Neural Language Modeling for Contextualized Temporal Graph Generation

Aman Madaan, Yiming Yang
Language Technologies Institute, Carnegie Mellon University
Pittsburgh, PA, USA
amadaan@cs.cmu.edu

Abstract

This paper presents the first study on using large-scale pre-trained language models for automated generation of an event-level temporal graph for a document. Despite the huge success of neural pre-training methods in NLP tasks, its potential for temporal reasoning over event graphs has not been sufficiently explored. Part of the reason is the difficulty in obtaining large training corpora with human-annotated events and temporal links. We address this challenge by using existing IE/NLP tools to automatically generate a large quantity (89,000) of system-produced document-graph pairs, and propose a novel formulation of the contextualized graph generation problem as a sequence-to-sequence mapping task. These strategies enable us to leverage and fine-tune pre-trained language models on the system-induced training data for the graph generation task. Our experiments show that our approach is highly effective in generating structurally and semantically valid graphs. Further, evaluation on a challenging hand-labeled, out-domain corpus shows that our method outperforms the closest existing method by a large margin on several metrics.¹

1 Introduction

Temporal reasoning is crucial for analyzing the interactions among complex events and producing coherent interpretations of text data (Duran et al., 2007). There is a rich body of research on the use of temporal information in a variety of important application domains, including topic detection and tracking (Makkonen et al., 2003), information extraction (Ling and Weld, 2010), parsing of clinical records (Lin et al., 2016), discourse analysis (Evers-Vermeulen et al., 2017), and question answering (Ning et al., 2020).

Graphs are a natural choice for representing the temporal ordering among events, where the nodes are the individual events, and the edges capture temporal relationships such as “before”, “after” or “simultaneous”. Representative work on automated extraction of such graphs from textual documents includes the early work by Chambers and Jurafsky (2009), where the focus is on the construction of event chains from a collection of documents, and the more recent CAEVO (Chambers et al., 2014) and Cogcomptime (Ning et al., 2018), which extract a graph for each input document instead. These methods focus on rule-based and statistical sub-modules to extract verb-centered events and the temporal relations among them. As an emerging area of NLP, large scale pre-trained language models have made strides in addressing challenging tasks like commonsense knowledge graph completion (Bosselut et al., 2019) and task-oriented dialog generation (Budzianowski and Vulić, 2019). These systems typically fine-tune large language models on a corpus of a task-specific dataset. However, these techniques have not been investigated for temporal graph extraction.

This paper focuses on the problem of generation of an event-level temporal graph for each document, and we refer to this task as contextualized graph generation. We address this open challenge by proposing a novel reformulation of the task as a sequence-to-sequence mapping problem (Sutskever et al., 2014), which enables us to leverage large pre-trained models for our task. Further, our proposed approach is completely end-to-end and eliminates the need for a pipeline of sub-systems commonly used by traditional methods.

We also address a related open challenge, which is a prerequisite to our main goal: the difficulty of obtaining a large quantity of training graphs with human-annotated events and temporal relations. To this end, we automatically produce a large collec-

¹Code and pre-trained models available at https://github.com/madaan/temporal-graph-gen
Radomir Markovic, the former head of Serbian intelligence under Slobodan Milosevic, was jailed for seven years for covering up the attempted murder of a leading opposition politician in a 1999 car crash. Markovic, who has been imprisoned since 2001 for revealing state secrets, had denied there was ever a plot to kill Vuk Draskovic, the opposition leader, who survived the crash with minor injuries. Mr. Draskovic’s brother-in-law and three others traveling in a convoy with him were killed.

Figure 1: Task overview: given a document (left), automatically extract a temporal graph (right).

Figure 1: Task overview: given a document (left), automatically extract a temporal graph (right).

GPT-2 on this dataset of document-graph pairs, which yields large performance gains over strong baselines on system generated test set and outperforms CAEVO on TimeBank-Dense (Cassidy et al., 2014) on multiple metrics. Figure 1 shows an example of the input document and the generated graph by our system. In summary, our main contributions are:

1. We present the first investigation on using large pre-trained language models for contextualized temporal event graph generation by proposing a new formulation of the problem as a sequence-to-sequence mapping task.

2. We address the difficulty of obtaining a large collection of human-annotated graphs, which is crucial for effective fine-tuning of pre-trained models, by automatically producing a collection of 89,000 document-graph pairs.

3. Our experimental results on both the system-generated test set (which allows us to compare the relative performance of different models) and a hand-labeled, out-of-domain dataset (TimeBank-Dense), show the advantage of our proposed approach over strong baselines.

2 Related Work

Temporal Graph Extraction  TempEval-3 (UzZaman et al., 2013) introduced the task of temporal graph extraction as “the ultimate task for evaluating an end-to-end system that goes from raw text to TimeML annotation”. Notable systems developed in response include CAEVO (Chambers et al., 2014), followed by the more recent Cogcomptime (Ning et al., 2018). Both CAEVO and Cogcomptime use several statistical and rule-based methods like event extractors, dependency parsers, semantic role labelers, and time expression identifiers for the task. Our work differs from these systems in both the methodology and desired result in the following ways: i) Instead of using specialized sub-systems, we transform the task into a sequence-to-sequence mapping problem and use a single language model to generate such temporal graphs in an end-to-end fashion from text, subsuming all the intermediate-steps. ii) We develop our system using a corpus of 89,000 documents, which is ~300x larger compared to datasets used by CAEVO (36 documents) and Cogcomptime on (276 documents); iii) We remove the noisy events included by CAEVO, but do not limit the extracted events to any specific semantic axis as done by Cogcomptime; and finally, iv) Our method generates graphs where the nodes are not simple verbs but augmented event phrases, containing the subject and the object of each verb. We use CAEVO over Cogcomptime to generate a large-scale corpus for our task and to evaluate our system. Two factors informed our decision: i) We found CAEVO to be
much more scalable, a critical feature for our task of annotating close to 100k documents, ii) CAEVO over-generates (and not excludes) verbs from its output, giving us the flexibility to filter out noisy events without inadvertently missing out on any critical events. However, our method makes no assumption specific to CAEVO and is adaptable to any other similar system (including Cogcomptime).

We note that the problem of temporal graph extraction is different from the more popular task of Temporal relation extraction (Temprel), which deals with classifying the temporal link between two already extracted events. State-of-the-art Temprel systems use neural methods (Ballesteros et al., 2020; Ning et al., 2019b; Goyal and Durrett, 2019), but typically use a handful of documents for their development and evaluation. Vashishtha et al. (2019) are a notable exception by using Amazon Mechanical Turks to obtain manual annotations over a larger dataset of 16,000 sentences. We believe that the techniques presented in our work can be applied to scale the corpus used for training Temprel systems.

Language Models for Graph Generation

The task of generating graphs using language models has gained a lot of attention. Recently, Bosselut et al. (2019) proposed COMET, a system that fine-tunes GPT (Radford et al., 2018) on commonsense knowledge graphs like ATOMIC (Sap et al., 2019) and conceptnet (Speer et al., 2017) for commonsense kb completion. Each node in a commonsense knowledge graph is a phrase or a sentence. Thus, commonsense kb completion is naturally a problem of conditional text generation. Similar to COMET, we adopt large-scale language models for such a conditional generation of text. However, our task differs from COMET in the complexity of both the conditioning text and generated text: we seek to generate temporal graphs conditioned on a document, whereas COMET generates a short event/concept phrase conditioned on a relation and an input event/concept phrase. You et al. (2018) formulate graphs as a sequence for learning generative models of synthetic and real-world graphs. Similar to their work, we formulate graph generation as an auto-regressive task. However, our goal is the conditional generation of temporal graphs, and not learning unconditional generative distributions. Finally, inspired by recent trends (Raffel et al., 2019), we don’t make any graph specific modifications to the model or the decoding process and formulate the problem as a straightforward sequence-to-sequence mapping task. While our approach does not rely on any particular language model, it would be interesting to see the gains achieved by the much larger GPT-3 (Brown et al., 2020) on the dataset produced by our method.  

3 Deriving Large-scale Dataset for the Temporal Graph Generation

Definitions and Notations: Let $G(V, E)$ be a temporal graph associated with a document $D$, such that vertices $V$ are the events in document $D$, and the edges $E$ are temporal relations (links) between the events $e \in V$. Every temporal link in $E$ takes the form $r'(e_p, e_q)$ where events $e_p, e_q \in V$, and $r$ is a temporal relation like before or after. In this work, we undertake two related tasks of increasing complexity: i) Node generation, and ii) Temporal graph generation:

Task 1: Node Generation: Let $r\{e_p, e_q\}$ be an edge in $E$. Let $C_r$ be the set of sentences in the document $D$ that contains the events $e_p$ or $e_q$ or are adjacent to them. Given a query consisting of $C_r$, $r$, and $e_p$, generate $e_q$.

Task 2: Temporal Graph Generation: Given a document $D$, generate the corresponding temporal graph $G(E, V)$.

Figure 1 illustrates the two tasks. Task 1 is similar to knowledge base completion, with a difference that the output events $e_q$ are generated, and not drawn from a fixed set of events. Task 2 is significantly more challenging, requiring the generation of both the structure and semantics of $G$. The training data for both the tasks consists of tuples $(x_i, y_i)$, where $x_i$ is the concatenation of the query tokens $(C_r, e_p, r)$, and $y_i$ consists of tokens of event $e_q$. For Task 2, $x_i$ is simply the $i^{th}$ document $D_i$, and $y_i$ is the corresponding temporal graph $G_i$.

We use the New York Times (NYT) Annotated Corpus 3 to derive our dataset of document-graph pairs. The corpus has 1.8 million articles written and published by NYT between 1987 and 2007. Each article is annotated with a hand-assigned list of descriptive terms capturing its subject(s). We filter articles with one of the following descriptors: “bomb”, “terrorism”, “murder”, “riots”, “hijacking”, “assassination”, “kidnapping”,

---

2 Not available for research as of September 2020.
3 https://catalog.ldc.upenn.edu/LDC2008T19
“arsen”, “vandalism”, “hate crime”, “serial murder”, “manslaughter”, “extortion”}, yielding 89,597 articles, with a total of 2.6 million sentences and 66 million tokens. For each document D, we use CAEVO (Chambers et al., 2014) to extract the dense temporal graph consisting of i) the set of verbs, and ii) the set of temporal relations between the extracted verbs. CAEVO extracts six temporal relations: before, after, includes, is included, simultaneous, and vague.

We process each dense graph extracted by CAEVO with a series of pruning and augmentation operations: i) We observed that some of the most frequent verbs extracted by CAEVO were the so-called reporting verbs (Liu et al., 2018), like said, say, and told, which do not contribute to the underlying events. For example, said formed nearly 10% of all the verbs extracted by CAEVO as an event. To remove such noisy events, we remove the five verbs with the lowest inverse document frequencies, as well as an additional set of light and reporting verbs (Liu et al., 2018; Recasens et al., 2010)\footnote{The full list of verbs is: i) low idf: “said”, “say”, “had”, “made”, “told”, ii) light: “appear”, “be”, “become”, “do”, “have”, “seem”, “get”, “give”, “go”, “have”, “keep”, “make”, “put”, “set”, “take”, iii) reporting: “argue”, “claim”, “say”, “suggest”, “tell”.
}

\textbf{Creating Sub-graphs using Event Communities:} We discovered that the (pruned) graph generated for a given document typically has several sub-graphs that are either completely disconnected or have high intra-link density. Further, each of these sub-graphs refers to different parts of the document. We exploit this phenomenon to map each sub-graph in its correct context, thus reducing the noise in the data.

Relying merely on connectivity for creating sub-graphs is still prone to noise, as largely unrelated sub-graphs are often connected via a single event. Instead, we propose a novel approach based on the detection of event communities to divide a graph into sub-graphs, such that the events in a sub-graph are more densely connected to each other. We learn these event communities using the concept of modularity, first introduced by (Newman and Girvan, 2004). We illustrate the clustering with Figure 3 and defer the derivation of modularity optimization to the Appendix. \textbf{Datasets for Task 1 and Task 2} After running the pruning and clustering operations outlined above on 89k documents, we obtain a corpus of over 890,677 text-graph pairs, with an average of 120.31 tokens per document, and 3.33 events and 4.91 edges per graph. These text-graph pairs constitute the training data for Task 2. We derive the data for Task 1 from the original (undivided) 89k graphs (each document-graph pair contributes multiple examples for Task 1). In Task 1 data, nearly 80% of the queries \((C_r, e_p, r)\) had a unique answer \(e_q\), and nearly 16% of the queries had two different true \(e_q\). We retain examples with multiple true \(e_q\) in the training data because they help the model learn diverse temporal patterns that connect two events. For fairness, we retain such cases in the test set. Table 2 lists the statistics of the dataset. The splits were created using non-overlapping documents.

\textbf{Graph Representation} We use language models to generate each graph as a sequence of tokens con-
Table 2: Dataset statistics.

| Task  | Split  | #Samples   |
|-------|--------|------------|
| Task 1| train  | 4,260,328  |
| Task 1| valid  | 542,994    |
| Task 1| test   | 541,844    |
| Task 1| total  | 5,345,166  |
| Task 2| train  | 709,929    |
| Task 2| valid  | 89,407     |
| Task 2| test   | 91,341     |
| Task 2| total  | 890,677    |

Figure 2: Event temporal graph and the extracted communities for a sample document. Each community is shown in different color. The singleton nodes (gray) are dropped. The nodes are only annotated with the verbs for brevity. The edge labels and directions are not used for community detection.

Figure 3: Temporal graph and the corresponding DOT representation for the sentence: In the southeastern city of Trebisov, Roma clashed fiercely with the police, leading to arrests in which Roma activists said excessive force was used.

\[ \text{slide} \] from left to right as the graph generation proceeds. Additionally, a fixed order makes the problem well defined, as the mapping between a document and a graph becomes deterministic.

4 Model

The training data \( X \) for both Tasks 1 and 2 comprises of tuples \( \{(x_i, y_i)\}_{i=1}^{N} \). We train a (separate) conditional language model to solve both the tasks. Specifically, given a training corpus of the form \( \{(x_i, y_i)\} \), we aim to estimate the distribution \( p_\theta(y_i | x_i) \). Given a training example \( (x_i, y_i) \) we set \( u_i = x_i \| y_i \). \( p_\theta(u_i) \) can then be

\[ \text{\^\}} \text{denotes concatenation} \]
We evaluate our method on two different datasets: a hand-labeled mixed-domain corpus derived from TimeBank-Dense (Cassidy et al., 2014) (TB-Dense). We adapt TB-Dense for our task by applying the same pre-processing operations as were used for TG-Gen, and only use the test splits of TimeBank-Dense for evaluation. In addition to being hand-labeled, TB-Dense forms a very challenging dataset for our task because of domain mismatch; our system was trained on a corpus of terrorism-related events, whereas TB-Dense includes documents from a wide array of domains, essentially forming a zero-shot evaluation scenario for our method.

5.2 Implementation Setup

- **GPT-2** We use GPT-2 medium (355M parameters) for our experiments with 24-layers, a hidden size of 1024, and 16 self-attention heads. All of our experiments were done on a single Nvidia GeForce RTX 2080 Ti. The models were fine-tuned for five epochs, with a runtime of 48 hours/epoch for Task 1 and 52 hours/epoch for Task 2. The details on hyperparameter tuning are deferred to the Appendix. We build on the implementation by Wolf et al. (2019) and draw samples from a fine-tuned model using nucleus sampling (Holtzman et al., 2019). We also experimented with GPT-2 without fine-tuning (i.e., by directly using pre-trained weights). The non-finetuned GPT-2 fared poorly for both the tasks across all the metrics, getting a BLEU score of near 0 for Task 1. This dismal performance underscores the importance of fine-tuning on the end task for large-scale pre-trained language models.

- **LSTM** We train an LSTM (Hochreiter and Schmidhuber, 1997) based sequence-to-sequence model (Bahdanau et al., 2015) with global attention (Luong et al., 2015) as a baseline to contrast with GPT-2. We use two-layered LSTM bi-directional encoder and uni-directional decoder with a hidden size of 500. The token embeddings initialized using 300-dimensional pre-trained Glove (Pennington et al., 2014). The training follows the typical sequence-to-sequence paradigm, where the input→output pairs are either query→event (Task 1) or document→graph (Task 2).

- **CAEVO** We use CAEVO (Chambers et al., 2014)⁶ as a baseline for the TB-Dense dataset.

5 Experiments and Results

5.1 Evaluation Datasets

We evaluate our method on two different datasets: i) Test splits of our dataset (TG-Gen), and ii) A hand-labeled mixed-domain corpus derived from TimeBank-Dense (Cassidy et al., 2014) (TB-Dense). We adapt TB-Dense for our task by factorizing as a sequence of auto-regressive conditional probabilities using the chain rule: \( p_\theta(u_i) = \prod_{k=1}^n p_\theta(u_{i,k}|u_{i,k-1}) \), where \( u_{i,k} \) denotes the \( k \)th token of the \( i \)th sequence, and \( u_{i,k} \) denotes the sequence of tokens \( \{u_1, u_2, ..., u_{k-1}\} \). Language models are typically trained by minimizing a cross-entropy loss \( -\log p_\theta(u_i) \) over each sequence \( u_i \) in \( X \). However, the cross-entropy loss captures the joint distribution \( p_\theta(x_i, y_i) \), and is not aligned with our goal of learning conditional distribution \( p_\theta(y_i|x_i) \). To circumvent this, we train our model by masking the loss terms corresponding to the input \( x_i \), similar to Bosselut et al. (2019). Let \( m_i \) be a mask vector for each sequence \( u_i \), set to 0 for positions corresponding to \( x_i \), and 1 otherwise i.e. \( m_{i,j} = 1 \) if \( j \neq |x_i| \), else 0. We combine the mask vector with our factorization of \( p_\theta(u_i) \) to formulate a masked language modeling loss, which is minimized over the training corpus \( X \) to estimate the optimal \( \theta \):

\[
\mathcal{L}_{\text{masked}}(X) = -\sum_{i=1}^{n} \sum_{j=1}^{l} m_{i,j} \log(p_\theta(u_{i,j}|u_{i,<j}))
\]

In practice, we use GPT-2 (Radford et al., 2019) based on transformer architecture (Vaswani et al., 2017) for our implementation. An input sequence \( u_i \) of length \( n \) is first embedded to a continuous representation denoted by \( u_i^{(0)} \in \mathbb{R}^{n \times d} \). \( u_i^{(0)} \) is then passed through a series of \( L \) transformer blocks to obtain the output sequence \( u_i^{(L)} \in \mathbb{R}^{n \times h} \). Each transformer block (Vaswani et al., 2017) consists of two operations: an auto-regressive version of the multiheaded self-attention (Vaswani et al., 2017) operation followed by a feed-forward operation. Both these operations are surrounded by layer normalization (Ba et al., 2016) and a residual connection (He et al., 2016). After obtaining \( u_i^{(L)} \), we set \( p_\theta(u_i) = \text{softmax}(u_i^{(L)} \ast W_e) \), where \( W_e \in \mathbb{R}^{h \times |V|} \). Finally, we calculate the masked loss as \( \mathcal{L}(u_i) = m_i^T \odot \log(p_\theta(u_i)) \), and the optimal \( \phi \) is obtained by minimizing \( \mathcal{L}_{\text{masked}}(X) = -\sum_{x_i \in X} \mathcal{L}(u_i) \).
We remove all the vague edges and the light verbs from the output of CAEVO for a fair comparison. Note that CAEVO is not used for evaluations on TG-Gen as it was used for creating TG-Gen. Cogcomtime uses a different temporal annotation scheme (MATRES) and thus cannot be used for evaluations.

5.3 Task 1: Node Generation

**Paragraph:** Mr. Grier, a former defensive lineman for the New York Giants who was ordained as a minister in 1986, testified on Dec. 9 that he had visited Mr. Simpson a month earlier.

**Query:** event after “Mr. Grier visited”

| CAEVO answer (incorrect): | Mr. Grier ordained |
|---------------------------|--------------------|
| GPT-2 answer (correct):   | Mr. Grier testified |

Table 3: An example of GPT-2 fixing the label by CAEVO.

**Metrics**

Given a query \((C_r, e_p, r)\), with \(C_r\) being the context (sentences containing events \(e_p, e_q\) and their neighboring sentences) and \(e_p\) as the source event, Task 1 is to generate an event \(e_q\) such that \(r(e_p, e_q)\). Let \(\hat{e}_q\) be the system generated event. We compare \(e_q\) vs. \(\hat{e}_q\) using BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), and ROUGE (Lin, 2004), and measure the accuracy (ACC) as the fraction of examples where \(e_q\) is equal to \(\hat{e}_q\).

**Results on TG-Gen**

The results are listed in Table 4. GPT-2 boasts high scores across the metrics showing that it is highly effective in generating correct events. Moreover, GPT-2 gets an accuracy of over 60%, about 3x higher than that of LSTM (21%). As a stress test for the generative capabilities of the models, we perform an ablation by removing the sentence containing \(e_q\) from \(C_r\), and show the results in Table 4 (indicated with -C). Removal of context causes a drop in performance for both GPT-2 and LSTM across the metrics, showing that it is crucial for generating temporal events. However, GPT-2 obtains higher relative gains with context present, indicating that it uses its large architecture and pre-training to use the context more efficiently. To understand the exact nature of errors, we analyzed 100 randomly sampled incorrect generations. We found that 53% of the errors were caused by GPT-2 generating a non-salient event, which nevertheless had the correct temporal relation with the query. Interestingly, for 10% of the events, we found that GPT-2 fixed the label assigned by CAEVO (Table 3), i.e., \(e_q\) was incorrect but \(\hat{e}_q\) was correct.

**Results on TB-Dense**

The performance of both LSTM and GPT-2 drops on the out-domain TB-Dense dataset. However, GPT-2 fares better as compared with LSTM in terms of drop in performance for metrics like accuracy (−21% vs. −33%) and BLEU (−16% vs. −21%), indicating that pre-training makes helps GPT-2 in generalizing across the domains.

5.4 Task 2: Graph Generation

**Metrics**

Let \(G_i(V_i, E_i)\) and \(\hat{G}_i(\hat{V}_i, \hat{E}_i)\) be the true and the predicted graphs for an example \(i\) in the test corpus. Let \(y_i\) and \(\hat{y}_i\) be their respective string representations in DOT. We evaluate our generated graphs with the following metrics:

- **\(y_i\) vs. \(\hat{y}_i\):** we use the token-overlap metrics BLEU, METEOR, and ROUGE used in Task 1, and also measure parse accuracy (DOT%) as the % of generated graphs \(\hat{y}_i\) which are valid DOT files.
- **\(G_i\) vs. \(\hat{G}_i\):** We use three graph comparison metrics, i) Graph edit distance (GED) (Abu-Aisheh et al., 2015) - the minimum numbers of edits required to transform the predicted graph to the true graph by addition/removal of an edge/node; ii) Graph isomorphism (ISO) (Cordeilla et al., 2001) - a binary measure set to 1 if the graphs are isomorphic (without considering the node or edge attributes); iii) The average graph size (|\(V_i|, |E_i|, |\hat{V}_i|, |\hat{E}_i|\)) and the average degree (\(\text{deg}(V)\)).
- **\(V_i\) vs. \(\hat{V}_i\):** For every example \(i\), we calculate node-set precision, recall, and \(F_1\) score. These metrics are averaged over the test set to obtain node precision (\(v_P\)), recall (\(v_R\)), and \(F_1\) (\(v_F\)).
- **\(E_i\) vs. \(\hat{E}_i\):** We evaluate the predicted edge set using temporal awareness (UzZaman and Allen, 2012; UzZaman et al., 2013). For a particular example \(i\), we define:

\[
\epsilon^i_P = \frac{|\hat{E}_i^+ \cap E_i^-|}{|\hat{E}_i^-|}
\]

\[
\epsilon^i_R = \frac{|\hat{E}_i^+ \cap E_i^-|}{|E_i^-|}
\]
Table 4: Task 1 results. $B_i$ is the BLEU score calculated with $i$-grams.

| Method    | Dataset       | $B_1$ | $B_2$ | $B_3$ | $B_4$ | MTR | RG | ACC |
|-----------|---------------|-------|-------|-------|-------|-----|----|-----|
| LSTM      | TG-Gen (-C)   | 29.44 | 25.21 | 22.43 | 20.20 | 14.62 | 31.95 | 19.68 |
| LSTM      | TG-Gen        | 33.93 | 28.46 | 24.46 | 21.23 | 16.48 | 35.54 | 20.99 |
| GPT-2     | TG-Gen (-C)   | 42.94 | 40.23 | 38.41 | 36.60 | 25.11 | 43.07 | 34.07 |
| GPT-2     | TG-Gen        | **68.64** | **64.43** | **64.62** | **62.53** | **43.78** | **69.10** | **61.35** |
| LSTM      | TB-Dense (-C) | 20.07 | 15.91 | 13.39 | 11.55 | 9.23  | 21.87 | 10.06 |
| LSTM      | TB-Dense      | 26.44 | 21.98 | 19.02 | 16.68 | 12.69 | 27.75 | 13.97 |
| GPT-2     | TB-Dense (-C) | 27.11 | 24.66 | 23.23 | 22.35 | 15.04 | 27.73 | 20.81 |
| GPT-2     | TB-Dense      | **58.07** | **55.73** | **53.61** | **52.21** | **25.11** | **57.98** | **57.98** |

Table 5: Task 2 results.

The $F_1$ score $e_{F_1}^t$ is the harmonic mean of $e_{P}^t$ and $e_{R}^t$. The symbol + denotes the temporal transitive closure (Allen, 1983) of the edge set. Similarly, − indicates the reduced edge set, obtained by removing all the edges that can be inferred from other edges transitively. $e_{P}^t$, $e_{R}^t$, and $e_{F_1}^t$ are averaged over the test set to obtain the temporal awareness precision ($e_{P}^t$), recall ($e_{R}^t$), and $F_1$ score ($e_{F_1}^t$). We present the results in Tables 5 and 6.

Results on TG-Gen: Table 5 shows that GPT-2 achieves impressive results on TG-Gen, outperforming LSTM on all the metrics by a large margin. The difference between the performances of LSTM and GPT-2 is stark, showing that GPT-2 is able to leverage its massive architecture and the powerful self-attention mechanism for the challenging task of graph generation. GPT-2 generated graphs are closer to the true graphs in size and topology, as shown by lower edit distance and a higher rate of isomorphism in Table 5, and by the results in Table 6. Further, LSTM produces about 2x smaller graphs, which also reflects in its deficient BLEU scores (Table 5), since BLEU exponentially penalizes predictions shorter than the references (brevity penalty). Both the systems achieve high parsing rates (DOT %), with GPT-2 generating valid DOT files 94.6% of the time. The high parsing rates are expected, as even simpler architectures like vanilla RNNs have been shown to generate syntactically valid complex structures like LATEXdocuments with ease (Karpathy, 2015). GPT-2 obtains a high $F_1$ score on the node-set, indicating that it learns to avoid picking noisy events (high $P$), and extracts salient events (high $R$). This is confirmed by manual analysis, done by randomly sampling 100 graphs from the GPT-2 generated graphs and isolating the main verb in each node (Table 7). We provide several examples of generated graphs in the Appendix. Finally, we note that while the dif-
Table 7: Verbs in GPT-2 generated graphs.

| Top 10 Verbs: | found, killed, began, called, want, took, came, used, trying, asked |
|-------------|-------------------------------------------------|
| Randomly Sampled Verbs: | shooting, caused, accused, took, conceived, visit, vowing, play, withdraw, seems |

Comparison of the node-set metrics, we see that GPT-2 leads CAEVO by over eight precision points ($v_F$), but loses on recall ($v_R$) as CAEVO extracts nearly every verb in the document as a potential event. On temporal awareness, GPT-2 outperforms both CAEVO and LSTM in terms of average precision score $e_F$ and achieves a competitive $e_F$ score. We also observe that edge extraction ($e_F$) is highly sensitive to node extraction ($v_F$); for GPT-2, a 27% drop in $v_F$ (66.34 on TG-Gen vs. 44.97 on TB-Dense) causes a 68% drop in $e_F$ (25.22 on TG-Gen vs. 7.96 on TB-Dense). We see a similar trend for LSTM, where a 39% drop in $v_F$ leads to a 95% drop in $e_F$. This phenomenon stems from the fact that each node is connected to multiple edges on average (Table 6). Thus, missing a node during the generation process might lead to multiple edges being omitted, affecting edge extraction metrics disproportionately.

6 Conclusion and Future Work

Current methods for generating event-level temporal graphs are developed with relatively small amounts of hand-labeled data. On the other hand, the possibility of using pre-trained language models for this task has not received sufficient attention, primarily due to the difficulty of obtaining large corpora of human-annotated graphs for temporal reasoning over events. This paper addresses this open challenge by first developing a data generation pipeline that uses existing IE/NLP/clustering techniques for automated acquisition of a large corpus of document-graph pairs, and by proposing a new formulation of the graph generation task as a sequence-to-sequence mapping task, allowing us to leverage and fine-tune pre-trained language models for our goal. Our experiments strongly support the effectiveness of the proposed approach, which significantly outperforms strong baselines, including traditional IE techniques. We plan to explore techniques for adapting large-scale language models on unseen domains and at multiple granularity levels in the future.

Acknowledgments

This material is based on research sponsored in part by the Air Force Research Laboratory under agreement number FA8750-19-2-0200. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. Thanks to Nathanael Chambers for the helpful discussion.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An exact graph edit distance algorithm for solving pattern recognition problems. In An exact graph edit distance algorithm for solving pattern recognition problems.

James F Allen. 1983. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11):832–843.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015.

Miguel Ballesteros, Rishita Anubhai, Shuai Wang, Nima Pourdamghani, Yogarshi Vyas, Jie Ma, Parminder Bhatia, Kathleen McKeown, and Yaser Al-Onaizan. 2020. Severing the edge between before and after: Neural architectures for temporal ordering of events. arXiv preprint arXiv:2004.04295.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In ACL.
Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Pawel Budzianowski and Ivan Vulić. 2019. Hello, it’s gpt-2—how can i help you? towards the use of pre-trained language models for task-oriented dialogue systems. arXiv preprint arXiv:1907.05774.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. Transactions of the Association for Computational Linguistics, 2:273–284.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In Proceedings of ACL-08: HLT, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 602–610.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. Physical review E, 70(6):066111.

Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2001. An improved algorithm for matching large graphs. In 3rd IAPR-TC15 workshop on graph-based representations in pattern recognition, pages 149–159.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In Proceedings of the sixth workshop on statistical machine translation, pages 85–91. Association for Computational Linguistics.

Nicholas D Duran, Philip M McCarthy, Art C Graesser, and Danielle S McNamara. 2007. Using temporal cohesion to predict temporal coherence in narrative and expository texts. Behavior Research Methods, 39(2):212–223.

Jacqueline Evers-Vermeul, Jet Hoek, and Merel CJ Scholman. 2017. On temporality in discourse annotation: Theoretical and practical considerations. Dialogue & Discourse, 8(2):1–20.

Emden Gansner, Eleftherios Koutsofios, and Stephen North. 2006. Drawing graphs with dot.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA.

Tanya Goyal and Greg Durrett. 2019. Embedding time expressions for deep temporal ordering models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4400–4406, Florence, Italy. Association for Computational Linguistics.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In Proceedings of the 7th Python in Science Conference, pages 11 – 15, Pasadena, CA USA.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751.

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks. Andrej Karpathy blog, 21:23.

Chen Lin, Dmitriy Dligach, Timothy A Miller, Steven Bethard, and Guergana K Savova. 2016. Multi-layered temporal modeling for the clinical domain. Journal of the American Medical Informatics Association, 23(2):387–395.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pages 74–81.

Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In AAAI, volume 10, pages 1385–1390.

Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard Hovy. 2018. Automatic event salience identification. arXiv preprint arXiv:1809.00647.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421.

Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. 2003. Topic detection and tracking with spatio-temporal evidence. In European Conference on Information Retrieval, pages 251–265. Springer.

Mark EJ Newman. 2004. Fast algorithm for detecting community structure in networks. Physical review E, 69(6):066133.

Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. Physical review E, 69(2):026113.
Qiang Ning, Zhili Feng, and Dan Roth. 2019a. A structured learning approach to temporal relation extraction. *arXiv preprint arXiv:1906.04943*.

Qiang Ning, Sanjay Subramanian, and Dan Roth. 2019b. An improved neural baseline for temporal relation extraction. *arXiv preprint arXiv:1909.00429*.

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, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018. Cogcomptime: A tool for understanding time in natural language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 72–77.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper.pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Marta Recasens, Eduard H Hovy, and Maria Antònia Martí. 2010. A typology of near-identity relations for coreference (nident). In *LREC*.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.
A Learning Event Communities Using Community Detection

In this section, we provide the details on the community detection algorithm used by our method. We define the temporal event communities to be a division of the temporal graph \( G(V, E) \) into sub-graphs \( G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_k(V_k, E_k) \) such that the events in a community (sub-graph) \( G_1 \) are more co-referential to each other as opposed to the other events in the temporal graph. We use the undirected link between two events \( e_j, e_i \) as a proxy for them being co-referential, and learn temporal event communities utilizing the concept of modularity, first introduced by (Newman and Girvan, 2004).

Formally, let \( A \) be the undirected adjacency matrix for a temporal graph \( G(V, E) \) such that \( A(e_i, e_j) = 1 \) if \( e_i \) and \( e_j \) are connected by a temporal relation, and 0 otherwise. Further, let \( \delta(e_i, e_j) = 1 \) if events \( e_i, e_j \) belong to the same temporal community, and 0 otherwise. For a given \( \delta \), we denote the fraction of the edges that exist between events that belong to the same communities by \( f_{\text{same}} = \frac{\sum_{e_i, e_j \in E} A(e_i, e_j) \delta(e_i, e_j)}{2|E|} \). Where the \( 2|E| \) in the denominator is due to the fact that \( A \) treats \( G \) as an undirected graph. Let the popularity \( p \) of an event \( e_i \) be the number of events that are linked to it i.e. \( p(e_i) = \sum_{e_j \in E} A(e_i, e_j) \).

The probability of randomly picking an event \( e_i \) when sampled by popularity is \( \frac{p(e_i)}{\sum_{e_j \in E} p(e_j)} = \frac{p(e_i)}{2|E|} \).

Thus, if edges are created randomly by sampling nodes by popularity \( p \) of the nodes, the fraction of edges within the communities, \( f_{\text{random}} \), is given by

\[
f_{\text{random}} = \frac{\sum_{e_i, e_j \in E} p(e_i)p(e_j)\delta(e_i, e_j)}{2|E| + 2|E|}.
\]

Finally, defining modularity, \( Q \), to be \( f_{\text{same}} - f_{\text{random}} \):

\[
Q = \frac{1}{2|E|} \sum_{e_i, e_j \in E} A(e_i, e_j) - \frac{p(e_i)p(e_j)\delta(e_i, e_j)}{2|E|}
\]

We want to learn community assignments \( \delta \) that maximize \( Q \). A high \( Q \) would promote \( f_{\text{same}} > f_{\text{random}} \) and thereby encourage highly inter-connected event communities. Calculating such \( \delta \) directly is not tractable, since the complexity of such an operation would be at least exponential in the number of events (Newman, 2004). We use the fast implementation provided by (Clauset et al., 2004) for calculating event communities iteratively. The algorithm converges at \( Q 0.3 \). We use a similar approximation at test time: given a document \( D \), we first break it down into sub-documents using CAEVO and then feed each sub-document to our method.

B Dataset Statistics

Tables 8, 9, and 10 list various statistics calculated from the source data.

| Descriptor | #Articles |
|------------|-----------|
| terrorism  | 40909     |
| murders and attempted murders | 25169 |
| united states international relations | 17761 |
| united states armament and defense | 16785 |
| airlines and airplanes | 16103 |
| world trade center (nc) | 15145 |
| demonstrations and riots | 14477 |
| hijacking | 14472 |
| politics and government | 6270 |
| bombs and explosives | 5607 |

Table 8: Top Descriptors for the filtered Dataset. Note that each article is typically assigned more than one descriptor.

| Event verb | Raw frequency | % Frequency |
|------------|---------------|-------------|
| said       | 647685        | 9.60        |
| say        | 57667         | 0.86        |
| had        | 47320         | 0.70        |
| killed     | 43369         | 0.64        |
| told       | 42983         | 0.64        |
| found      | 41733         | 0.62        |
| made       | 40544         | 0.60        |
| war        | 35257         | 0.52        |
| get        | 30726         | 0.46        |
| make       | 29407         | 0.44        |

Table 9: Most frequent events extracted by CAEVO.

C Examples

Figures 4-9 show randomly picked examples from the test corpus. Each figure shows the text, the corresponding true graph, and the graph predicted by GPT-2.
They were parents and grandparents who thought they had fully grasped the perils facing their teenagers in the tough working-class streets north of Kennedy Airport in Queens. They said that they understood the power of peers, the lure of gangs and drugs, the impact of movies and music and television that with relentless repetition depicted a culture of casual violence. And, they said, they believed they had taken the necessary precautions. They gave their children beepers and cellular phones to keep better track of them. They lined up mentors, and set curfews, and encouraged faith.

| Relation       | Raw Frequency | % Frequency |
|----------------|---------------|-------------|
| BEFORE         | 2436201       | 54.51       |
| AFTER          | 1772071       | 39.65       |
| IS INCLUDED    | 131052        | 2.93        |
| SIMULTANEOUS   | 112509        | 2.52        |
| INCLUDES       | 17465         | 0.39        |

Table 10: Relation Frequency in our Corpus

| Relation       | Frequency |
|----------------|-----------|
| BEFORE         | 98715     |
| AFTER          | 68582     |
| IS INCLUDED    | 6179      |
| SIMULTANEOUS   | 6209      |
| INCLUDES       | 285       |

Table 11: Edges in Generated Graphs: Top
Iran, with its 65 million Shiites, its powerful army and its ancient civilization, is the de facto master of the Persian Gulf. Tehran is clearly pleased that Iraq’s 15 million Shiites will more or less control their country eventually. In Lebanon, with one million Shiites, the well-armed Hezbollah militia has proved itself a most effective military-social-political group, which even forced both American and Israeli armed forces from the country. There are 400,000 Shiites in Bahrain and several million more in pockets from Pakistan to Saudi Arabia.

Figure 5

In the last month, Eastman Kodak, which has been shrinking for the last decade, announced plans to lay off an additional 6,000 to 8,000 workers. The Rochester police vowed to step up efforts after it reported a homicide rate that was the worst in years. Candidates competing for the region’s top offices are holding regular news conferences to criticize one another.

Figure 6
He dreams of being a biochemist and revels in science fiction novels, insect collecting and dismantling broken electronic equipment. He acknowledges that his actions were potentially dangerous. 'People can't see the difference between showing off, testing the boundaries and killing people,' he said. 'I went into the school to see what I could get away with, not because I wanted to kill anyone.' Michael said the most difficult part of his punishment was the six weeks he spent at the Essex County Youth House in Newark. One classmate, he said, beat him and tried to strangle him.

Figure 7

In a dispatch that Mr. Altgess wrote for the agency later that day, he said: 'There was a burst of noise, the second one I heard, and pieces of flesh appeared to fly from President Kennedy's rear. Blood covered the whole left side of his head. Mrs. Kennedy saw what had happened to her husband, she grabbed him, exclaiming, 'Oh, no!' The Associated Press, in its book on the assassination, 'The Torch is Passed ...' which was published soon afterward, republished Mr. Altgess's photograph of the First Lady and the agent with a caption saying it shows Secret Service Agent Clint Hill leaping toward Mrs. Kennedy as she desperately moves for help in the first moment of horror.'

Figure 8
An article yesterday about the release of an Egyptian chemist who had been held by Egyptian police as a suspect in the July 7 London transit bombings misstated the date he was arrested in Cairo. It was July 14, not June 14.

Figure 9