□ (Missing)		Missing Role Filler	Introduce Role Filler	Target: —	Target: [local garrison, garrison]
vi)		■ Role Error		Incorrect Role	Alter Role	PerpInd: — Victim: [gonzalo rodriguez gacha]	PerpInd: [gonzalo rodriguez gacha] Victim: —
vii)			■	Incorrect Role + Partially Matched Filler	Alter Span + Alter Role	PerpInd: — Victim: [gonzalo rodriguez]	PerpInd: [gonzalo rodriguez gacha] Victim: —
viii)	Within and Cross Template	■ Template Error		Wrong Template Role Filler	Remove Cross Template Spurious Role Filler	T1:Target: [public bus] T2:Target: —	T1:Target: — T2:Target: [public bus, bus]
ix)			■	Wrong Template For Partially Matched Role Filler	Alter Span + Remove Cross Template Spurious Role Filler	T1:Target: [public] T2:Target: —	T1:Target: — T2:Target: [public bus, bus]
x)		■■ Role + Template Error		Wrong Template + Wrong Role	Alter Role + Remove Cross Template Spurious Role Filler	T1: Victim: — Weapon: — T2: Victim: [adolfo spezua] Weapon: [thomas pellisier]	T1: Victim: [thomas pellisier] Weapon: — T2: Victim: [adolfo spezua] Weapon: —
xi)			■	Wrong Template + Wrong Role + Partially Matched Filler	Alter Span + Alter Role + Remove Cross Template Spurious Role Filler	T1: Victim: — Weapon: — T2: Victim: [adolfo spezua] Weapon: [thomas]	T1: Victim: [thomas pellisier] Weapon: — T2: Victim: [adolfo spezua] Weapon: —
xii)	Template Detection	■ Spurious		Spurious Template	Remove Template	T1: PerpOrg: [fmln]	—
xiii)		□ (Missing)		Missing Template	Introduce Template	—	T1: PerpOrg: [fmln]

Figure 3: Error Types with examples from the MUC-4 dataset. For each template, in every role, the role fillers in brackets refer to the same entity, while role fillers in different brackets refer to different entities. The underlined text indicates the error in the prediction.

SCS is between 0 and 1 (not inclusive), and if there is no overlap between the spans, the SCS is 1. The order of comparison of the spans doesn't change the SCS score for both modes.

As the absolute mode is less sensitive to changes in span indices as compared to the geometric mean, we chose geometric mean for our analysis, as tiny changes in index positions result in a bigger change in the SCS score.

- **Alter Role** changes the role of a role filler to another role within the same template.
- **Remove Duplicate Role Filler** removes a role filler that is coreferent to an already matched role filler.
- **Remove Cross Template Spurious Role Filler** removes a role filler that would be correct if present in another template (in the same role).
- **Remove Spurious Role Filler** removes a role filler that has not been mentioned in any of the gold templates for a given document.

- **Introduce Role Filler** introduces a role filler that was not present in the predicted template but was required to be present in the matching gold template.
- **Remove Template** removes a predicted template that could not be matched to any gold template for a given document.
- **Introduce Template** introduces a template that can be matched to an unmatched gold template for a given document.

For a given document, all singleton **Alter Span** and **Alter Role** transformations, as well as sets of **Alter Span + Alter Role** transformations, are applied first. The other transformations are applied in the order in which they were detected, which is dependent on the order of predicted and gold template pairs in the optimized matching and the order of the slots/roles in the template.

4.3 Error Type Mappings

The transformations in Section 4.2 are mapped onto a set of IE-specific error types as shown in Figure 3. In some cases, a single transformation maps onto a single error, while in others a sequence of transformations is associated with a single error. Full details are in Appendix A.

5 Document-level IE Datasets

Our experiments employ three document-level information extraction datasets. We briefly describe each below. Dataset statistics are summarized in Table 1.

MUC-4 (MUC-4, 1992) consists of newswire describing terrorist incidents in Latin America provided by the FBIS (Federal Broadcast Information Services). We converted the optional templates to required templates and removed the subtypes of the incidents as done in previous work (Chambers, 2013; Du et al., 2021b) so that the dataset is transformed into standardized templates. The roles chosen from the MUC-4 dataset are PERPIND (individual perpetrator), PERPORG (organization perpetrator), TARGET (physical target), VICTIM (human target), and WEAPON which are all string-fill roles, as well as INCIDENT TYPE which is a set-fill role with six possible role fillers: `attack`, `kidnapping`, `bombing`, `arson`, `robbery`, and `forced work stoppage`. As seen in Table 1, 44.59% of the documents have no templates, which makes the identification of relevant vs. irrelevant documents critical to the success of any IE model for this dataset.

ProMED[8] (Patwardhan and Riloff, 2009) consists of just 125 annotated tuning examples and 120 annotated test examples, describing global disease outbreaks by subject matter experts from ProMED. We use the tuning data as training data and reserve 10% of the test data, i.e. 12 examples, to create a development/validation set. 19.83% of the documents in the dataset have no templates. The roles that we extract from the dataset are STATUS, COUNTRY, DISEASE, and VICTIMS. DISEASE, VICTIMS, and COUNTRY are string-fill roles[9]; STATUS is a set-fill role with `confirmed`, `possible`, and `suspected` as the possible role filler options.

SciREX (Jain et al., 2020) consists of annotated computer science articles from Papers with Code[10]. We focus specifically on its 4-ary relation extraction subtask. The roles present in each relation are MATERIAL (DATASET), METRIC, TASK, and METHOD which are all string-fills. We convert the dataset from its original format to templates for our models, and remove individual role fillers (entities) that have no mentions in the text.[11] We also remove any duplicate templates.[12] During preprocessing, we remove malformed words longer than 25 characters, as the majority of these consist of concatenated words that are not present in the corresponding text.

6 IE Modeling Details

In our experiments, we train and test two neural-based IE models, described briefly below, on the MUC-4, ProMED, and SciREX datasets. Note that

[8] `http://www.promedmail.org`

[9] In the ProMED dataset, COUNTRY is a set-fill role, but since countries are explicitly mentioned in most of the documents, we can treat this role as a string-fill.

[10] `https://paperswithcode.com`

[11] According to Jain et. al., around 50% of relations in the SciREX dataset contain one or more role fillers that do not appear in the corresponding text. These relations are removed during evaluation for our end-to-end task. https://github.com/allenai/SciREX/blob/master/README.md

[12] Removing unmentioned entities sometimes eliminates differences between templates. This results in some templates becoming identical or making some templates contain information that is a subset of the information present in another template. Thus, we only keep one of these processed templates.

	# docs (train/val/test/unannot.)	# tokens per doc (min/max/avg.)	# templates per relevant doc (max/avg.)	% docs with 0 templates
- MUC-4 (MUC-4, 1992)	1300 / 200 / 200 / 0	31 / 1695 / 362	14 / 1.61	44.59
- ProMED[7]	125 / 12 / 108 / 4979	57 / 4417 / 621	9 / 1.55	19.83
- SciREX (Jain et al., 2020)	304 / 66 / 66 / 0	1153 / 13155 / 5401	16 / 2.28	0.00

Table 1: Dataset Statistics. A relevant document has one or more templates.

to create the training data for both the DyGIE++ and GTT models, we use the first mention of each role filler in the document as the mention to be extracted.

DyGIE++ with Clustering We use DyGIE++ — a span-based, sentence-level extraction model — to identify role fillers in the document and associate them with certain role types. During training, the maximum span length enumerated by the model is set to 8 tokens as in Wadden et al. (2019) for the SciREX dataset and 11 tokens for the ProMED dataset. We use *bert-base-cased* and *allenai/scibert_scivocab_uncased* for the base BERT and SciBERT models respectively, which both have a maximum input sequence length of 512 tokens.

To aggregate entities detected by DyGIE++ into templates, we use a clustering algorithm. For the SciREX dataset, we adopt a heuristic approach that assumes there is only one template per document, and in that template, we assign the named entities predicted by DyGIE++ for a document to the predicted role types. For the ProMED dataset, we use a different clustering heuristic that ensures that each template has exactly one role filler for the COUNTRY and DISEASE roles, as detailed in the dataset annotation guidelines. Also, since STATUS has the value `confirmed` in the majority of the templates, every template predicted has its STATUS assigned as `confirmed`.

GTT is an end-to-end document-level template-generating model. For the MUC-4 and SciREX datasets, GTT is run for 20 epochs, while for ProMED it is run for 36 epochs, to adjust for the smaller size of the dataset. All other hyperparameters are set as in Du et al. (2021b). We use the same BERT and SciBERT base models as described in the DyGIE++ architecture above, both with a maximum input sequence length of 512 tokens.

The computational budget and optimal hyperparameters for these models can be found in Appendix sections D and E, respectively.

7 Experimental Results and Analysis

We first discuss the results of DyGIE++ and GTT on SciREX, ProMED, and MUC-4; and then examine the performance of these newer neural models on the 1992 MUC-4 dataset vs. a few of the best-performing IE systems at the time.

7.1 DyGIE++ vs. GTT

Table 2 shows the results of evaluating DyGIE++ and GTT on the SciREX, ProMED, and MUC-4 datasets. We can see that **all models perform substantially worse on scientific texts (ProMED, SciREX) as compared to news (MUC-4)**, likely because the model base is pretrained for general-purpose NLP applications (BERT) or there are not enough examples of scientific-style text in the pretraining corpus (SciBERT). In addition, models seem to perform better on the news-style ProMED dataset than the scientific-paper-based long-text SciREX dataset. This can be explained by the fact that all four models handle a maximum of 512 tokens as inputs, while the average length of a SciREX document is 5401 tokens. Thus, a majority of the text is truncated and, hence, unavailable to the models.

Nevertheless, we see an increase in F1 scores for all SciBERT-based models when compared to their BERT counterparts for the SciREX dataset. The same trend is seen for DyGIE++ for ProMED, but not for GTT. This can be explained by the fact that GTT (SciBERT) has more Missing Template errors than GTT (BERT). So even if GTT (SciBERT) performs better on the scientific slot VICTIMS, i.e. it extracts more scientific information, it does not identify relevant events as well as GTT (BERT), reducing the F1 score across the remaining slots.

From the error count results in Figure 4, we see that **GTT makes fewer Missing Template errors than DyGIE++ on the MUC-4 dataset** (86 vs. 97). However, there is no significant difference

	SciREX	ProMED	MUC-4
DyGIE++ (BERT)	22.47%	35.01%	45.79%
DyGIE++ (SciBERT)	25.39%	38.15%	-
GTT (BERT)	21.54%	**44.64%**	**49.00%**
GTT (SciBERT)	**27.68%**	42.96%	-

Table 2: F1 Scores for the Neural Models on SciREX, ProMED, and MUC-4

	Precision	Recall	F1
GE NLToolset	56.69%	**52.09%**	**54.29%**
NYU PROTEUS	34.23%	31.28%	32.69%
SRI FASTUS	48.47%	38.42%	42.86%
UMass CIRCUS	48.62%	39.04%	43.30%
GTT (BERT)	**63.18%**	40.02%	49.00%
DyGIE++ (BERT)	61.90%	36.33%	45.79%

Table 3: Precision, Recall, and F1 scores for models on the MUC-4 dataset. The first four models were developed in 1992, while the last two models are recent and use neural-based methods.

in the number of missing templates between the two models on the ProMED and SciREX datasets. This could be because DyGIE++ is prone to over-generation — there are significantly more Spurious Role Filler and Spurious Template errors as compared to the results of GTT. Since we use heuristics that create templates based on the extracted role fillers, this increases the probability that there was a possible match to a gold template, reducing the number of Missing Template Errors.

We can also conclude that **DyGIE++ is worse at coreference resolution when compared to GTT** as DyGIE++ makes more Duplicate Role Filler errors across all datasets.

Overall, we find that **the major source of error for both GTT and DyGIE++ across all the datasets is missing recall** in the form of Missing Role Filler and Missing Template errors.

7.2 Early IE Models vs. DyGIE++ and GTT

Table 3 presents the precision, recall, and F1 performance on the MUC-4 dataset for early models from 1992 alongside those of the more recent DyGIE++ and GTT models. We summarize key findings below.

The best of the early models (GE NLToolset) performs better than either of the modern models. It does so by doing a better job balancing precision and recall, whereas GTT and DyGIE++ exhibit much higher precision and much lower recall.

Predicted	Gold Match
power lines **along the road**	power lines
enrique ruiz**, retired**	enrique ruiz
maoist shining path group	shining path
group of unidentified individuals **who hurled a bomb ... passing vehicle**	group of unidentified individuals

Table 4: Span Errors in early models. The differences between the predicted mention and its best gold mention match according to our analysis tool are in bold.

The early models have more span errors than the modern DyGIE++ and GTT models. The representative kinds of span errors from the 1992 model outputs are shown in Table 4. One interesting difference between the span errors in the early models and the modern models is that the predicted mentions include longer spans with more information than is indicated in the best gold mention match. Some could be due to errors in dataset annotation; for example, *maoist shining path group* versus *shining path* but a significant number of the span errors occur as the early models seem to extract the entire sentence or clause which contains the desired role filler mention. The modern models tend to leave off parts of the desired spans, and if they do predict larger spans than required, are only off by a few words.

The early models have fewer Missing Template and Missing Role Filler errors as compared to the modern models. However, the former also have more Spurious Template and Spurious Role Filler errors than the latter, indicating these models mitigate the issue of Missing Templates through over-generation.

The early models have fewer Incorrect Role errors as compared to modern models. However, since all the models make relatively few such errors, it suggests that role classification for predicted mentions is not a major problem for modern models.

The main source of error for both early and modern models is missing recall due to missing templates and missing role fillers. This strongly suggests future systems can maximize their performance by being less conservative in

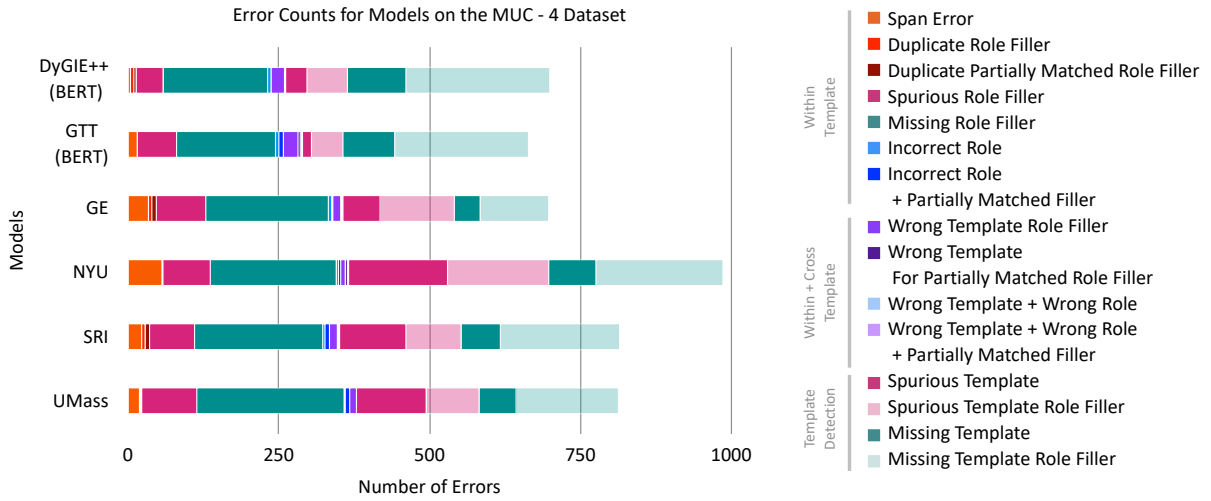

Figure 4: Automated Error Analysis Results (Error Counts) for Models on the MUC-4 dataset.

role filler detection and focusing on improvement of the recall, even at the expense of potentially decreasing some precision.

8 Limitations and Future Work

This work explores subtypes of Spurious Role Filler errors extensively, however, we would like to further analyze Missing Role Filler and template-level errors for more fine-grained error subtypes and the linguistic reasons behind why they occur.

Due to the pairwise comparisons between all predicted and gold mentions in a role for all pairs of predicted and gold templates in an example, the error analysis tool is slow when the number of both the predicted and gold templates as well as the number of role fillers in the templates is high. Thus, we would also like to improve the time complexity of our template (and mention) matching algorithms using an approach like bipartite matching (Yang et al., 2021).

Currently, the error analysis tool reports exact match precision/recall/F1 which is more suitable for string-fill roles. We would like to extend the tool to further analyze set-fill roles by implementing metrics such as false-positive rate.

We used a limited number of models in this paper as we aimed to develop and test the usability of our error analysis tool. In the future, however, we would like to test our tool on a wider range of models, in addition to running more experiments in order to reach more generalizable conclusions about the behavior of IE models.

9 Conclusion

As new models for information extraction continue to be developed, we find that their predicted error types contain insights regarding their shortcomings. Analyzing error patterns within model predictions in a more fine-grained manner beyond scores provided by commonly used metrics is important for the progress of the field. We introduce a framework for the automatic categorization of model prediction errors for document-level IE tasks. We used the tool to analyze the errors of two state-of-the-art models on three datasets from varying domains and compared the error profiles of these models to four of the earliest systems in the field on a dataset from that era. We find that state-of-the-art models, when compared to the earlier manual feature-based models, perform better at span extraction but worse at template detection and role assignment. With a better balance between precision and recall, the best early model outperforms the relatively high-precision, low-recall modern models. Missing role fillers remain the main source of errors, and scientific corpora are the most difficult for all systems, suggesting that improvements in these areas should be a priority for future system development.

Acknowledgments

We thank the anonymous reviewers and Ellen Riloff for their helpful comments(!) and Sienna Hu for converting the 1992 model outputs to a format compatible with our error analysis tool. Our research was supported, in part, by NSF CISE Grant 1815455 and the Cornell CS Department CSURP grants for undergraduate research.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569, Online. Association for Computational Linguistics.

Dustin Arendt, Zhuanyi Shaw, Prasha Shrestha, Ellyn Ayton, Maria Glenski, and Svitlana Volkova. 2021. CrossCheck: Rapid, reproducible, and interpretable model evaluation. In *Proceedings of the Second Workshop on Data Science with Human in the Loop: Language Advances*, pages 79–85, Online. Association for Computational Linguistics.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA. Association for Computational Linguistics.

Nancy Chinchor. 1991. MUC-3 evaluation metrics. In *Third Message Uunderstanding Conference (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*.

Xinya Du. 2021. *Towards More Intelligent Extraction of Information from Documents*. Ph.D. thesis, Cornell University. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021a. GRIT: Generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 634–644, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021b. Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 909–914, Online. Association for Computational Linguistics.

Mireia Farrús, Marta R. Costa-jussà, José B. Mariño, and José A. R. Fonollosa. 2010. Linguistic-based evaluation criteria to identify statistical machine translation errors. In *Proceedings of the 14th Annual conference of the European Association for Machine Translation*, Saint Raphaël, France. European Association for Machine Translation.

Ralph Grishman. 2019. Twenty-five years of information extraction. *Natural Language Engineering*, 25(6):677–692.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. SciREX: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2021. Speech and language processing, 3rd ed. draft, chapter 17, information extraction.

Ahmed Kholy and Nizar Habash. 2011. Automatic error analysis for morphologically rich languages.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Jonathan K. Kummerfeld and Dan Klein. 2013. Error-driven analysis of challenges in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 265–277, Seattle, Washington, USA. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Sebastian Martschat, Thierry Göckel, and Michael Strube. 2015. Analyzing and visualizing coreference resolution errors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 6–10, Denver, Colorado. Association for Computational Linguistics.

Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2070–2081, Doha, Qatar. Association for Computational Linguistics.

MUC-3. 1991. Third Message Understanding Conference (MUC-3): Proceedings of a conference held in San Diego, California, May 21-23, 1991.

MUC-4. 1992. Fourth message understanding conference (MUC-4). In *Proceedings of FOURTH MESSAGE UNDERSTANDING CONFERENCE (MUC-4)*, McLean, Virginia.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160, Singapore. Association for Computational Linguistics.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Computational Linguistics*, 37(4):657–688.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182, Valencia, Spain. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Olga Uryupina. 2008. Error analysis for learning-based coreference resolution. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. 2017. Error analysis in an automated narrative information extraction pipeline. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(4):342–353.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

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.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP interpret: A framework for explaining predictions of NLP models. 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): System Demonstrations*, pages 7–12, Hong Kong, China. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy. Association for Computational Linguistics.

Hang Yang, Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Taifeng Wang. 2021. Document-level event extraction via parallel prediction networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6298–6308, Online. Association for Computational Linguistics.

Daniel Zeman, Mark Fishel, Jan Berka, and Ondrej Bojar. 2011. Addicter: What is wrong with my translations? In *Prague Bull. Math. Linguistics*.

Junchi Zhang, Yanxia Qin, Yue Zhang, Mengchi Liu, and Donghong Ji. 2019. Extracting entities and events as a single task using a transition-based neural model. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5422–5428. ijcai.org.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency

trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Ming Zhou, Bo Wang, Shujie Liu, Mu Li, Dongdong Zhang, and Tiejun Zhao. 2008. Diagnostic evaluation of machine translation systems using automatically constructed linguistic check-points. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1121–1128, Manchester, UK. Coling 2008 Organizing Committee.

A Detailed Error Types Mappings

The specific list of transformations applied in the error correction process.

(1) **Span Error.** Each singleton Alter Span transformation is mapped to a Span Error. A Span Error occurs when a predicted role filler becomes an exact match to the a gold role filer only upon span alteration.

(2) **Duplicate Role Filler.** Each singleton Remove Duplicate Role Filler transformation is mapped to a Duplicate Role Filler error. A Duplicate Role Filler error occurs when a spurious role filler is coreferent to an already matched role filler and is treated as a separate entity. This happens when the system fails at coreference resolution.

(3) **Duplicate Partially Matched Role Filler (Spurious).** Same as (2) above, but with an added Alter Span transformation applied first to account for partial matching. This happens when the system fails at correct span extraction and coreference resolution.

(4) **Spurious Role Filler.** Each singleton Remove Spurious Role Filler transformation is mapped to a Spurious Role Filler error. A Spurious Role Filler error occurs when a mention is extracted from the text with no connection to the gold templates.

(5) **Missing Role Filler.** Each singleton Introduce Role Filler transformation is mapped to a Missing Role Filler error. A Missing Role Filler error occurs when a role filler is present in the gold template but not the predicted template for a given role.

(6) **Incorrect Role.** Each singleton Alter Role transformation is mapped to an Incorrect Role. An Incorrect Role occurs when a spurious role filler is assigned to the incorrect role within the same template, i.e. the role filler would have been correct if present filled in another slot/role in the same template. This happens when the system fails at correct role assignment.

(7) **Incorrect Role + Partially Matched Filler.** Same as (4) above, but with an added Alter Span transformation applied first to account for partial matching. This happens when the system fails at correct span extraction and role assignment.

(8) **Wrong Template for Role Filler.** Each singleton Remove Cross Template Spurious Role Filler transformation is mapped to a Wrong Template for Role Filler error. A Wrong Template for Role Filler occurs when a spurious role filler in one template can be assigned to the correct role in another template, i.e. it would be correct if it had been placed in another template. This happens when the system fails at correct event assignment.

(9) **Wrong Template for Partially Matched Role Filler.** Same as (6) above, but with an added Alter Span transformation applied first to account for partial matching. This happens when the system fails at correct span extraction and event assignment.

(10) **Wrong Template + Wrong Role.** An Alter Role and Remove Cross Template Spurious Role Filler transformation are applied to the same predicted role filler in that order to be mapped to a Wrong Template + Wrong Role error. A Wrong Template + Wrong Role error occurs when a spurious role filler can be assigned to another role in another template. This happens when the system fails at correct role assignment and event assignment.

(11) **Wrong Template + Wrong Role + Partially Matched Filler.** Same as (8) above, but with an added Alter Span transformation applied first to account for partial matching. This happens when the system fails at correct span extraction, role assignment and event assignment.

(12) **Spurious Template.**[13] Each singleton Remove Template is mapped to a Spurious Template error. A Spurious Template error occurs when an extra predicted template is present that cannot be matched to a gold template.

(13) **Missing Template.**[14] Each singleton Introduce Template transformation is mapped to a Missing Template error. A Missing Template error occurs when there is a gold template remaining that has no matching predicted template.

[13]The role fillers in the Spurious Templates are not added to the Spurious Role Filler error counts but are accounted for in the Spurious Template Role Filler counts.

[14]The role fillers in the Missing Templates are not added to the Missing Role Filler error counts but are accounted for in the Missing Template Role Filler counts.

B Example Error Types with ProMED

We also provide example error types with the ProMED dataset.

	Error Types	Transformations(s)	Predicted	Gold
i)	Span Error	Alter Span	Victims: **[young fattening cattle]**	Victims: [young fattening cattle and sheep]
ii)	Duplicate Role Filler	Remove Duplicate Role Filler	Disease: [west nile fever], **[west nile virus]**	Disease: [west nile fever, west nile virus]
iii)	Within Template Incorrect Role	Alter Role	**T1:** Disease: **[2 humans]** Victims: —	**T1:** Disease: — Victims: [2 humans]
iv)	Wrong Template For Role Filler	Remove Cross Template Spurious Role Filler	**T1:** Country: [netherlands] Victims: **[770 cases]**	**T1:** Country: [netherlands] Victims: [its 11th case] **T2:** Country: [united kingdom] Victims: [770 cases]
v)	Spurious Template	Remove Spurious Template	**T1: Country: [china]**	—
vi)	Missing Template	Introduce Missing Template	—	**T1**: Country: [germany] Disease: [fmd] Victims: [2 pigs]

Table 5: Some examples of the Error Types taken from the ProMED dataset. For each template, in every role, the role fillers within brackets refer to the same entity, while role fillers in different brackets refer to different entities. The text in bold black indicates the error in the prediction.

C Precision, Recall, and F1 Scores for All Models on all Three Datasets

We also provide additional precision, recall scores along with the F1 scores.

Models	SciREX	ProMED	MUC-4
DyGIE++ (BERT)	27.85 / 18.83 / 22.47	51.13 / 26.62 / 35.01	61.90 / 36.33 / 45.79
DyGIE++ (SciBERT)	30.47 / 21.76 / 25.39	52.55 / 29.94 / 38.15	-
GTT (BERT)	52.86 / 13.53 / 21.54	68.58 / 33.09 / **44.64**	63.18 / 40.02 / **49.00**
GTT (SciBERT)	53.68 / 18.65 / **27.68**	64.68 / 32.16 / 42.96	-

Table 6: Precision, Recall and F1 Scores (%).

D Computational Budget

The GTT (BERT) model on the MUC-4 dataset took 1 hour and 21 minutes to train and around 11 minutes to test on Google Colab (GPU).

The GTT (BERT) model on the ProMED dataset took around 24 minutes to train and 4 minutes to test, while the GTT (SciBERT) model on the ProMED dataset took around 13 minutes to train and 4 minutes to test, both on Google Colab (GPU). The DyGIE++ (BERT) model on the ProMED dataset took around 50 minutes to train, while the DyGIE++ (SciBERT) model on the ProMED dataset took around 1 hour and 30 minutes to train, both on a NVIDIA V100 GPU.

For the SciREX dataset, it took around 10-20 minutes to run the GTT (BERT) and GTT (SciBERT) models on a NVIDIA V100 GPU. It is worth noting that since the GTT model embeds all inputs before training and SciREX documents are extremely long, more than 25 GB of memory needs to be allocated at the embedding phrase. The training process has normal memory usage. The DyGIE++ (BERT) model took around 2 hours to train, while the DyGIE++ (SciBERT) model took around 4 hours to train, both on a NVIDIA V100 GPU.

Our error analysis tool can be run completely on a CPU and takes a couple of minutes to run, depending on the size of the dataset and the predicted outputs.

E Hyperparameters and Model Configurations

We did not run the DyGIE++ model on the MUC-4 dataset as the model output was made available to us by Xinya Du.

	GTT (BERT)
Hyperparameter Name	Value
number of gpus	1
number of tpu cores	0
max_grad_norm	1.0
gradient_accumulation_steps	1
seed	1
base_model	bert_base_uncased
learning_rate	5e-05
weight_decay	0.0
adam_epsilon	1e-08
warmup_steps	0
num_train_epochs	20
train_batch_size	1
eval_batch_size	1
max_seq_length_src	435
max_seq_length_tgt	75
threshold	80.0

Table 7: GTT on the MUC-4 dataset

	GTT (BERT)	GTT (SciBERT)
Hyperparameter Name	Value	Value
number of GPUs	1	1
number of TPU cores	0	0
max_grad_norm	1.0	1.0
gradient_accumulation_steps	1	1
seed	1	1
base_model	bert_base_uncased	allenai_ scibert_ scivocab_uncased
learning_rate	5e-05	5e-05
weight_decay	0.0	0.0
adam_epsilon	1e-08	1e-08
warmup_steps	0	0
num_train_epochs	36	36
train_batch_size	1	1
eval_batch_size	1	1
max_seq_length_src	435	435
max_seq_length_tgt	75	75
threshold	80.0	80.0

Table 8: GTT Models on the ProMED dataset

	GTT (BERT)	GTT (SciBERT)
Hyperparameter Name	Value	Value
number of GPUs	1	1
number of TPU cores	0	0
max_grad_norm	1.0	1.0
gradient_accumulation_steps	1	1
seed	1	1
base_model	bert_base_uncased	allenai_ scibert_ scivocab_uncased
learning_rate	5e-05	5e-05
weight_decay	0.0	0.0
adam_epsilon	1e-08	1e-08
warmup_steps	0	0
num_train_epochs	20	20
train_batch_size	1	1
eval_batch_size	1	1
max_seq_length_src	435	435
max_seq_length_tgt	75	75
threshold	80.0	80.0

Table 9: GTT Models on the SciREX dataset

	DyGIE++ (BERT)	DyGIE++ (SciBERT)
Hyperparameter Name	Value	Value
number of GPUs	1	1
max_span_width	11	11
base_model	bert_base_cased	allenai_ scibert_ scivocab_cased
learning_rate	5e-04	5e-04
patience	5	5
num_train_epochs	20	20
train_batch_size	32	32
num_dataloader_workers	2	2
max seq length	512	512
ner loss weight	1.0	1.0
relation loss weight	0.0	0.0
coreference loss weight	0.2	0.2
events loss weight	0.0	0.0
target task	ner	ner

Table 10: DyGIE++ Models on the ProMED dataset

	DyGIE++ (BERT)	DyGIE++ (SciBERT)
Hyperparameter Name	Value	Value
number of GPUs	1	1
max_span_width	8	8
base_model	bert_base_cased	allenai_ scibert_ scivocab_cased
learning_rate	5e-04	5e-04
patience	5	5
num_train_epochs	20	20
train_batch_size	32	32
num_dataloader_workers	2	2
max seq length	512	512
ner loss weight	1.0	1.0
relation loss weight	0.0	0.0
coreference loss weight	0.2	0.2
events loss weight	0.0	0.0
target task	ner	ner

Table 11: DyGIE++ Models on the SciREX dataset