Doc2EDAG: An End-to-End Document-level Framework for Chinese Financial Event Extraction

Shun Zheng[†*], **Wei Cao**[‡], **Wei Xu**[†], **Jiang Bian**[‡]
[†]Institute of Interdisciplinary Information Sciences, Tsinghua University
[‡]Microsoft Research
zhengs14@mails.tsinghua.edu.cn;
{Wei.Cao, Jiang.Bian}@microsoft.com;
weixu@mail.tsinghua.edu.cn

Abstract

Most existing event extraction (EE) methods merely extract event arguments within the sentence scope. However, such sentence-level EE methods struggle to handle soaring amounts of documents from emerging applications, such as finance, legislation, health, etc., where event arguments always scatter across different sentences, and even multiple such event mentions frequently co-exist in the same document. To address these challenges, we propose a novel end-to-end model, Doc2EDAG, which can generate an entity-based directed acyclic graph to fulfill the document-level EE (DEE) effectively. Moreover, we reformalize a DEE task with the no-trigger-words design to ease document-level event labeling. To demonstrate the effectiveness of Doc2EDAG, we build a large-scale real-world dataset consisting of Chinese financial announcements with the challenges mentioned above. Extensive experiments with comprehensive analyses illustrate the superiority of Doc2EDAG over state-of-the-art methods. Data and codes can be found at https://github.com/dolphin-zs/Doc2EDAG.

1 Introduction

Event extraction (EE), traditionally modeled as detecting trigger words and extracting corresponding arguments from plain text, plays a vital role in natural language processing since it can produce valuable structured information to facilitate a variety of tasks, such as knowledge base construction, question answering, language understanding, etc.

In recent years, with the rising trend of digitalization within various domains, such as finance, legislation, health, etc., EE has become an increasingly important accelerator to the development of business in those domains. Take the financial domain as an example, continuous economic growth has witnessed exploding volumes of digital financial documents, such as financial announcements in a specific stock market as Figure 1 shows, specified as *Chinese financial announcements* (ChFinAnn). While forming up a gold mine, such large amounts of announcements call EE for assisting people in extracting valuable structured information to sense emerging risks and find profitable opportunities timely.

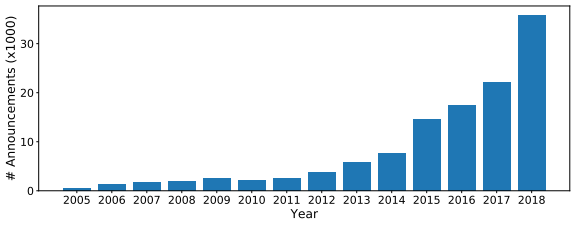

Figure 1: The rapid growth of event-related announcements considered in this paper.

Given the necessity of applying EE on the financial domain, the specific characteristics of financial documents as well as those within many other business fields, however, raise two critical challenges to EE, particularly *arguments-scattering* and *multi-event*. Specifically, the first challenge indicates that arguments of one event record may scatter across multiple sentences of the document, while the other one reflects that a document is likely to contain multiple such event records. To intuitively illustrate these challenges, we show a typical ChFinAnn document with two *Equity Pledge* event records in Figure 2. For the first event, the entity[1] "[SHARE1]" is the correct *Pledged Shares* at the sentence level (*ID 5*). However, due to the capital stock increment (*ID 7*),

[*] This work was done during the internship of Shun Zheng at Microsoft Research Asia, Beijing, China.

[1] In this paper, we use "entity" as a general notion that includes named entities, numbers, percentages, etc., for brevity.

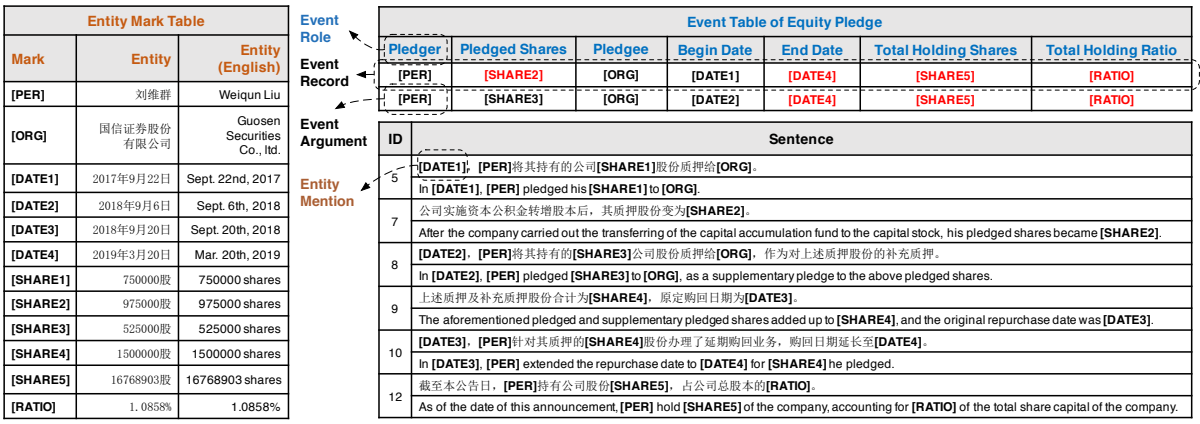

Figure 2: A document example with two *Equity Pledge* event records whose arguments scatter across multiple sentences, where we use *ID* to denote the sentence index, substitute entity mentions with corresponding marks, and color event arguments outside the scope of key-event sentences as red.

the correct *Pledged Shares* at the document level should be "[SHARE2]". Similarly, "[DATE3]" is the correct *End Date* at the sentence level (*ID 9*) but incorrect at the document level (*ID 10*). Moreover, some summative arguments, such as "[SHARE5]" and "[RATIO]", are often stated at the end of the document.

Although a great number of efforts (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Riedel and McCallum, 2011; Li et al., 2013, 2014; Chen et al., 2015; Yang and Mitchell, 2016; Nguyen et al., 2016; Liu et al., 2017; Sha et al., 2018; Zhang and Ji, 2018; Nguyen and Nguyen, 2019; Wang et al., 2019) have been put on EE, most of them are based on ACE 2005[2], an expert-annotated benchmark, which only tagged event arguments within the sentence scope. We refer to such task as the **sentence-level EE** (SEE), which obviously overlooks the *arguments-scattering* challenge. In contrast, EE on financial documents, such as ChFinAn, requires **document-level EE** (DEE) when facing *arguments-scattering*, and this challenge gets much harder when coupled with *multi-event*.

The most recent work, DCFEE (Yang et al., 2018), attempted to explore DEE on ChFinAnn, by employing *distant supervision* (DS) (Mintz et al., 2009) to generate EE data and performing a two-stage extraction: 1) a sequence tagging model for SEE, and 2) a key-event-sentence detection model to detect the key-event sentence, coupled with a heuristic strategy that padded missing arguments from surrounding sentences, for DEE. However, the sequence tagging model for SEE cannot handle multi-event sentences elegantly, and even worse, the context-agnostic arguments-completion strategy fails to address the arguments-scattering challenge effectively.

In this paper, we propose a novel end-to-end model, Doc2EDAG, to address the unique challenges of DEE. The key idea of Doc2EDAG is to transform the event table into an *entity-based directed acyclic graph* (EDAG). The EDAG format can transform the hard table-filling task into several sequential path-expanding sub-tasks that are more tractable. To support the EDAG generation efficiently, Doc2EDAG encodes entities with document-level contexts and designs a memory mechanism for path expanding. Moreover, to ease the DS-based document-level event labeling, we propose a novel DEE formalization that removes the trigger-words labeling and regards DEE as directly filling event tables based on a document. This no-trigger-words design does not rely on any predefined trigger-words set or heuristic to filter multiple trigger candidates, and still perfectly matches the ultimate goal of DEE, mapping a document to underlying event tables.

To evaluate the effectiveness of our proposed Doc2EDAG, we conduct experiments on a real-world dataset, consisting of large scales of financial announcements. In contrast to the dataset used by DCFEE where 97%[3] documents just contained one event record, our data collection is ten times larger where about 30% documents include multiple event records. Extensive experiments demonstrate that Doc2EDAG can significantly outper-

[2] https://www.ldc.upenn.edu/collaborations/past-projects/ace

[3] Estimated by their Table 1 as $\frac{2*\text{NO.ANN}-\text{NO.POS}}{\text{NO.ANN}}$.

form state-of-the-art methods when facing DEE-specific challenges.

In summary, our contributions include:

- We propose a novel model, Doc2EDAG, which can directly generate event tables based on a document, to address unique challenges of DEE effectively.
- We reformalize a DEE task without trigger words to ease the DS-based document-level event labeling.
- We build a large-scale real-world dataset for DEE with the unique challenges of *arguments-scattering* and *multi-event*, the extensive experiments on which demonstrate the superiority of Doc2EDAG.

Note that though we focus on ChFinAnn data in this work, we tackle those DEE-specific challenges without any domain-specific assumption. Therefore, our general labeling and modeling strategies can directly benefit many other business domains with similar challenges, such as criminal facts and judgments extraction from legal documents, disease symptoms and doctor instructions identification from medical reports, etc.

2 Related Work

Recent development on information extraction has been advancing in building the joint model that can extract entities and identify structures (relations or events) among them simultaneously. For instance, (Ren et al., 2017; Zheng et al., 2017; Zeng et al., 2018a; Wang et al., 2018) focused on jointly extracting entities and inter-entity relations. In the meantime, the same to the focus of this paper, a few studies aimed at designing joint models for the entity and event extraction, such as handcrafted-feature-based (Li et al., 2014; Yang and Mitchell, 2016; Judea and Strube, 2016) and neural-network-based (Zhang and Ji, 2018; Nguyen and Nguyen, 2019) models. Nevertheless, these models did not present how to handle argument candidates beyond the sentence scope. (Yang and Mitchell, 2016) claimed to handle event-argument relations across sentences with the prerequisite of well-defined features, which, unfortunately, is nontrivial.

In addition to the modeling challenge, another big obstacle for democratizing EE is the lack of training data due to the enormous cost to obtain expert annotations. To address this problem, some researches attempted to adapt distant supervision (DS) to the EE setting, since DS has shown promising results by employing knowledge bases to automatically generate training data for relation extraction (Mintz et al., 2009). However, the vanilla EE required the trigger words that were absent on factual knowledge bases. Therefore, (Chen et al., 2017; Yang et al., 2018) employed either linguistic resources or predefined dictionaries for trigger-words labeling. On the other hand, another recent work (Zeng et al., 2018b) showed that directly labeling event arguments without trigger words was also feasible. However, they only considered the SEE setting and their methods cannot be directly extended to the DEE setting, which is the main focus of this work.

Traditionally, when applying DS to relation extraction, researchers put huge efforts into alleviating labeling noises (Riedel et al., 2010; Lin et al., 2016; Feng et al., 2018; Zheng et al., 2019). In contrast, this work shows that combining DS with some simple constraints can obtain pretty good labeling quality for DEE, where the reasons are two folds: 1) both the knowledge base and text documents are from the same domain; 2) an event record usually contains multiple arguments, while a common relational fact only covers two entities.

3 Preliminaries

We first clarify several key notions: 1) **entity mention**: an entity mention is a text span that refers to an entity object; 2) **event role**: an event role corresponds to a predefined field of the event table; 3) **event argument**: an event argument is an entity that plays a specific event role; 4) **event record**: an event record corresponds to an entry of the event table and contains several arguments with required roles. For example, Figure 2 shows two event records, where the entity "[PER]" is an event argument with the *Pledger* role.

To better elaborate and evaluate our proposed approach, we leverage the ChFinAnn data in this paper. ChFinAnn documents contain firsthand official disclosures of listed companies in the Chinese stock market and have hundreds of types, such as annual reports and earnings estimates. While in this work, we focus on those event-related ones that are frequent, influential, and mainly expressed by the natural language.

4 Document-level Event Labeling

As a prerequisite to DEE, we first conduct the DS-based event labeling at the document level. More specifically, we map tabular records from an event knowledge base to document text and regard well-matched records as events expressed by that document. Moreover, we adopt a no-trigger-words design and reformalize a novel DEE task accordingly to enable end-to-end model designs.

Event Labeling. To ensure the labeling quality, we set two constraints for matched records: 1) arguments of predefined key event roles must exist (non-key ones can be empty) and 2) the number of matched arguments should be higher than a certain threshold. Configurations of these constraints are event-specific, and in practice, we can tune them to directly ensure the labeling quality at the document level. We regard records that meet these two constraints as the well-matched ones, which serve as distantly supervised ground truths. In addition to labeling event records, we assign roles of arguments to matched tokens as token-level entity tags. Note that we do not label trigger words explicitly. Besides not affecting the DEE functionality, an extra benefit of such no-trigger-words design is a much easier DS-based labeling that does not rely on predefined trigger-words dictionaries or manually curated heuristics to filter multiple potential trigger words.

DEE Task Without Trigger Words. We reformalize a novel task for DEE as directly filling event tables based on a document, which generally requires three sub-tasks: 1) **entity extraction**, extracting entity mentions as argument candidates, 2) **event detection**, judging a document to be triggered or not for each event type, and 3) **event table filling**, filling arguments into the table of triggered events. This novel DEE task is much different from the vanilla SEE with trigger words but is consistent with the above simplified DS-based event labeling.

5 Doc2EDAG

The key idea of Doc2EDAG is to transform tabular event records into an EDAG and let the model learn to generate this EDAG based on document-level contexts. Following the example in Figure 2, Figure 3 typically depicts an EDAG generation process and Figure 4 presents the overall workflow of Doc2EDAG, which consists of two key stages:

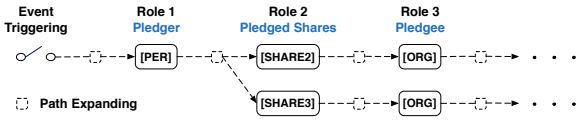

Figure 3: An EDAG generation example that starts from event triggering and expands sequentially following the predefined order of event roles.

document-level entity encoding (Section 5.1) and EDAG generation (Section 5.2). Before elaborating each of them in this section, we first describe two preconditioned modules: input representation and entity recognition.

Input Representation. In this paper, we denote a document as a sequence of sentences. Formally, after looking up the token embedding table $\boldsymbol{V} \in \mathbb{R}^{d_w \times |V|}$, we denote a document $\boldsymbol{d}$ as a sentence sequence $[\boldsymbol{s_1}; \boldsymbol{s_2}; \cdots; \boldsymbol{s_{N_s}}]$ and each sentence $\boldsymbol{s_i} \in \mathbb{R}^{d_w \times N_w}$ is composed of a sequence of token embeddings as $[\boldsymbol{w_{i,1}}, \boldsymbol{w_{i,2}}, \cdots, \boldsymbol{w_{i,N_w}}]$, where $|V|$ is the vocabulary size, N_s and N_w are the maximum lengths of the sentence sequence and the token sequence, respectively, and $\boldsymbol{w_{i,j}} \in \mathbb{R}^{d_w}$ is the embedding of j^{th} token in i^{th} sentence with the embedding size d_w.

Entity Recognition. Entity recognition is a typical sequence tagging task. We conduct this task at the sentence level and follow a classic method, BI-LSTM-CRF (Huang et al., 2015), that first encodes the token sequence and then adds a conditional random field (CRF) layer to facilitate the sequence tagging. The only difference is that we employ the Transformer (Vaswani et al., 2017) instead of the original encoder, LSTM (Hochreiter and Schmidhuber, 1997). Transformer encodes a sequence of embeddings by the multi-headed self-attention mechanism to exchange contextual information among them. Due to the superior performance of the Transformer, we employ it as a primary context encoder in this work and name the Transformer module used in this stage as Transformer-1. Formally, for each sentence tensor $\boldsymbol{s_i} \in \mathbb{R}^{d_w \times N_w}$, we get the encoded one as $\boldsymbol{h_i} = \text{Transformer-1}(\boldsymbol{s_i})$, where $\boldsymbol{h_i} \in \mathbb{R}^{d_w \times N_w}$ shares the same embedding size d_w and sequence length N_w. During training, we employ roles of matched arguments as entity labels with the classic `BIO` (Begin, Inside, Other) scheme and wrap $\boldsymbol{h_i}$ with a CRF layer to get the entity-recognition loss L_{er}. As for the inference, we use the Viterbi

decoding to get the best tagging sequence.

5.1 Document-level Entity Encoding

To address the arguments-scattering challenge efficiently, it is indispensable to leverage global contexts to better identify whether an entity plays a specific event role. Consequently, we utilize document-level entity encoding to encode extracted entity mentions with such contexts and produce an embedding of size d_w for each entity mention with a distinct surface name.

Entity & Sentence Embedding. Since an entity mention usually covers multiple tokens with a variable length, we first obtain a fixed-sized embedding for each entity mention by conducting a max-pooling operation over its covered token embeddings. For example, given l^{th} entity mention covering j^{th} to k^{th} tokens of i^{th} sentence, we conduct the max-pooling over $[\boldsymbol{h_{i,j}}, \cdots, \boldsymbol{h_{i,k}}]$ to get the entity mention embedding $\boldsymbol{e}_l \in \mathbb{R}^{d_w}$. For each sentence $\boldsymbol{s_i}$, we also take the max-pooling operation over the encoded token sequence $[\boldsymbol{h_{i,1}}, \cdots, \boldsymbol{h_{i,N_w}}]$ to obtain a single sentence embedding $\boldsymbol{c_i} \in \mathbb{R}^{d_w}$. After these operations, both the mention and the sentence embeddings share the same embedding size d_w.

Document-level Encoding. Though we get embeddings for all sentences and entity mentions, these embeddings only encode local contexts within the sentence scope. To enable the awareness of document-level contexts, we employ the second Transformer module, Transformer-2, to facilitate the information exchange between all entity mentions and sentences. Before feeding them into Transformer-2, we add them with sentence position embeddings to inform the sentence order. After the Transformer encoding, we utilize the max-pooling operation again to merge multiple mention embeddings with the same entity surface name into a single embedding. Formally, after this stage, we obtain document-level context-aware entity mention and sentence embeddings as $\boldsymbol{e^d} = [\boldsymbol{e^d_1}, \cdots, \boldsymbol{e^d_{N_e}}]$ and $\boldsymbol{c^d} = [\boldsymbol{c^d_1}, \cdots, \boldsymbol{c^d_{N_s}}]$, respectively, where N_e is the number of distinct entity surface names. These aggregated embeddings serve the next stage to fill event tables directly.

5.2 EDAG Generation

After the document-level entity encoding stage, we can obtain the document embedding $\boldsymbol{t} \in \mathbb{R}^{d_w}$ by operating the max-pooling over the sentence tensor $\boldsymbol{c^d} \in \mathbb{R}^{d_w \times N_s}$ and stack a linear classifier over $\boldsymbol{t}$ to conduct the event-triggering classification for each event type. Next, for each triggered event type, we learn to generate an EDAG.

EDAG Building. Before the model training, we need to build the EDAG from tabular event records. For each event type, we first manually define an event role order. Then, we transform each event record into a linked list of arguments following this order, where each argument node is either an entity or a special empty argument `NA`. Finally, we merge these linked lists into an EDAG by sharing the same prefix path. Since every complete path of the EDAG corresponds to one row of the event table, recovering the table format from a given EDAG is simple.

Task Decomposition. The EDAG format aims to simplify the hard table-filling task into several tractable path-expanding sub-tasks. Then, a natural question is how the task decomposition works, which can be answered by the following EDAG recovering procedure. Assume the event triggering as the starting node (the initial EDAG), there comes a series of path-expanding sub-tasks following a predefined event role order. When considering a certain role, for every leaf node of the current EDAG, there is a path-expanding sub-task that decides which entities to be expanded. For each entity to be expanded, we create a new node of that entity for the current role and expand the path by connecting the current leaf node to the new entity node. If no entity is valid for expanding, we create a special `NA` node. When all sub-tasks for the current role finish, we move to the next role and repeat until the last. In this work, we leverage the above logic to recover the EDAG from path-expanding predictions at inference and to set associated labels for each sub-task when training.

Memory. To better fulfill each path-expanding sub-task, it is crucial to know entities already contained by the path. Hence, we design a memory mechanism that initializes a memory tensor $\boldsymbol{m}$ with the sentence tensor $\boldsymbol{c^d}$ at the beginning and updates $\boldsymbol{m}$ when expanding the path by appending either the associated entity embedding or the zero-padded one for the `NA` argument. With this design, each sub-task can own a distinct memory tensor, corresponding to the unique path history.

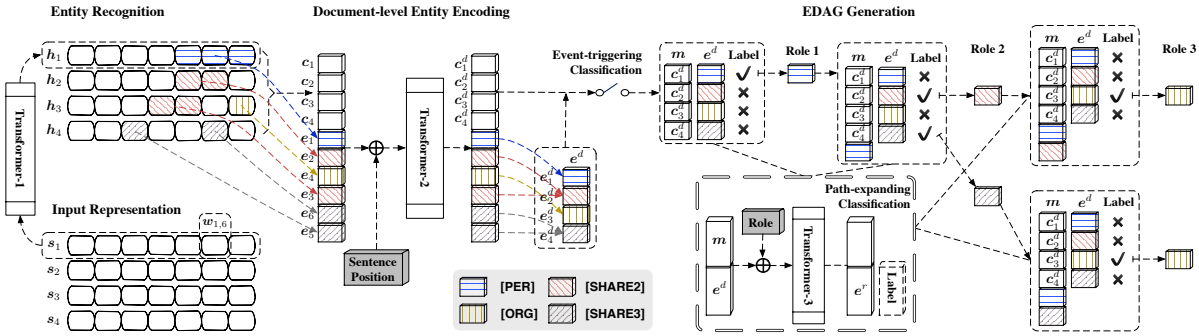

Figure 4: The overall workflow of Doc2EDAG, where we follow the example in Figure 2 and the EDAG structure in Figure 3, and use stripes to differentiate different entities (note that the number of input tokens and entity positions are imaginary, which do not match previous ones strictly, and here we only include the first three event roles and associated entities for brevity).

Path Expanding. For each path-expanding subtask, we formalize it as a collection of multiple binary classification problems, that is predicting *expanding* (1) or *not* (0) for all entities. To enable the awareness of the current path state, history contexts and the current event role, we first concatenate the memory tensor $\boldsymbol{m}$ and the entity tensor $\boldsymbol{e^d}$, then add them with a trainable event-role-indicator embedding, and encode them with the third Transformer module, Transformer-3, to facilitate the context-aware reasoning. Finally, we extract the enriched entity tensor $\boldsymbol{e^r}$ from outputs of Transformer-3 and stack a linear classifier over $\boldsymbol{e^r}$ to conduct the path-expanding classification.

Optimization. For the event-triggering classification, we calculate the cross-entropy loss L_{tr}. During the EDAG generation, we calculate a cross-entropy loss for each path-expanding subtask, and sum these losses as the final EDAG-generation loss L_{dag}. Finally, we sum L_{tr}, L_{dag} and the entity-recognition loss L_{er} together as the final loss, $L_{all} = \lambda_1 L_{er} + \lambda_2 L_{tr} + \lambda_3 L_{dag}$, where λ_1, λ_2 and λ_3 are hyper-parameters.

Inference. Given a document, Doc2EDAG first recognizes entity mentions from sentences, then encodes them with document-level contexts, and finally generates an EDAG for each triggered event type by conducting a series of path-expanding sub-tasks.

Practical Tips. During training, we can utilize both ground-truth entity tokens and the given EDAG structure. While at inference, we need to first identify entities and then expand paths sequentially based on embeddings of those entities to recover the EDAG. This gap between training and inference can cause severe error-propagation problems. To mitigate such problems, we utilize the scheduled sampling (Bengio et al., 2015) to gradually switch the inputs of document-level entity encoding from ground-truth entity mentions to model recognized ones. Moreover, for path-expanding classifications, false positives are more harmful than false negatives, because the former can cause a completely wrong path. Accordingly, we can set $\gamma(> 1)$ as the negative class weight of the associated cross-entropy loss.

6 Experiments

In this section, we present thorough empirical studies to answer the following questions: 1) to what extent can Doc2EDAG improve over state-of-the-art methods when facing DEE-specific challenges? 2) how do different models behave when facing both *arguments-scattering* and *multi-event* challenges? 3) how important are various components of Doc2EDAG?

6.1 Experimental Setup

Data Collection with Event Labeling. We utilize ten years (2008-2018) ChFinAnn[4] documents and human-summarized event knowledge bases to conduct the DS-based event labeling. We focus on five event types: *Equity Freeze* (EF), *Equity Repurchase* (ER), *Equity Underweight* (EU), *Equity Overweight* (EO) and *Equity Pledge* (EP), which belong to major events required to be disclosed by the regulator and may have a huge impact on the company value. To ensure the labeling quality, we set constraints for matched document-record pairs

[4] Crawling from http://www.cninfo.com.cn/new/index

Event	#Train	#Dev	#Test	#Total	MER (%)
EF	806	186	204	1,196	32.0
ER	1,862	297	282	3,677	16.1
EU	5,268	677	346	5,847	24.3
EO	5,101	570	1,138	6,017	28.0
EP	12,857	1,491	1,254	15,602	35.4
All	25,632	3,204	3,204	32,040	29.0

Table 1: Dataset statistics about the number of documents for the train (#Train), development (#Dev) and test (#Test), the number (#Total) and the multi-event ratio (MER) of all documents.

Precision	Recall	F1	MER (%)
98.8	89.7	94.0	31.0

Table 2: The quality of the DS-based event labeling evaluated on 100 manually annotated documents (randomly select 20 for each event type).

as Section 4 describes. Moreover, we directly use the character tokenization to avoid error propagations from Chinese word segmentation tools.

Finally, we obtain 32,040 documents in total, and this number is ten times larger than 2,976 of DCFEE and about 53 times larger than 599 of ACE 2005. We divide these documents into train, development, and test set with the proportion of 8 : 1 : 1 based on the time order. In Table 1, we show the number of documents and the multi-event ratio (MER) for each event type on this dataset. Note that a few documents may contain multiple event types at the same time.

Data Quality. To verify the quality of DS-based event labeling, we randomly select 100 documents and manually annotate them. By regarding DS-generated event tables as the prediction and human-annotated ones as the ground-truth, we evaluate the labeling quality based on the metric introduced below. Table 2 shows this approximate evaluation, and we can observe that DS-generated data are pretty good, achieving high precision and acceptable recall. In later experiments, we directly employ the automatically generated test set for evaluation due to its much broad coverage.

Evaluation Metric. The ultimate goal of DEE is to fill event tables with correct arguments for each role. Therefore, we evaluate DEE by directly comparing the predicted event table with the ground-truth one for each event type. Specifically, for each document and each event type, we pick one predicted record and one most similar ground-truth record (at least one of them is non-empty) from associated event tables without replacement to calculate event-role-specific true positive, false positive and false negative statistics until no record left. After aggregating these statistics among all evaluated documents, we can calculate role-level precision, recall, and F1 scores (all reported in percentage format). As an event type often includes multiple roles, we calculate micro-averaged role-level scores as the final event-level metric that reflects the ability of end-to-end DEE directly.

Hyper-parameter Setting. For the input, we set the maximum number of sentences and the maximum sentence length as 64 and 128, respectively. During training, we set $\lambda_1 = 0.05$, $\lambda_2 = \lambda_3 = 0.95$ and $\gamma = 3$. We employ the Adam (Kingma and Ba, 2015) optimizer with the learning rate $1e^{-4}$, train for at most 100 epochs and pick the best epoch by the validation score on the development set. Besides, we leverage the decreasing order of the non-empty argument ratio as the event role order required by Doc2EDAG, because more informative entities in the path history can better facilitate later path-expanding classifications.

Note that, due to the space limit, we leave other detailed hyper-parameters, model structures, data preprocessing configurations, event type specifications and pseudo codes for EDAG generation to the appendix.

6.2 Performance Comparisons

Baselines. As discussed in the related work, the state-of-the-art method applicable to our setting is DCFEE. We follow the implementation described in (Yang et al., 2018), but they did not illustrate how to handle multi-event sentences with just a sequence tagging model. Thus, we develop two versions, *DCFEE-O* and *DCFEE-M*, where *DCFEE-O* only produces one event record from one key-event sentence, while *DCFEE-M* tries to get multiple possible argument combinations by the closest relative distance from the key-event sentence. To be fair, the SEE stages of both versions share the same neural architecture as the entity recognition part of Doc2EDAG. Besides, we further employ a simple decoding baseline of Doc2EDAG, *Greedy-Dec*, that only fills one event table entry greedily by using recognized entity roles to verify the necessity of end-to-end modeling.

Main Results. As Table 3 shows, Doc2EDAG achieves significant improvements over all base-

Model	EF			ER			EU			EO			EP		
	P.	R.	F1	P.	R.	F1	P.	R.	F1	P.	R.	F1	P.	R.	F1
DCFEE-O	66.0	41.6	51.1	84.5	81.8	83.1	62.7	35.4	45.3	51.4	42.6	46.6	64.3	63.6	63.9
DCFEE-M	51.8	40.7	45.6	83.7	78.0	80.8	49.5	39.9	44.2	42.5	47.5	44.9	59.8	66.4	62.9
GreedyDec	**79.5**	46.8	58.9	83.3	74.9	78.9	68.7	40.8	51.2	69.7	40.6	51.3	**85.7**	48.7	62.1
Doc2EDAG	77.1	**64.5**	**70.2**	**91.3**	**83.6**	**87.3**	**80.2**	**65.0**	**71.8**	**82.1**	**69.0**	**75.0**	80.0	**74.8**	**77.3**

Table 3: Overall event-level precision (P.), recall (R.) and F1 scores evaluated on the test set.

Model	EF		ER		EU		EO		EP		Avg.		
	S.	M.	S.	M.	S.	M.	S.	M.	S.	M.	S.	M.	S. & M.
DCFEE-O	56.0	46.5	86.7	54.1	48.5	41.2	47.7	45.2	68.4	61.1	61.5	49.6	58.0
DCFEE-M	48.4	43.1	83.8	53.4	48.1	39.6	47.1	42.0	67.0	60.6	58.9	47.7	55.7
GreedyDec	75.9	40.8	81.7	49.8	62.2	34.6	65.7	29.4	**88.5**	42.3	74.8	39.4	60.5
Doc2EDAG	**80.0**	**61.3**	**89.4**	**68.4**	**77.4**	**64.6**	**79.4**	**69.5**	85.5	**72.5**	**82.3**	**67.3**	**76.3**

Table 4: F1 scores for all event types and the averaged ones (Avg.) on single-event (S.) and multi-event (M.) sets.

Model	EF	ER	EU	EO	EP	Avg.
Doc2EDAG	70.2	87.3	71.8	75.0	77.3	76.3
-PathMem	-11.2	-0.2	-10.1	-16.3	-10.9	-9.7
-SchSamp	-5.3	-4.8	-5.3	-6.6	-3.0	-5.0
-DocEnc	-4.7	-1.5	-1.6	-1.1	-1.5	-2.1
-NegCW	-1.4	-0.4	-0.7	-1.3	-0.4	-0.8

Table 5: Performance differences of Doc2EDAG variants for all event types and the averaged ones (Avg.).

lines for all event types. Specifically, Doc2EDAG improves 19.1, 4.2, 26.5, 28.4 and 13.4 F1 scores over DCFEE-O, the best baseline, on EF, ER, EU, EO and EP events, respectively. These vast improvements mainly owe to the document-level end-to-end modeling of Doc2EDAG. Moreover, since we work on automatically generated data, the direct document-level supervision can be more robust than the extra sentence-level supervision used in DCFEE, which assumes the sentence containing most event arguments as the key-event one. This assumption does not work well on some event types, such as EF, EU and EO, on which DCFEE-O is even inferior to the most straightforward baseline, GreedyDec. Besides, DCFEE-O achieves better results than DCFEE-M, which demonstrates that naively guessing multiple events from the key-event sentence cannot work well. By comparing Doc2EDAG with GreedyDec that owns high precision but low recall, we can clearly see the benefit of document-level end-to-end modeling.

Single-Event vs. Multi-Event. We divide the test set into a single-event set, containing documents with just one event record, and a multi-event set, containing others, to show the extreme difficulty when *arguments-scattering* meets *multi-event*. Table 4 shows F1 scores for different scenarios. Although Doc2EDAG still maintains the highest extraction performance for all cases, the multi-event set is extremely challenging as the extraction performance of all models drops significantly. Especially, GreedyDec, with no mechanism for the *multi-event* challenge, decreases most drastically. DCFEE-O decreases less, but is still far away from Doc2EDAG. On the multi-event set, Doc2EDAG increases by 17.7 F1 scores over DCFEE-O, the best baseline, on average.

Ablation Tests. To demonstrate key designs of Doc2EDAG, we conduct ablation tests by evaluating four variants: 1) *-PathMem*, removing the memory mechanism used during the EDAG generation, 2) *-SchSamp*, dropping the scheduled sampling strategy during training, 3) *-DocEnc*, removing the Transformer module used for document-level entity encoding, and 4) *-NegCW*, keeping the negative class weight as 1 when doing path-expanding classifications. From Table 5, we can observe that 1) the memory mechanism is of prime importance, as removing it can result in the most drastic performance declines, over 10 F1 scores on four event types except for the ER type whose MER is very low on the test set; 2) the scheduled sampling strategy that alleviates the mismatch of entity candidates for event table filling between training and inference also contributes greatly, improving by 5 F1 scores on average; 3) the document-level entity encoding that enhances global entity representations contributes

2.1 F1 scores on average; 4) the larger negative class weight to penalize false positive path expanding can also make slight but stable contributions for all event types.

Case Studies. Let us follow the example in Figure 2, Doc2EDAG can successfully recover the correct EDAG, while DCFEE inevitably makes many mistakes even with a perfect SEE model, as discussed in the introduction. Due to the space limit, we leave another three fine-grained case studies to the appendix.

7 Conclusion and Future Work

Towards the end-to-end modeling for DEE, we propose a novel model, Doc2EDAG, associated with a novel task formalization without trigger words to ease DS-based labeling. To validate the effectiveness of the proposed approach, we build a large-scale real-world dataset in the financial domain and conduct extensive empirical studies. Notably, without any domain-specific assumption, our general labeling and modeling strategies can benefit practitioners in other domains directly.

As this work shows promising results for the end-to-end DEE, expanding the inputs of Doc2EDAG from pure text sequences to richly formatted ones (Wu et al., 2018) is appealing, and we leave it as future work to explore.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (NSFC) Grant 61532001 and the Zhongguancun Haihua Institute for Frontier Information Technology.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *AAAI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Alex Judea and Michael Strube. 2016. Incremental global event extraction. In *COLING*.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Qi Li, Heng Ji, HONG Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *ACL*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *AAAI*.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. CoType: Joint extraction of typed entities and relations with knowledge bases. In *WWW*.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *EMNLP*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

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

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial training for weakly supervised event detection. In *NAACL*.

Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *SIGMOD*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL*.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: A document-level chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018a. Extracting relational facts by an end-to-end neural model with copy mechanism. In *ACL*.

Ying Zeng, Yansong Feng, Rong Ma, Zheng Wang, Rui Yan, Chongde Shi, and Dongyan Zhao. 2018b. Scale up event extraction learning via automatic training data generation. In *AAAI*.

Tongtao Zhang and Heng Ji. 2018. Event extraction with generative adversarial imitation learning. *arXiv preprint arXiv:1804.07881*.

Shun Zheng, Xu Han, Yankai Lin, Peilin Yu, Lu Chen, Ling Huang, Zhiyuan Liu, and Wei Xu. 2019. DIAG-NRE: A neural pattern diagnosis framework for distantly supervised neural relation extraction. In *ACL*.

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. In *ACL*.