

Modeling Preconditions in Text with a Crowd-sourced Dataset

Heeyoung Kwon¹, Mahnaz Koupaee¹,
 Pratyush Singh¹, Gargi Sawhney¹, Anmol Shukla¹, Keerthi Kumar Kallur¹,
 Nathanael Chambers² and Niranjan Balasubramanian¹

¹ Stony Brook University, Stony Brook, New York

² US Naval Academy, Annapolis, MD

{heekwon, mkoupaee, niranjan}@cs.stonybrook.edu

nchamber@usna.edu

Abstract

Preconditions provide a form of logical connection between events that explains why some events occur together and information that is complementary to the more widely studied relations such as causation, temporal ordering, entailment, and discourse relations. Modeling preconditions in text has been hampered in part due to the lack of large scale labeled data grounded in text. This paper introduces PeKo, a crowd-sourced annotation of *preconditions* between event pairs in newswire, an order of magnitude larger than prior text annotations. To complement this new corpus, we also introduce two challenge tasks aimed at modeling preconditions: (i) Precondition Identification – a standard classification task defined over pairs of event mentions, and (ii) Precondition Generation – a generative task aimed at testing a more general ability to reason about a given event. Evaluation on both tasks shows that modeling preconditions is challenging even for today’s large language models (LM). This suggests that precondition knowledge is not easily accessible in LM-derived representations alone. Our generation results show that fine-tuning an LM on PeKo yields better conditional relations than when trained on raw text or temporally-ordered corpora.

1 Introduction

Recognizing logical connections between events in text is important for comprehensive document understanding and to improve global coherence in language generation systems. There is a rich body of work in identifying relations between textual events which covers causation (Mirza et al., 2014), temporal relations (Pustejovsky et al., 2003), textual entailment (Dagan et al., 2005), and discourse relations (Blair-Goldensohn and McKeown, 2006).

In this work, we focus on the precondition relation, which offers a general view of *why* cer-

tain events occur together in the world. This is not easily deduced from other event-event relations. Temporal ordering systems can sequence the order in which events occurred (Bethard, 2013; Chambers et al., 2014; Han et al., 2019) but can’t explain why they occurred at all. Which events in a sequence were by chance, and which were required? Textual entailment identifies event paraphrases (Berant et al., 2015) and some causation (Girju, 2003a), but their view misses the broader look at enabling events like preconditions. Let the following serve as an example:

I heard a bird **sing** above as I **turned** the key in the door. It **opened** with a **push**.

You can sequence these four events in order, but an ordering does not understand the *why* of the situation. One of these events (*sing*) is clearly not relevant to the door opening. How do we know that *turning* the key is a precondition to *opened* and not *push*? Turning the key usually doesn’t cause the door to open (perhaps on some doors, but here a *push* was needed). Turning is simply a precondition. Causation and entailment do not apply to *turn* either. Preconditions thus provide a unique and still fine-grained understanding of this situation.

How do we build models that can recognize (and learn from) this type of common-sense knowledge in text? Do language models trained on vast amounts of data already capture it? Since there are no large scale datasets that can effectively answer these questions, we introduce **PeKo**, the **Precondition Knowledge** dataset. We also introduce two tasks – one aimed at recognizing preconditions in text, and the other at generating precondition events for any given target event.

The core contribution in this paper is this new publicly available crowd-sourced PeKo dataset. It

consists of 28,948 event pairs annotated with precondition relations. We will first present our working definition of preconditions, and then discuss how to practically get crowd workers to identify them in text. We provide analysis of the new corpus and compare it against other existing corpora.

In addition to the corpus, this paper proposes two new challenge tasks. The first is a traditional classification task on the corpus itself. We thus address critical questions of how to model precondition knowledge. For instance, do today’s large language models (e.g., BERT or XLNet) already capture precondition knowledge, and how do they perform on a precondition prediction task? Second, does textual context assist in precondition prediction? We experiment with varying levels of context and show that identifying preconditions requires careful modeling of the context.

The second proposed task is a precondition *generation* evaluation: models must generate necessary preconditions for a given target event. This is a test for how well models can reason about the necessary preconditions for a given situation, which is a useful capability for story generation and learning generalized scripts. We show how PeKo can be used to train (fine-tune) standard generative models, such as GPT-2, for this task. Empirical results show that fine-tuning on the PeKo-derived training set generates at least twice as many preconditions as compared to training on general instances.

All code and data are available at <https://stonybrooknlp.github.io/PeKo/>.

2 Related Work

There has been a vast amount of research on extracting different types of relations between events including temporal (Pustejovsky et al., 2003), causal (Girju, 2003b), and paraphrasal relationships (Lin and Pantel, 2001), but relatively less research into precondition relationships. One of the early definitions and computational use of preconditions comes from the STRIPS program (Fikes and Nilsson, 1971). Preconditions were defined as a set of conditions that MUST be met in order for the action (event) to be allowed to take place.

Later work focused on aggregating precondition knowledge for a small class of action words, leveraging FrameNet and a text corpus to generate candidate precondition words using a PMI-based

heuristic (Sil et al., 2010; Sil and Yates, 2011). Using small amounts of labeled data, they use hand-crafted PMI and wordnet based features to learn a SVM-based classifier that scores preconditions for a given action. Branavan et al. (2012) learned domain-specific preconditions from written instructions for the game of Minecraft. The instructions are procedural and well suited for identification. These mostly target preconditions that are event-state relations as opposed to our goals of textual event-event identification.

ATOMIC (Sap et al., 2019) is a related crowd-sourced dataset of event-event relations, where given a simple target event (verb phrase and its arguments), crowd workers provided various types of common-sense knowledge. This included ‘NEED’ events analogous to our precondition events for a target. The main difference is our work grounds both target and precondition events in news text, whereas ATOMIC elicits general world knowledge, a complementary approach with different trade-offs. Interestingly, we find that the precondition relations learnt from textually grounded news events generalize to story events in ATOMIC for our generation task.

Annotated Text Corpora Three existing datasets capture some form of precondition knowledge in their annotations: the Rich Event Description (RED) dataset (O’Gorman et al., 2016), CaTeRS (Mostafazadeh et al., 2016), and Event StoryLine (Caselli and Vossen, 2017). These are generally too small for learning text classifiers as we briefly describe now.

RED is the most directly related, created to model a broad set of event-event relations in news. Preconditions are not their sole focus, though, so this dataset only contains ~1000 precondition instances. CaTeRs shares a similar problem to RED. It has an enables relation similar to precondition, but since the domain is 5-sentence short stories and preconditions aren’t the main focus, it only has ~400 instances. The Event StoryLine dataset is small in size too, but also doesn’t have a precise precondition relation. The dataset instead has RISING_ACTION that includes preconditions in its definition, but the same label captures other concepts like subevents and entailment. There are ~5000 instances, but only a fraction are preconditions and it is not possible to separate them out.

This paper is thus unique to prior work by annotating grounded written text at a scale large enough

to enable machine learning solutions. This enables our target tasks: text classification and generation.

3 Preconditions as Relations

Our goal is to develop a resource that can help models reason about the necessary preconditions for events mentioned in text. This is useful for planning towards a goal, explaining how a certain situation came about, and predicting what future events are plausible. We make two important design choices in building such a resource: **Grounding** – the resource is grounded in text, particularly over events in the news domain, and **Framing** – we construct the resource with preconditions framed as event-to-event relation pairs in a specific context.

Grounding: We ground the resource to text so that we can leverage the full context of the events, and we choose the news domain due to its common use in other event-related tasks such as event extraction, schema generation, and temporal reasoning.

Framing: Broadly speaking, preconditions specify what must exist/happen before something else can exist/happen (Fikes and Nilsson, 1971; Sap et al., 2019). It is natural to think of a precondition as a state of the world that must be satisfied for an event to happen i.e. a state-event relation. However, the state of the world is hard to circumscribe for most real world events, and more importantly the precondition state is often left unsaid in a story. Rather, the author will more often mention an event from which it follows that the precondition state is satisfied. Thus, it makes sense to frame preconditions as relations between *two events* described in their specific textual context.

We first present a formal definition based on this notion and then describe a crowdsourcing methodology for obtaining this knowledge at scale.

Definition: Given a target event mention t and a candidate event mention p , we assert p is a precondition event for t if p is necessary for t to happen i.e., t likely would not have occurred without p , in the current text context.

Using the example of opening a door from the Introduction, turning the key is a precondition event (for opening the door) because it results in

a state where the door is unlocked. The opening event cannot occur without such a state. Importantly, we do not define a precondition event as an absolute requirement for the target (the door opening) to occur in all scenarios. However, we do require that the target event *likely* would *not* have occurred in the *current context*. This allows another story with an alternate event, such as “I picked the lock”. Both picking-lock and turning-key are preconditions in their own story contexts. Strict logicians might take issue, but language understanding requires a looser definition that uses likelihood of occurrence when interpreting real-world scenarios.

4 Preconditions Dataset

This section describes our methodology to annotate news articles with the previous section’s definitions. One problem with annotating preconditions in text is the large number of event mentions in each article, which means annotation of all possible event pairs is infeasible. The temporal community has struggled with this same dilemma (Chambers et al., 2014; Vashishta et al., 2019).

We address the question of which pairs to annotate with two approaches. First, instead of attempting a dense annotation of *few* articles, we sub-sample candidate pairs of events across *many* articles. Second, we use an automatic temporal relation classifier to filter pairs by identifying possible candidates. We then ask crowd-workers to annotate the resulting pairs for preconditions.

4.1 Candidate Event Pair Extraction

Sub-sampling event pairs at random from a document can result in a large number of pairs that are not preconditions. Because precondition event pairs ought to be temporally related (i.e., the precondition should precede the target event), we can filter the candidate event pairs to only those that are in a BEFORE or AFTER relationship.

As a first step, we extract events and their temporal relations from news articles using CAEVO (Chambers et al., 2014), a temporal relation extraction system. We chose CAEVO over other available systems for two main reasons, although it’s not the only option out there: (1) it automatically extracts *both* events and their temporal relations, and (2) it extracts events in any form (verbs, nouns, and adjectives), which gives a broader coverage than some other recent systems

1. She was hit by a drunken driver six years ago, further complicating the multiple sclerosis that had been diagnosed years earlier.			
Is hit a valid event?	<input type="radio"/> Yes	<input type="radio"/> No	<input type="radio"/> Not Sure
Is complicating a valid event?	<input type="radio"/> Yes	<input type="radio"/> No	<input type="radio"/> Not Sure
Is hit necessary for complicating to happen?	<input type="radio"/> Yes	<input type="radio"/> No	<input type="radio"/> Not Sure
2. This month, Ms. Burton goes to family court to seek custody of the twins, William Brandon Sykes and Breanna Geneva Sykes, who turned 9 on Saturday.			
Is turned a valid event?	<input type="radio"/> Yes	<input type="radio"/> No	<input type="radio"/> Not Sure
Is seek a valid event?	<input type="radio"/> Yes	<input type="radio"/> No	<input type="radio"/> Not Sure
Is turned necessary for seek to happen?	<input type="radio"/> Yes	<input type="radio"/> No	<input type="radio"/> Not Sure

Figure 1: Example instances annotated by crowd-workers. Each HIT included ten such instances.

that only consider verbs as events (Ning et al., 2018). We used CAEVO on a random sample of 6,837 articles in the New York Times Annotated Corpus (Sandhaus, 2008).

On average CAEVO extracted around 63 events per article, which yielded a total of 3,906 possible relation candidates per document. We filtered these to retain only pairs of events that have a BEFORE or AFTER temporal relation between them. We call the temporally preceding event the *candidate precondition*, and the temporally subsequent event in the pair the *target event*. We filtered out pairs involving causative targets or reporting verb preconditions to remove trivial context independent preconditions (see Appendix for examples).

From the remaining, we randomly sampled 40,500 pairs for annotation. We used the first 500 in a pilot annotation to help us improve the task instructions. We then used the remainder for the actual annotation.

4.2 Crowdsourcing

The annotators were presented with a text snippet and two event mentions highlighted. Figure 1 shows two examples. To prune out event extraction errors from CAEVO, the annotators were first asked if the highlighted text denoted valid events. An event was deemed valid only if it describes an action that occurs in the world.¹ If both triggers

¹We left the decision for event validity up to annotators on their own. We asked annotators to consider an event with its context rather than the meaning of the word alone. This includes the negation of an event, which might imply a prevention relation.

	Precond.	Non-Precond.
#Evaluated	200	200
Errors	13.5%	9%
- Event Validity	1.5%	3.5%
- Relation	12%	5.5%

Table 1: Expert review of PeKo annotations. "Event Validity" indicates annotation error on validity labels, "Relation" indicates errors on identifying the event-event relation.

were deemed valid, then the annotators evaluated whether or not the candidate precondition event was an actual precondition for the target event. Specifically they check if the candidate event is necessary for the target event to happen.²

We used a pilot task to refine the instructions and the examples to improve consistency amongst the annotators. For the main annotation task, we used *four* crowd-workers to annotate each instance. For quality control, each HIT included control instances whose labels we knew *a priori*. We retained only those event pairs where a majority (i.e., at least three) of the annotators agreed on the label and use the majority label as the gold label for each instance.

4.3 Dataset Quality and Analysis

The resulting dataset, which we call PeKo, contains more than 30K annotated relations (~10k preconditions, ~20k not).

Annotation Quality: The annotators had fair *inter-annotator agreement* with a Fleiss Kappa value $\kappa = 0.387$. We used 4 Turkers per event pair to ensure accuracy and filter out disagreements. To further measure the quality of the annotation, we randomly sub-sampled 400 instances from the annotated data and re-annotated them using four “expert” graduate students trained to recognize preconditions. A post-analysis of the expert and crowd annotations shows the annotation to be of high quality. Table 1 summarizes the quality statistics. Experts disagree with the crowd-sourced annotations in only 11.75% of the cases, with a slightly higher disagreement for precondition instances at 13.5%. A small percentage of these disagreements are on determining when an event is valid.

²We expected annotators to make decisions on the given CAEVO output, and they were not allowed to suggest a directional change. We limited the number of labeling options to keep the annotation instructions as straightforward as possible.

We also analyzed the discarded instances that received conflicting votes. Only 10% of these instances can be considered as preconditions and some of them are arguable based on their context. Here’s an example:

Before he was **hired** in 2005, before his team upset Texas last season, he **educated** himself on the college culture.

According to the context with discourse cues, one can reasonably conclude that **educated** is necessary for the event **hired** to happen. However, one might also disagree based on the fact that the connection is not perfectly clear.

Text Position: As with temporal and other event-event relations, one might ask if position in text is an indicator of a precondition relation. We thus tallied our annotations and identified how many intervening verbs occurred between the annotated event pairs, as well as how far apart they are in the document based on token distance. Figure 2 shows these distributions. The negative numbers indicate distance when the precondition event occurs *after* the target event. As the graphs show, the majority of preconditions occur first in the text, but a sizable amount are actually reversed with an evenly spread out distribution over distance.

Precondition Predicate → Target Predicate	
pay → provide	try → get
know → miss	ask → make
use → provide	love → miss
go → provide	delay → mean
look → find	find → use
take → get	ask → take
work → make	tell → take
use → find	know → get
born → die	agree → pay
use → help	touch → miss
go → find	get → help
move → take	lose → help
leave → take	

Table 2: The 25 most frequent predicate pairs in the annotated event pairs.

For further insight into the dataset, Table 2 lists the most frequent verbs that were annotated as precondition-target pairs. While there are a few pairs that can be readily interpreted without other context (e.g. everyone is born before they can die, and you must look before you can find), most other pairs require additional context from the text itself.

4.4 Comparison to Other Datasets

Section 2 described how this new dataset differs from prior work. We now include Table 3 to fur-

ther illustrate the size difference, showing an order of magnitude more precondition instances than prior corpora with specific precondition annotations.

We consider our precondition as a broader concept than that in the RED. We focus on *necessary* events, which covers both precondition and causal relations in the RED dataset.

Dataset	#Instances	#Precond.
RED (news/forums)	4,969	1,055
CaTeRS (stories)	2,715	488
StoryLine (news)	12,423	< 5,519*
PeKo (news)	28,948	10,806

Table 3: Comparison of labeled corpora. The *instances* are how many total labels, and *precondition* is how many precondition-related instances. We included causation+precondition labels in the total counts if causation exists. *Event StoryLine mixes preconditions with many other relations, so the 5,519 is an upper bound.

5 PeKo Tasks and Evaluation

Having created the PeKo annotated corpus, we now propose two tasks that test for the ability to recognize and generate preconditions in textual contexts. Here we describe evaluations to benchmark the performance of current models on these tasks and to better understand the challenges involved.

5.1 PeKo Task 1: Precondition Identification

Given a text snippet with a target and candidate event pair, the task is to classify if the candidate event is a precondition for the target in the context described by the text snippet. This is a standard sentence-level classification task. We evaluate two strong and widely-used large transformer-based language models – fine-tuned BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) base models. For each model, we take the final representation of each event trigger, concatenate together, and then feed into a linear classification layer. We also evaluate a 1-layer GRU sequence model (Cho et al., 2014) with GloVe embeddings (Pennington et al., 2014) to calibrate against a much simpler baseline. See the Appendix for more details on parameters, layer sizes, and training time.

Precondition identification is a difficult task.

Table 4 shows the results. The GRU-based sequence model trained from scratch on PeKo is bet-

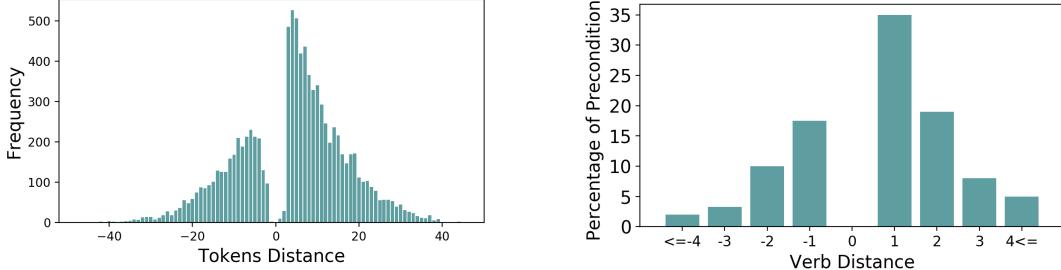


Figure 2: Distribution of distances between preconditions and target events. Negative numbers correspond to cases where the target precedes the precondition, and positives for the other way around. The left plot shows the number of intervening tokens and the right shows percentages of verb distances between precondition and target events.

Precondition Identification				Text Context Ablation			
Model	Precision	Recall	F1	Context	Precision	Recall	F1
Random	37.34	50.00	42.75	Event Trigger	54.06	75.68	63.07
GloVe-GRU	56.25	73.38	63.68	Event Tuple	64.02	76.97	69.90
BERT-feature	59.80	81.15	68.84	Event Tuple(± 1)	63.84	78.95	70.59
XLNet	66.69	77.10	71.52	Sentence	64.65	81.92	71.91
BERT	64.65	81.02	71.91	Sentence(± 1)	62.69	76.92	68.47
				Sentence(± 2)	61.65	77.33	68.60

Table 4: Benchmarking performance of existing models on the precondition identification task. Simply fine-tuning large language models is not enough.

ter than a prior-based random baseline³ but still leaves a large room for improvement. BERT and XLNet both perform substantially better (> 71 F1) than the GRU model (63.7 F1) but their F1 score of 71 illustrates that this is a difficult task not readily solved by simply fine-tuning large LMs.

Precondition information is not readily available in BERT.

One premise for our work is that distributional knowledge alone is insufficient to capture precondition relations. We conduct two sets of *inoculation-based probing experiments* (similar to Liu et al. (2019)) to get at how the information in the pre-trained LM representations relate to preconditions. We use BERT in the fine-tuning and feature-extractor mode (the parameters for BERT are fixed and only those in the classification layer are updated) and measure performance with increasing amounts of data. If the performance peaks early with only small amounts of data then it tells us that most of the information necessary for recognizing preconditions is in a readily accessible form in the original LM representations. On the other hand, if performance keeps increasing then it suggests that PeKo provides extra information.

³Chooses a label at random from a binomial distribution of labels estimated from the training data

Context	Precision	Recall	F1
Event Trigger	54.06	75.68	63.07
Event Tuple	64.02	76.97	69.90
Event Tuple(± 1)	63.84	78.95	70.59
Sentence	64.65	81.92	71.91
Sentence(± 1)	62.69	76.92	68.47
Sentence(± 2)	61.65	77.33	68.60

Table 5: Precondition identification results with varying levels of context using our BERT classifier.

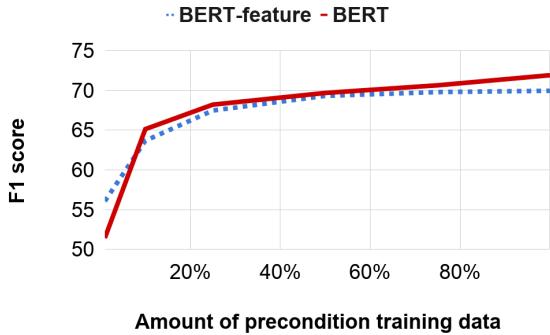


Figure 3: Inoculation: Performance of fine-tuning (solid) and feature extractor (dotted) modes of BERT with increasing amounts of PeKo training data. Neither plateaus quickly suggesting that precondition knowledge is not readily accessible in BERT.

Figure 3 shows that neither model plateaus quickly. BERT, as a feature-extractor (dashed line) plateaus around 50% of the data. The fixed features from the LM pre-training BERT hits a performance ceiling. Whereas fine-tuning BERT, which fine-tunes the representation to the PeKo task, provides continuous improvements for increasing amounts of data. These together suggest that a substantial amount of precondition knowledge is not easily adapted from the language modeling information captured in BERT but can be learned from PeKo.

Role of Context. Table 5 compares the perfor-

mance of BERT when using different levels of context. Using event triggers alone achieves moderate performance. This suggests that the verb trigger does carry a lot of the precondition knowledge regardless of event arguments (e.g., canceling requires scheduling first, but in most cases it doesn’t matter what is canceled). However, if we use event tuples⁴, which also captures the main entities of the event, then we see a significant improvement in performance (+6.9 points). In addition to the tuples of the event pair, adding tuple representations of neighboring events provides an additional gain (+1.5 points). Further inspection of the tuple-based representation shows that automatic tuple extraction sometimes introduces errors and misses critical context and other important discourse cues. The best results come from using the sentence(s) that contain the event pair in its entirety – adding further sentences leads to worse performance.

When is it difficult to identify preconditions?

The first plot in Figure 4 shows that F1 score is highest where the target event is in the same sentence as the precondition event, higher where the target event is in the sentence that follows the precondition event, and lowest when the target event is in the previous sentence. A similar trend holds for different verb distances as well, as seen in the second plot. As the distance increases, the F1 score decreases in either direction. However, on the negative side, F1 scores are lower compared to the positive side showing the difficulty of the task when the target verb precedes the precondition.

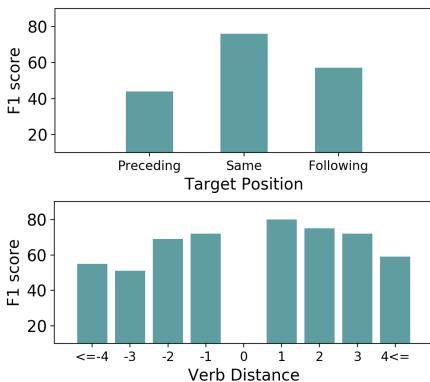


Figure 4: F1 scores across different contexts. Top: F1 when the target event precedes, is in the same, or follows the precondition’s sentence. Lower: F1 for varying # of intervening verbs between the event pair.

⁴We used OpenIE(Stanovsky et al., 2018) to extract event tuples implemented in AllenNLP(Gardner et al., 2018)

5.2 PeKo Task 2: Precondition Generation

Here we introduce Precondition Generation as a more general challenge that a dataset like PeKo now enables. Given a target event t , generate an event p that is a precondition for t . We first show how to create instances for this task using the PeKo dataset and then benchmark performance on evaluation instances drawn from both PeKo and an out-of-domain dataset ATOMIC (Sap et al., 2019).

Generation Training Task. We created precondition generation training instances by transforming each PeKo instance as follows. The *input* is the entire snippet of a PeKo instance (i.e, the entire text snippet with a pair of events where one is marked as a precondition of the other) but with the precondition portion of the snippet replaced by a [BLANK] slot. The *output* for the generation instance is the entire sentence where the [BLANK] is to be filled in with text representing a precondition event. See Table 7 for examples. Note that because the precondition portion can occur anywhere (earlier or later) in the sentence, we do not frame this as a typical left-to-right language model completion task. Instead, the models have to generate the entire sentence in addition to filling in the [BLANK] slot with a plausible precondition. We use the text chunk spanned by the precondition trigger node in the constituency parse as the precondition portion.

We benchmark three variations of a large language model GPT-2 (Radford et al., 2019) to show how much of precondition information can be generated directly from general language models and from temporal knowledge in comparison to learning from PeKo: (i) **LM-GPT-2** – training instances created from a random collection of sentences to mimic fine-tuning GPT only for the format of this task but with no special constraint on the relation between the events in the instance. We randomly select sentences with a pair of events, and choose at random one event as target and the other as precondition and then create the generative training instances as described earlier. (ii) **Temp-GPT-2** – training on instances created from temporally BEFORE events, randomly sampled from the non-precondition portion of PeKo dataset. (iii) **PeKo-GPT-2** – training on generation instances created from the training portion of the PeKo dataset. LM-GPT-2 trains on 18,000 instances (since it is not limited by PeKo data),

whereas Temp-GPT-2 and PeKo-GPT-2 train on 6000 instances.

Testing on PeKo For testing, we transform instances from the testing portion of PeKo. Because precondition instances can sometimes contain strong linguistic and syntactic cues for preconditions, we create test instances only from the non-preconditions in PeKo. This is a stronger test of models’ abilities that mitigates some of the confounds of how the sentence is structured.

Testing on ATOMIC We used the following heuristics to address the peculiarities of ATOMIC and improve compatibility with training. We filtered instances such that they are full sentences, with fully-specified arguments for events, and with single participant instances. We replace Person variable mentions with third-person pronouns.

Benchmarking Precondition Generation. Table 6 shows results of a manual evaluation of the generated preconditions⁵. Three of the authors of this paper evaluated 150 instances of generated text snippets from three systems. The snippets from the systems were randomly swapped during the blind evaluation. Each output was first rated for sensibility on a scale of 0 to 3, where 3 means the output is perfectly sensible as English, and 0 means nonsensical. The output, which contains the marked target and precondition event pairs, were then rated on a binary scale – 1 if the precondition relation holds; 0 otherwise. The same annotation guidelines described in Section 4.2 were taken to ignore invalid events, hypotheticals, and other noisy output.

Results in Table 6 shows that LM-GPT-2, the version that trains on random event pairs, struggles. It produces the least precondition outputs. PeKo-GPT-2 generates plausible preconditions nearly **twice as often** as the Temp-GPT-2 baseline. These results illustrate the need for PeKo as preconditions do not easily fall out from today’s large LMs. The trends also hold for the out-of-domain ATOMIC instances indicating generalization to everyday events in the ATOMIC dataset. On ATOMIC we see more preconditions than on the original PeKo dataset. We hypothesize that this

⁵ Automatic evaluation against reference preconditions is not informative since there can be multiple preconditions for any given event. We found that using BLEU for instance showed no difference between Temp-GPT2 and PeKo-GPT2 despite the huge difference in manual evaluation.

Precondition Generation			
Dataset	Model	Sense	Precond.
PeKo	LM-GPT-2	1.69	12.00% (12.87)
	Temp-GPT-2	2.19	17.56% (17.21)
	PeKo-GPT-2	2.32	35.81% (37.96)
ATOMIC	LM-GPT-2	2.20	10.40% (10.78)
	Temp-GPT-2	2.30	21.33% (28.97)
	PeKo-GPT-2	2.12	39.33% (50.43)

Table 6: Human evaluation of generation. Sense: Average sensibility rating on a 0-3 scale. Precond.: Percentage of instances with valid precondition outputs. Parenthetical numbers are percentages within instances with sensible score ≥ 2 . Bold face indicates best results.

is in part because in the PeKo test set, we created harder cases where the models have to generate preconditions to fit in text that originally contained a non-precondition event.

Table 7 shows some examples that illustrate the differences between training on PeKo and otherwise. As expected, the non-precondition trained model outputs events that temporally precede the target event but not necessarily preconditions.

Error Analysis. We evaluated the outputs for 50 instances from Peko-GPT-2 and found three main categories of failures: (i) Difficulty in handling input context (56%) – In some cases the input target event context is too limited, whereas in others the context is too complex with many intervening entities or a chained set of events after which the model is supposed to generate a precondition. Another set of cases have to do with the sentence structure of the context sets up for a hypothetical precondition event, or a reporting verb. (ii) Common Language Generation Errors (28%) – Cases like repetition or semantically implausible text and hallucinating new entities whose relation to the original context is not clear. (iii) Temporally related (16%) – Cases where the outputs are temporally and topically related but are not preconditions, indicating failures in generalizing precondition knowledge.

Overall, these first results on PeKo suggests that training on this new dataset enables a generative model to learn some common-sense precondition knowledge beyond basic language modeling cues. We see room for improvement both in terms of modeling as well as training approaches.

PeKo	
INPUT	[BLANK] that will <i>enable</i> consumers to quickly download videos...
Temp-G2	The company has <i>said</i> that it will be able to sell more phones in the coming months
PeKo-G2	The company also <i>agreed</i> on a plan
INPUT	And finally a third will <i>rebuild</i> homes in the historic older district. The foundations of a temporary market are in place, to house stores and stalls [BLANK]
Temp-G2	that have been <i>sold</i> for decades.
PeKo-G2	that were <i>built</i> in the 1880s.

ATOMIC	
INPUT	He <i>is</i> in dire need of money [BLANK]
Temp-G2	he <i>said</i> in an interview with The Daily.
PeKo-G2	because he has <i>lost</i> his job.
INPUT	She <i>moves</i> to cambridge in 2013 [BLANK]
Temp-G2	when she <i>became</i> the first woman to walk the halls of Congress.
PeKo-G2	she <i>took</i> a job as a waitress at a local restaurant.

Table 7: Generation Examples on PeKo and ATOMIC test instances: **INPUT** is the system input: text with the target event (italicised) and placeholder [BLANK]. Temp-G2 and PeKo-G2 are the generated outputs from the Temporal and PeKo GPT-2 systems, with the precondition event in bold.

6 Conclusions

Knowing what conditions are necessary for an event to happen is critical for understanding and reasoning about events mentioned in text. In this work, we address the lack of a large scale resource for learning precondition knowledge about events. Our crowdsourcing methodology yielded more than 10,000 precondition event relations (and 20,000 negative examples) from news domain texts. We showed in both classification and generation that these relations are not readily accessible in distributional knowledge encoded by large language models, highlighting the challenges in learning common-sense knowledge from text. We also proposed two new challenge tasks based on PeKo and hope it helps drive further research into rich event understanding that touches a variety of areas from schema learning, information extraction, and even story generation.

Acknowledgments

This material is based on research that is supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes.

References

- Jonathan Berant, Noga Alon, Ido Dagan, and Jacob Goldberger. 2015. Efficient global learning of entailment graphs. *Computational Linguistics*, 41(2):249–291.
- Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14.
- Sasha Blair-Goldensohn and Kathleen McKeown. 2006. Integrating rhetorical-semantic relation models for query-focused summarization. In *Document Understanding Workshop (DUC-2006)*.
- S.R.K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 126–135, Jeju Island, Korea. Association for Computational Linguistics.
- Tommaso Caselli and Piek Vossen. 2017. The event storyline corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86.
- Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.
- Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Richard E Fikes and Nils J Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

- Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.
- Roxana Girju. 2003a. Automatic detection of causal relations for question answering. In *ACL*.
- Roxana Girju. 2003b. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 76–83, Sapporo, Japan. Association for Computational Linguistics.
- Rujun Han, Qiang Ning, and Nanyun Peng. 2019. Joint event and temporal relation extraction. In *EMNLP*.
- Dekang Lin and Patrick Pantel. 2001. Dirt – discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328.
- Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. Annotating causality in the tempeval-3 corpus. In *Workshop on Computational Approaches to Causality in Language at EACL*.
- Nasrin Mostafazadeh, Alyson Greathouse, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California. Association for Computational Linguistics.
- Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018. Improving temporal relation extraction with a globally acquired statistical resource. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 841–851, New Orleans, Louisiana. 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.
- 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.
- James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TimeBank corpus. *Proceedings of Corpus Linguistics*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Evan Sandhaus. 2008. The New York Times Annotated Corpus.
- 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.
- Avirup Sil, Fei Huang, and Alexander Yates. 2010. Extracting action and event semantics from web text. In *2010 AAAI Fall Symposium Series*.
- Avirup Sil and Alexander Yates. 2011. Extracting STRIPS representations of actions and events. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 1–8, Hissar, Bulgaria. Association for Computational Linguistics.
- Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *NAACL-HLT*.
- Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2906–2919, Florence, Italy. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

A Appendix

A.1 Candidate Filtering for Crowdsourcing

We discard event pairs that come from the same sentence when the candidate precondition is a causative verb or when the target is a reporting verb. This is because both cases are always true regardless of their context. Consider the following examples:

(A) He *said* that his birth mother *lived* nearby.

(B) The president *made* his secretary *create* copies of the report

As these examples show – A is a reporting verb (‘said’) in the target position, and B is a causative (‘made’) as the candidate precondition – the candidates in both cases are reliable preconditions independent of the context. For instance, in example in (B) if we use a new context “not *create* copies of the report”, the precondition relation would still hold. Since we aim to collect precondition knowledge that can be obtained at least partially from context, we excluded these reporting and causative precondition verb instances from our candidate pool.

A.2 Experimental Details

A.2.1 Data Split

We split our dataset into train/dev/test set with the ratio of 6:2:2. Since the number of instances in each class is imbalanced, we split the data separately based on the class and then randomly shuffle instances in each set together.

A.2.2 Infrastructure

All models are trained using NVIDIA Titan RTX (24GB of GDDR6 VRAM).

A.2.3 Parameters

Identification Task: All models for identification task are trained for 50 epochs with 16 of the batch size. A model is picked based on the performance (i.e., F1 score) on the dev set among 5 different random seeds. All other parameters are described in Table 8.

Generation Task: All three models use the same GPT-2 architecture, which has 163,047,936 trainable parameters. The epochs are set to 100 with 16 as the batch size. Models are picked based on loss on the dev set.

We use AdamW (Loshchilov and Hutter, 2019) for the optimizer in both tasks.

Model	Hidden size	#Parameters
GloVe-GRU	512	9,675,154
BERT-feature	768	3,074
XLNet	768	116,721,410
BERT	768	108,313,346

Table 8: Parameters for the identification models. For GloVe-GRU model, we use GloVe embeddings with the size of 300.

A.2.4 Training Time

Table 9 shows the training time for each model. The time is measured by the average elapsed time for each epoch excluding testing time on the dev set.

Task	Model	Time
Identification	GloVe-GRU	25.29s
	BERT-feature	154.18s
	XLNet	204.15s
	BERT	235.85s
Generation	LM-GPT-2	574.99s
	PeKo-GPT-2	126.83s
	TEMP-GPT-2	130.20s

Table 9: Average training time for each model on an epoch.

A.3 Testing on ATOMIC

We follow heuristics to address the peculiarities of the ATOMIC dataset and improve compatibility with training: 1) We remove instances that do not have a fully specified argument for the event (referred to as placeholders in their paper (Sap et al., 2019)). 2) We only use ‘simple’ instances that mention a single participant because the context often contains enough information to fully understand the target event. 3) We only use instances that are complete sentences and not fragments. 4) To make the inputs more natural, we replace the Person variable mentions with a third-person pronoun and added markers to the main verb and the placeholder [BLANK] at the end:

“PersonX is in dire need of money” to “He <target> is </target> in dire need of money [BLANK]”