An AMR-based Link Prediction Approach for Document-level Event Argument Extraction

Yuqing Yang[1]*†, Qipeng Guo[2]†, Xiangkun Hu[2], Yue Zhang[3], Xipeng Qiu[1]‡, Zheng Zhang[2]
[1]School of Computer Science, Fudan University
[2]Amazon AWS AI, [3]School of Engineering, Westlake University
yuqingyang21@m.fudan.edu.cn, {gqipeng, xiangkhu, zhaz}@amazon.com
xpqiu@fudan.edu.cn, zhangyue@westlake.edu.cn

Abstract

Recent works have introduced Abstract Meaning Representation (AMR) for Document-level Event Argument Extraction (Doc-level EAE), since AMR provides a useful interpretation of complex semantic structures and helps to capture long-distance dependency. However, in these works AMR is used only implicitly, for instance, as additional features or training signals. Motivated by the fact that all event structures can be inferred from AMR, this work reformulates EAE as a link prediction problem on AMR graphs.

Since AMR is a generic structure and does not perfectly suit EAE, we propose a novel graph structure, Tailored AMR Graph (TAG), which compresses less informative subgraphs and edge types, integrates span information, and highlights surrounding events in the same document. With TAG, we further propose a novel method using graph neural networks as a link prediction model to find event arguments.

Our extensive experiments on WikiEvents and RAMS show that this simpler approach outperforms the state-of-the-art models by 3.63pt and 2.33pt F1, respectively, and do so with reduced 56% inference time. The code is available at https://github.com/ayyyq/TARA.

1 Introduction

Event Argument Extraction (EAE) is a long-standing information extraction task to extract event structures composed of arguments from unstructured text (Xiang and Wang, 2019). Event structures can serve as an intermediate semantic representation and be further used for improving downstream tasks, including machine reading comprehension (Han et al., 2021), question answering (Costa et al., 2020), dialog system (Zhang et al., 2020), and recommendation system (Li et al., 2020). Despite the large performance boost by Pre-trained Language Models (PLMs), extracting complex event structures across sentences is still challenging (Ebner et al., 2020).

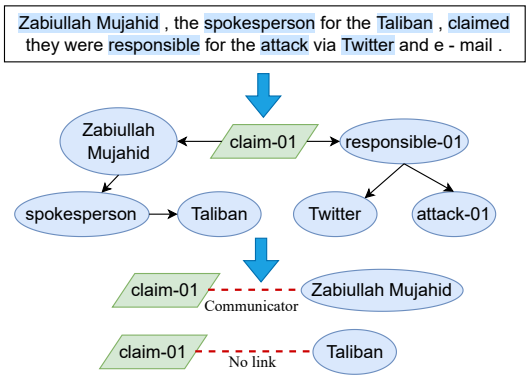

Figure 1: Treating EAE as a link prediction problem, where the green box means the event trigger and blue circles are non-trigger nodes. The highlighted text means they are captured by the graph. We parse the document to a tailored AMR graph and apply a GNN model to predict edges between the trigger and other nodes. In this example, the model predicts one argument, "Zabiullah Mujahid", with the role of "Communicator".

In real-world text, event structures are usually distributed in multiple sentences (Li et al., 2021). To capture cross-sentence and multi-hop structures, Xu et al. (2022) introduces Abstract Meaning Representation (AMR) graphs to assist the model in understanding the document. Their main idea is to take AMR as additional features to enrich span representations. Xu and Huang (2022) and Wang et al. (2021) utilize AMR graphs to provide training signals via self-training and contrastive learning, respectively. These methods exemplify that introducing AMR information facilitates the model's understanding of complex event structures. However, previous works implicitly use AMR information by enriching neural sequential models rather than making explicit use of discrete structures. Intuitively,

*Work done during internship at Amazon Shanghai AI Lab.
†Equal contribution.
‡Corresponding author.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 12876–12889
July 9-14, 2023

discrete AMR structures can force the model to better focus on predicate-argument structures and the content most related to EAE, therefore having stronger effect than implicit AMR.

We aim to exploit the potentials of explicit AMR for improving EAE by formulating EAE as a link prediction task, and Figure 1 illustrates the framework. We parse the input document to a graph structure and adopt a link prediction model to find event arguments. We determine if a node is an argument by whether it is connected to the trigger node or not. The advantages of formulating EAE as a link prediction problem are three-fold: 1) AMR graph is typically more compact than raw text (see Sec-2.2), so processing AMR to find arguments would be simple and efficient. 2) Dependencies among multiple arguments and events are explicitly captured, while previous works (Liao and Grishman, 2010; Du et al., 2022) have pointed out the importance of these dependencies which are only implicitly considered in the feature space. 3) The simpler model architecture and sparse graphs can lead to improvement over efficiency, as our experiments show (up to 56% inference time saving).

The proposed method assumes that AMR graphs contain all necessary information for EAE. However, the original AMR graphs generated by off-the-shelf AMR parsers do not meet this assumption. First, they cover only 72.2% event arguments in WikiEvents, impeding the performance of EAE models directly on the parsed AMR graphs. The primary problem is that AMR graphs are defined at word-level, but an event argument could be a text span. Second, the Smatch score of SOTA AMR parsers is around 85 (Bai et al., 2022), which causes information loss as well. To address the above issue, we propose a novel Tailored AMR Graph (TAG), which compresses information irrelevant to EAE, merges words into text spans via a span proposal module, and highlights the surrounding events in the same document to encourage their communication. Particularly, the number of nodes in TAG equals around 47% of words in WikiEvents, which is a significant reduction. Since too much distracting information is a major challenge of document-level tasks, we also expect performance gains from focusing on TAG, which is evidenced by our experiment results. TAG can cover all EAE samples if the span proposal module adds enough text spans, and we will discuss the trade-off between the recall of spans and model efficiency in Appendix-A.3.

Although there is a large design space for the link prediction model, we choose a simple architecture that stacks GNN layers on top of pre-trained text encoders. The whole model is called TARA for **T**ailored **AMR**-based **A**rgument Extraction. We conduct extensive experiments on latest document-level EAE datasets, WikiEvents (Li et al., 2021) and RAMS (Ebner et al., 2020). TARA achieves 3.63pt and 2.33pt improvements of F1 against the SOTA, respectively. Since interactions in GNN are sparse, the computation cost of our model is also lower, saving up to 56% inference time.

To our knowledge, we are the first to formulate EAE as a link prediction problem on AMR graphs.

2 Methodology

In this section, we first explain how to formulate EAE as a link prediction problem and discuss the benefits of doing so (Sec-2.1). To make AMR graphs better suit the EAE task and ensure the reformulation is lossless, we provide a series of modifications for AMR graphs, resulting in a compact and informative graph, named Tailored AMR Graph (TAG) (Sec-2.2).

2.1 EAE as Link Prediction

Formally, given a document $\mathbf{D}$ and an event trigger τ with its event type e, the goal of Doc-level EAE is to extract a set of event arguments $\mathbf{A}$ related to τ. We formulate EAE as a link prediction problem, which is defined on TAG. Suppose all nodes in TAG are aligned with text spans in the input sequence, triggers and arguments are captured in the graph, and the node corresponding to the event trigger is marked (we will discuss how to satisfy these in Sec-2.2).

Thus, we apply a link prediction model to the tailored AMR graph $\mathcal{G}_t$ of the document $\mathbf{D}$. If the model predicts there is an edge connecting a node u and the event trigger τ with the type r, we say the corresponding text span of u is an argument, and it plays the role r in the event with trigger τ. We illustrate this procedure in Figure 1, and it also shows the tailored AMR graph removes a large amount of distracting information in the input text. Note that the removed text participates in constructing initial node representations, so the model can still access their information as context. Detailed implementation is shown in Sec-2.3.

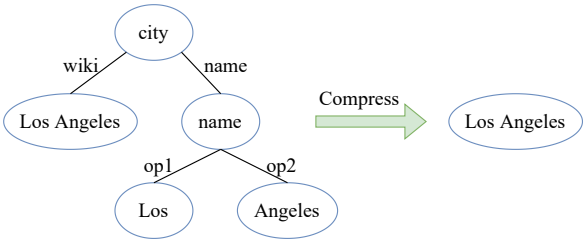

Figure 2: An example of the compression process in Tailored AMR Graph. A subgraph will be replaced by a node with the merged content.

Table 1: Coalescing edge types of TAG, where each *ARGx* is treated as an individual cluster.

Categories	AMR edge types
Spatial	location, destination, path
Temporal	year, time, duration, decade, weekday
Means	instrument, manner, topic, medium
Modifiers	mod, poss
Operators	op-X
Prepositions	prep-X
Core Roles	ARG0, ARG1, ARG2, ARG3, ARG4
Others	*Other AMR edge types*

2.2 Tailored AMR Graph for EAE

TAG can be built on vanilla AMR graphs generated by an off-the-shelf AMR parser (Bai et al., 2022; Astudillo et al., 2020), which also provides the alignment information between nodes and words. As mentioned above, vanilla AMR graphs are insufficient to solve EAE, so we clean the graph by compressing bloated subgraphs, enrich the graph with span boundary information derived by a span proposal module, and highlight the surrounding events to encourage interactions among multiple events.

Coalescing edges We follow previous works (Zhang and Ji, 2021; Xu et al., 2022) and cluster the fine-grained AMR edge types into main categories as shown in Table 1 and parse the document sentence by sentence before fully connecting the root nodes of all the sentences.

Compressing Subgraphs AMR is rigorous and tries to reflect all details as much as possible. For example, Figure 2 shows that a vanilla AMR graph uses five nodes to represent an entity "*Los Angeles*". Since EAE does not require such detailed information, we can compress the subgraph to a single node. We find that about 36% of nodes and 37% of edges can be removed by compression. Note that all incoming and outgoing edges of the subgraph to be compressed will be inherited, so that the compression does not affect the rest of the graph. A streamlined graph not only improves efficiency and saves memory but also promotes the training of GNN since a larger graph often requires a deeper GNN. The compression procedure only relies on the vanilla AMR graph, so it is a one-time overhead for each sample. The detailed compression rules are described in Appendix-B.

Missing Spans The vanilla AMR graph fails to cover span-form arguments since it is defined at the word level, harming the performance on more than 20% of EAE samples. To overcome this issue, we add the span information $\mathbf{S}$, which is generated by a span proposal module, to $\mathcal{G}_t$ as shown in Figure 3. We follow the idea introduced in Zhang and Ji (2021) to merge the generated spans with existing AMR nodes. If a generated span perfectly matches a node's position in the text sequence according to the alignment information, we add a special node-type embedding to the node's initial representation so that the model can know the span proposal module announces this node. If a generated span partially matches a node, we add a new node to represent this span and inherit connectives from the partially matched node. We also add a special edge between this node and the new node to indicate their overlap. If a generated span fails to match any existing nodes, we add a new node and connect it to the nearest nodes to its left and right with a special edge.

Surrounding Events Events in a document are not isolated. A recent work (Du et al., 2022) augments the input with the text that contains other events, but the utilization of AMR graphs offers a simpler solution. We add node-type embeddings to indicate that a node is the current trigger or surrounding event triggers in the same document. This modification encourages communication between multiple event structures, and the consistency between event structures can help to extract as many correct arguments as possible. For example, the *Victim* of an *Attack* event is likely to be the *Victim* of a *Die* event, while less likely to be the *Defendant* of an *ChargeIndict* event in the same document.

2.3 Implementation

We propose a novel model to find event arguments based on TAG, and Figure 3 gives an overview of our method. We first parse the input document with an AMR parser and aligner to obtain the vanilla

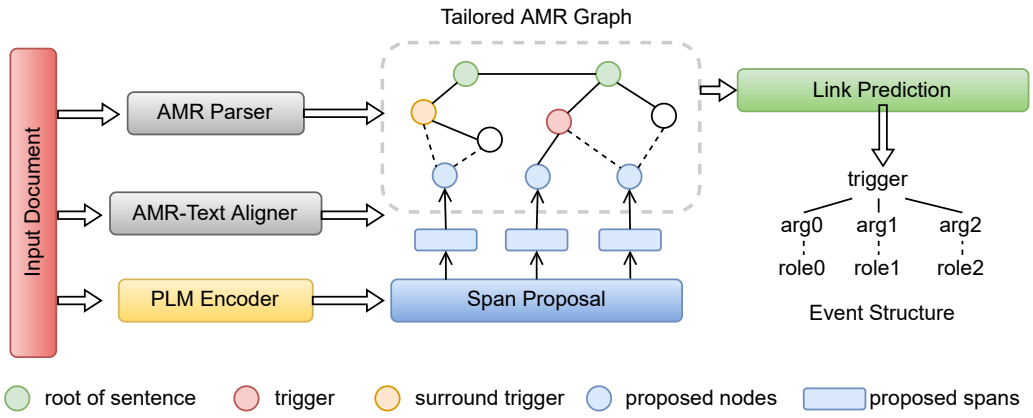

Figure 3: Overview of our method TARA. We adopt an AMR paser and aligner to parse the input document for AMR information, and it is combined with spans generated by a span proposal module to form the Tailored AMR Graph. This graph also contains task-relevant information, which is detailed in Sec-2.2. We further apply a link prediction model on top of the graph to find event arguments.

AMR graph, and coalesce edges and compress subgraphs to preprocess it as described in Sec-2.2. We then enrich the graph with spans generated by a span proposal module. Next, we use token-level features output by a pre-trained text encoder to initialize node representation according to the alignment information. Finally, a GNN-based link prediction model is applied to predict event arguments.

Encoder Module Given an input document $\mathbf{D} = \{w_1, w_2, \ldots, w_n\}$, we first obtain the contextual representation $\mathbf{h}_i$ for each word w_i using a pre-trained language model such as BERT or RoBERTa:

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n] = \text{PLM}([w_1, w_2 \ldots, w_n]).$$

For a text span s_{ij} ranging from w_i to w_j, we follow Xu et al. (2022) to calculate its contextual representation $\mathbf{x}_{s_{ij}}$ by concatenating the start representation $\mathbf{h}_i$, the end representation $\mathbf{h}_j$, and the average pooling of hidden states of the span, which would inject span boundary information. Formally,

$$\mathbf{x}_{s_{ij}} = \mathbf{W}_0 \left[\mathbf{W}_1 \mathbf{h}_i; \mathbf{W}_2 \mathbf{h}_j; \frac{1}{j-i+1} \sum_{t=i}^{j} \mathbf{h}_t \right],$$

where $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2$ are trainable parameters.

Span Proposal Module To find as many arguments as possible, we enumerate all spans up to a length of m. Following Zaporojets et al. (2022), we apply a simple span proposal step to keep only the top-k spans based on the span score $\Phi(s)$ from a feed-forward neural net (FFNN):

$$\Phi(s) = \text{FFNN}(\mathbf{x}_s).$$

Then the generated k candidate spans, tipped as argument spans most likely, will insert to the AMR graph $\mathcal{G}$ to construct our proposed tailored AMR graph $\mathcal{G}_t$. We analyze the influence of the choice of k in Appendix-A.3 on the recall and efficiency.

We also minimize the following binary cross entropy loss to train the argument identification:

$$\mathcal{L}_{span} = -(y \log(\Phi(\mathbf{x})) + (1-y) \log(1-\Phi(\mathbf{x}))),$$

where y is assigned the true label when the offsets of corresponding span match the golden-standard argument span, otherwise, the false label.

AMR Graph Module As introduced in Sec-2.2, the embedding of each node u_s in $\mathcal{G}_t$ is initialized by the aligned span representation $\mathbf{x}_s$ and its type embedding:

$$\mathbf{g}_{u_s}^0 = \text{LayerNorm}(\mathbf{x}_s + \mathcal{T}_{node}(u_s)),$$

where $\mathcal{T}_{node}$ refers to the lookup table about node types, composed of {*trigger, surrounding trigger, candidate span, others*} four types. The newly inserted nodes are connected to their neighbor nodes, which are close in the text sequence, with a new edge type `context`.

We use L-layer stacked R-GCN (Schlichtkrull et al., 2018) to model the interactions among different nodes through edges with different relation types. The hidden states of nodes in $(l+1)^{th}$ layer can be formulated as:

$$\mathbf{g}_u^{l+1} = \text{ReLU}(\mathbf{W}_0^{(l)} \mathbf{g}_u^{(l)} + \sum_{r \in R} \sum_{v \in N_u^r} \frac{1}{c_{u,r}} \mathbf{W}_r^{(l)} \mathbf{g}_v^{(l)}),$$

where R is the clusters of AMR relation types in Table 1, N_u^r denotes the set of neighbor nodes of node u under relation $r \in R$ and $c_{u,r}$ is a normalization constant. $\mathbf{W}_0^{(l)}$, $\mathbf{W}_r^{(l)}$ are trainable parameters.

We concatenate hidden states of all layers and derive the final node representation $\mathbf{g}_u = \mathbf{W}_g[g_u^0; g_u^1; \ldots, g_u^L]$.

Classification Module We perform multi-class classification to predict what role a candidate span plays, or it does not serve as an argument. As mentioned in Sec-2.1, we take the node representation $\mathbf{g}_{u_s}$ and $\mathbf{g}_{u_\tau}$ which denote the aligned candidate span s and trigger τ, respectively. Following Xu et al. (2022), we also concatenate the event type embedding. The final classification representation can be formulated as:

$$\mathbf{z}_s = [\mathbf{g}_{u_s}; \mathbf{g}_{u_\tau}; \mathcal{T}_{event}(e)].$$

We adopt the cross entropy loss function:

$$\mathcal{L}_{cls} = -\sum_s y_s \log P(\hat{r}_s = r_s),$$

where $\hat{r}_s$ is logits obtained by a FFNN on $\mathbf{z}_s$, and r_s is the gold argument role of span s.

We train the model using the multi-task loss function $\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{span}$ with hyperparameter λ. As a result, argument classification can be positively affected by argument identification.

3 Experiments

3.1 Datasets and Evaluation Metrics

We evaluate our model on two commonly used document-level event argument extraction datasets, WikiEvents (Li et al., 2021) and RAMS (Ebner et al., 2020). WikiEvents contains more than 3.9k samples, with 50 event types and 59 argument roles. RAMS is a benchmark that emphasizes the cross-sentence events, which has 9124 annotated events, containing 139 event types and 65 kinds of argument roles. We follow the official train/dev/test split for WikiEvents and RAMS, and leave the detailed data statistics in Appendix-A.1.

For WikiEvents, we evaluate two subtasks of event argument extraction. **Arg Identification**: An argument span is correctly identified if the predicted span boundary match the golden one. **Arg Classification**: If the argument role also matches, we consider the argument is correctly classified. Following Li et al. (2021), we report two metrics, Head F1 and Coref F1. Head F1 measures the correctness of the head word of an argument span, the word that has the smallest arc distance to the root in the dependency tree. For Coref F1, the model is given full credit if the extracted argument is coreferential with the reference as used in Ji and Grishman (2008). In addition, for RAMS dataset, we mainly concern Arg Classification and report the Span F1 and Head F1. For a sufficient comparison, We follow Ma et al. (2022) and additionally evaluate Span F1 for Arg Identification on the test set.

Table 2: **Main results on the WikiEvents test set.** Rows in gray are results of our proposed models that perform best on the development set, and subscripts denote the standard deviation computed from 3 runs. Models under the double line are based on large models with similar model sizes. The best results are in **bold** and the previous best results are underlined.

Model	Arg Identification		Arg Classification	
	Head F1	Coref F1	Head F1	Coref F1
BERT-base				
BERT-CRF	69.83	72.24	54.48	56.72
BERT-QA	61.05	64.59	56.16	59.36
BERT-QA-Doc	39.15	51.25	34.77	45.96
EEQA	-	-	56.9	-
TSAR	75.52	73.17	68.11	66.31
TARA	76.49	74.44	**70.52**	68.47
$\text{TARA}_{\text{compress}}$	**76.76**	**74.88**	70.18	**68.67**
BART-large				
BART-Gen	71.75	72.29	64.57	65.11
PAIE	-	-	68.4	-
EA^2E	74.62	75.77	68.61	69.70
RoBERTa-large				
EEQA	-	-	59.3	-
TSAR	76.62	75.52	69.70	68.79
TARA	$\mathbf{78.64}_{0.16}$	$76.40_{0.23}$	$72.89_{0.27}$	$70.95_{0.23}$
$\text{TARA}_{\text{compress}}$	$78.50_{0.34}$	$\mathbf{76.71}_{0.14}$	$\mathbf{73.33}_{0.41}$	$\mathbf{71.55}_{0.25}$

3.2 Settings

We adopt the transition-based AMR parser proposed by Astudillo et al. (2020) to obtain the AMR graph with node-to-text alignment information, which can achieve satisfactory results for downstream tasks. We also show the performance using another state-of-the-art AMR parser, AMRBART (Bai et al., 2022), in Appendix-A.4. Besides, we use $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{large}}$ provided by huggingface[1] as the backbone. The models are trained with same hyper-parameters as Xu et al. (2022), details listed in Appendix-A.2. Experiments based on base models are conducted on a single Tesla T4 GPU, and large models on 4 distributed Tesla T4 GPU in parallel.

[1] https://huggingface.co/

Table 3: **Main results on the RAMS dataset. Arg-I** denotes Span F1 for the Arg Identification subtask, and other metrics are for the Arg Classification subtask. * indicates results reported by Ma et al. (2022).

Model	Dev		Test		
	Span F1	Head F1	Span F1	Head F1	Arg-I
BERT-base					
BERT-CRF	38.1	45.7	39.3	47.1	-
BERT-CRF$_{TCD}$	39.2	46.7	40.5	48.0	-
Two-Step	38.9	46.4	40.1	47.7	-
Two-Step$_{TCD}$	40.3	48.0	41.8	49.7	-
FEAE	-	-	47.40	-	**53.49**
TSAR	45.23	51.70	**48.06**	55.04	-
TARA	45.81	**53.22**	**48.06**	55.23	52.82
TARA$_{compress}$	**45.89**	53.15	47.43	**55.24**	52.34
BART-large					
BART-Gen	-	-	48.64	57.32	51.2*
PAIE	-	-	52.2	-	56.8
RoBERTa-large					
TSAR	49.23	56.76	51.18	58.53	-
TARA	$50.01_{0.20}$	$58.17_{0.16}$	$\mathbf{52.51}_{0.05}$	$\mathbf{60.86}_{0.12}$	$\mathbf{57.11}_{0.10}$
TARA$_{compress}$	$\mathbf{50.33}_{0.17}$	$\mathbf{58.49}_{0.30}$	$52.28_{0.15}$	$60.73_{0.10}$	$56.91_{0.17}$

3.3 Main Results

We compare our model with several baselines and the following previous state-of-the-art models. (1) QA-based models: **EEQA** (Du and Cardie, 2020b) and **FEAE** (Wei et al., 2021). (2) Generation-based models: **BART-gen** (Li et al., 2021), **PAIE** (Ma et al., 2022), and **EA^{2}E** (Zeng et al., 2022). (3) Span-based models: **TSAR** (Xu et al., 2022). TSAR is the first and sole work utilizing AMR for Doc-level EAE.

Table 2 illustrates the results on the WikiEvents test set. As is shown, our proposed methods consistently outperform previous works with different sized backbone models. TARA$_{compress}$ achieves comparable results with TARA, with more than 30% nodes and edges being pruned, which suggests that the compression process is effective. We compare the better one with other models in the following analysis.

More than 4pt Head F1 for Arg Classification against approaches that do not use AMR indicates the value of deep semantic information. TSAR is the only work to introduce AMR to document-level EAE tasks, but utilizes AMR graphs in an implicit way of decomposing the node representations to contextual representations. The 3.63pt performance gain compared to TSAR shows that our method, which explicitly leverages AMR graphs to perform link prediction, can make better use of rich semantic structures provided by AMR. Besides, EA^{2}E learns event-event relations by augmenting the context with arguments of neighboring events, which may bring noises in the inference iteration, while we simply mark nodes of other event triggers in the graph and yields an improvement of 4.72pt Head F1.

Table 4: Ablation study on the WikiEvents test set based on RoBERTa$_{large}$. "wo" denotes "without".

Model	Arg Identification		Arg Classification	
	Head F1	Coref F1	Head F1	Coref F1
TARA	**78.64**	**76.40**	**72.89**	**70.95**
(a) wo AMR	75.04	73.79	68.94	68.04
(b) implicit AMR	76.34	73.98	70.00	68.36
(c) wo span proposal	70.84	67.71	64.38	61.84
(d) wo surrounding events	77.15	75.76	71.48	70.27
(e) homogeneous graph	77.87	75.88	71.54	69.74
(f) fully-connected graph	76.95	75.30	70.52	69.42

Comparing the identification and classification scores, we find that the performance gain of the latter is always higher, which indicates that our method not only helps the model find more correct arguments but also increases the accuracy of classifying argument roles. Another finding is that our method contributes more to Head F1 instead of Coref F1 in most cases. The main difference between the two metrics is boundary correctness. The result suggests that although our method helps less in identifying the span boundary, it enhances the capability of finding arguments. Our model is less powerful in span boundary identification is reasonable since the span proposal module only takes the textual information, and we will consider upgrading the span proposal module with AMR information in future work.

Similar conclusion can be drawn from Table 3[2], which compares our method with previous works in both dev and test sets of RAMS. Our method achieves new state-of-the-art results using the large model with 2.33pt Head F1 improvement on the test set compared with TSAR, and yields comparable results based on BERT$_{base}$. PAIE manually creates a set of prompts containing event descriptions for each event type, providing additional knowledge which benefits most for classification with numerous classes. In contrast, our method improves up to 0.31/0.31pt Span F1 for Arg Identification/Classification with the help of explicit AMR information.

4 Analysis

4.1 Ablation Study

We perform ablation study to explore the effectiveness of different modules in our proposed model. Table 4 provides results on the WikiEvents test

[2]We did not mark surrounding events for RAMS due to the lack of annotations.

set based on RoBERTa$_{\text{large}}$ when excluding various modules at a time, which helps us answer the following three crucial questions:

What is the effect of explicit AMR graphs? (a): When we throw away the whole AMR graph and depend solely on the contextual representations from PLM to extract arguments, the Head F1 of Arg Classification decreases by a large margin of 3.95pt, due to the lack of deep semantic information provided by AMR. Besides, (b): implicitly utilizing AMR by taking AMR edge classification as an auxiliary task, leads to a performance drop by 2.89pt. It suggests that explicitly using AMR graphs is more practical for document understanding and argument extraction.

What is the effect of tailored AMR graphs for EAE? (c): Once we drop spans that are not aligned with an AMR node, there is a sharp decrease up to 8.51pt Head F1, demonstrating the necessity of span proposal. (d): If we do not mark surrounding event triggers in the AMR graph, the Head F1 gains a rise by 2.54pt compared to (a), but drops by 1.41pt compared to TARA using the unabridged tailored AMR graph, which shows that barely indicating surrounding events benefits to make full use of event-event relations.

What is the effect of heterogeneous graph structures? (e): The removal of different edge types in the AMR graph, causes a slight performance drop by 1.35pt, illustrating the effectiveness of various edge types. In addition, (f): when we further remove the edge relations and replace the graph structure with a fully-connected layer, the performance decreases by 2.37pt. It suggests that the edge relations are also useful. Moreover, we find that (f) outperforms (a) with an improvement of 1.58pt Head F1, which indicates that the node-level representations are more expressive than word-level representations.

4.2 Efficiency

Table 5 reports the efficiency of different models using AMR graphs. TSAR encodes the input document from local and global perspectives and obtains AMR-enhanced representations by fusing contextual and node representations, while TARA directly utilize AMR graphs to perform link prediction. Though the two models share similar model sizes, TARA runs approximately 2 times faster than TSAR and saves up to 53% training time. When compressing the AMR graph in the pre-processing stage, with more than 30% nodes and edges omitted, TARA$_{\text{compress}}$ speeds up further, resulting in 56% inference time saving.

Table 5: Time cost (in seconds) of different AMR-guided models based on RoBERTa$_{\text{large}}$ on the test set of WikiEvents. Experiments are run for one epoch on 4 same Tesla T4 GPUs.

Model	Training Time	Inference Time
TSAR	603.52	33.43
TARA	319.63	15.56
TARA$_{\text{compress}}$	**281.92**	**14.70**

Table 6: Error analysis on the WikiEvents test set based on RoBERTa$_{\text{large}}$.

Model	Missing Head	Overpred Head	Wrong Role	Wrong Span
Baseline	137	110	37	18
TSAR	136	104	34	20
TARA	120	98	33	19

4.3 Error Analysis

To compare different models in greater detail, we explore four specific error types listed in Table 6. *Missing Head* refers to the number of missing arguments, those the model dose not predict that they are arguments, and we only consider the head word to relax the boundary constraint. *Overpred Head* denotes the number of spans that the model falsely assumes that they play a role. Besides, even though the model succeeds to identify the head word of a golden span, it can still assign wrong argument role to it, which we call *Wrong Role*; or it cannot predict the true start and end position of the span, named *Wrong Span*. We suppose extracting coreferential arguments is reasonable. Baseline refers to (a) in Sec-4.1, which performs worse than TSAR and TARA.

As shown in Table 6, TARA misses fewer argument spans compared to TSAR. In addition, while finding more correct argument spans, TARA dose not predict more wrong roles, that is, it will improve more on the Arg Classification subtask. The first three error types are usually attributed to the severe class imbalance problem. With few examples of one label, the model cannot successfully learn the meaning of it and thus is hard to assign it to the true arguments. Moreover, our proposed model does not do better in recognizing correct span boundary considering *Wrong Span*. We observe that most Wrong Span errors result from the inconsistency of the annotation in the dataset, e.g.,

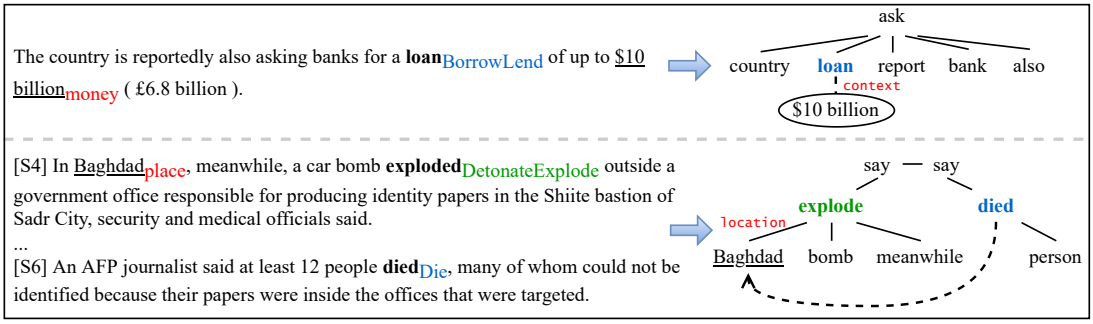

Figure 4: Case study for TAG. The left are excerpted text sequences, with **triggers** in bold, arguments underlined, and event types and argument roles as subscripts. The right are corresponding AMR graphs.

whether articles (such as *the* and *a*), adjectives and quantifiers before a noun should be included to a span.

4.4 Case Study for TAG

In this section, we look into specific examples to explore how tailored AMR graphs work. Firstly, the top part of Figure 4 illustrates the effect of adding missing spans to AMR graphs. Though AMR compresses the text sequence to a deep semantic structure, it may have a different focus from event structures. For instance, "*$10 billion*", which plays an argument role of *Money* for event *BorrowLend*, is left out by the vanilla AMR graph. In contrast, TAG will add the span and successfully serve for link prediction. Additionally, as shown in the bottom part of the figure, there are two events share the same argument "*Baghdad*", and Baseline can not correctly identify the argument for the further event "*died*" while TARA does both right. That is because when indicating surrounding event triggers in the graph, the event "*died*" would pay attention to the subgraph of the event "*explode*" and identify the implicit argument through a closer path in the graph than in the text sequence.

5 Related Work

Doc-level EAE is a frontier direction of Event Extraction and has received broad attention from industry and academia in recent years. Unlike the well-developed sentence-level event extraction (Xi et al., 2021; Ma et al., 2020), the Doc-level EAE faces more challenges. Li et al. (2021) proposes an end-to-end generation-based approach for Doc-level EAE. Fan et al. (2022) and Xu et al. (2021) construct an entity-based graph to model dependencies among the document. Du and Cardie (2020a) chooses the hierarchical method to aggregate information from different granularity.

Recently, there has been a rising trend of utilizing AMR information to assist event extraction. Xu et al. (2022) employs node representations derived by AMR graphs. Lin et al. (2022) and Xu and Huang (2022) introduce AMR path information as training signals to correct argument predictions. Wang et al. (2021) pre-trains the EAE model with a contrastive loss built on AMR graphs. However, previous works have only treated AMR as an auxiliary feature or supervised signal and has not fully exploited the correlation between AMR and EAE. As the scheme of the AMR graph is very similar to the event structure (predicate-arguments vs. trigger-arguments), EAE can be reformulated as an AMR-based problem. With TAG, we can define EAE as a task only related to graphs and conditionally independent of documents, thus achieving a simpler and more efficient model.

Previous works also explore the ways of enriching AMR graphs to suit information extraction tasks. Fan et al. (2022) trains a learnable module to add nodes and edges to the AMR graph. Zhang and Ji (2021) discusses different ways to integrate missing words with the AMR graph. While these methods tend to enlarge AMR graphs, causing a larger graph size and increasing the training difficulty, our method compresses the irrelevant information in AMR to improve efficiency and help the model to be concentrated.

6 Conclusion

We propose to reformulate document-level event argument extraction as a link prediction problem on our proposed tailored AMR graphs. With adding missing spans, marking surrounding events, and removing noises, AMR graphs are tailored to EAE tasks. We also introduce a link prediction model based on TAG to implement EAE. Elaborate exper-

iments show that explicitly using AMR graphs is beneficial for argument extraction.

Limitations

Firstly, as analyzed in Sec-4.3, our proposed method fails to make a significant improvement on span boundary identification. For one thing, the annotation inconsistency in the dataset hinders the model's understanding. For another, our span proposal module leverages the contextual information alone with implicit training signals for span boundary information. We will consider enhancing the span proposal module with AMR information in the future. Secondly, though TARA saves up to 56% inference time compared to the previous AMR-guided work, its entire training requires more than 7h on 4 Tesla T4 GPUs. The bottleneck is the incongruity of pre-trained language models and non-pre-trained GNNs. We leave the problem for future work. Finally, arguments on Wikievents and RAMS are still relatively close to its event trigger (e.g., RAMS limits the scope of arguments in a 5-sentence window), and thus connecting sentence-level AMR graphs is enough to model the long-distance dependency. Otherwise, document-level AMR graphs with coreference resolution are in demand.

Ethics Statement

Our work complies with the ACL Ethics Policy. As document-level event argument extraction is a standard task in NLP, we do not see any critical ethical considerations. We confirm that the scientific artifacts used in this paper comply with their license and intended use. Licenses are listed in Table 7.

Acknowledgement

We would like to express our sincere gratitude to the reviewers for their thoughtful and valuable feedback. This work was supported by the National Key Research and Development Program of China (No.2020AAA0106700) and National Natural Science Foundation of China (No.62022027).

References

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 1001–1007. Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *ACL (1)*, pages 6001–6015. Association for Computational Linguistics.

Tarcísio Souza Costa, Simon Gottschalk, and Elena Demidova. 2020. Event-qa: A dataset for event-centric question answering over knowledge graphs. In *CIKM*, pages 3157–3164. ACM.

Xinya Du and Claire Cardie. 2020a. Document-level event role filler extraction using multi-granularity contextualized encoding. In *ACL*, pages 8010–8020. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020b. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 671–683. Association for Computational Linguistics.

Xinya Du, Sha Li, and Heng Ji. 2022. Dynamic global memory for document-level argument extraction. In *ACL (1)*, pages 5264–5275. Association for Computational Linguistics.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *ACL*, pages 8057–8077. Association for Computational Linguistics.

Siqi Fan, Yequan Wang, Jing Li, Zheng Zhang, Shuo Shang, and Peng Han. 2022. Interactive information extraction by semantic information graph. In *IJCAI*, pages 4100–4106. ijcai.org.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677.

Rujun Han, I-Hung Hsu, Jiao Sun, Julia Baylon, Qiang Ning, Dan Roth, and Nanyun Peng. 2021. ESTER: A machine reading comprehension dataset for reasoning about event semantic relations. In *EMNLP (1)*, pages 7543–7559. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 254–262. The Association for Computer Linguistics.

Manling Li, Alireza Zareian, Ying Lin, Xiaoman Pan, Spencer Whitehead, Brian Chen, Bo Wu, Heng Ji, Shih-Fu Chang, Clare R. Voss, Daniel Napierski, and Marjorie Freedman. 2020. GAIA: A fine-grained multimedia knowledge extraction system. In *ACL (demo)*, pages 77–86. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *NAACL-HLT*, pages 894–908. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *ACL*, pages 789–797. The Association for Computer Linguistics.

Jiaju Lin, Qin Chen, Jie Zhou, Jian Jin, and Liang He. 2022. CUP: curriculum learning based prompt tuning for implicit event argument extraction. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4245–4251. ijcai.org.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jie Ma, Shuai Wang, Rishita Anubhai, Miguel Ballesteros, and Yaser Al-Onaizan. 2020. Resource-enhanced neural model for event argument extraction. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 3554–3559. Association for Computational Linguistics.

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. Prompt for extraction? PAIE: prompting argument interaction for event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6759–6774. Association for Computational Linguistics.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Ziqi Wang, Xiaozhi Wang, Xu Han, Yankai Lin, Lei Hou, Zhiyuan Liu, Peng Li, Juanzi Li, and Jie Zhou. 2021. CLEVE: contrastive pre-training for event extraction. In *ACL/IJCNLP (1)*, pages 6283–6297. Association for Computational Linguistics.

Kaiwen Wei, Xian Sun, Zequn Zhang, Jingyuan Zhang, Zhi Guo, and Li Jin. 2021. Trigger is not sufficient: Exploiting frame-aware knowledge for implicit event argument extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4672–4682. Association for Computational Linguistics.

Xiangyu Xi, Wei Ye, Shikun Zhang, Quanxiu Wang, Huixing Jiang, and Wei Wu. 2021. Capturing event argument interaction via A bi-directional entity-level recurrent decoder. In *ACL/IJCNLP (1)*, pages 210–219. Association for Computational Linguistics.

Wei Xiang and Bang Wang. 2019. A survey of event extraction from text. *IEEE Access*, 7:173111–173137.

Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. 2021. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. In *ACL/IJCNLP (1)*, pages 3533–3546. Association for Computational Linguistics.

Runxin Xu, Peiyi Wang, Tianyu Liu, Shuang Zeng, Baobao Chang, and Zhifang Sui. 2022. A two-stream amr-enhanced model for document-level event argument extraction. In *NAACL-HLT*, pages 5025–5036. Association for Computational Linguistics.

Zhiyang Xu and Lifu Huang. 2022. Improve event extraction via self-training with gradient guidance. *CoRR*, abs/2205.12490.

Klim Zaporojets, Johannes Deleu, Yiwei Jiang, Thomas Demeester, and Chris Develder. 2022. Towards consistent document-level entity linking: Joint models for entity linking and coreference resolution. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 778–784. Association for Computational Linguistics.

Qi Zeng, Qiusi Zhan, and Heng Ji. 2022. Ea^{2}e: Improving consistency with event awareness for document-level argument extraction. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2649–2655. Association for Computational Linguistics.

Tianran Zhang, Muhao Chen, and Alex A. T. Bui. 2020. Diagnostic prediction with sequence-of-sets representation learning for clinical events. In *AIME*, volume 12299 of *Lecture Notes in Computer Science*, pages 348–358. Springer.

Zixuan Zhang and Heng Ji. 2021. Abstract meaning representation guided graph encoding and decoding for joint information extraction. In *NAACL-HLT*, pages 39–49. Association for Computational Linguistics.

Table 7: Licenses of scientific artifacts used in this paper.

Scientific Artifact	License
WikiEvents	MIT License
RAMS	Apache License 2.0
bert-base-uncased	Apache License 2.0
roberta-large	MIT License

Table 8: Statistics of WikiEvents and RAMS datasets.

Dataset	Split	#Docs	#Events	#Arguments
WikiEvents	Train	206	3,241	4,542
	Dev	20	345	428
	Test	20	365	566
RAMS	Train	3,194	7,329	17,026
	Dev	399	924	2,188
	Test	400	871	2,023

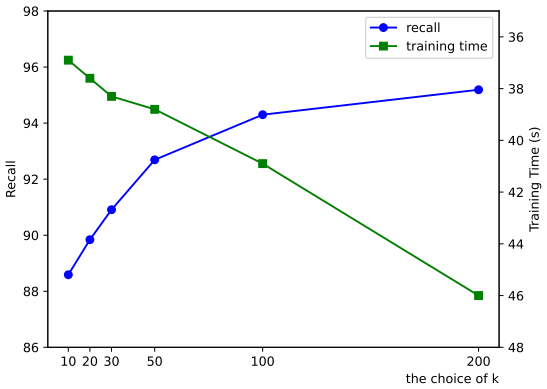

Figure 5: Recall and Training Time with respect to the choice of k. Experiments are run for one epoch on a single Tesla T4 GPU.

A Appendix

A.1 Statistics of Datasets

The details of statistics of WikiEvents and RAMS datasets are listed in Table 8.

A.2 Hyperparameters

We set batch size to 8, and train the model using AdamW (Loshchilov and Hutter, 2019) optimizer and a linearly decaying scheduler (Goyal et al., 2017) with 3e-5 learning rate for pre-trained language encoders and 1e-4 for other modules. For Wikievents, we train the model for 100 epochs, and set λ to 1.0 and L to 3. For RAMS, we train the model for 50 epochs, and set λ to 0.05 and L to 4.

Table 9: Comparison between transition-based AMR parser (abbrev. transition-AMR) and AMRBART on WikiEvents test set based on RoBERTa$_{\text{large}}$.

Model	AMR 2.0	Arg Identification		Arg Classification	
	Smatch	Head F1	Coref F1	Head F1	Coref F1
transition-AMR	81.3	**78.64**	76.40	72.89	70.95
transition-AMR$_{\text{compress}}$	81.3	78.50	**76.71**	**73.33**	**71.55**
AMRBART	**85.4**	78.35	76.29	73.07	70.83

A.3 The choice of k

Span proposal module is of great importance to construct the tailored AMR graph, and intuitively, selecting different number of spans as candidates for Arg Classification will exert an influence on performance and efficiency. Therefore, we present visually the trend of recall and inference time when ranging k, which denotes the number of proposed spans. As illustrated in Figure 5, as k becomes larger, recall is higher, while inference is lower. Moreover, when recall of span proposal is low, a number of positive examples for Arg Classification would be dropped, which impedes the model to learn argument roles. On the other hand, too many candidate spans aggravate the problem of class imbalance. As a consequence, we make a trade-off to set $k = 50$.

A.4 AMR parsers

TARA, as the name implies, relies on automatic AMR parsers to build signals of message passing. To explore the effect of different AMR parsing performance, we compare test results of TARA using transition-based AMR parser and a latest state-of-the-art parser AMRBART (Bai et al., 2022) in Table 9. We implement a simple node-to-text aligner and compress the obtained AMR graph as described in Sec-B for AMRBART. As shown in the table, though AMRBART brings better AMR parsing performance, it dose not gain more improvements for EAE. It demonstrates that there is still a gap between AMR graphs and event structures. Nonetheless, TARA equipped with AMRBART consistently outperforms previous models, which indicates the robustness of our proposed model.

B Subgraph Compression

As mentioned in the main text, we compress the subgraph to make the graph compact. Figure 6 illustrates how we compress a subgraph. Firstly, we will find a subgraph that has an AMR label in pre-defined entity types. The type

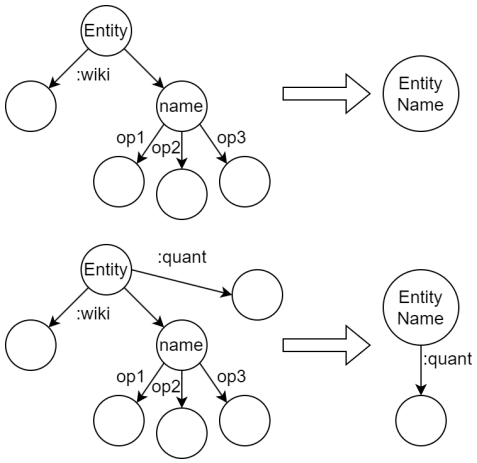

Figure 6: Compression rules. See text for details.

list is induced from the AMR parser configurations, and we also give the list here, *Country, Quantity, Organization, Date-attrs, Nationality, Location, Entity, Misc, Ordinal-entity, Ideology, Religion, State-or-province, Cause-of-death, Title, Date, Number, Handle, Score-entity, Duration, Ordinal, Money, Criminal-charge, Person, Thing, State, Date-entity, Name, Publication, Province, Government-organization, City-district, City, Criminal-organization, Group, Religious-group, String-entity, Political-party, World-region, Country-region, String-name, URL-entity, Festival, Company, Broadcast-program*. If such a node has a child node with the label "name" and outgoing edges like "op1", "op2", we will compress this subgraph. The compression merges labels of all nodes connected with "op1", "op2", "op3" edges as a phrase according to the ascending order of edges. The text alignment information of the merged node becomes the range from the most left position to the most right position of nodes in the subgraph, which means there is a little chance to enlarge the corresponding text span if the original positions are discontinuous. The compression will preserve all incoming and outgoing edges except the edge ":wiki". As shown in the Figure 6, we keep the ":quant" edge but remove the ":wiki" edge.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract; 1. Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Ethics Statement

☑ B1. Did you cite the creators of artifacts you used?
Introduction; 4. Experiments; References

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Ethics Statement

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Ethics Statement

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
4. Experiments

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
4. Experiments; Appendix

C ☑ Did you run computational experiments?

4. Experiments

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4. Experiments; Appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4. Experiments

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4. Experiments

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.