Toward Diverse Precondition Generation

Heeyoung Kwon[1], **Nathanael Chambers**[2], **and Niranjan Balasubramanian**[1]
[1]Stony Brook University, Stony Brook, New York
[2] US Naval Academy, Annapolis, MD
{heekwon, niranjan}@cs.stonybrook.edu
nchamber@usna.edu

Abstract

Language understanding must identify the logical connections between events in a discourse, but core events are often unstated due to their commonsense nature. This paper fills in these missing events by generating *precondition events*. Precondition generation can be framed as a sequence-to-sequence problem: given a target event, generate a possible precondition. However, in most real-world scenarios, an event can have *several* preconditions, requiring diverse generation – a challenge for standard seq2seq approaches. We propose DiP, a **Di**verse **P**recondition generation system that can generate unique and diverse preconditions. DiP uses a generative process with three components – an event sampler, a candidate generator, and a post-processor. The event sampler provides control codes (precondition triggers) which the candidate generator uses to focus its generation. Unlike other conditional generation systems, DiP automatically generates control codes without training on diverse examples. Analysis against baselines reveals that DiP improves the diversity of preconditions significantly while also generating more preconditions.

1 Introduction

Preconditions are an important part of language understanding with numerous applications, ranging from event understanding to story generation. They provide the semantic glue to understand (or generate) the chains of events common in narrative text. How can we build intelligent systems to fill in these chains, or to identify semantically related events in context? Kwon et al. (2020) took a first step by introducing a precondition generation task, where given a target event mention the goal is to generate text that describes a precondition for the target. They released the 'PeKo' dataset for training, and showed that a GPT-2 model can be fine-tuned on input/output sequence pairs.

TARGET: [BLANK] to **fill** Mr. Lavelle 's seat, for a term that expires on Dec. 31, 2008.
The Senate **voted** overwhelmingly on Thursday
The Senate **voted** on Wednesday
The Senate **voted** overwhelmingly on Wednesday
The Senate **voted** overwhelmingly on Tuesday
Mr. Lavelle was **appointed** by Gov. Eliot Spitzer

Table 1: Top 5 preconditions generated from GPT-2 with beam search decoding. Key problem: the top 4 preconditions are almost identical.

While PeKo is useful, it is constrained by annotating a single relation for each target event. This is contrast to the real-world where most events have many preconditions. For example, "*opening* a door" has several preconditions like *approaching* the door, *turning* a key in the door, and *pushing* the door. PeKo's annotation limits the ability of models to learn to generate multiple and *diverse* preconditions[1]. In this work, we address the challenge of generating more preconditions for each target event while still maintaining quality.

Generating non-repetitive diverse outputs is a challenge for any conditional language generation system. Our analysis of the GPT-2 based model shows that this is also the case for preconditions. Table 1 shows such top preconditions for an example event. Standard sampling techniques produce high-levels of lexical and semantic redundancy. In the absence of any explicit mechanisms to force diversity, the model just produces minor variations of the same event as preconditions. To obtain diverse candidate preconditions, we have to start looking lower in the model's ranked lists of probable preconditions, thereby sacrificing quality.

How can we induce a model to generate diverse outputs without losing quality? Context sensitivity might help with quality, but it also hinders diversity. To address this we introduce a three-stage

[1]In order to observe diverse preconditions for the same (or similar) target event we would need a much larger training set.

Proceedings of the 10th Conference on Lexical and Computational Semantics, pages 160–172
August 5–6, 2021, Bangkok, Thailand (online)

generative process, which we call DiP. In the first stage, DiP uses an event sampler whose only goal is to generate event trigger words as precondition candidates. In the second stage, DiP forces the generative model to use the candidate triggers from the first stage to produce the full description of the precondition event. In the third stage, DiP reranks and filters the generated descriptions using a precondition classifier (also trained from the same training data).[2] A brief example is shown here:

Target Event: I apologized for the debacle of the day before, and `[BLANK]` to **help** me make it right

Stage 1	**Stage 2**	**Stage 3**
trying	I am **trying** now	*delete*
use	I **use** my time	*delete*
used	I **used** my time	#2
asked	**asked** me	#3
hired	**hired** a new staff	#1

Experiments on the PeKo dataset show that DiP produces more diverse and better quality preconditions compared to standard beam decoding, as well as an iterative filtering extension that applies a standard repetition penalty in a sampling strategy. Analyses show that DiP is able to better balance the need for diversity against quality. While the iterative repetition penalty method generates lexically diverse outputs, it often introduces irrelevant information rather than producing distinct types of preconditions. Our human evaluation shows that DiP on the other hand is able to produce text that is more likely to be preconditions.

All code and data are available at `https://stonybrooknlp.github.io/DiP/`.

2 Related Work

Most work on logical preconditions has focused on identification/extraction from text. For example, Sil et al. (2010) identified preconditions using a SVM-based score function with hand-crafted PMI and WordNet based features. Branavan et al. (2012) extracted domain-specific precondition relations from instructions for the game of Minecraft. This paper is instead focused on generating novel preconditions. To the best of our knowledge, only the prior PeKo work (Kwon et al., 2020) has attempted this. We are building on those initial ideas.

There has been research for diverse generation using control codes or latent variables. Some works use explicit cues to control text generation. Huang et al. (2018) used emotion embeddings to generate dialogue responses in a specific mood. Keskar et al. (2019) trained a LM with human readable control codes, which describe domain, style, or topics. Then the model learns to generate text conditioned on a given code. The model requires manually predefined control codes and a corresponding training corpus for each code.

Other diverse generation works learn latent representations or codes from input text, and then generate text conditioned on those codes. Shu et al. (2019) applied a sentence embedding to generate syntactically diverse translations. They find that syntax-based encoding with TreeLSTM (Socher et al., 2011) yields better diversity than a contextual encoding using BERT (Devlin et al., 2019) or FastText (Bojanowski et al., 2017). Bao et al. (2020) used K categorical latent variables to control the generation context of dialogue responses and pick the highest probability response from the responses generated using the latent variables. COD3S (Weir et al., 2020) is designed to generate diverse causal relations. It uses locality-sensitive hashing (LSH) (Indyk and Motwani, 1998) on representations from Sentence-BERT (Reimers and Gurevych, 2019). Conditioning on these 16-bit LSH signatures, it generates cause/effect sentences using a Transformer architecture (Vaswani et al., 2017) but with a limited vocabulary size of 10K.

These previous approaches have some drawbacks – they either require explicit control codes and training examples, or they have low interpretability of their codes. Our approach addresses these two limitations: control codes are learned from non-diverse input text and the codes are human-readable events. And these approaches are not directly comparable to our method without proper modification, which would not be fair comparisons. Thus, we present our own baselines for evaluation, and these baselines serve as a proxy of ablation studies as well.

3 Diverse Precondition Generation

This section describes our diverse precondition generation task and our methodology for solving it. Our proposed approach does not require additional diverse training examples.

3.1 Generation Task

This paper follows the precondition definitions from Kwon et al. (2020):

Precondition Definition – "Given a target event mention t and a candidate event mention p, we

[2] We will release the source code upon acceptance.

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."

Precondition Generation – "Given a target event *t*, generate an event *p* that is a precondition for *t*."

The precondition generation task is defined over sentences that contain both a target and a precondition event. The precondition part is masked and a model is asked to reconstruct the sentence including its precondition. For masking, the syntactic subtree of a precondition is replaced with `[BLANK]`. In order to indicate the events of interest – target and precondition – we use special tokens `<event>` .. `</event>` and `<pre>` .. `</pre>`.

For our new task, instead of generating the entire sentence, we only generate a precondition clause that would fit into the input's `[BLANK]`. Since a precondition could be stated in either preceding or succeeding position of its target event, we modeled this as a text infilling task. This approach is inspired by Donahue et al. (2020) and this modification allows the model to focus solely on generating preconditions because the model doesn't need to copy over its input text. Thus, the model can learn faster and more efficiently.

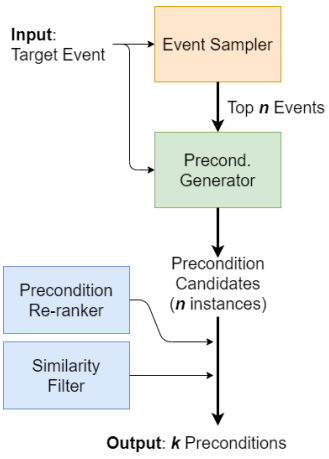


Figure 1: The DiP pipeline. Candidates are generated conditioned on the Event Sampler. The Re-ranker and Similarity Filter improve quality/diversity.

3.2 Diverse Precondition Generator

Generating preconditions is a difficult task even for a single output setting (Kwon et al., 2020). With the training data derived from existing news articles, generative models only get to see one possible precondition for each target event. Not surprisingly the top candidates in beam search tend to be focused towards a specific type of precondition event with minor variations. This suggests that we need to provide explicit guidance to the model to explore diverse candidates.

How can we get such diverse guidance? A main strength of large generative language models is that they learn to generate text that fits with the input context. If we can get the input context to be less specific then we can aim to get more general outputs. We can exploit this behavior by training a separate event sampler that is fed a reduced version of the target event description. For example, we can denote the target event by just the event trigger and its arguments. The event sampler learns to predict possible precondition event triggers based on this reduced context. This task forces the sampler to learn a more general mapping between target and precondition events that can produce a diverse set of starting points for generating the precondition events. We can then train another generative model to condition on the precondition trigger in addition to the input sentence. This gives us a model whose outputs we can control by providing different possible precondition triggers. Not all precondition triggers may yield high quality preconditions. To further assist the model, we also devise a precondition re-ranker.

Our overall system, shown in Figure 1, consists of three components – an event sampler, a candidate generator, and a post processor (Precondition re-ranker and Similarity filter). The first two stages are used for generation – they use two separate generation models, and the last is employed to improve the quality of generated preconditions. We refer to this system as DiP short for Diverse Preconditions.

3.2.1 Event Sampler

The event sampler provides possible precondition event triggers given a target event. This can be formulated as a sequence to sequence problem where the input sequence is a target event and the output sequence is a precondition event trigger. Since our goal here is to get diverse precondition events, we can experiment with input contexts of different levels of detail. To get more general precondition events, we use just the target event triggers as the input. To get more specific preconditions, we can use larger contexts surrounding the target event trigger as the input. During inference, we sample top n event triggers based on their probability.

Formally, let x' be a subset of the full description x of the target event. The sampler can be seen as a generative model that outputs event triggers e for

the preconditions of the target event.

$$\hat{e} = \arg\max_{e} \log p(e|x')$$

The generative model is trained to maximize the probability for the correct precondition trigger e and during inference can be used to sample a candidate set of top n precondition event triggers.

3.2.2 Candidate Generator

The candidate generator, as the name suggests, is a language model that we fine-tune for generating precondition candidates. We want this model to generate preconditions corresponding to the triggers from the event sampler. To this end, in addition to the full target event description x, we also provide a precondition trigger marked by a special token – `<E> precondition_event` – at the end of the input. This can be seen as a form of a control code similar to those used in Keskar et al. (2019); Weir et al. (2020). The crucial difference, however, is that the codes in our case are dynamically generated conditioned on the input and not restricted to a predefined set.

Formally, the candidate generator is a language model that generates a description of the precondition event c_i conditioning on the full description of the target event x and a given precondition trigger e_i from the event sampler.

$$\hat{c_i} = \arg\max_{c_i} \log p(c_i|x, e_i)$$

The model is trained to maximize the probability of the observed precondition text for the target event when provided with the correct precondition trigger. Note that during training, the precondition trigger provided as input always appears in the correct precondition description output ($\hat{c_i}$). This encourages the model to learn to incorporate the trigger provided at the end of the input as part of its output. During inference, the model generates a set of preconditions one for each of the top n triggers obtained from the event sampler.

3.2.3 Post Processor

Precondition Re-ranker We use a precondition re-ranker to reorder the generated candidates based on how likely they are to be preconditions of the target event. Note that the generative model is implicitly trained for a similar objective. However, the model is also forced to include the input precondition trigger which could make it harder to focus on ensuring the result is indeed a precondition. Therefore, we introduce a separate precondition classifier that scores the generated candidates. Note that the original PeKo dataset is already setup for training such a classifier (Kwon et al., 2020). Each instance in this dataset consists of an input text that includes a pair of marked event triggers (target, candidate) and a label that indicates whether the candidate is a precondition of the event denoted by the target trigger. The output from the precondition generator is essentially equivalent to an instance from this dataset. We build a classifier that scores a pair of events in text, and we use this score as an indicator of the precondition quality of the generated candidates and re-rank them based on this score.

Iterative Redundancy Filtering The resulting candidates are a mix of candidate precondition events from different triggers. To further avoid redundancy we also include an explicit filtering step, where we post-process the generated text based on their pairwise similarity. Specifically, we start with the highest ranked instance in the output set, and iteratively walk down the ranked list and add instances to the output if the highest similarity score they have with any of the current output set is lower than a certain threshold.

4 Evaluation

Our goal is to investigate the impact of our DiP approach for generating diverse and high-quality preconditions. We closely follow Kwon et al. (2020) for the experimental setup and the GPT-2 based generation system for our evaluation.

4.1 Datasets

For the fine-tuning task, we use the precondition generation instances in the PeKo dataset. In addition, we also create a large additional pre-training dataset that includes temporal generation instances. With this additional dataset we can perform a form of domain adaptive pre-training (DAPT) introduced by Gururangan et al. (2020). The main idea here is to create generation instances where the model gets to see a target event but now is required to produce an event that temporally precedes the target event. Since preconditions are supposed to be temporally preceding this temporal generation task can be seen as a more permissive yet related generation task, which is then subsequently restricted to only preconditions in the fine-tuning stage. We use the CAEVO (Chambers et al., 2014) system

Target Event	The Metropolitan Transportation Authority recently ***canceled*** some large projects
Trigger Only	took, canceled, succeeded, began, scheduled, died, rejected, decision, filed, pushed
±3 tokens	planned, took, began, expected, needed, approved, designed, intended, completed, brought
±5 tokens	planned, intended, took, began, aimed, devised, completed, expected, designed, needed
Target Event	They tried to ***rebuild*** their shattered nation
Trigger Only	took, rebuilt, losing, sustained, lost, opened, came, bought, died, used
±3 tokens	took, lost, war, losing, reached, brought, moved, began, fled, came
±5 tokens	took, lost, war, collapsed, left, failed, abandoned, came, laid, began

Table 2: Top 10 generated event triggers from the event sampler. As more context is provided, the model generate more specific events related to the provided context.

Input Text	Generation Target
`[BLANK]` that donations be ***made*** to the Crohn's and Colitis Foundation or NYBOT Futures and Options for Kids in memory of Harry. `<E>` requests	In lieu of flowers, the family ***requests***
`[BLANK]` to ***start*** trading an important Nymex product, West Texas intermediate crude oil. `<E>` inspired	Nymex's foray also ***inspired*** ICE
Mr. Robbins played hard and fluidly, `[BLANK]` to ***give*** his solos funk and shape. `<E>` landing	***landing*** heavily on unexpected notes

Table 3: Examples of training instance pairs for the candidate generator. Unlike Kwon et al. (2020), we add the precondition event at the end of the input to help the model utilize the event trigger when generating a precondition.

to obtain temporally related event pairs from the NYT corpus (Sandhaus, 2008). This yields 1.1 million instances and each instance contains one temporal relation (BEFORE/AFTER). Note that all systems are trained using the same pre-training and fine-tuning strategy using both datasets.

4.2 Baselines

Beam Search As a baseline, we use text infilling GPT-2 system (inspired by (Donahue et al., 2020) with a standard beam search decoding strategy. This beam search decoder can provide multiple responses up to its beam size. We expect this simple baseline to contain high-levels of redundancy in its outputs.

Repetition Penalized Sampling (RPS) For a stronger baseline, we use a decoding strategy that can generate diverse preconditions by penalizing previously generated precondition event triggers. This is done by an iterative decoding process applied to the same GPT-2 generation model. Given a target event, the model generates k preconditions in an iterative manner. When the model generates a precondition trigger – after `<pre>` token – a repetition penalty is applied to deter the model from selecting previously generated precondition events. We adopt the penalized sampling from Keskar et al. (2019). Instead of using a list of all generated tokens, we use a list of precondition event triggers that are generated in the previous iterations. Given a list of generated precondition events t, the probability distribution p_i for the next trigger token x_i is defined as:

$$p_i = \frac{\exp\left(x_i / I(i \in t)\right)}{\sum_j \exp(x_j / I(j \in t))}$$

$$I(c) = \lambda \quad \text{if } c \text{ is true else } 1$$

We set $\lambda = 1.2$ as in Keskar et al. (2019). For decoding, we use Nucleus Sampling (Holtzman et al., 2020) which has been claimed to generate a higher quality of text. Finally, we test the RPS model with the post-processor from DiP, to confirm that the major gain of DiP is from the Event Sampler.

4.3 DiP Model

DiP has three modules – Event sampler, Candidate generator, and Precondition re-ranker. We train each module separately.

Event sampler We use the GPT-2 model for the event sampler. The model is trained on the same data instances described in Section 4.1, but instead of using the entire target-precondition pairs, we use target-precondition *event trigger* pairs. We train three event samplers with different levels of context – trigger only, 3 neighboring tokens, and 5 neighboring tokens – to understand how different context affect candidate precondition sampling. As Table 2 shows, adding more context help the model to generate more specific events related to describe situations while the model provides more general events if only a trigger is given.

Candidate generator The GPT-2 model is also used for the candidate generator. For training, as described in 3.2, we add `<E> precondition_event` at the end of input so

Model Diversity Evaluation

Model	Self-BLEU	Self-BLEURT
Beam Search	0.234	-0.450
RPS	0.016	-1.273
RPS+Post-proc.	0.013	-1.280
DiP	0.038	-1.111

Table 4: Diversity evaluation for different models. We evaluate top 10 preconditions for each model. RPS+Post-proc. produces the most diverse outputs followed by RPS and DiP with a small margin.

that the model can learn how to utilize the provided event trigger as a control code. Table 3 shows the training examples for the candidate generator.

Post-processor We train a precondition re-ranker using BERT (Devlin et al., 2019). The F1 score of the classifier is 71.91 with 64.65 of the precision. To remove possibly redundant preconditions using iterative redundancy filtering, we need to compute cosine similarity between the generated preconditions. We take the precondition classifier's [CLS] token representation as the embedding for preconditions. Since the similarity score distributions are different from instance to instance, instead of using a fixed value as the threshold, we set the threshold as $\mu + \sigma$ of each instance (the mean and the standard deviation of pairwise similarity scores). This filters out $\sim$16% of the most similar generated preconditions. For comparison with the baselines, we take top 10 preconditions from remaining outputs.

4.4 Automatic Evaluation Metrics

We use Self-BLEU (Zhu et al., 2018) and Self-BLEURT score to measure the diversity of generated preconditions. Self-BLEU measures how similar a set of sentences is to each other using BLEU score – the average of BLEU scores for the all pairs of sentences in the set. In addition to direct lexical overlap, we also measure semantic overlap using BLEURT (Sellam et al., 2020), which is a BERT-based learned evaluation metric that is trained on human ratings of sentence pairs. We refer to this metric as Self-BLEURT. For both metrics, a lower score implies more diverse preconditions.

4.5 Results

We compare the models on both diversity and quality. For diversity, we use an automatic evaluation, and for quality we used human annotators.

Automatic Diversity Evaluation:

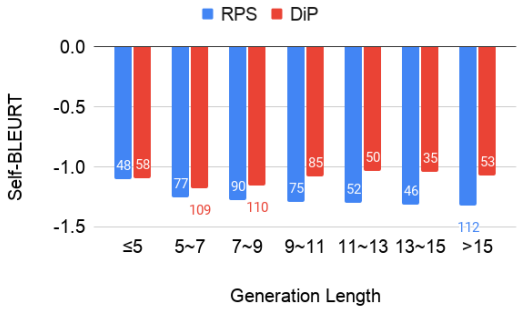


Figure 2: The Self-BLEURT scores across different lengths of generated text. The numbers indicate the number of instances in each bucket.

Quality of Preconditions

Model	Average Score	#Wins
RPS	0.954	30
DiP	**1.101**	**56**

Table 5: The Top 10 generated preconditions for each target event were scored on a 0-2 scale. A model "wins" a target if its average is highest. Using Bootstrapping with $n = 1000$ the 95% conf-interval for the RPS mean is (0.89, 1.01) and DiP is (1.05, 1.15).

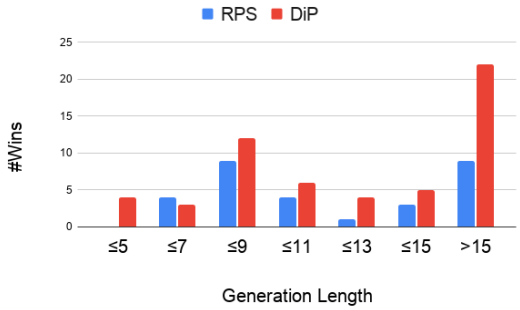


Figure 3: The number of wins across generation lengths. DiP wins more as the generations lengthen.

Table 4 shows the diversity metrics for all methods. We evaluated 5,000 preconditions generated for 500 target events. Comparing RPS+Post-proc to RPS, Post-proc shows little effect, we compare just RPS to DiP in the rest of the evaluations (See Appendix for more details between RPS and RPS+Post-proc).

In both metrics, DiP and RPS generate more diverse output than the beam search decoder. DiP is compatible to RPS in shorter preconditions, and RPS produces more diverse outputs when the generated text gets longer, as shown in Figure 2.

Manual Quality Evaluation:

The automatic evaluation only measures diversity. To see if the models generate legitimate preconditions, we conducted a manual evaluation for quality. We evaluated 960 generated outputs covering 96

distinct target events for both DiP and the RPS baseline. For each instance the annotators were presented with the top ten generated outputs from two systems. For each output the annotators provided a rating on a scale from 0 to 2, where 0 means not a precondition, 1 is a maybe, and 2 is definitely a precondition. We split the 96 instances across 8 different annotators[3].

Table 5 shows the results in terms of two metrics: one is the average score across all 96 instances, and another is the number of "Wins" where a model gets +1 point if the sum of its 10 precondition scores is higher than the other. In both metrics, DiP outperforms RPS. Moreover, as shown in Figure 3, DiP produces better preconditions across most output lengths and is best on longer outputs.

4.6 Analysis

Examples: Table 8 shows the top 5 generations from our main three systems. The beam search's failure on the diversity metric is easy to see with its repetitive output. Most verbs are the same. Both RPS and DiP are notably better in terms of diversity, but RPS introduces lots of irrelevant information that may have artificially increased its diversity score. Long irrelevant phrases are clear to see, and verb synonyms are common. In contrast, DiP generates more succinct and general preconditions, as well as fewer direct synonyms.

Context Specificity: Table 7 shows the diversity scores when different levels of context are provided to the event sampler. Diversity gets slightly worse with *more* context. This aligns with our observation from Table 2 that the event sampler with more context generates more context-specific precondition events, which now appear to be closer to each other semantically. This makes intuitive sense if you view context as closing a model's view of broader options.

Errors: We categorize 4 types of frequently observed errors from DiP. 1) **Rare target events**: some target event triggers are polysemious, and some meanings are rarely used in the newswire. Thus, the event sampler is biased toward generating precondition events based on more frequent usages of triggers. 2) **Conflict with sentence structure**: the candidate generator has no ability to modify the trigger for its syntactic context because it is trained only as copying over the event trigger. 3) **Re-ranking issue**: the re-ranker can induce errors due to its performance – 64.65 precision. 4) **Limited available context**: when the provided context of a target event is too limited, the model often fails to generate preconditions. Table 6 shows examples for each type of error.

[3]These were computer science graduate students

Rare target events	
`[BLANK]` my homework," and exercising like a demon at a nearby gym. She'd ***lost*** weight and viewed herself as training for a fight.	
Event	Generated Precondition
war	I ***war*** with the other women in my school, who had been working together on
died	She ***died*** of cancer in 1998, after her husband was killed by an electrician who had been working with him"
The word ***lost*** is usually used in the context of war, election, or obituary in the newswire. That may lead the event sampler to fail to generate proper precondition event triggers.	

Conflict with sentence structure	
The proposal from Mr. Levy that the Justice Department ***struck*** down, she added, would have violated nondiscrimination provisions of the consent decree and the Civil Rights Act of 1964 by `[BLANK]`.	
Event	Generated Precondition
created	***created*** a new state law requiring that a judge be present on the court
called	***called*** for a "state level ban on political activity" in New York City
The candidate generator is not able to modify the provided triggers according to the context.	

Re-ranking issue	
China's markets nearly ***disintegrated*** in 2005, and one 2003 poll found `[BLANK]`.	
Event	Generated Precondition
became (#4)	that the market ***became*** more popular than in 2000 (#2)
named (>#10)	that the country was ***named*** the world's largest economy (#4)
Generated sentences are not preconditions but ranked high after re-ranking (#4 $\rightarrow$ #2 and >#10 $\rightarrow$ #4).	

Limited available context	
`[BLANK]` to ***hit*** a short forehand and guided it wide.	
Event	Generated Precondition
batted	In the third round, he ***batted*** the ball with his left hand
pitched	In the first inning, the Mets ***pitched*** three consecutive hits
The target context is related to tennis, but preconditions are generated in the context of baseball because the provided context is too limited.	

Table 6: Examples from each type of errors. There are 4 types of frequently observed errors and the first 3 types are caused by each stage in DiP. The last one is due to data instances.

5 Conclusion

Real-world events often have multiple preconditions, but today's datasets do not, including the

Context	Self-BLEU	Self-BLEURT
Trigger only	0.038	-1.111
±3 tokens	0.039	-1.103
±5 tokens	0.040	-1.098

Table 7: Diversity evaluation for different samplers. Precondition candidates are generated from the event samplers given the input with a trigger only, a trigger with neighboring 3 tokens, or a trigger with 5 tokens.

latest PeKo, presenting a challenge for text-driven models. Vanilla generative models have high-levels of redundancy in their outputs and are thus not well suited for diverse generation. This work introduced an event sampler that overcomes the issue of target context specificity to provide diverse guidance to the generator. Coupled with a precondition ranker and similarity filter, this multi-stage generation setup yields more diverse and higher quality preconditions. Further, a new training corpus was not required. More generally, this approach can be seen as an instance of controllable diverse output generation for conditional language models.

Acknowledgements

This material is based on research that is supported in part by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003 and in part by the National Science Foundation under the award IIS #2007290.

References

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96, Online. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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.

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.

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.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Chenyang Huang, Osmar Zaïane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 49–54, New Orleans, Louisiana. Association for Computational Linguistics.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Heeyoung Kwon, Mahnaz Koupaee, Pratyush Singh, Gargi Sawhney, Anmol Shukla, Keerthi Kumar Kallur, Nathanael Chambers, and Niranjan Balasubramanian. 2020. Modeling preconditions in text with a crowd-sourced dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3818–3828, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Target Event: [BLANK] to ***maintain*** below-market rents.		
Beam Search	RPS	DiP
In the last few years, real estate prices have ***risen*** and real estate prices have dropped	Wilhelmina's stock ***dropped*** 8 percent in early 2005 after a lull, but more actively revived the strong dollar, lending to an expected influx of funds	The city has ***reached*** a deal with the tenants
In the last few years, real estate prices have ***risen*** and real estate investment trusts have grown	The City Council's 10-member City Planning Committee ***voted*** 11 to 6 yesterday	The city will ***use*** the money
In the last few years, real estate prices have ***risen*** and real estate prices have dropped, but rents have continued	property values are ***rising***, rising as the vacancy rate is expected	In the 1980s, the city ***moved*** its building to a new site in the East River
In the last few years, real estate prices have ***risen*** and real estate investment trusts have grown in size	In less than a year, such improvements have ***increased*** through acquisitions and capitalization at New York City police stations, legal firms and cruise ships, suggesting that housing can be bought	The City Council ***passed*** a bill on Wednesday that would give the city the authority to build a new building at the site of the old Erez subway station, and
In the last few years, real estate prices have ***risen*** and real estate prices have dropped, and rents have risen in the last few years	Matthew Hallico, president of the General Electric Company in Manhattan, and Robert Chrisin, a sales vice partner at Ira G. Albrecht His comments about the incentive package ***raised*** many concerns about how it works, as well as what shareholders might do	In the 1980s, the city ***began*** a program
Target Event: By about 10 p.m., the proposals ***appeared*** dead for now [BLANK].		
Beam Search	RPS	DiP
Mr. Spitzer ***took*** office	after the judge, Col Richard Kultura of Thailand, ***signaled*** the end of his sentence	after the commission ***filed*** a proposal to provide $ 2 million in new money for the project
Mr. Spitzer ***took*** over	after city Hall ***learned*** it would begin public comment on ways it could add 27,000 new jobs to the island	after the Senate 's Democratic majority has ***taken*** over control of the House
the City Council ***voted*** on them	as they were ***rejected*** by legislative leaders	after the State Legislature ***put*** them on a vote
Mr. Spitzer ***took*** office in January	when Mayor Mark Meehan ***heeded*** all of his smaller complaints	when the State Legislature ***used*** the budget to cut a $ 2 billion tax break
the City Council ***passed*** them to the City Council	after the State Senate ***voted*** yes on key issues	after a suicide bomber ***killed*** a man in an Internet chat room
Target Event: [BLANK] to ***scout*** potential recruits.		
Beam Search	RPS	DiP
The N.F.L. and the N.B.A. have ***taken*** steps	The pending replacement of Carl Crawford has ***enticed*** some intelligence officials and top Qaeda leaders	The police ***took*** over the department 's operations , and they began
The N.F.L. and the N.F.L. have ***taken*** steps	Employees are ***giving*** them the opportunity	The department is ***sending*** a new system
The N.C.A.A. has ***taken*** steps	Most Somalis ***want*** a law that would enable them	The New York State Department of Education ***began*** a program last year
The N.F.L. and the N.B.A. have ***taken*** similar steps	Shortly after Katrina , Post servicemen were ***chasing*** selectors after the storm 's onset	The department has ***sent*** a handful of officers to the police
The N.F.L. and the N.F.L. have ***taken*** similar steps	Ever since college opened in 1983 , he ***shopped*** for school assignments	The N.C.A.A. ***set*** up a task force

Table 8: Top 5 generations from 3 systems. Red cells are invalid preconditions. Greyed out cells are repetitions from previous cells. DiP produces both valid *and* diverse preconditions.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Evan Sandhaus. 2008. The New York Times Annotated Corpus.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Raphael Shu, Hideki Nakayama, and Kyunghyun Cho. 2019. Generating diverse translations with sentence

codes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1823–1827, Florence, Italy. Association for Computational Linguistics.

Avirup Sil, Fei Huang, and Alexander Yates. 2010. Extracting action and event semantics from web text. In *2010 AAAI Fall Symposium Series*.

Richard Socher, Eric Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in neural information processing systems*, 24:801–809.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Nathaniel Weir, João Sedoc, and Benjamin Van Durme. 2020. COD3S: Diverse generation with discrete semantic signatures. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5199–5211, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

A Appendix

A.1 Experimental Details

A.1.1 Data Split

For the dataset for pre-training, we split into train/dev/test with the ratio of 8:1:1. For PeKo dataset, we follow the setting from Kwon et al. (2020).

A.1.2 Infrastructure

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

A.1.3 Parameters

We use Wolf et al. (2020) library for all transformer models. For the beam search baseline and RPS model we use the GPT-2 architecture, which has 124,445,184 trainable parameters. DiP model consists of two GPT-2 models for the event sampler and the candidate generator – 2 $\times$ 124,445,184 – and one BERT model for the re-ranker – 108,313,346 parameters. In total, DiP has 357,203,714 trainable parameters.

Optimizer: We use AdamW (Loshchilov and Hutter, 2019) for the optimizer across all models. For pre-trianing, we fix the learning rate as 1e-3. For fine-tuning, we experiment with [1e-4, 1e-5, 1e-6].

Event sampler: For pre-training, the epochs are set to 100 with the batch size of 128 for the trigger only, 64 for the ± 3 tokens, and 32 for the ± 5 tokens model. For fine-tuning, the epochs are set to 10 with the batch size of 32.

Candidate generator: For pre-training, the epochs are set to 100 with the batch size of 16. For fine-tuning, the epochs are set to 10 with the batch size of 16.

Precondition re-ranker: We use the classifier provided by the authors of Kwon et al. (2020) – https://stonybrooknlp.github.io/PeKo/.

All Models are picked based on the losses from the dev set.

A.2 Comparison between RPS and RPS+Post-processing

Table 9 shows the comparison between RPS and RPS+Post-processing. The effect of Post-processor on RPS system is considered neutral. There are some cases where the generation qualities are improved but also other cases where the qualities are compromised.

A.3 Manual Evaluation

Evaluation Instruction Figure 4 shows the evaluation instruction that we provided to annotators.

Evaluation rating distribution Table 10 shows the distribution of voted ratings by annotators. On average, DiP got higher ratings than RPS and RPS got highest votes in "Not a Precondition."

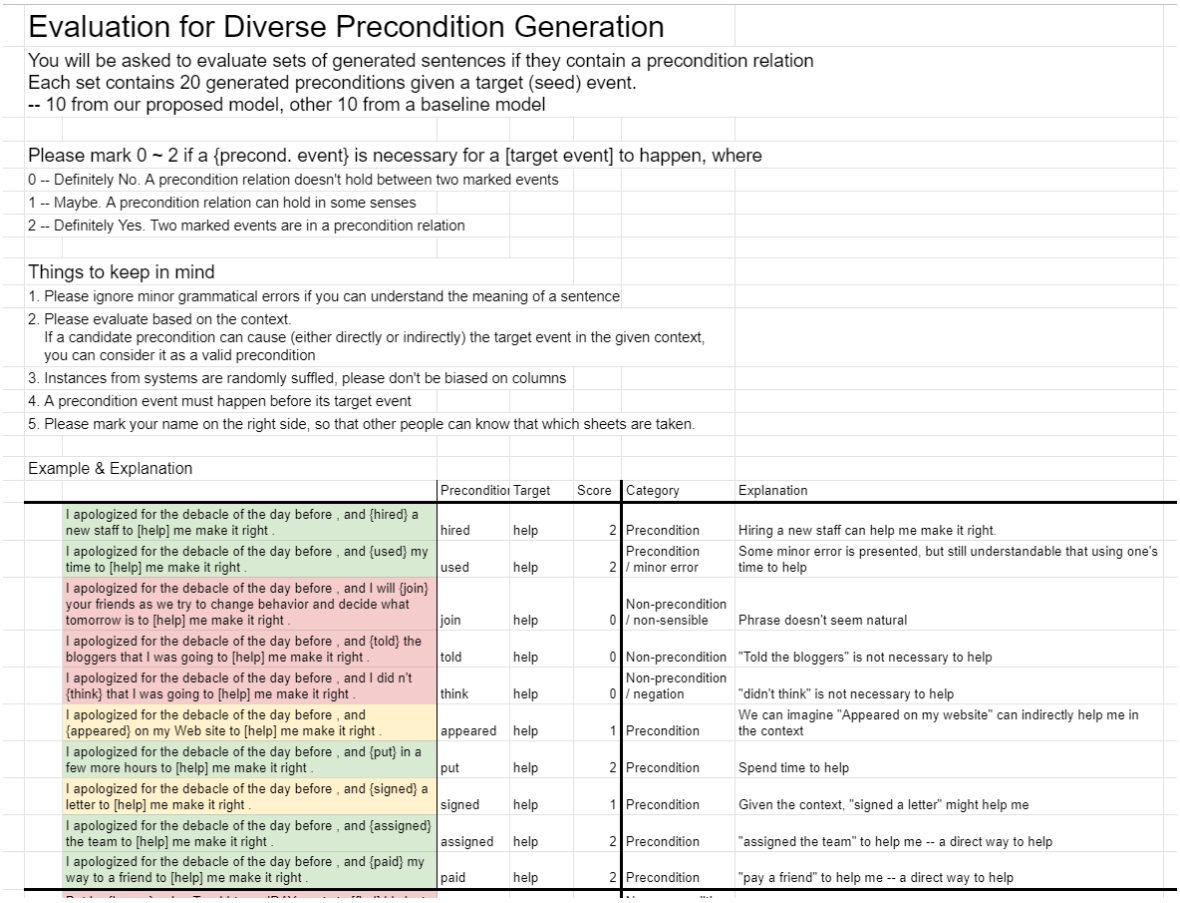
Target Event: [BLANK] to ***maintain*** below-market rents.	
RPS	RPS+Post-processor
Wilhelmina's stock ***dropped*** 8 percent in early 2005 after a lull, but more actively revived the strong dollar, lending to an expected influx of funds	Mr. Berg last year ***signed*** a two-year contract with the firm
The City Council's 10-member City Planning Committee ***voted*** 11 to 6 yesterday	Another landlord is ***assembling*** a plan
property values are ***rising***, rising as the vacancy rate is expected	Ms. Cianci, 32, is ***building*** an intercom tower
In less than a year, such improvements have ***increased*** through acquisitions and capitalization at New York City police stations, legal firms and cruise ships, suggesting that housing can be bought	Her workers ***hit*** a brick floor a few years ago
Matthew Hallico, president of the General Electric Company in Manhattan, and Robert Chrisin, a sales vice partner at Ira G. Albrecht His comments about the incentive package ***raised*** many concerns about how it works, as well as what shareholders might do	Last week, Lloyd Pound, the influential Wall Street analyst, ***gave*** 75 percent shares of his money
Target Event: By about 10 p.m. , the proposals ***appeared*** dead for now [BLANK].	
RPS	RPS+Post-processor
after the judge, Col Richard Kultura of Thailand, ***signaled*** the end of his sentence	after negotiators from both parties ***reconvened*** in Davis Park to talk things out
after city Hall ***learned*** it would begin public comment on ways it could add 27,000 new jobs to the island	after the researchers ***analyzed*** DEMIC data on children and early adults whose ages began at 8 or 15
as they were ***rejected*** by legislative leaders	after Google ***released*** its pie-in-pie template during an extensive public presentation
when Mayor Mark Meehan ***heeded*** all of his smaller complaints	after the Council ***passed*** it on Monday
after the State Senate ***voted*** yes on key issues	after the developer, Trivata Films of New Orleans, ***agreed*** to pay up to $14 million over seven years
Target Event: [BLANK] to ***scout*** potential recruits.	
RPS	RPS+Post-processor
The pending replacement of Carl Crawford has ***enticed*** some intelligence officials and top Qaeda leaders	About the same time, Mr. Booker ***elicited*** state financing for another program that provided some of the funds through the Police Department's National Guard to help workers find mental illness or
Employees are ***giving*** them the opportunity	In 2005 , Mr. SCAD ***sent*** students from Iowa and Ohio to visit Johns Hopkins
Most Somalis ***want*** a law that would enable them	In championing the elite classes last week, public school teachers ***mounted*** an extensive publicity campaign to persuade parents
Shortly after Katrina , Post servicemen were ***chasing*** selectors after the storm 's onset	Joel Packer, a Detroit Pistons and assistant coach with Brigham ***captured*** a larger campus and invited the scouts
Ever since college opened in 1983 , he ***shopped*** for school assignments	As the trend forward moves into next season , larger colleges are ***beginning*** with faculty members from 75 sites on an extensive bioharker scholarship site

Table 9: Top 5 generation examples from RPS and RPS+Post-processor. Green colored events are considered legitimate preconditions and red colored ones are not. A red colored cell indicates invalid precondition text. As the examples show, the effect of Post-processor on RPS system is neutral – in some cases, the generation qualities are improved but compromised in other cases.

Evaluation rating distribution

Model	Not a Precond.	Maybe	Def. a Precond.
RPS	38.6%	27.3%	34.1%
DiP	29.8%	30.3%	39.9%

Table 10: Evaluation rating distribution. On average, DiP got higher ratings than RPS.



Evaluation for Diverse Precondition Generation

You will be asked to evaluate sets of generated sentences if they contain a precondition relation
Each set contains 20 generated preconditions given a target (seed) event.
-- 10 from our proposed model, other 10 from a baseline model

Please mark 0 ~ 2 if a {precond. event} is necessary for a [target event] to happen, where

0 -- Definitely No. A precondition relation doesn't hold between two marked events
1 -- Maybe. A precondition relation can hold in some senses
2 -- Definitely Yes. Two marked events are in a precondition relation

Things to keep in mind

1. Please ignore minor grammatical errors if you can understand the meaning of a sentence
2. Please evaluate based on the context.
   If a candidate precondition can cause (either directly or indirectly) the target event in the given context, you can consider it as a valid precondition
3. Instances from systems are randomly suffled, please don't be biased on columns
4. A precondition event must happen before its target event
5. Please mark your name on the right side, so that other people can know that which sheets are taken.

Example & Explanation

	Precondition	Target	Score	Category	Explanation
I apologized for the debacle of the day before , and {hired} a new staff to [help] me make it right .	hired	help	2	Precondition	Hiring a new staff can help me make it right.
I apologized for the debacle of the day before , and {used} my time to [help] me make it right .	used	help	2	Precondition / minor error	Some minor error is presented, but still understandable that using one's time to help
I apologized for the debacle of the day before , and I will {join} your friends as we try to change behavior and decide what tomorrow is to [help] me make it right .	join	help	0	Non-precondition / non-sensible	Phrase doesn't seem natural
I apologized for the debacle of the day before , and {told} the bloggers that I was going to [help] me make it right .	told	help	0	Non-precondition	"Told the bloggers" is not necessary to help
I apologized for the debacle of the day before , and I did n't {think} that I was going to [help] me make it right .	think	help	0	Non-precondition / negation	"didn't think" is not necessary to help
I apologized for the debacle of the day before , and {appeared} on my Web site to [help] me make it right .	appeared	help	1	Precondition	We can imagine "Appeared on my website" can indirectly help me in the context
I apologized for the debacle of the day before , and {put} in a few more hours to [help] me make it right .	put	help	2	Precondition	Spend time to help
I apologized for the debacle of the day before , and {signed} a letter to [help] me make it right .	signed	help	1	Precondition	Given the context, "signed a letter" might help me
I apologized for the debacle of the day before , and {assigned} the team to [help] me make it right .	assigned	help	2	Precondition	"assigned the team" to help me -- a direct way to help
I apologized for the debacle of the day before , and {paid} my way to a friend to [help] me make it right .	paid	help	2	Precondition	"pay a friend" to help me -- a direct way to help

Figure 4: Manual evaluation instruction