# A Trigger-Sense Memory Flow Framework for Joint Entity and Relation Extraction

Yongliang Shen, Xinyin Ma, Yechun Tang, Weiming Lu
Zhejiang University
{syl,maxinyin,tangyechun,luwm}@zju.edu.cn



## ABSTRACT

Joint entity and relation extraction framework constructs a unified model to perform entity recognition and relation extraction simultaneously, which can exploit the dependency between the two tasks to mitigate the error propagation problem suffered by the pipeline model. Current efforts on joint entity and relation extraction focus on enhancing the interaction between entity recognition and relation extraction through parameter sharing, joint decoding, or other ad-hoc tricks (e.g., modeled as a semi-Markov decision process, cast as a multi-round reading comprehension task). However, there are still two issues on the table. First, the interaction utilized by most methods is still weak and uni-directional, which is unable to model the mutual dependency between the two tasks. Second, relation triggers are ignored by most methods, which can help explain why humans would extract a relation in the sentence. They're essential for relation extraction but overlooked. To this end, we present a **Trigger-Sense Memory Flow Framework (TriMF)** for joint entity and relation extraction. We build a memory module to remember category representations learned in entity recognition and relation extraction tasks. And based on it, we design a multi-level memory flow attention mechanism to enhance the bi-directional interaction between entity recognition and relation extraction. Moreover, without any human annotations, our model can enhance relation trigger information in a sentence through a trigger sensor module, which improves the model performance and makes model predictions with better interpretation. Experiment results show that our proposed framework achieves state-of-the-art results by improves the relation F1 to 52.44% (+3.2%) on SciERC, 66.49% (+4.9%) on ACE05, 72.35% (+0.6%) on CoNLL04 and 80.66% (+2.3%) on ADE.

**ACM Reference Format:**
Yongliang Shen, Xinyin Ma, Yechun Tang, Weiming Lu. 2021. A Trigger-Sense Memory Flow Framework for Joint Entity and Relation Extraction . In *Proceedings of The Web Conference 2021 (WWW '21).* ACM, New York, NY, USA, 12 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

## 1 INTRODUCTION

Entity recognition and relation extraction aim to extract structured knowledge from unstructured text and hold a critical role in information extraction and knowledge base construction. For example, given the following text: *Ruby shot Oswald to death with the 0.38-caliber Colt Cobra revolver in the basement of Dallas City Jail on Nov. 24, 1963, two days after President Kennedy was assassinated.*, the goal is to recognize entities about *People*, *Location* and extract relations about *Kill*, *Located in* held between recognized entities. There are two things of interest to humans when carrying out this task. First, potential constraints between the relation type and the entity type, e.g., the head and tail entities of the *Kill* are of *People* type, and the tail entity of the *Located in* is of *Location* type. Second, triggers for relations, e.g. with words *shot* and *death*, the fact *(Ruby, Kill, Oswald)* can be easily extracted from the above example.

Current entity recognition and relation extraction methods fall into two categories: pipeline methods and joint methods. Pipeline methods label entities in a sentence through an entity recognition model and then predict the relation between them through a relation extraction model [9, 25]. Although it is flexible to build pipeline methods, there are two common issues with these methods. First, they are more susceptible to error prorogation wherein prediction errors from entity recognition can affect relation extraction. Second, they lack effective interaction between entity recognition and relation extraction, ignoring the intrinsic connection and dependency between the two tasks. To address these issues, many joint entity and relation extraction methods are proposed and have achieved superior performance than traditional pipeline methods. In these methods, an entity recognition model and a relation extraction model are unified through different strategies, including constraint-based joint decoding [22, 34], parameter sharing [5, 11, 26], cast as a reading comprehension task [23, 41] or hierarchical reinforcement learning [32]. Current joint extraction models have made great progress, but the following issues still remain:

(1) **Trigger information is underutilized in entity recognition and relation extraction.** Before neural information extraction models, rule-based entity recognition and relation extraction framework were widely used. They were devoted to mine hard template-based rules or soft feature-based rules from text and match them with instances [1–3, 13, 17, 19, 28]. Such methods provide good explanations for the extraction work, but the formulation of rules requires domain expert knowledge or automatic discovery from a large corpus, suffering from tedious data processing and incomplete rule coverage. End-to-end neural network methods have made great progress in the field of information extraction in recent years. To exploit the rules, many works have begun to combine traditional rule-based methods by introducing a neural matching module [24, 35, 43]. However, these methods still need to formulate seed rules or label seed relation triggers manually, and iteratively expand them.
(2) **The interaction between entity recognition and relation extraction is insufficient and uni-directional.** Entity recognition and relation extraction tasks are supposed to


*WWW '21, April 19-23, 2021, Ljubljana, Slovenia*

ACM ISBN 978-1-4503-XXXX-X/18/06.
https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

be mutually beneficial, but joint extraction methods do not take full advantage of dependency between the two tasks. Most joint extraction models are based on parameter sharing, where different task modules share input features or internal hidden layer states. However, these methods usually use independent decoding algorithms, resulting in a weak interaction between the entity recognition module and the relation extraction module. The joint decoding-based extraction model strengthens the interaction between modules, but it requires a trade-off between the richness of features for different tasks and joint decoding accuracy. Other joint extraction methods, such as modeling the task as a reading comprehension problem [23, 41] or a semi-Markov process [32], still suffer from a lack of bi-directional interaction due to the sequential order of subtasks. More specifically, if relation extraction follows entity recognition, the entity classification task will ignore the solution of the relation classification task.

(3) **There is no distinction between the syntactic and semantic importance of words in a sentence.** We note that some words have a significant syntactic role but contribute little to the semantics of a sentence, such as prepositions and conjunctions. While some words are just the opposite, they contribute significantly to the semantics, such as nouns and notional verbs. When encoding context, most methods are too simple to inject syntactic features into the word vector, ignoring the fact that words differ in their semantic and syntactic importance. For example, some methods concatenate part of speech tags of words onto their semantic vectors via an embedding layer [12, 29]. Other methods combine the word, lexical, and entity class features of the nodes on the shortest entity path in the dependency tree to get the final features, which are then concatenated onto the semantic vector [8, 29]. These methods do not distinguish the two roles of a word for sentence semantics and syntax, but rather treat both roles of all words as equally important.

In this paper, we propose a novel framework for joint entity and relation extraction to address the issues mentioned above. First, our model makes full use of relation triggers, which can indicate a specific type of relation. Without any relation trigger annotations, our model can extract relation triggers in a sentence and provide them as an explanation for model predictions. Second, to enhance the bi-directional interaction between entity recognition and relation extraction tasks, we design a Memory Flow Attention module. It stores the already learned entity category and relation category representations in memory. Then we adopt a memory flow attention mechanism to compute memory-aware sentence encoding, and make the two subtasks mutually boosted by enhancing task-related information of a sentence. The Memory Flow Attention module can easily be extended to multiple language levels, enabling the interaction between the two subtasks at both subword-level and word-level. Finally, we distinguish the syntactic and semantic importance of a word in a sentence and propose a node-wise Graph Weighted Fusion module to dynamically fuse the syntactic and semantic information of words.

Our main contributions are as follow:

- Considering the relation triggers, we propose the Trigger Sensor module, which implicitly extracts the relation triggers from a sentence and then aggregates the information of triggers into span-pair representation. Thus, it can improve the model performance and strengthens the model interpretability.
- To model the mutual dependency between entity recognition and relation extraction, we propose the Multi-level Memory Flow Attention module. This module constructs entity memory and relation memory to preserve the learned representations of entity and relation categories. Through the memory flow attention mechanism, it enables the bi-directional interaction between entity recognition and relation extraction tasks at multiple language levels.
- Since the importance of semantic and syntactic roles that words play in a sentence are different, we propose a node-wise Graph Weighted Fusion module to dynamically fuse semantic and syntactic information.
- Experiments show that our model achieves state-of-the-art performance consistently on the SciERC, ACE05, CoNLL04, and ADE datasets, and outperforms several competing baseline models on relation F1 score by 3.2% on SciERC, 4.9% on ACE05, 0.6% on CoNLL04 and 2.3% on ADE.

## 2 RELATED WORK

### 2.1 Rule-based Relation Extraction

Traditional relation extraction methods utilize template-based rules [2, 13, 28], which are first formulated by domain experts or automatically generated from a large corpus based on statistical methods. Then, they apply hard matching to extract the corresponding relation facts corresponding to the rules. Later on, some works change the template-based rules to feature-based rules (such as TF-IDF, CBOW) and extract relations by soft matching [7, 18, 20, 40], but still could not avoid mining the rule features from a large corpus using statistical methods. In short, rule-based relation extraction models typically suffer from a number of disadvantages, including tedious efforts on the rule formulation, a lack of extensibility, and low accuracy due to incomplete rule coverage, but they can provide a new idea for neural relation extraction systems.

Some recent efforts on neural extraction systems attempt to focus on rules or natural language explanations [35]. NERO [43] explicitly exploits labeling rules over unmatched sentences as supervision for training RE models. It consists of a sentence-level relation classifier and a soft rule matcher. The former learns the neural representations of sentences and classifies which relation it talks about. The latter is a learnable module that produces matching scores for unmatched sentences with collected rules. NERO labels sentences according to predefined rules, and makes full use of information from unmatched instances. However, it is still a tedious process to formulate seed rules manually. And the quality of rule-making affects the performance of the entire system.

### 2.2 Joint Entity and Relation Extraction

Previous entity and relation extraction models are pipelined [9, 25]. In these methods, an entity recognition model first recognize entities of interest, and a relation extraction model then predicts the

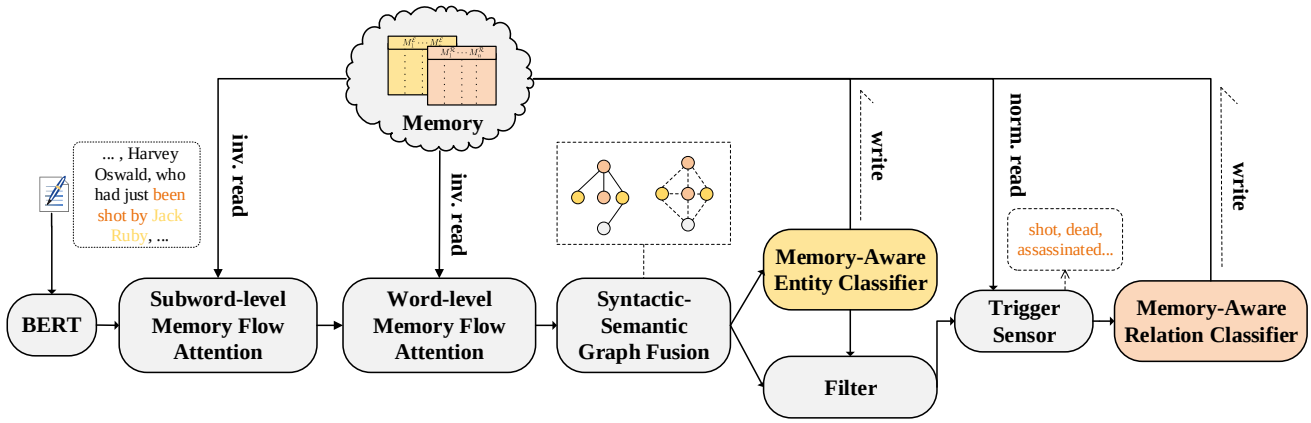


**Figure 1: Trigger-Sense Memory Flow Framework (TriMF) Overview**

relation type between the recognized entities. Although pipeline models have the flexibility of integrating different model structures and learning algorithms, they suffer significantly from error propagation. To tackle this issue, joint learning models have been proposed. They fall into two main categories: parameter sharing and joint decoding methods.

Most methods jointly model the two tasks through parameter sharing [29, 42]. They unite entity recognition and relation extraction modules by sharing input features or internal hidden layer states. Specifically, these methods use the same encoder to provide sentence encoding for both the entity recognition module and the relation extraction module. Some methods [4, 26, 27, 33] perform entity recognition first and then pair entities of interest for relation classification. While other methods [32, 38] are the opposite, they predict possible relations first and then recognize the entities in the sentence. DygIE [27] constructs a span-graph and uses message propagation methods to enhance interaction between entity recognition and relation extraction. HRL [32] models the joint extraction problem as a semi-Markov decision process, and uses hierarchical reinforcement learning to extract entities and relations. CASREL [36] considers the general relation classification as a tagging task. Each relation corresponds to a tagger which recognizes the tail entities based on a head entity and context. CopyMTL [39] casts the extraction task as a generation task and proposes an encoder-decoder model with a copy mechanism to extract relation tuples with overlapping entities. Although entity recognition and relation extraction modules can adopt different structures in these methods, their independent decoding algorithms result in insufficient interaction between the two modules. Furthermore, subtasks are performed sequentially in these methods, so the interaction between two tasks is uni-directional.

To enhance the bi-directional interaction between entity recognition and relation extraction tasks, some joint decoding algorithms have been proposed. [37] proposes to use integer linear planning to enforce constraints on the prediction results of the entity and relation models. [21] uses conditional random fields for both entity and relation models and obtains the output results of the entity and relation by the Viterbi decoding algorithm. Although the joint decoding-based extraction model strengthens the interaction between two modules, it still requires a trade-off between the richness of features required for different tasks and the accuracy of joint decoding.

## 3 TRIGGER-SENSE MEMORY FLOW FRAMEWORK

### 3.1 Framework Overview

In this section, we will introduce the **Trigger-Sense Memory Flow Framework (TriMF)** for joint entity and relation extraction, which consists of five main modules: **Memory** module, **Multi-Level Memory Flow Attention** module, **Syntactic-Semantic Graph Weighted Fusion** module, **Trigger Sensor** module, and **Memory-Aware Classifier** module.

The overall architecture of the TriMF is illustrated in Figure 1. We first initialize the Memory, including an Entity Memory $\mathrm{M}^{\mathcal{E}} \in \mathbb{R}^{n^e \times h_{me}}$ and a Relation Memory $\mathrm{M}^{\mathcal{R}} \in \mathbb{R}^{n^r \times h_{mr}}$, where $n^e$ and $n^r$ denote the number of entity categories and relation categories, $h_{me}$ and $h_{mr}$ denote the slot size of entity memory and the relation memory.

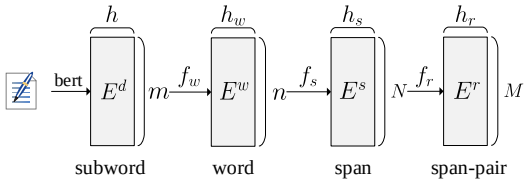


**Figure 2: Four Levels Encoding**

Our model performs a four-level sentence encoding (subword, word, span, and span-pair, as shown in Figure 2) and two-step classification (entity classification and relation classification). More specifically, a sentence is encoded by BERT [10] to obtain subword sequence encoding $\mathrm{E}^d = \mathbb{R}^{m \times h}$, where $m$ denotes the number of subwords in the sentence, and $h$ denotes the hidden state size of

BERT. Based on $\mathrm{M}^{\mathcal{R}}$, $\mathrm{M}^{\mathcal{E}}$ and $\mathrm{E}^{d}$, we perform the first Memory Flow Attention at the subword-level. Then we use $f_w$ to aggregate the subword sequence encoding into a word sequence encoding $\mathrm{E}^{w} = \mathbb{R}^{n \times h_w}$, where $n$ denotes the number of words in the sentence, and $h_w$ denotes the size of the word vector. Here for $f_w$, we adopt the max-pooling function. Based on $\mathrm{M}^{\mathcal{R}}$, $\mathrm{M}^{\mathcal{E}}$ and $\mathrm{E}^{w}$, we perform the second Memory Flow Attention at the word-level. After that, the word sequence encoding is fed into the Syntactic-Semantic Graph Weighted Fusion module to fuse semantic and syntactic information at word-level. Then, we combine the word sequence encodings by $f_s$ to obtain the span sequence encodings $\mathrm{E}^{s} = \mathbb{R}^{N \times h_s}$, where $N$ denotes the number of spans in the sentence, and $h_s$ denotes the size of the span vector. Here for $f_s$, we adopt a method of concatenating a span-size embedding on max-pooled word embeddings. We filter out the spans which are classified as the *None* category by a Memory-Aware Entity Classifier. After pairing the spans of interest, We compute local-context representation $g_{local}$ and full-contextual span-pair specific trigger representation $\mathrm{g}_{trigger}$ using the Trigger Sensor. We combine the encodings of the head span, tail span, $g_{local}$ and $\mathrm{g}_{trigger}$ to obtain the encoding $\mathrm{E}^{r} \in \mathbb{R}^{M \times h_r}$, where $\mathrm{E}^{r}_{(ij)}$ denotes the span pair encoding consisting of the $i^{th}$ and $j^{th}$ spans, $M$ denotes the number of candidate span pairs, and $h_r$ denotes the size of the span pair encoding.

Lastly, we input the candidate span-pair representation to the Memory-Aware Relation Classifier and predict the relation type between the two spans.

In the next sections, we'll cover five main modules of our model in detail.

## 3.2 Memory

Memory holds category representations learned from historical training examples, consist of entity memory and relation memory. Each slot of these two memories indicates an entity category and a relation category respectively. The category representation is hold in the corresponding memory slot, which can be used by Memory Flow Attention module to enhance information related to the tasks in a sentence, or by Trigger Sensor module to sense triggers.

On the Memory module, we define two types of processes, Memory Read Process and Memory Write Process, to manipulate the memory.

**Memory Read Process** Given an input E and our memory M, we define two processes to read memory: *normal read process* and *inverse read process*. The normal read process takes the input as *query*, the memory as *key* and *value*. First, we calculate the attention weights of the input E on the memory M by bilinear similarity function, and then we weight the memory by the weights.

$$\mathrm{A}_{norm}\left(\mathrm{E}, \mathrm{M}\right) = \mathrm{softmax}\left(\frac{\mathrm{EWM}^{T}}{\sqrt{h}}\right) \tag{1}$$

$$\mathrm{Read}_{norm}\left(\mathrm{E}, \mathrm{M}\right) = \mathrm{A}_{norm}\left(\mathrm{E}, \mathrm{M}\right)\mathrm{M} \tag{2}$$

where W is a learnable parameter for the bilinear attention mechanism. While the inverse read process takes the memory as *query*, the input as *key* and *value*. We first compute 2d-attention weight matrix through bilinear similarity function, and then sum the 2d-attention weight matrix on the memory-slot dimension to obtain a 1d-attention weight vector on the input E. The more relevant element in input with the memory has a larger weight. We then multiply the 1d-attention weight vector with E to get a memory-aware sequence encoding:

$$\mathrm{A}_{inv}\left(\mathrm{E}, \mathrm{M}\right) = \sum_{i=1}^{|\mathrm{M}|} \mathrm{softmax}\left(\frac{\mathrm{M}_i \mathrm{WE}^{T}}{\sqrt{h}}\right) \tag{3}$$

$$\mathrm{Read}_{inv}\left(\mathrm{E}, \mathrm{M}\right) = \mathrm{A}_{inv}\left(\mathrm{E}, \mathrm{M}\right)\mathrm{E} \tag{4}$$

where W is a learnable parameter for the bilinear attention mechanism and |M| denotes the number of slots in the memory M.

**Memory Write Process** We write entity memory using gradients of entity classification losses and write relation memory using gradients of relation classification losses. If the gradient of current instance's classification loss is large, it means that the classified instance (span or span-pair) representation is far away from the corresponding memory slot (entity or relation category representation of ground truth) while closer to the memory slots of the other categories, and we need assign a large weight to this instance when writing it into memory. This makes the representations of the categories stored in memory more accurate. The write process for entity memory and relation memory is described below:

$$\mathrm{M}^{\mathcal{E}}_{e} = \mathrm{M}^{\mathcal{E}}_{e} - \mathrm{E}^{s}_{i}\mathrm{W}^{e}\frac{\partial \mathcal{L}^{e}}{\partial logit_e} lr \tag{5}$$

$$\mathrm{M}^{\mathcal{R}}_{r} = \mathrm{M}^{\mathcal{R}}_{r} - \mathrm{E}^{r}_{(ij)}\mathrm{W}^{r}\frac{\partial \mathcal{L}^{r}}{\partial logit_r} lr \tag{6}$$

$$logit_e = log\left(\frac{p(s_i = e)}{1 - p(s_i = e)}\right) \tag{7}$$

$$logit_r = log\left(\frac{p(r_{ij} = r)}{1 - p(r_{ij} = r)}\right) \tag{8}$$

where $\mathcal{L}^{e}$ and $\mathcal{L}^{r}$ denote entity classification loss and relation classification loss, $lr$ denotes the learning rate, $\mathrm{W}^{e}$ and $\mathrm{W}^{r}$ are two weight matrices, $p(s_i = e)$ denotes the probability of span $s_i$ belonging to entity type $e$, $p(r_{ij} = r)$ denotes the probability of span-pair's relation $r_{ij}$ belonging to relation type $r$, and $\mathrm{E}^{s}_{i}$, $\mathrm{E}^{r}_{ij}$ denote candidate span and span-pair encoding, respectively. The above symbols are specifically defined in defined at Sec.3.6.

## 3.3 Multi-level Memory Flow Attention

We perform a memory flow attention mechanism between the memory and the input sequence to enhance task-relevant information, such as entity surface names and trigger words. Entity memory and relation memories can enhance entity-related and relation-related information in the input instance for the two tasks respectively, thus they can help to strengthen bi-directional interaction between tasks.

**Memory Flow Attention** In order to enhance the task-relevant information in a sentence, we designed the Memory Flow Attention based on the Memory. Given a memory M and a sequence encoding E, We calculate the memory-aware sequence encoding by runing *memory inverse read process*:

$$\mathrm{MFA}_{s}\left(\mathrm{E}, \mathrm{M}\right) = \mathrm{Read}_{inv}\left(\mathrm{E}, \mathrm{M}\right) \tag{9}$$

A single memory flow can be extended to multiple memory flows. We consider two types in our work: relation memory flow and entity memory flow. So we design a Multi-Memory Flow Attention mechanism, which is calculated as follows:

$$\mathrm{MFA}_m(\mathrm{E}, \mathrm{M}^{\mathcal{R}}, \mathrm{M}^{\mathcal{E}}) = \mathrm{mean}\left(\mathrm{MFA}_s\left(\mathrm{E}, \mathrm{M}^{\mathcal{R}}\right), \mathrm{MFA}_s\left(\mathrm{E}, \mathrm{M}^{\mathcal{E}}\right)\right) \tag{10}$$

where $\mathrm{M}^{\mathcal{E}}$ and $\mathrm{M}^{\mathcal{R}}$ denote entity and relation memory respectively. we know that languages are hierarchical, and different levels represent semantic information at different levels of granularity. As shown in Figure 3, we extend the multi-memory flow attention mechanism to multiple levels ( subword-level and word-level ), and design a Multi-Level Multi-Memory Flow Attention mechanism:

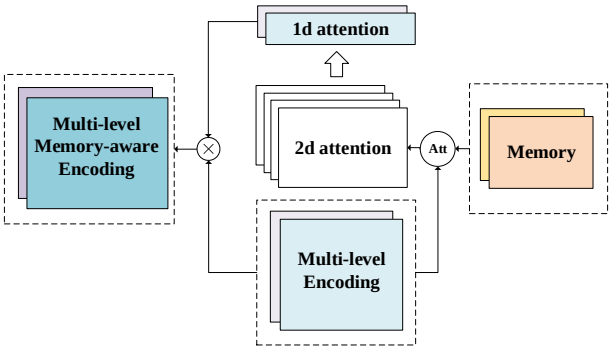


**Figure 3: Multi-Level Multi-Memory Flow Attention**

$$\overline{\mathrm{E}}^{d} = \mathrm{MFA}_m(\mathrm{E}^{d}, \mathrm{M}^{r}, \mathrm{M}^{e}) \tag{11}$$

$$\mathrm{E}^{w} = f_w\left(\overline{\mathrm{E}}^{d}\right) \tag{12}$$

$$\overline{\mathrm{E}}^{w} = \mathrm{MFA}_m(\mathrm{E}^{w}, \mathrm{M}^{r}, \mathrm{M}^{e}) \tag{13}$$

where $\overline{\mathrm{E}}^{d}$ and $\overline{\mathrm{E}}^{w}$ denote memory-aware sequence encoding at subword-level and word-level respectively.

## 3.4 Syntactic-Semantic Graph Weighted Fusion

The semantic information and syntactic structure of a sentence are important for both entity recognition and relation extraction. We consider both by constructing semantic and syntactic graphs from a sentence, with nodes in the graph refer to words in the sentence. We update a node representation based on its neighbor nodes' representations and the graph structure in the two graphs. We note that some words have a significant syntactic role but contribute little to the semantics of a sentence, such as prepositions and conjunctions. While some words are just the opposite, they contribute significantly to the semantics, such as nouns and notional verbs. Therefore, we need to fuse syntactic and semantic graphs based on the relative importance of the syntactic role and semantic role. First, the nodes in the two graphs are initialized as:

$$\mathrm{H}^{(0)} = \overline{\mathrm{E}}^{w} \tag{14}$$

**Syntactic Graph** We construct a directed syntactic graph from a sentence based on dependency parsing, with the word as a node and the dependency between words as an edge. We then use the R-GCN [31] to update node representations. The node representations of the syntactic graph $\widehat{\mathrm{H}}^{(l)}$ in $l^{th}$ layer are calculated as:

$$\widehat{\mathrm{H}}_i^{(l)} = \sigma\left(\sum_{r \in \mathcal{R}_{dep}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} \widehat{\mathrm{W}}_r^{(l)} \mathrm{H}_j^{(l)} + \widehat{\mathrm{W}}_0^{(l)} \mathrm{H}_i^{(l)}\right) \tag{15}$$

where $\widehat{\mathrm{W}}_r^{(l)}$ and $\widehat{\mathrm{W}}_0^{(l)}$ denote two learnable weight matrices, and $\mathcal{N}_i^r$ denotes the set of neighbor indices of node $i$ under relation $r \in \mathcal{R}_{dep}$.

**Semantic Graph** We compute the dense adjacency matrix based on semantic similarity and randomly sample from the fully connected graph to construct the semantic graph:

$$\alpha = \mathrm{LeakyReLU}\left(\widetilde{\mathrm{W}}\mathrm{H}^{(l)}\right)^{T} \mathrm{LeakyReLU}\left(\widetilde{\mathrm{W}}\mathrm{H}^{(l)}\right) \tag{16}$$

where $\widetilde{\mathrm{W}}$ denotes a trainable weight matrix. Then we compute a weighted average for aggregation of neighbor nodes $\mathcal{N}(i)$, where the weights come from the normalized adjacency matrices $\overline{\alpha}$. We update the node representations of semantic graph $\widetilde{\mathrm{H}}_i^{(l)}$ in $l^{th}$ layer, which are calculated as follows:

$$\overline{\alpha} = \mathrm{softmax}\,(\alpha) \tag{17}$$

$$\widetilde{\mathrm{H}}_i^{(l)} = \overline{\alpha}_{i,i} \widetilde{\mathrm{W}}\mathrm{H}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \overline{\alpha}_{i,j} \widetilde{\mathrm{W}}\mathrm{H}_j^{(l)} \tag{18}$$

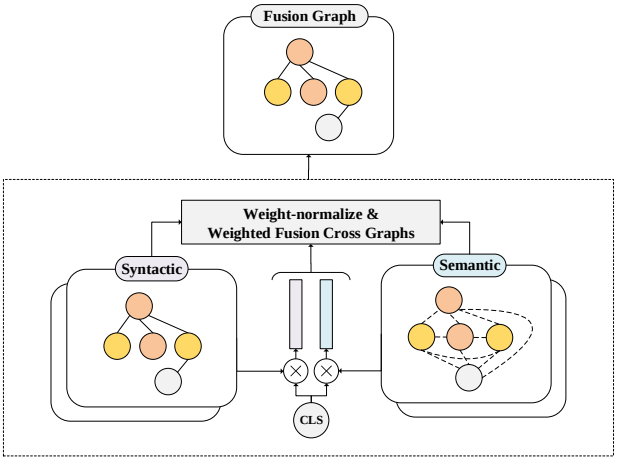


**Figure 4: Syntactic-Semantic Graph Weighted Fusion**

**Node-Wise Graph Weighted Fusion** We design a graph weighted fusion module to dynamically fuse two graphs according to the relative semantic and syntactic importance of words in a sentence. The [CLS] vector, denote as $\mathrm{e}^{cls}$, is often used for sentence-level tasks and contains information about the entire sentence. We first calculate the bilinear similarity between $\mathrm{e}^{cls}$ and each node of semantic and syntactic graphs. Then we normalize the similarity vectors

across two graphs to obtain two sets of weights, which indicate semantic and syntactic importance respectively. Finally, we fuse all nodes across the graphs based on the weights, as shown in Figure 4.

$$\widetilde{\mathrm{w}}, \widehat{\mathrm{w}} = \mathrm{softmax}\left(\left\{\mathrm{e}^{cls}\mathrm{W}\widetilde{\mathrm{H}}^{(l)}, \mathrm{e}^{cls}\mathrm{W}\widehat{\mathrm{H}}^{(l)}\right\}\right) \tag{19}$$

$$\mathrm{H}_i^{(l+1)} = \widetilde{\mathrm{w}}_i \cdot \widetilde{\mathrm{H}}_i^{(l)} + \widehat{\mathrm{w}}_i \cdot \widehat{\mathrm{H}}_i^{(l)} \tag{20}$$

where W is a learnable weight matrix, $\widetilde{\mathrm{w}}$ and $\widehat{\mathrm{w}}$ denote the node importance weights of syntactic and semantic graphs, respectively. Then we map the node representations $\mathrm{H}^{(l+1)}$ to the corresponding word representations $E^g$ using mean-pooling:

$$\mathrm{E}^g = \mathrm{mean}\left(\mathrm{H}^{(l+1)}, \overline{\mathrm{E}}^w\right) \tag{21}$$

## 3.5 Trigger Sensor

We know that a particular relation usually occurs in conjunction with a particular set of words, which we call relation triggers. They can help explain why humans would extract a relation in the sentence and play an essential role in relation extraction. We present a Trigger Sensor module that senses and enhances the contextual trigger information without any trigger annotations.

Relation triggers typically appear in local context between a pair of spans $(s_i, s_j)$, and some approaches encode local context directly into the span-pair representation for relation classification. However, these approaches do not consider the case where the triggers are outside the span-pair, resulting in the model ignoring useful information from other contexts. We design both a Local-Context Encoder and a Full-Context Trigger Sensor to compute the local-context representation $\mathrm{g}_{local}$ and the full-context trigger representation $\mathrm{g}_{trigger}$.

**Local-Context Encoder** We aggregate local-context information between spans of interest using max-pooling. The local-context representation $\mathrm{g}_{local}$ is calculated as:

$$\mathrm{g}_{local} = \max\left(\mathrm{E}_k^g, \mathrm{E}_{k+1}^g, \cdots, \mathrm{E}_h^g\right) \tag{22}$$

where $\mathrm{E}_{\mathrm{k}}^{\mathrm{g}}, \mathrm{E}_{\mathrm{k+1}}^{\mathrm{g}}, \cdots, \mathrm{E}_{\mathrm{h}}^{\mathrm{g}}$ are the encodings of words between the two spans $(s_i, s_j)$.

**Full-Context Trigger Sensor** Full-context trigger sensor aims to sense and enhance span-pair specific triggers. Given a pair of spans $(s_i, s_j)$, we use head span and tail span as queries respectively and execute *normal read process* on the relation memory. After obtaining two span-specific memory representations, we perform mean-pooling across them to get the span-pair specific relation representation $m_{(ij)}^r$:

$$m_{(ij)}^r = \mathrm{mean}\left(\mathrm{Read}_{norm}\left(\mathrm{E}_i^s, \mathrm{M}^{\mathcal{R}}\right), \mathrm{Read}_{norm}\left(\mathrm{E}_j^s, \mathrm{M}^{\mathcal{R}}\right)\right) \tag{23}$$

We calculate the similarity between $\mathrm{m}_{(ij)}^r$ and each word representation of a word sequence, and then weight the word sequence to get the full-context trigger representation $\mathrm{g}_{trigger}$.

$$\mathbf{g}_{trigger} = \mathrm{softmax}\left(\mathrm{m}_{(ij)}(\mathrm{E}^g)^T\right)\mathrm{E}^g \tag{24}$$

We incorporate the local-context representation $\mathrm{g}_{local}$ and the full-context trigger representation $\mathrm{g}_{trigger}$ into the span-pair encoding $\mathrm{E}_{ij}^r$ using $f_r$:

$$\mathrm{E}_{ij}^r = f_r\left(\mathrm{E}_i^s, \mathrm{E}_j^s, \mathrm{g}_{local}, \mathrm{g}_{trigger}\right) \tag{25}$$

for $f_r$ we adopt the concatenate function.

**Trigger Extraction** Using the trigger sensor, we can also extract relation triggers and provide a reasonable explanation for model predictions. Based on the similarity of each word representation with the span-pair specific relation representations $m_{(ij)}^r$, we rank the words. The top-ranked words can be used as relation triggers to explain the model's predictions. We will show the trigger extraction ability of our model in the case study section.

## 3.6 Memory-Aware Classifier

Representations of the entity and relation categories are stored in entity memory and relation memory, respectively. Based on the bilinear similarity between instance (span or span-pair) representation and categories representations, we compute the probability of candidate span $s_i$ being an entity $e$:

$$p\left(s_i = e\right) = \frac{\exp\left(\mathrm{E}_i^s \mathrm{W}^e M_e^{\mathrm{E}}\right)}{\sum_{k\in\mathcal{E}}\exp\left(\mathrm{E}_i^s \mathrm{W}^e \mathrm{M}_k^{\mathrm{E}}\right)} \tag{26}$$

and the probability of candidate span-pair $(s_i, s_j)$ having a relation $r$:

$$p\left(r_{(ij)} = r\right) = \frac{\exp\left(\mathrm{E}_{(ij)}^r \mathrm{W}^r \mathrm{M}_r^{\mathcal{R}}\right)}{\sum_{k\in\mathcal{R}}\exp\left(\mathrm{E}_{(ij)}^r \mathrm{W}^r \mathrm{M}_k^{\mathcal{R}}\right)} \tag{27}$$

where $\mathrm{W}^e \in \mathbb{R}^{h_s \times h_{me}}$ and $\mathrm{W}^r \in \mathbb{R}^{h_r \times h_{mr}}$ denote two learnable weight matrices. Finally, we define a joint loss function for entity classification and relation classification:

$$\mathcal{L} = \mathcal{L}^s + \mathcal{L}^r$$

where $\mathcal{L}^s$ denotes the cross-entropy loss over entity categories(including the *None* category), and $\mathcal{L}^r$ denotes the binary cross-entropy loss over relation categories.

## 3.7 Two-Stage Training

At the start of training, since the memory is randomly initialized, the Memory Flow Attention module and Trigger Sensor module will introduce noises to the sequence encoding. These noises further corrupt the semantic information of the pre-trained BERT [10] through the gradient descent. We therefore divide the model training procedure into two stages. In the first stage, we aim to learn more accurate category representations and store them into the corresponding memory slots. We only train Memory-Aware Classifier and Graph Weighted Fusion modules and update the memory through the *memory write process*. In the second stage, we add Memory Flow Attention and Trigger Sensor modules to the training procedure. Based on the more accurate representations of the categories stored in the memory, we can strengthen the contextual task-related features and relation triggers through *memory read process*.

# 4 EXPERIMENTS

## 4.1 Datasets

We evaluate TriMF described above using the following four datasets:

- **SciERC**: The SciERC [26] includes annotations for scientific entities, their relations, and coreference clusters for 500 scientific abstracts. The dataset defines 6 types for annotating scientific entities and 7 relation categories. We adopt the same data splits as in [26].
- **ACE05**: ACE05 was built upon ACE04, and is commonly used to benchmark NER and RE methods. ACE05 defines 7 entity categories. For each pair of entities, it defines 6 relation categorie. We adopt the same data splits as in [29].
- **CoNLL04**: The CoNLL04 dataset [30] consists of 1,441 sentences with annotated entities and relations extracted from news articles. It defines 4 entity categories and 5 relation categorie. We adopt the same data splits as in [14], which contains 910 training, 243 dev, and 288 test sentences.
- **ADE**: The Adverse Drug Events (ADE) dataset [15] consists of 4, 272 sentences and 6, 821 relations extracted from medical reports. These sentences describe the adverse effects arising from drug use. ADE dataset contains two entity categories and a single relation category.

## 4.2 Compared Methods

Our model is compared with current advanced joint entity and relation extraction models, divided into three types: general parameter-sharing based models (Multi-head AT, SPtree, SpERT, SciIE), span-graph based models (DyGIE, DyGIE++), and reading-comprehension based models (multi-turn QA, MRC4ERE).

**Multi-head + AT** [4] treats the relation extraction task as a multi-head selection problem. Each entity is combined with all other entities to form entity pairs that can be predicted which relations to have. In addition, instead of being a multi-category task where each category is mutually exclusive, the relation classification is treated as multiple bicategorical tasks where each relation is independent, which allows more than one relation to be predicted.

**SPTree** [29] shares parameters of the encoder in joint entity recognition and relation extraction tasks, which strengthens the correlation between two tasks. SPTree is the first model that adopts a neural network to solve a joint extraction task for entities and relations.

**SpERT** [11] is a simple and effective model for joint entity and relation extraction. It uses BERT [10] to encode a sentence, and enumerates all spans in the sentence. Then it performs span classification and span-pair classification to extract entities and relations.

**SciIE** [26] is a framework for extracting entities and relations from the scientific literature. It reduces error propagation between tasks and leverages cross-sentence relations through coreference links by introducing a multi-task setup and a coreference disambiguation task.

**DyGIE/DYGIE++** [27, 33] dynamically build a span graph, and iteratively refine the span representations by propagating coreference and relation type confidences through the constructed span graph. Also, DyGIE++ takes event extraction into account.

**Multi-turn QA** [23] treats joint entity and relation extraction task as a multiple-round question-and-answer task. Each entity and each relation is depicted using a question-and-answer template, so that these entities and relations can be extracted by answering these templated questions.

**MRC4ERE++** [41] introduces a diversity question answering mechanism based on Multi-turn QA. Two answering selection strategies are designed to integrate different answers. Moreover, MRC4ERE++ proposes to predict a subset of potential relations to filter out irrelevant ones to generate questions effectively.

## 4.3 Evaluation Metrics

We evaluate these models on both entity recognition and relation extraction tasks. An entity is considered correct if its predicted span and entity label match the ground truth. When evaluating relation extraction task, previous works have used different metrics. For convenience of comparison, we report multiple evaluation metrics consistent with them. We define a **strict evaluation**, where a relation is considered correct if its relation type as well as the two related entities are both correct, and a **boundary evaluation**, where entity type correctness is not considered. We reported strict relation f1 on Conll04 and ADE, boundary relation f1 on SciERC, and both on ACE05. Our experiments on these datasets all report a micro-F1 score, except for the ADE dataset, where we report the macro-F1 score.

## 4.4 Experiment Settings

In most experiments, we use BERT [10] as the encoder, pre-trained on an English corpus. On the SciERC dataset, we replace BERT with SciBERT [6]. We perform the four-level encoding with a subword encoding size $h = 768$, a word encoding size $h_w = 768$, a span encoding size $h_s = 793$, and a span-pair encoding size $h_r = 2354$. We set both entity memory slot size $h_{me}$ and relation memory slot size $h_{mr}$ to 768. We just use a single graph neural layer in semantic and syntactic graphs. We initialize entity memory and relation memory using the normal distribution $\mathcal{N}(0.0, 0.02)$. We use the Adam Optimizer with a linear warmup-decay learning rate schedule (with a peak learning rate of 5e-5), a dropout before the entity and relation bilinear classifier with a rate of 0.5, a batch size of 8, span width embeddings of 25 dimensions and max span-size of 10. The training is divided into two stages with the first stage of 18 epochs, and the second stage of 12 epochs.

## 4.5 Results and Analysis

**Main Results** We report the average results over 5 runs on SciERC, ACE05 and CoNLL04 datasets. For ADE, we report metrics averaged across the 10 folds. Table 1 illustrates the performance of the proposed method as well as baseline models on SciERC, ACE05, CoNLL04 and ADE datasets. Our model consistently outperforms the state-of-the-art models for both entity and relation extraction on all datasets. Specifically, the relation F1 scores of our model advances previous models by +3.2%, +4.9%, +0.6%, +2.3% on SciERC, ACE05, CoNLL04 and ADE respectively. We attribute the improvement to three reasons. First, our model can share learned information between tasks through the Memory module, enhancing task interactions in both directions(from NER to RE, and from

| Dataset | Model | Entity | | | Relation | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F1 | Precision | Recall | F1 |
| SciERC | SciIE† | 67.20 | 61.50 | 64.20 | 47.60 | 33.50 | 39.30 |
| | DyGIE† | - | - | 65.20 | - | - | 41.60 |
| | DYGIE++† | - | - | 67.50 | - | - | 48.40 |
| | SpERT† (using SciBERT [6]) | 70.87 | 69.79 | 70.33 | 53.40 | 48.54 | 50.84 |
| | TriMF† (using SciBERT) | 70.18 (±0.65) | 70.17 (±0.94) | 70.17 (±0.56) | 52.63 (±1.24) | 52.32 (±1.73) | **52.44 (±0.40)** |
| ACE05 | DyGIE† | - | - | 88.40 | - | - | 63.20 |
| | DYGIE++† | - | - | 88.60 | - | - | 63.40 |
| | TriMF† | 87.67 (±0.17) | 87.54 (±0.29) | 87.61 (±0.21) | 65.87 (±0.55) | 67.12 (±0.63) | **66.49 (±0.32)** |
| | Multi-turn QA‡ | 84.70 | 84.90 | 84.80 | 64.80 | 56.2 | 60.20 |
| | MRC4ERE++‡ | 85.90 | 85.20 | 85.50 | 62.00 | 62.20 | 62.10 |
| | TriMF‡ | 87.67 (±0.17) | 87.54 (±0.29) | **87.61 (±0.21)** | 62.19 (±0.52) | 63.37 (±0.52) | **62.77 (±0.22)** |
| CoNLL04 | Multi-head + AT [4] ‡ | | | 83.9 | | | 62.04 |
| | Multi-turn QA‡ | 89.00 | 86.60 | 87.80 | 69.20 | 68.20 | 68.90 |
| | SpERT‡ | 88.25 | 89.64 | 88.94 | 73.04 | 70.00 | 71.47 |
| | MRC4ERE++‡ | 89.30 | 88.50 | 88.90 | 72.20 | 71.50 | 71.90 |
| | TriMF‡ | 90.26 (±0.62) | 90.34 (±0.60) | **90.30 (±0.24)** | 73.01 (±0.21) | 71.63 (±0.26) | **72.35 (±0.23)** |
| ADE | Multi-head + AT [4] ‡* | - | - | 86.73 | - | - | 75.52 |
| | SpERT‡* | 88.99 | 89.59 | 89.28 | 77.77 | 79.96 | 78.84 |
| | TriMF‡* | 89.50 | 91.29 | **90.38** | 74.22 | 83.43 | **80.66** |

**Table 1: Precision, Recall, and F1 scores on the SciERC, ACE05, CoNLL04 and ADE datasets. (macro-average=*, boundary evaluation=†, strict evaluation=‡)**

RE to NER). Second, the Trigger Sensor module can enhance the relation trigger information, which is essential for relation classification. Lastly, taking a step further from introducing structure information through syntactic graphs, we distinguish the semantic and syntactic importance of words to fuse two-way information through a dynamic Graph Weighted Fusion module. We conduct ablation studies to further investigate the effectiveness of these modules.

## 4.6 Ablation Study

**Effect of Different Modules** To prove the effects of each proposed modules, we conduct the ablation study. As shown in Table 2, all modules contribute to the final performance. Specifically, removing the Trigger Sensor module has the most significant effect, causing the relation F1 score to drop from 52.44% to 51.23% on SciERC, from 62.77% to 61.60% on ACE05. Comparing the effects of Memory-Flow Attention at subword-level and word-level on the two datasets, we find that the improvement of MFA at subword-level is more significant. We thus believe that fine-grained semantic information is more effective for relation extraction. The performance of the Syntactic-Semantic Graph Weighted Fusion module varies widely across datasets, achieving an improvement of 1.09% on ACE05, but only 0.61% on SciERC. This may be related to the different importance of syntactic information for relation extraction on different domains.

**Effect of Interaction Between Two Subtasks** There is mutual dependency between the entity recognition and relation extraction tasks. Our framework models this relationship through the Multi-level Memory Flow Attention module. Depending on the memory that attention mechanism rely on, it can be divided into Relation-specific MFA and Entity-specific MFA. The Relation-specific MFA module enhances the relation-related information based on the relation memory, allowing the entity recognition task to utilize the information already captured in the relation extraction task, as does Entity-specific MFA. To verify that the Memory Flow Attention module can facilitate the interaction between entity recognition and relation extraction, we perform ablation studies, as shown in the Table 3. On ACE05 and SciERC, both Entity-specific MFA and Relation-specific MFA bring significant performance improvement. In addition, the Relation-specific MFA improves more compared

| Method | Entity | | Relation | |
|---|---|---|---|---|
| | F1 | Δ | F1 | Δ |
| SciERC | | | | |
| TriMF | 70.17 | - | **52.44** | - |
| w/o Graph Weighted Fusion | 70.12 | -0.05 | 51.83 | -0.61 |
| w/o Trigger Sensor | 70.19 | +0.02 | 51.23 | -1.21 |
| w/o Subword-level MFA | 70.11 | -0.06 | 51.27 | -1.17 |
| w/o Token-level MFA | **70.21** | +0.04 | 51.78 | -0.66 |
| ACE05 | | | | |
| TriMF | **87.61** | - | **62.77** | - |
| w/o Graph Weighted Fusion | 87.55 | -0.06 | 61.68 | -1.09 |
| w/o Trigger Sensor | 87.45 | -0.16 | 61.60 | -1.17 |
| w/o Subword-level MFA | 87.09 | -0.52 | 61.68 | -1.09 |
| w/o Token-level MFA | 87.42 | -0.19 | 62.02 | -0.75 |

**Table 2: Effect of Different Modules**

| Method | Entity | | Relation | |
|---|---|---|---|---|
| | F1 | Δ | F1 | Δ |
| SciERC | | | | |
| TriMF | **70.17** | - | **52.44** | - |
| w/o MFA | 70.04 | -0.13 | 50.78 | -1.66 |
| w/o Relation MFA | 70.07 | -0.10 | 51.28 | -1.16 |
| w/o Entity MFA | 70.17 | 0 | 51.84 | -0.60 |
| ACE05 | | | | |
| TriMF | **87.61** | - | **62.77** | - |
| w/o MFA | 87.42 | -0.19 | 62.19 | -0.58 |
| w/o Relation MFA | 87.37 | -0.24 | 62.06 | -0.71 |
| w/o Entity MFA | 87.38 | -0.23 | 62.64 | -0.13 |

**Table 3: Effect of Interaction between NER and RE**

with Entity-specific MFA. We think the reason may be that our model performs entity recognition first and then relation extraction. This order determines that information from entity recognition has been used by relation extraction, but the information from relation extraction is not fed back to entity recognition. When using Relation-specific MFA, a bridge for bi-directional information flow is built between the two tasks. Furthermore, when we use both Entity-specific MFA and Relation-specific MFA, the experiment achieves the best performance, indicating that MFA can enhance the bi-directional interaction between entity recognition and relation extraction.

**Effect of Different Graph Fusion Methods** Our proposed graph weighted fusion module employs a node-wise weighted fusion approach based on attention, which enables a flexible fusion of node representations according to words' syntactic importance and semantic importance. To demonstrate the effectiveness of our approach, we compare other node-wise fusion methods, including no-fusion, max-fusion, mean-fusion and sum-fusion, as shown in the Table 4. Comparing the two experiments which only use semantic graph or syntactic graph, we find that the syntactic graph provide a greater improvement in model performance, probably because the initial encodings of the nodes of the syntactic graph have already contained semantic information. Compared to max-fusion, mean-fusion, and sum-fusion, the node-wise weight-fusion method brings more improvement on relation F1 scores of both SciERC and ACE05, which proves the effectiveness of our method.

**Effect of Different Stage Divisions for Memory** We explored the effect of different two-stage divisions on the relation classification, as shown in Figure 5 (x-axis is the number of epochs for the first stage and the total number of epochs is 30). We can note that if our model skips the first stage (x=0) or ignores the second stage (x=30), the performance of the model degrades significantly. Specifically, as the proportion of first stage epochs to total epochs increases, our model performs better. But at a certain point, the performance degrades significantly. We believe this is due to a decrease in epochs of the second stage and the memory already written in the first stage is not utilized effectively. Therefore the two-stage training strategy is effective, and a good balance of the two stages can bring out a better model performance.

| Method | Entity | | | Relation | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| SciERC | | | | | | |
| No Graph | 69.87 | 70.33 | 70.10 | 52.56 | 49.59 | 51.03 |
| Semantic Graph | 68.47 | 69.61 | 49.04 | 52.00 | 50.62 | 51.30 |
| Syntactic Graph | 72.18 | 70.68 | **71.42** | 54.02 | 48.97 | 51.37 |
| Mean-fusion | 69.77 | 69.02 | 69.39 | 53.56 | 49.39 | 51.38 |
| Sum-fusion | 69.45 | 69.57 | 69.51 | 52.94 | 49.65 | 51.24 |
| Max-fusion | 69.12 | 69.64 | 69.38 | 53.01 | 49.45 | 51.17 |
| Weighted fusion | 70.18 | 70.17 | 70.17 | 52.63 | 52.32 | **52.44** |
| ACE05 | | | | | | |
| No Graph | 87.24 | 87.18 | 87.21 | 60.11 | 61.83 | 60.96 |
| Semantic Graph | 87.57 | 87.69 | 87.63 | 59.45 | 62.47 | 60.92 |
| Syntactic Graph | 87.47 | 87.36 | 87.41 | 59.29 | 62.96 | 61.07 |
| Mean-fusion | 87.32 | 87.78 | 87.55 | 59.74 | 62.90 | 61.28 |
| Sum-fusion | 87.85 | 87.47 | **87.66** | 60.12 | 62.26 | 61.17 |
| Max-fusion | 87.51 | 87.62 | 87.56 | 60.22 | 62.25 | 61.22 |
| Weighted fusion | 87.67 | 87.54 | 87.61 | 62.19 | 63.37 | **62.77** |

**Table 4: Effect of Different Graph Fusion Methods**

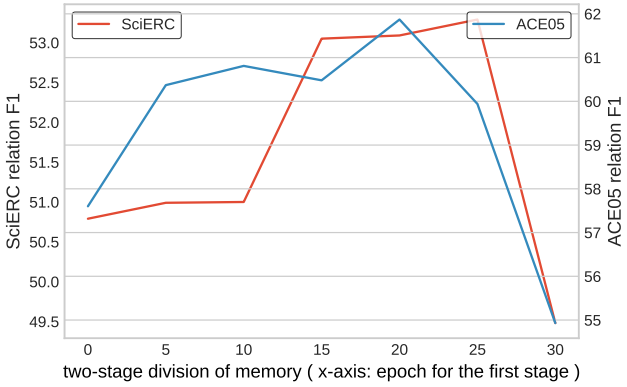


**Figure 5: Effect of Train Stage Division**

**Effect of Different Gradients Flow to Memory** Our model primarily write the memory in Memory-Aware Classifier. Furthermore, we can also tune the memory in MFA and Trigger Sensor modules through the back propagation of gradients. The gradient flows are divided into three types: Trigger Sensor gradients, Subword-level MFA gradients and Word-level MFA gradients, and we investigated the effects of different gradients, as shown in Table 6. We see that on the ACE05 dataset, when we block any of the gradient flows, the model performance decreases significantly, by 1.35%, 1.54%, and 0.92% on relation F1 score, which indicates that tuning the memory during the second stage is effective. However, On the SciERC dataset, there is no significant drop, and we believe that the model has learned accurate representations of the categories in the first training stage.

**Effect of Relation Filtering Threshold** The precision and recall of relation classification are correlated with predefined thresholds. We investigate the impact of the relation filtering threshold on

| Original Text | Relation | Top-5 Relation Triggers |
|---|---|---|
| Urutigoechea and the others were arrested Wednesday in the cities of Bayonee and **Bonloc** in southwestern **France** in Poitiers in west-central France. | (Bonloc, Located in, France) | southwestern, west-central, cities, of, in |
| **Kleber Elias Gia Bustamante**, accused by the police of being a member of the "Red Sun" central committee, has been living clandestinely since his escape from the Garcia Moreno Prison, where he was held accused of assassinating the industrialist, **Jose Antonio Briz Lopez**. | (Kleber Elias Gia Bustamante, Kill, Jose Antonio Briz Lopez) | Prison, assassinating, held, of, accused |

**Table 5: Results of Trigger Words Extraction**

| Method | Entity F1 | Entity Δ | Relation F1 | Relation Δ |
|---|---|---|---|---|
| SciERC | | | | |
| TriMF | 70.17 | - | **52.44** | - |
| w/o Trigger Sensor Grad. | 70.14 | -0.03 | 52.28 | -0.16 |
| w/o Subword-level MFA Grad. | **70.23** | +0.08 | 52.03 | -0.41 |
| w/o Word-level MFA Grad. | 70.12 | -0.05 | 52.14 | -0.30 |
| ACE05 | | | | |
| TriMF | **87.61** | - | **62.77** | - |
| w/o Trigger Sensor Grad. | 87.55 | -0.06 | 61.42 | -1.35 |
| w/o Subword-level MFA Grad. | 87.43 | -0.18 | 61.23 | -1.54 |
| w/o Word-level MFA Grad. | 87.34 | -0.27 | 61.85 | -0.92 |

**Table 6: Effect of Gradient Flow to Memory**

relation F1. Figure 6 shows the relation F1 score on the SciERC and ACE05 test sets, plotted against relation filtering threshold. We see that the performance of our model is stable for the choice of relation filtering thresholds. Our model is able to achieve good results on relation classification except for extreme thresholds of 0.0 or 1.0. Therefore, within a reasonable range, our model is not sensitive to choose a threshold.

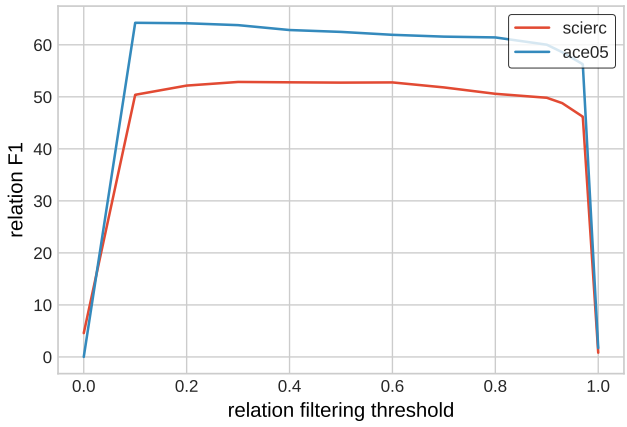


**Figure 6: Effect of Relation Filtering Threshold**

## 4.7 Case Study

**Trigger Words Extraction** With the Trigger Sensor module, our model has the ability to extract the relation triggers. We rank the similarities of each word representation with the span-pair specific relation representation, which have been calculated in the Trigger Sensor. Filtering out the entity surface words and stopwords, the top k words are picked as relation triggers and used to interpret the results of the relation extraction. We show two cases in Table 5.

**Memory Flow Attention Visualization** We visualize the weights of attention to provide a straightforward picture of how the entity and relation memory flow attention we designed can both enhance the interaction between entity recognition and relation extraction. Also, it can enhance the information about relation triggers in context, to some extent explaining the model's predictions. Figure 7 shows two cases of how attention weights on context from a relation memory flow can help the model recognize entities and highlight relation triggers. Each example is split into two visualizations, with the top showing the original attention weights and the bottom showing the attention weights after masking the entities. In the top figure, we can see that the darker words belong to an entity, for example, *"Urutigoechea"*, *"Bayonee"*, *"Bonloc"* in case 1, *"Dallas"*, *"Jack Ruby"* in case 2, illustrating that the attention of our relation memory flow attention can highlight relevant entity information. Consistent with [16], our attention distribution also illustrates that entity names provide more valid information for relation classification compared to context. To more clearly visualize the attention weights of different contextual words, we mask all entities, formalize the weights of the remaining words, and then visualize them. As shown in the bottom figure, the relation memory flow pays more attention on the words that indicate the type of relation, i.e., relation triggers, such as *"in"*, *"southwestern"*, *"west-central"* in case 1 can indicate *"Located in"* relation, and *"assassin"*, *"murdering"* in case 2 can indicate *"Kill"* relation. This shows that our relation memory flow is able to highlight relation triggers, helping the model with better performance on relation extraction.

**Error Cases** In addition to visualizing Memory Flow Attention weights on true positives, we also analyze a number of **false positives** and **false negatives**. These error cases include: relation requiring inference, ambiguous entity recognition and long entity recognition, as shown in the Table 7. In the first case, although our model is able to recognize the four entities about *Location*, it incorrectly extracts the relation *"(Guernsey, Located in, France)"* and does not extract the correct one *"(Guernsey, Located in, Channel Islands)"*. This is because the model does not infer the complex location relation between the four entities. Our model is prone to make

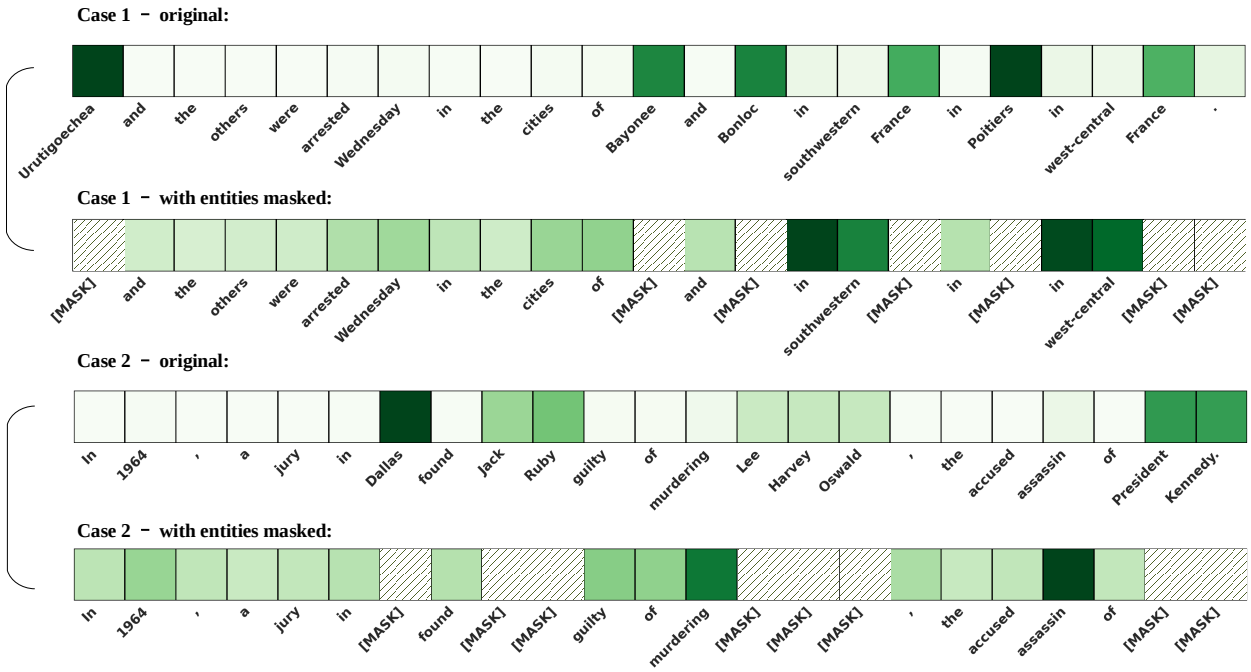


**Figure 7: Two case studies of relation memory flow attention during inference. The darker cells have higher attention weights.**

| |
|---|
| The problem is not unusual in [[ Guernsey ]$_{H\_Located\text{-}in}$]$_{H\_Located\text{-}in}$, one of [ Britain ]$_{T\_Located\text{-}in}$ 's [ Channel Islands ]$_{T\_Located\text{-}in}$ off the coast of [[ France ]]$_{T\_Located\text{-}in}$ |
| ... and former [[ CBS ]$_{T\_Work\text{-}for}$ News ] commentator [[ Eric Sevareid ]$_{H\_Live\text{-}in}$ ]$_{H\_Work\text{-}for}$ , who was born in [ Velva ]$_{T\_Live\text{-}in}$ , several miles southeast of [ Minot ]. |
| Text of the statement issued by the [ Organization of the Oppressed on Earth ] claiming [[ U. S. ]$_{T\_Live\text{-}in}$]$_{T\_Live\text{-}in}$ Marine Lt.[[ William R. Higgins ]$_{H\_Live\text{-}in}$]$_{H\_Live\text{-}in}$ was hanged. |

**Table 7: Typical error examples. Red brackets indicate entities predicted by the model, blue brackets indicate true entities, and the labels in the lower right corner indicate the type of relation between the corresponding entities and the head or tail type (T for tail entity; H for head entity)**

mistakes when classifying ambiguous entities, and False Positive and False Negative often occur together. For example, in the second row of the Table 7, the model does not recognize *"CBS News"* as a Location entity, but recognizes *"CBS"* which is not labeled in the test set. Furthermore, recognition of long entities is a challenge for our model due to the fact that long entities are sparse in the dataset. For example, in the third row of the Table 7, the model fails to recognize the long entity *"Organization of the Oppressed on Earth"*.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we propose a Trigger-Sense Memory Flow Framework (TriMF) for joint entity and relation extraction. We use the memory to boost the task-related information in a sentence through Multi-level Memory Flow Attention module. This module can effectively exploit the mutual dependency and enhance the bi-directional interaction between entity recognition and relation extraction tasks. Also, focusing on the relation triggers, we design a Trigger Sensor to sense and enhance triggers based on memory. Our model can extract the relation triggers without any trigger annotations, which can better assist the relation extraction and provide an explanation. Furthermore, we distinguish the semantic and syntactic importance of a word in a sentence and fuse semantic and syntactic graphs dynamically based on the attention mechanism. Experiments on SciERC, ACE05, CoNLL04 and ADE datasets show that our proposed model TriMF achieves state-of-the-art performance.

In the future, we will improve our work along with two directions. First, we plan to impose constraints on the representations of entity categories and relation categories written in the memory, due to the fact that relations and entities substantively satisfy specific constraints at the ontology level. Second, for improving the model's ability on sensing the trigger, we plan to add weak supervision (e.g. word frequency, entity boundary) to the Trigger Sensor module.

## ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Project of China (No. 2018AAA0101900), the Fundamental Research Funds for the Central Universities, the Chinese Knowledge Center of Engineering Science and Technology (CKCEST) and MOE Engineering Research Center of Digital Library.

## REFERENCES

[1] Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*. 85–94.

[2] Chinatsu Aone, Lauren Halverson, Tom Hampton, and Mila Ramos-Santacruz. 1998. SRA: Description of the IE2 system used for MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

[3] David S Batista, Bruno Martins, and Mário J Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 499–504.

[4] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. *arXiv preprint arXiv:1808.06876* (2018).

[5] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications* 114 (2018), 34–45.

[6] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).

[7] Quoc-Chinh Bui, Sophia Katrenko, and Peter MA Sloot. 2011. A hybrid approach to extract protein–protein interactions. *Bioinformatics* 27, 2 (2011), 259–265.

[8] Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 724–731.

[9] Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 551–560.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Markus Eberts and Adrian Ulges. 2019. Span-based Joint Entity and Relation Extraction with Transformer Pre-training. *arXiv preprint arXiv:1909.07755* (2019).

[12] Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1409–1418.

[13] Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. RelEx-Relation extraction using dependency parse trees. *Bioinformatics* 23, 3 (2007), 365–371.

[14] Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2537–2547.

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

[16] Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. More Data, More Relations, More Context and More Openness: A Review and Outlook for Relation Extraction. *arXiv preprint arXiv:2004.03186* (2020).

[17] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Coling 1992 volume 2: The 15th international conference on computational linguistics*.

[18] Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 113–120.

[19] Rosie Jones, Andrew McCallum, Kamal Nigam, and Ellen Riloff. 1999. Bootstrapping for text learning tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, Vol. 1.

[20] Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. 22–es.

[21] Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 919–929.

[22] Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 402–412.

[23] Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. *arXiv preprint arXiv:1905.05529* (2019).

[24] Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang Ren. 2020. Triggerner: Learning with entity triggers as explanations for named entity recognition. In *ACL*.

[25] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2124–2133.

[26] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602* (2018).

[27] Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. *arXiv preprint arXiv:1904.03296* (2019).

[28] Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

[29] Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770* (2016).

[30] Dan Roth and Wen-tau Yih. 2004. *A linear programming formulation for global inference in natural language tasks*. Technical Report. ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE.

[31] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[32] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7072–7079.

[33] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546* (2019).

[34] Shaolei Wang, Yue Zhang, Wanxiang Che, and Ting Liu. 2018. Joint extraction of entities and relations based on a novel graph scheme.. In *IJCAI*. 4461–4467.

[35] Ziqi Wang, Yujia Qin, Wenxuan Zhou, Jun Yan, Qinyuan Ye, Leonardo Neves, Zhiyuan Liu, and Xiang Ren. 2019. Learning from explanations with neural execution tree. In *International Conference on Learning Representations*.

[36] Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A Novel Cascade Binary Tagging Framework for Relational Triple Extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 1476–1488.

[37] Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1640–1649.

[38] Yue Yuan, Xiaofei Zhou, Shirui Pan, Qiannan Zhu, Zeliang Song, and Li Guo. 2020. A Relation-Specific Attention Network for Joint Entity and Relation Extraction. In *International Joint Conference on Artificial Intelligence 2020*. Association for the Advancement of Artificial Intelligence (AAAI), 4054–4060.

[39] Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 506–514.

[40] Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. 288–295.

[41] Tianyang Zhao, Zhao Yan, Y. Cao, and Zhoujun Li. 2020. Asking Effective and Diverse Questions: A Machine Reading Comprehension based Framework for Joint Entity-Relation Extraction. In *IJCAI*.

[42] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. *arXiv preprint arXiv:1706.05075* (2017).

[43] Wenxuan Zhou, Hongtao Lin, Bill Yuchen Lin, Ziqi Wang, Junyi Du, Leonardo Neves, and Xiang Ren. 2020. Nero: A neural rule grounding framework for label-efficient relation extraction. In *Proceedings of The Web Conference 2020*. 2166–2176.