

DEER: A Data Efficient Language Model for Event Temporal Reasoning

Rujun Han^{1,2} Xiang Ren^{1,2} Nanyun Peng^{1,2,3}

¹Information Sciences Institute, University of Southern California

²Department of Computer Science, University of Southern California

³Department of Computer Science, University of California, Los Angeles

rujunhan@isi.edu; xiangren@usc.edu; violetpeng@cs.ucla.edu

Abstract

Pretrained language models (LMs) such as BERT, RoBERTa and ELECTRA are effective at improving performances of a variety of downstream NLP tasks. Recently, researchers have incorporated domain and task-specific knowledge in these LMs’ training objectives and further enhanced models’ capability of handling downstream tasks. However, none of these LMs are designed specifically for event temporal reasoning. We propose *DEER*, a language model that is trained to focus on event temporal relations and performs better under low-resource settings than original LMs. More specifically, we create a large amount of training samples to simulate the machine reading comprehension and information extraction tasks for event temporal understanding and leverage a generator-discriminator structure to reinforce the LMs’ capability of event temporal reasoning. Our experimental results show that *DEER* can achieve SOTA results and works particularly well in low-resource settings across 5 widely used datasets.

1 Introduction

Event temporal ordering is crucial for natural language understanding, which benefits many real-world applications such as tracking biomedical histories, generating stories, and forecasting social events. Currently, there are two types of tasks related to event temporal reasoning: information extraction (IE) (Chambers et al., 2014; Ning et al., 2018; O’Gorman et al., 2016) and machine reading comprehension (MRC) (Ning et al., 2020; Zhou et al., 2019). Figure 1 shows an input text and its IE-style annotation sample: events (**preparing** and **transfer**) as well as their pairwise temporal relations are marked by the annotators, which produces samples in the form of triplets such as (**preparing**, **transfer**, *BEFORE*). Figure 1 also shows an MRC-style sample where an event temporal ordering question is presented and the answer is selected

Input Text	He was preparing the paperwork for the move, following the course of an absolutely standard transfer .
IE Sample	Triplet: (preparing , transfer , <i>BEFORE</i>)
QA Sample	Question: What happened before he began preparing the paperwork? Answer: transfer. Prediction: No relation
Masked Temporal token	He was preparing the paperwork for the move, _____ the course of an absolutely standard transfer .
Masked Event	He was preparing the paperwork for the move, following the course of an absolutely standard _____.

Figure 1: An example illustrating the difference between IE and QA / MRC samples of event temporal reasoning, as well as two different ways to mask tokens for pre-training temporal knowledge aware LMs.

from the input text.

Recent approaches leveraging large pre-trained language models such as RoBERTa (Liu et al., 2019) achieved SOTA results (Ning et al., 2020; Pereira et al., 2020; Wang et al., 2020; Zhou et al., 2020b; Han et al., 2019a) for both IE and MRC tasks. Despite this progress, the current fine-tuning LMs approaches do not explicitly capture event temporal knowledge. As a result, the reported performances of the SOTA pretrained LMs models and their variations are significantly below human performance. Through our error analysis, we observe plenty of cases where a fine-tuned pretrained LM such as RoBERTa, fails to capture temporal indicators and leverage them to infer event relations. In Figure 1 again, an annotator can easily infer from the temporal indicator “following” that “**preparing** the paperwork” happens *BEFORE* **transfer**, but a fine-tuned *RoBERTa-large* model predicts that **transfer** has no such relation with the event “**preparing** the paperwork”.

In order to inject **event temporal knowledge** in pre-trained LMs, we propose to create a large number of distant training examples with masked

temporal indicators and events. For instance, in Figure 1, we can either mask out “following” or “transfer.” In the first case, we hope to encourage LMs to pay closer attention to temporal indicators when events are presented. In the second case, we attempt to simulate the target task in MRC samples where a question consists of a temporal relation and a query event, and the model needs to predict qualified events that answer the question.

With masked samples prepared, we adopt a continuous training technique based on RoBERTa. The training objectives are inspired by ELECTRA’s generator and discriminator architecture (Clark et al., 2020). As shown in Figure 2, the generator predicts the masked temporal indicators or events and the discriminator determines whether the masked token is corrupted or original. Mask prediction reinforces LMs’ attention to temporal indicators and simulates the downstream IE and MRC tasks while the discriminator help models to distinguish correct event temporal relations from noises. To our best knowledge, no prior work in event temporal reasoning has attempted to leverage such model architecture to incorporate **event temporal knowledge** in pre-trained LMs.

While training models to understand event temporal relation is important, current approaches demand expensive data collection efforts. The IE style annotations require a formal definition of event temporal relations. As a result, an expert level of training is needed for annotators. This process is time-consuming and data is relatively small in scale. MRC datasets attempt to bypass this issue using natural language QA pairs to collect large scale of data, which inevitably incurs a large number of monetary costs. In both cases, we show our proposed method is more robust than original LMs in low-resource settings, which provides hope that small-scale data collection can be sufficient for event temporal reasoning.

We summarize our contributions below,

- We created large-scale masked samples as distant supervision signals for event temporal reasoning and adapted a generator-discriminator architecture to enrich event temporal knowledge in language models.
- We applied our proposed method on 5 widely adopted event temporal commonsense datasets, and achieve either new or on-par performances with the SOTA systems.

Categories	words
[before]	‘before’, ‘previous to’, ‘prior to’, ‘preceding’, ‘followed by’, ‘until’
[after]	‘after’, ‘following’, ‘since’, ‘once’, ‘now that’, ‘soon after’
[during]	‘during’, ‘while’, ‘when’, ‘at the time’, ‘at the same time’, ‘meanwhile’
[past]	‘earlier’, ‘previously’, ‘formerly’, ‘in the past’, ‘yesterday’, ‘last time’
[future]	‘consequently’, ‘subsequently’, ‘in turn’, ‘henceforth’, ‘later’, ‘then’
[beginning]	‘initially’, ‘originally’, ‘at the beginning’, ‘to begin’, ‘starting with’, ‘to start with’
[ending]	‘finally’, ‘in the end’, ‘at last’, ‘lastly’

Table 1: The full list of the temporal lexicon. Categories are manually created based on authors’ expertise and best judgment.

- We show that our method works well in low-resource settings, which suggests a promising direction for future few-shot learning in this research area.

2 Method

Our method is inspired by ELECTRA’s generator and discriminator architecture to pre-train LMs. The key differences of our method are that we mask tokens specifically for temporal indicators and events, and train mask prediction and discriminative task jointly with shared representations. In other words, we only use a single transformer for both generator and discriminator; without the discriminator component, our model resembles the mask prediction task in BERT or RoBERTa.

2.1 Mask Prediction for Temporal Indicators

We design a generator to predict masked temporal indicators that are crucial for event temporal reasoning.

Mask Creation. We first compile a lexicon of commonly used temporal indicators such **before**, **after**, **during**, **following**, **while**, etc. The complete list of 40 temporal indicators can be found in the Table 1. We then conduct string match with the temporal lexicon over the 20-year’s New York Times news articles¹ and obtain over 10 million 1-2 sentence passages that contain at least 1 temporal indicators. Finally, we replace these temporal indicators with a [mask] token.

Training Objective. Adapting the notations for the masked language modeling from the ELECTRA paper,

¹NYT news articles are public from 1987-2007.

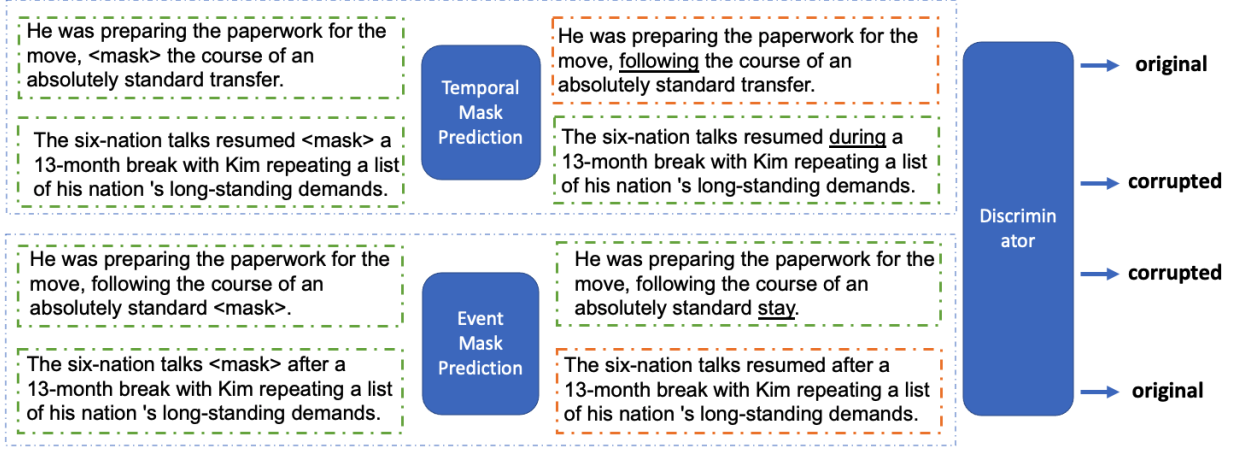


Figure 2: The proposed generator-discriminator architecture for event temporal reasoning. The upper block is the mask prediction task for temporal indicators and the bottom block is the mask prediction task for events. Both generators and the discriminator share the same representations.

$$p_{\mathcal{T}}(x_t|\mathbf{x}) = \text{Softmax}(f_{\mathcal{T}}(h_G(\mathbf{x})_t)) \quad (1)$$

where $\mathbf{x} = [x_1, \dots, x_n]$ is a sequence of input tokens and x_t is the masked token. $h(\mathbf{x})$ is the corresponding encoded representations using a Transformer and f is a linear layer module that maps the masked token representation into label space \mathcal{T} consisting of temporal indicators.

The training objective $\mathcal{L}_{\mathcal{T}}$ is negative log likelihood loss with $p_{\mathcal{T}}$ as input.

2.2 Mask Prediction for Events

Mask Creation. Instead of conducting string matching, we trained a highly accurate event detection model using a large number of event annotations in TORQUE. We then apply this model to automatically label events from the 10 million passages mentioned above. Similarly, we replace these events with [mask] tokens.

Training Objective. This is similar to Eq. 1 except that the label space is now \mathcal{E} which is a set of all the event tokens from the data.

$$p_{\mathcal{E}}(x_t|\mathbf{x}) = \text{softmax}(f_{\mathcal{E}}(h_G(\mathbf{x})_t)) \quad (2)$$

The training objective $\mathcal{L}_{\mathcal{E}}$ is negative log likelihood loss with $p_{\mathcal{E}}$ as input.

2.3 Discriminator

For a given position t , the discriminator predicts whether the token x_t produced by the generator is

original or corrupted, with a sigmoid output layer,

$$D(\mathbf{x}, t) = \text{sigmoid}(f_D(h_D(\mathbf{x})_t)) \quad (3)$$

The training objective for the discriminator \mathcal{L}_{Disc} is again the negative log likelihood loss with $D(\mathbf{x}, t)$ as input.

2.4 Joint Training

The final training loss is,

$$\mathcal{L} = \mathcal{L}_{\mathcal{T}} + \alpha\mathcal{L}_{\mathcal{E}} + \beta\mathcal{L}_{Disc} \quad (4)$$

where α and β are hyper-parameters that balance different training objectives. Our training method differs from ELECTRA in that 1) we share the hidden representations for generator and discriminator, i.e. h_G and h_D but allowing task specific final linear layers $f_{\mathcal{T}}$, $f_{\mathcal{E}}$ and f_D ; 2) We do not train our LMs from scratch and instead continuing to train transformer parameters provided by the *RoBERTa-large* model.

3 Implementations

In this section, we provide details of implementing our methods including data creation as well as training *DEER*.

3.1 Temporal Lexicon

Table 1 shows all temporal indicators (words and phrases) we include in the temporal lexicon. Those indicators in the [before], [after] and [during] categories can be used to represent the most common temporal relation between events. The associated words in each of these categories can be considered

as synonyms for each other. Training LMs to predict these masked indicators simulates the IE-style tasks where a pair of events are given and temporal relations need to be predicted.

Temporal indicators in the **[past]**, **[future]**, **[beginning]** and **[ending]** categories probably do not represent pairwise event relations directly, but predicting these masked tokens may still be helpful for models to understand time anchors and hence facilitates temporal reasoning. Also, in TORQUE and MCTACO datasets, there are a large number of questions simply asking event occurrence time rather than pairwise event relations, so adding these types of temporal indicators can help.

When selecting input passages from NYT corpus, we filter out instances without any string match in the temporal lexicon and we end up with 10 million passages. We randomly select only 200,000 out of them to speed up our experiments and found the results can be as good as using a lot more data.

3.2 Event Prediction Model

As mentioned briefly in Section 2, we train an event prediction model using event annotations in TORQUE. We finetune *RoBERTa-large* model on the training set and select models based on the performance on the dev set. The model achieves $> 92\%$ event prediction F_1 score with $> 95\%$ precision score after training with just 1 epoch, which indicates that this is a highly accurate model.

We then apply this model to all selected instances that contain temporal words described above and mask out the event that is closest to the temporal indicators. Predicting event tokens simulates the task where a temporal relation and an event (A) is given, another event B (with the temporal relation to A) needs to be predicted. This setup is closer to samples in TORQUE and MCTACO where questions usually contain a temporal relation and an event.

3.3 Generator (mask prediction)

For the temporal generator, the entire temporal indicators (whether a single word or a phrase) are masked out; for the event generator, only the event head tokens are masked out. The generator is trained to recover the identity of the masked tokens. Therefore, our generators are effectively multi-label classifiers. In each training instance, only one type of mask is created, and each sample is associated with a task id (0 for the temporal task

and 1 for the event task) so that the model can tell which linear layer ($f_{\mathcal{T}}$ or $f_{\mathcal{E}}$) to use.

The temporal and event generators share the same RoBERTa encoder. Once the representation for the [mask] tokens are obtained, we feed them into separate MLP classifiers for predictions. In Eq. 4, there is a hyper-parameter α that controls the weight of event loss. We found its impact is minimal in our experiments, so we fix $\alpha = 1.0$.

3.4 Discriminator

Our mask prediction set-up is quite different from many SOTA LMs where typically 15% of the tokens are masked out. We instead focus on temporal and event tokens. As a result, our mask prediction can be easier than the one in BERT or ELECTRA particularly for the temporal generator, which achieves $> 70\%$ accuracy. This could make the contrastive loss used to train the discriminator not so powerful.

Perturbed samples. To deal with this issue, for $r\%$ of the generator’s output, instead of using the recovered tokens, we replace them with a token randomly sampled from either the temporal lexicon or the event vocabulary. These $r\%$ samples will serve as additional perturbed or corrupted samples that hopefully could make the contrastive loss more effective. We experiment with $r = 0, 20, 50$, and found $r = 50$ works the best.

3.5 Other Implementation Details

As mentioned, we continue to train the *RoBERTa-large* model with 100K steps with a batch size of 8. The training process takes 10 hours on a single GeForce RTX 2080 GPU. Due to computation constraints, we have to limit the grid search range for other hyper-parameters. So for learning rate, we tried $(1e^{-6}, 2e^{-6})$; for weights on the contrastive loss, we tried $(1.0, 2.0)$. We leverage the dev set of TORQUE to find the best hyperparameters.

4 Experiments on Downstream Tasks

After training *DEER*, there is another fine-tuning step on our target datasets: TORQUE, MCTACO, TB-Dense, MATRES and RED. We conduct grid search for learning rate on the range $(5e^{-6}, 1e^{-5})$ and for batch size on the range $(2, 4, 6, 12)$. We fine-tune *DEER* for 10 epochs for three random seeds (5, 7, 23). Dev set performances are used for early-stop and average dev performances over

three random seeds are used to pick the best hyper-parameters.

4.1 Data Statistics

We briefly summarize the data statistics in Table 2. Note that due to different ways of data construction, the numbers of TORQUE refers to the number of questions as answers are tokens in the input text. The numbers of MCTACO are valid question and answer pairs and there is no training set provided. So we train on the dev set and report the evaluation results on the test set following Pereira et al. (2020). The numbers for TB-Dense, MATRES and RED are all (event, event, relation) triplets. The standard dev set is not provide by MATRES and RED, so we follow the split used in Zhou et al. (2020b) and Han et al. (2019a).

Data	#Train	#Dev	#Test	#Label
TORQUE	24,523	1,483	4,668	2
MCTACO	-	3,783	9,442	2
TB-Dense	4,032	629	1,427	6
MATRES	5,412	920	827	4
RED	3,336	400	473	11

Table 2: The numbers of TORQUE refers to number of questions; the numbers for MCTACO are valid question and answer pairs; and the numbers for TB-Dense, MATRES and RED are all (event, event, relation) triplets.

4.2 Evaluation Metrics

Three evaluation metrics are used to evaluate the fine-tuning performances.

F_1 : for TORQUE, we follow the data paper (Ning et al., 2020) to report macro average of each instance’s F_1 score. Since in the dev and test set, multiple annotations are provided, max F_1 score per instance is used to calculated the final average F_1 score. For TB-Dense, MATRES and RED, we report standard micro-average F_1 scores to be consistent with the baselines.

Exact-match (EM): for TORQUE, $EM = 1$ if the model predictions match perfectly with one of the annotations in an instance; otherwise, $EM = 0$. For MCTACO, $EM = 1$ if all answers to the same questions are predicted correctly; otherwise, $EM = 0$.

EM-consistency (C): in TORQUE, some questions can be clustered into the same group due to

the data collection process where 1) warm-up questions follow the same format; 2) users can modify existing questions to derive new ones. This metric measures the average EM ratio within a group as opposed to an instance in the original EM metrics.

5 Results and Analysis

As shown in Table 3, we report two baselines. The first one is currently the SOTA results reported to the best of the authors’ knowledge. The second one is our implementation of fine-tuning *RoBERTa-large* model to compare fairly with *DEER*. The last two rows in the table show the performances of fine-tuning *DEER*. In Figure 3-7, we show performances of fine-tuning *DEER* by using different ratio training data.

5.1 Overall Results

TORQUE. The current SOTA system reported in Ning et al. (2020) finetunes *RoBERTa-large*, which achieves 75.2%, 51.1% and 34.5% per F_1 , EM and C scores respectively. Our implementation *RoBERTa-large* achieves on-par F_1 scores with the SOTA F_1 score, under-performs the SOTA EM score by 1.5%, and outperforms the SOTA C score by 0.8%. *DEER* outperforms our implementation of *RoBERTa-large* by 1.0%, 2.0% and 1.5% per F_1 , EM and C scores respectively, which demonstrates the effectiveness of continuing to train the original *RoBERTa-large* with our **event temporal knowledge** aware objectives. Our final gains against the current SOTA performances are 0.9%, 0.5% and 2.3% per F_1 , EM and C metrics.

MCTACO. For our *RoBERTa-large* baseline, we obtained 75.5% and 50.5% per F_1 and EM scores, and *DEER* outperforms it by 0.8% and 2.4%. The current SOTA system ALICE reported in Pereira et al. (2020) also uses *RoBERTa-large* as the text encoder, but leverages adversarial attacks on input samples. ALICE achieves 79.5% and 56.5% per F_1 and EM metrics on the test set, but note that these are the best single model performance. The best single model performance we achieved using *DEER* are 76.8% and 54.7% per F_1 and EM scores, which were not able to outperform ALICE. However, our results against the *RoBERTa-large* baseline again show the benefits of introducing **event temporal knowledge** in additional training objectives.

	TORQUE			MCTACO		TB-Dense	MATRES	RED
	F ₁	EM	C	F ₁	EM	F ₁	F ₁	F ₁
SOTA systems	75.2*	51.1	34.5	79.5[†]	56.5	65.2 ^{††}	78.8 [‡]	34.0 ^{‡‡}
RoBERTa-large (fine-tuned)	75.1	49.6	35.3	75.5	50.4	62.8	78.3	39.4
Event-temporal LM (average)	76.1	51.6	36.8	76.3	52.8	64.8	78.8	42.8
Event-temporal LM (single model)	-	-	-	76.8	54.7	66.8	79.3	43.8

Table 3: Overall experimental results. The SOTA performances for TORQUE* are provided by Ning et al. (2020) and the numbers are average over 3 random seeds. The SOTA performances for MCTACO[†], TB-Dense^{††}, MATRES[‡] and RED^{‡‡} are provided by Pereira et al. (2020), Zhou et al. (2020b), Wang et al. (2020) and Han et al. (2019a) respectively. Note that these numbers are the best single model results. To make a fair comparison with the baseline, we report both average and best single model performances. Our *RoBERTa-large* (fine-tuned) baseline also reports an average over three random seeds.

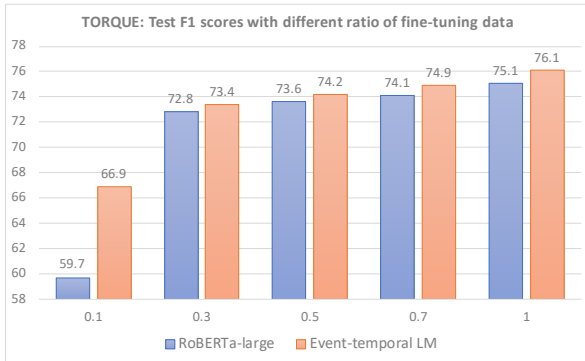


Figure 3: Performances comparison on TORQUE: fine-tuning RoBERTa-large vs. event-temporal LM.

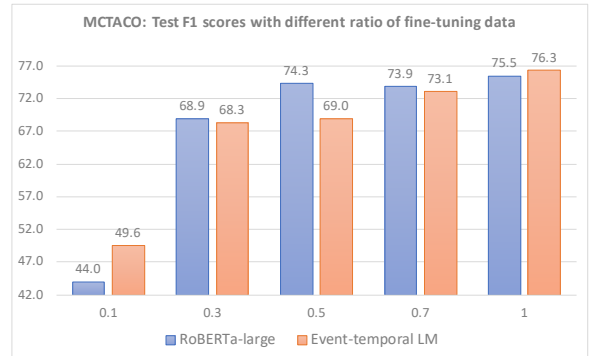


Figure 4: Performances comparison on MCTACO: fine-tuning RoBERTa-large vs. event-temporal LM.

TB-Dense. The current SOTA system reported in Zhou et al. (2020b) also uses *RoBERTa-large* as text encoder, but leverage additional probabilistic logic in the training loss. The single best model achieved 65.2% F_1 score per event relation extraction task. Our *RoBERTa-large* baseline achieved 62.8% average F_1 score which is outperformed by 2% using *DEER*. The single best model performance of fine-tuning *DEER* is 66.8%, which is 1.6% higher than the current SOTA baseline.

MATRES. The current SOTA system reported in Wang et al. (2020) uses *RoBERTa-large* as word representations and BiLSTM as feature extractor, but incorporates additional knowledge constraints in the training loss. The single best model achieved 78.8% F_1 score per event relation extraction task. Our fine-tuned *RoBERTa-large* baseline achieved 78.3% average F_1 score which is outperformed by 0.5% using *DEER*. The single best model performance of fine-tuning *DEER* is 79.3%, which is 0.5% higher than the current SOTA baseline.

RED. The current SOTA system reported in Han et al. (2019a) uses BERT-base as word representations and BiLSTM as feature extractor. The single

best model achieved 34.0% F_1 score. Our fine-tuned *RoBERTa-large* baseline achieved 39.4% average F_1 score which is outperformed by 3.4% using *DEER*. The single best model performance by fine-tuning *DEER* is 43.8%, which is 9.8% higher than the current SOTA baseline.

5.2 Fine-tuning under low-resource setting.

Now we show the results of our proposed method using a smaller amount of data, which demonstrates the data efficiency property of *DEER*.

In Figure 3, we can observe that when using only 10% of the training data ($\approx 2.5K$ questions) to fine-tune LMs, *DEER* outperforms the *RoBERTa-large* baseline by more than 7%. Similar pattern exists in MCTACO in Figure 4: using only 10% of data (≈ 400 question-answer pairs), *DEER* outperforms *RoBERTa-large* by 5.6%. However, we recognize that the overall trend of MCTACO is less stable compared with the other four datasets, which can be caused by two reasons. 1) MCTACO contains 80% of question-answer pairs about event stationarity, duration, frequency, and typical time, which is not strictly related to pairwise event relations; 2) the 3783 questions in the dev set consists of only

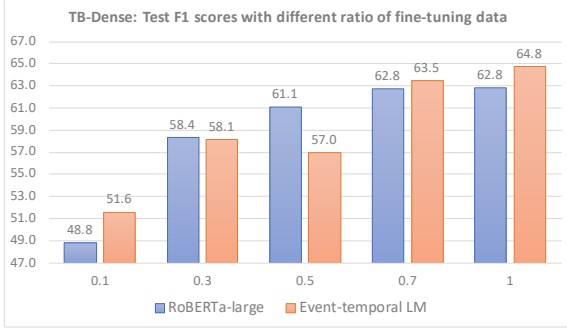


Figure 5: Performances comparison on TB-Dense: fine-tuning RoBERTa-large vs. event-temporal LM.

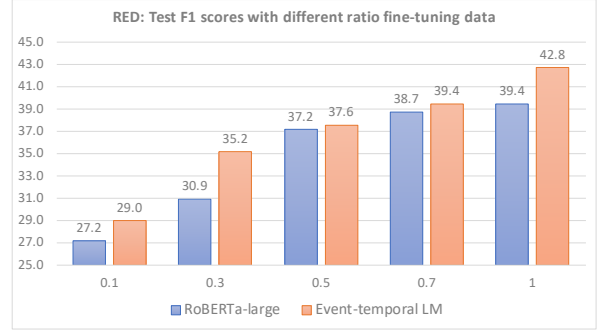


Figure 7: Performances comparison on RED: fine-tuning RoBERTa-large vs. event-temporal LM.

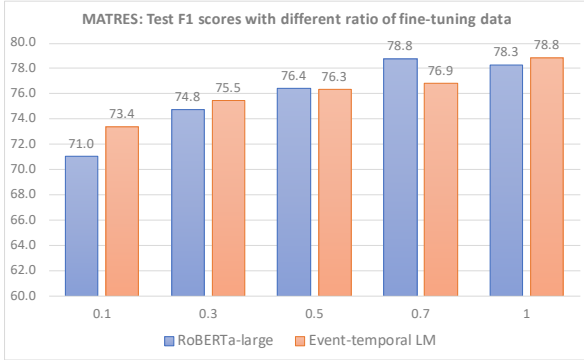


Figure 6: Performances comparison on MATRES: fine-tuning RoBERTa-large vs. event-temporal LM.

≈560 unique questions. When sampling the dev set randomly, we could likely encounter a biased data split that *DEER* is not particularly helpful.

Figure 5, 6 and 7 show that *DEER* outperforms *RoBERTa-large* by 2.4%, 2.8% and 1.8% per F_1 measure using only 10% of the training data for TB-Dense, MATRES and RED respectively. For RED, the gain is 4.3% when using 30% of the training data. The overall pattern for these three IE style datasets suggests that if there is only a small amount of samples available (<1K), training LMs with **event temporal aware knowledge** objective functions can significantly alleviate data sparsity issue, which helps improve or stabilize the fine-tuning performances for large LMs.

5.3 Ablation studies

To better understand the contribution of each component in our proposed model, we ran a set of experiments using the full training data of TORQUE. Results can be found in Table 4.

Generator trains the LM using only the mask prediction loss (i.e. excluding contrastive loss) for 25K steps. Model performance on TORQUE’s test set degrades afterward.

Generator + Discriminator means initializing the transformer with the generator above that produces the results in the second row in Table 4 and continuing to train **only** the discriminator for another for 25K steps.

Generator + Joint means initializing the shared transformers with the same generator’s parameters and continuously training the generator and discriminator **jointly** for another 25K steps.

RoBERTa + Joint (DEER) is our proposed method where we initialize the shared transformer with the original *RoBERTa-large*’s parameters and continue to train the generator and discriminator **jointly** on our masked input samples.

As we can observe, pre-training using only the generator provides noticeable lift over F_1 and EM scores, and the proposed ‘RoBERTa + Joint’ architecture continue to push all average scores up particularly for the **C** measure, which demonstrates the benefits of using both mask prediction and contrastive loss.

Another interesting finding from Table 4 is that the most effective way to incorporate discriminator is training generator and discriminator jointly from scratch. This seems to contradict the suggestions in Zhou et al. (2020a) and Clark et al. (2020) that the generator needs to be first trained for a few steps in order to obtain a good prediction distribution for the discriminator. We speculate that this is due to the fact that our temporal and event mask predictions are easier tasks than those in the previous related papers, which makes separate training of generator not crucial.

Impact of training steps. In Table 5, we show the average test set performance on TORQUE by varying the training steps of *DEER*. We observe that under-training (25K and 50K steps) or over-

	F_1	EM	C
RoBERTa-large	75.1	49.6	35.3
Generator	75.8	51.2	35.8
Generator + Discriminator	75.2	50.7	34.2
Generator + Joint	75.4	49.8	35.6
RoBERTa + Joint (<i>DEER</i>)	76.1	51.6	36.8

Table 4: Average test set performances by fine-tuning different event-temporal LMs using full TORQUE training data. All numbers are average scores over 3 random seeds. *RoBERTa-large* is the same reported in Table 3.

training (125K, 150K) could all lead to sub-optimal performances on the downstream tasks. We leave more rigorous studies over training steps for future research.

Epoch	F_1	EM	C
25K	75.0	50.0	34.6
50K	75.5	50.9	35.8
100K	76.1	51.6	36.8
125K	75.0	49.4	34.5
150K	65.1	45.8	27.4

Table 5: Test performance on TORQUE by training *DEER*. All numbers are average F_1 scores over 3 random seeds.

6 Related Work

6.1 Pre-trained Language Models

Since the breakthrough of BERT (Devlin et al., 2018), pre-trained LMs have become SOTA models for a variety of NLP applications. There have also been several modifications/improvements built on the original BERT model. RoBERTa (Liu et al., 2019) removes the next sentence prediction in BERT and train with longer text inputs and more steps. ELECTRA (Clark et al., 2020) proposes a generator-discriminator architecture, which allows the model to predict each token output as an original or a corrupted one, and addresses the sample-inefficiency issue in BERT and RoBERTa.

Recent research explored methods to continue to train pre-trained LMs so that they can adapt better to downstream tasks. For example, TANDA (Garg et al., 2019) adopts an intermediate training on modified Natural Questions dataset (Kwiatkowski et al., 2019) so that it performs better for the Answer Sentence Selection task. (Zhou et al., 2020a) proposed pre-training objective functions that require a model to distinguish correct sentences from

those with concepts randomly shuffled or generated by LMs. This continuous training technique enables LMs to capture large-scale commonsense knowledge.

6.2 Temporal Reasoning Tasks

There has been a surge of attention to event temporal reasoning research recently. Some noticeable datasets include TB-Dense (Chambers et al., 2014), MATRES (Ning et al., 2018) and RED (O’Gorman et al., 2016) where samples are annotated in IE format, i.e. (event, event, relation) triplets. Previous SOTA systems on these data leveraged pre-trained LMs and structured learning (Han et al., 2019b; Wang et al., 2020; Zhou et al., 2020b) and have substantially improved model performances, though none of them have tackled the problem by incorporating event temporal knowledge in LMs.

TORQUE (Ning et al., 2020) and MCTACO (Zhou et al., 2019) are the first machine reading comprehension dataset that attempts to reason about event temporal relation using natural language rather IE formalism. To our best knowledge, no continuous training on LMs methods has been explored or applied to these data, which is one of our contributions in this paper.

7 Conclusion and Future Work

In summary, we explored methods to enable pre-trained LM models to capture temporal knowledge by adopting a generator-discriminator structure. Extensive experimental results show that both generator and discriminator components can be helpful to improve fine-tuning performances on 5 commonly used data in event temporal reasoning. The improvements of our methods are much more pronounced in low-resource settings, which points out a promising research direction for few-shot learning in this research area.

References

- Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. [Dense event ordering with a multi-pass architecture](#). In *ACL*.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [Electra: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep

- bidirectional transformers for language understanding. In *NAACL-HLT*.
- Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2019. [Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection](#).
- Rujun Han, Mengyue Liang, Bashar Alhafni, and Nanyun Peng. 2019a. Contextualized word embeddings enhanced event temporal relation extraction for story understanding. *arXiv preprint*, arXiv:1904.11942.
- Rujun Han, Qiang Ning, and Nanyun Peng. 2019b. [Joint event and temporal relation extraction with shared representations and structured prediction](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 434–444, Hong Kong, China. Association for Computational Linguistics.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*, arXiv:1907.11692.
- Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. 2020. Torque: A reading comprehension dataset of temporal ordering questions. *arXiv preprint arXiv:2005.00242*.
- Qiang Ning, Hao Wu, and Dan Roth. 2018. [A multi-axis annotation scheme for event temporal relations](#). In *ACL*. Association for Computational Linguistics.
- Tim O’Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. [Richer event description: Integrating event coreference with temporal, causal and bridging annotation](#). In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56, Austin, Texas. Association for Computational Linguistics.
- Lis Pereira, Xiaodong Liu, Fei Cheng, Masayuki Asahara, and Ichiro Kobayashi. 2020. [Adversarial training for commonsense inference](#). In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 55–60, Online. Association for Computational Linguistics.
- Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. [Joint constrained learning for event-event relation extraction](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 696–706, Online. Association for Computational Linguistics.
- Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. [“going on a vacation” takes longer than “going for a walk”: A study of temporal commonsense understanding](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.
- Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, Bill Yuchen Lin, and Xiang Ren. 2020a. [Pre-training text-to-text transformers for concept-centric common sense](#).
- Yichao Zhou, Yu Yan, Rujun Han, J. Harry Caufield, Kai-Wei Chang, Yizhou Sun, Peipei Ping, and Wei Wang. 2020b. Clinical temporal relation extraction with probabilistic soft logic regularization and global inference. *arXiv preprint*, arXiv:2012.08790.