What do Large Language Models Learn about Scripts?

Abhilasha Sancheti
University of Maryland, College Park
Adobe Research
sancheti@{umd.edu,adobe.com}

Rachel Rudinger
University of Maryland, College Park
rudinger@umd.edu

Abstract

Script Knowledge (Schank and Abelson, 1975) has long been recognized as crucial for language understanding as it can help in filling in unstated information in a narrative. However, such knowledge is expensive to produce manually and difficult to induce from text due to reporting bias (Gordon and Van Durme, 2013). In this work, we are interested in the scientific question of whether explicit script knowledge is present and accessible through pre-trained generative language models (LMs). To this end, we introduce the task of generating full event sequence descriptions (ESDs) given a scenario as a natural language prompt. Through zero-shot probing, we find that generative LMs produce poor ESDs with mostly omitted, irrelevant, repeated or misordered events. To address this, we propose a pipeline-based script induction framework (SIF) which can generate good quality ESDs for unseen scenarios (e.g., bake a cake). SIF is a two-staged framework that fine-tunes LM on a small set of ESD examples in the first stage. In the second stage, ESD generated for an unseen scenario is post-processed using RoBERTa-based models to filter irrelevant events, remove repetitions, and reorder the temporally misordered events. Through automatic and manual evaluations, we demonstrate that SIF yields substantial improvements (1-3 BLEU points) over a fine-tuned LM. However, manual analysis shows that there is great room for improvement, offering a new research direction for inducing script knowledge[1].

1 Introduction

Scripts are structured commonsense knowledge in the form of event sequences that characterize commonplace scenarios, such as, eating at a restaurant (Schank and Abelson, 1975). Scripts are fundamental pieces of commonsense knowledge that humans share and assume to be tacitly understood by each other. When someone says "I went to a restaurant for lunch", our script knowledge allows us to infer that a waiter would have taken the order, the speaker would have eaten the lunch, payed for it, and tipped the waiter, even if these events are not explicitly mentioned. Knowledge of scripts, whether implicit or explicit, has been recognized as important for language understanding tasks (Miikkulainen, 1995; Mueller, 2004).

Scenario Baking a cake

Event Sequence Description (ESD) 1. gather ingredients 2. mix cake mix, eggs and water in bowl 3. pour into pan 4. turn on oven 5. put in oven and bake at specified temperature 6.remove cake from oven to cool 7. turn off oven 8. mix frosting 9. frost cake 10. serve cake 11. refrigerate any leftovers.

Input to LM $\underbrace{\text{Here is a sequence of events that happen when you}}_{\textbf{Natural Language Prompt}}$ $\underbrace{\text{bake a cake}}_{\textbf{Scenario}}$:

Figure 1: Sample event sequence description (ESD) from Wanzare et al. (2016) for BAKING A CAKE scenario. We use natural language prompts (Table 2) to *generate completely ordered* ESDs for evaluating extent of script knowledge accessible through LMs.

Earlier efforts to automatically induce scripts from text on a large scale include Chambers and Jurafsky (2008) who treat the problem of script induction as one of learning narrative chains using textual co-occurrence statistics. However, reporting bias (Gordon and Van Durme, 2013) remains an obstacle for script induction as many events are not mentioned explicitly in text, relying on the reader's ability to infer missing script-related events. Moreover, manual creation of such knowledge resources is challenging due to the wide coverage and complexity of relevant scenario knowledge. Although crowdsourced efforts (Singh et al., 2002; Regneri et al., 2010; Modi et al., 2017; Wanzare et al., 2016; Ostermann et al., 2018, 2019) address these issues and acquire script knowledge in the form of ESDs, the collected datasets are small, domain-specific, and crowdsourcing is not scalable.

With the success of pre-trained language mod-

[1]Code and dataset are available at https://github.com/abhilashasancheti/script-generation

Proceedings of the 11th Joint Conference on Lexical and Computational Semantics, pages 1 - 11
July 14-15, 2022 ©2022 Association for Computational Linguistics

els (henceforth, PLMs) (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2019) in various natural language understanding tasks, we are interested in investigating the extent and accessibility of explicit script knowledge present in PLMs. In this work, unlike cloze-based script evaluations (Chambers and Jurafsky, 2008; Mostafazadeh et al., 2016) which LMs are uniquely optimized for (Rudinger et al., 2015), we evaluate PLMs on the ability to *fully generate* event sequence descriptions (ESDs) (Regneri et al., 2010) in free-form natural language (Figure 1). This is a challenging task as scripts are complex structures with varied granularity of describing a scenario (e.g., starting from going to grocery store to buy ingredients or starting with finding a recipe for BAKING A CAKE scenario), and the requirement to produce all the scenario-*relevant* events in the correct temporal *order*.

To this end, we first probe LMs via carefully crafted prompts to analyze the quality of ESDs generated in a zero-shot setting (§3) and find that the generated ESDs are of poor quality with many scenario-irrelevant, repeated, temporally misordered, and missing events. To address this we propose a, LM-agnostic, pipeline-based script induction framework (§4), `SIF`, which can generate good quality ESDs for novel scenarios that LM has not seen during the training phase of the framework. `SIF` is a two-staged framework with fine-tuning LM on a small set of ESDs as the first stage followed by a three-stepped post-processing stage which corrects the ESDs generated from a fine-tuned LM for irrelevant, repeated, and temporally misordered events. This work makes the following **contributions**:

- We present an analysis of the extent of script knowledge accessible through LMs using probing techniques, in a zero-shot setting, via the task of generating full ESDs from natural language prompts.
- We propose script induction framework that can generate ESDs for held-out and novel scenarios drawn from a different distribution.
- We present automatic and manual evaluation of the generated ESDs, establishing the viability of our framework and paving way for future research in this direction.

2 Related Work

Narrative Chain Induction There has been a growing body of research into statistical script learning systems which can automatically infer implicit events from text. Seminal work by (Chambers and Jurafsky, 2008, 2009) describe a number of simple event co-occurrence based systems that infer (verb, dependency) pairs (known as narrative events) with partial-ordering related to one or multiple participants (Pichotta and Mooney, 2014) in discourse (known as narrative chains). As statistical co-occurrences cannot capture long-range dependencies between events, Pichotta and Mooney (2016a) represent events using LSTM leading to improved narrative cloze task performance. However, much of the information about events is usually left implicit in text. Moreover, narrative events are highly abstracted (Ostermann, 2020) and cloze task is insufficient to evaluate script knowledge (Chambers, 2017). Therefore efforts have been made to acquire crowdsourced ESDs (Singh et al., 2002; Regneri et al., 2010; Modi et al., 2017; Wanzare et al., 2016; Ostermann et al., 2018, 2019) and to learn similar events in a scenario using unsupervised (Regneri et al., 2010) and semi-supervised (Wanzare et al., 2017a) approaches.

Temporal Ordering and Relevance Previous works (Modi and Titov, 2014; Wanzare et al., 2017b; Lyu et al., 2020) have investigated induction or prediction of temporal ordering of prototypical events. Others have predicted next (Pichotta and Mooney, 2016b) or related (Lyu et al., 2020) events in natural language form. Zhou et al. (2019) acquire commonsense procedural knowledge directly from natural language source, like wikiHow, by learning representations for scenarios and events which are predictive of both relevance of event to the scenario and temporal ordering. Zhang et al. (2020) propose a non-learning based approach to predict fixed-length events given an unseen scenario and related scenarios with their events. A recent work (Sakaguchi et al., 2021) generates partially-ordered scripts using PLMs by predicting events and edges for partial-order while we are interested in completely ordered event descriptions. Lyu et al. (2021) propose the task of goal oriented script construction for multilingual wikiHow dataset and propose generation and retrieval-based approaches. However, their generation-based approach using LM only involves fine-tuning. We focus on different LMs to evaluate them on the task

Prompt Beginnings
here is a sequence of events that happen while baking a cake:
these are the things that happen when you bake a cake:
describe baking a cake in small sequences of short sentences:
here is an ordered sequence of events that occur when you bake a cake:

⊕

Continuations
None
1.
1. get a cake mix
1. get a cake mix 2. gather together other ingredients

Figure 2: Different prompt formulations for BAKING A CAKE scenario for probing. 16 prompts are created by combining a prompt beginning with a continuation.

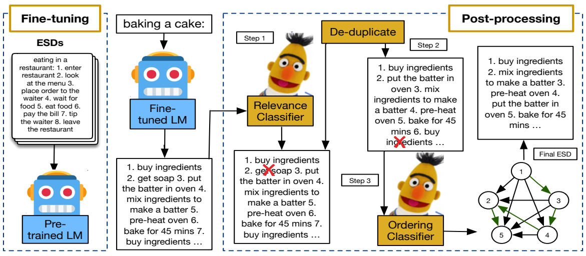

Figure 3: SIF: Pre-trained LM is fine-tuned on De-Script (Wanzare et al., 2016). Generated scripts are then post-processed with RoBERTa-based classifiers to correct for event relevance (Step 1), repetition (Step 2), and temporal ordering (Step 3).

of generating scripts both in zero-shot and fine-tuning settings. Our proposed framework is shown to outperform the fine-tuning approach.

Knowledge-acquisition from PLMs With the success of PLMs (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2019) in various natural language understanding tasks, a number of works investigate how commonsense knowledge is captured in these models (Feldman et al., 2019; Petroni et al., 2020; Weir et al., 2020; Shwartz et al., 2020). Successful efforts have been made to induce relational (Bouraoui et al., 2020), numerical (Lin et al., 2020), temporal (Zhou et al., 2020) and commonsense knowledge in PLMs using fine-tuning.

Unlike prior works, we focus on investigating the extent and accessibility of explicit script knowledge from PLMs via probing techniques and inducing such knowledge in them using a pipeline-based framework to generate full ESDs for novel scenarios in free-form natural language.

3 Probing for Script Knowledge

We design a zero-shot probing experiment to evaluate PLMs' ability to generate ESDs by carefully selecting natural language prompts, which LMs are known to be sensitive to (Bouraoui et al., 2020). We experiment with 16 manually crafted prompts[2] (Table 2) with different phrasing and levels of conditioning to enquire large versions of GPT2, BART, and T5 for script knowledge. The intuition behind these prompts is similar to asking questions (prompts) to a knowledge source in various ways to get the required answer (ESD for a scenario).

BART and T5 were not able to output anything except the input prompt or start, end, and pad tokens and hence we only present qualitative outputs from GPT2, when probed with various prompts for BAKING A CAKE scenario, in Table 2. We observe that the quality of generated ESDs vary for different prompts. Although GPT2 is able to generate some scenario-relevant events with just the prompt beginnings and no continuations (e.g., 1 and 2 in Table 2), the ESDs are incomplete with many auxiliary details, and incorrect event ordering (e.g., '3. The cake is served at the table' before '6. The cake is transferred to the oven.' in 2). It sometimes outputs (e.g., 4) narrations rather than procedural descriptions. As generation from scratch is an open-ended task, we use a prompt with a number to guide GPT2 to generate a procedural script. Although 4 and 5 are more procedural, the events are still at a coarse-grained level with most of the intermediate events missing. To further guide the generation towards a fine-grained level, we condition the generation on a few events (manually curated by authors looking at sample ESDs) along with the prompt beginning. This helps us in examining whether GPT2 has temporal knowledge about the events related to a scenario. Conditioning on the events results in a better quality ESD (e.g., 6, 7, 8). However, there is a repetition of events ('let it cool for another 10 minutes' in 6, 'add in your flour and mix by hand' in 7) in addition to wrong event ordering, irrelevant (e.g., 'is it hot?' in 8) and missing events. As GPT2 produces poor quality ESDs in this zero-shot setting with BART and T5 not even being able to output any events, we propose a script induction framework detailed in the following section.

4 SIF: Script Induction Framework

In this section, we provide details on our pipeline-based script induction framework, SIF (Figure 3), which addresses the limitations of zero-shot ESD generation. SIF is a two-staged framework which fine-tunes LM on a small set of ESDs in the first

[2]We also experiment with capitalized prompts but did not find significant change in the quality of generations.

1	**these are the things that happen when you bake a cake:** 3 tablespoons (30ml) of unsalted butter, at room temperature 2 tablespoons (30ml) of granulated sugar 2 tablespoons (30ml) of brown sugar, sifted 1 large egg, at room temperature 1 teaspoon vanilla extract 3 large egg yolks 1 cup (350ml) buttermilk To the bowl of an electric mixer fitted with a paddle attachment, add the butter, sugar, brown sugar, egg yolks, and vanilla extract and beat until light and fluffy. Add the buttermilk, then continue beating until fully incorporated. Sift the flour, baking powder, and salt into the bowl of the mixer and beat
2	**here is an ordered sequence of events that occur when you bake a cake:** 1. The cake is baked. 2. The cake is cooled. 3. The cake is served at the table. 4. The dessert is placed on the table. 5. The cake is placed in a baking pan. 6. The cake is transferred to the oven. 7. The cake is put to the middle of the oven and baked. 8. The cake is left in the oven until it is done. 9. The cake is removed from the oven and placed on a plate. 10. The cake is transferred to a cake rack. 11. The cake is placed on a cake plate.
3	**describe baking a cake in small sequences of short sentences:** "I want my cake to be soft and moist." (And, you know, don't do this: "If I use a pastry bag, I can get it back to me in less than two minutes.") You may be looking for the most simple recipe for cake, but baking requires skill and a lot of practice. You don't need to learn how to make a cake recipe. You don't even need to know the difference between the cake and the sponge. All you need is a great recipe and a couple of ovens and an oven thermometer. 2. Use a mixer. Mixers are really, really good. (A great idea is to invest in one of those mini, little, portable ones.
4	**here is a sequence of events that happen while baking a cake: 1.** The cake goes into the oven. 2. The cake cools down in the oven, because the water vapor from the oven condenses onto the top of the cake. 3. The cake is cooled and the frosting set (the sugar crystals are not allowed to fall off). 4. The frosting is set (a little more water vapor condenses onto the top of the frosting). 5. The frosting is set (more water vapor condenses onto the top of the frosting). 6. The cake cools down (the water vapor cools off). 7. The cake comes out of the oven and it is ready to eat. How is this cake not a cake? The first step is that the cake is a cake.
5	**here is an ordered sequence of events that occur when you bake a cake: 1.** You have a cake. 2. You have a batch of flour (or other ingredient, such as water, sugar or corn syrup) 3. You have your flour prepared 4. You have your flour mixture 5. You have your flour mixture pre-floured 6. You have your batter prepared 7. You have your batter made 8. You have your batter placed on a baking sheet 9. You have your batter removed from the oven 10. You have your cake cooled in the tray (you must place your cake in the tray before you take it out of the oven) 11. You take your cake out of the tray and remove it from the oven 12. You eat your cake, or leave it out to cool.
6	**here is a sequence of events that happen while baking a cake: 1. get a cake mix** 2. pour the cake mix into the pan 3. let it bake for 10 minutes 4. get out the oven lid and turn it on 5. let it bake for another 10 minutes 6. get out the oven lid and flip it on 7. let it bake for another 10 minutes 8. get out the oven lid and switch it back on 9. let it bake for another 10 minutes 10. get out the oven lid, turn it back on, and let it bake again 11. get out the oven lid and get out the knife 12. bake the cake again, and then let it cool on a rack 13. use the cake mix to make cupcakes 14. cut the cake, and then use the cupcake for serving!
7	**these are the things that happen when you bake a cake: 1. get a cake mix 2. gather together other ingredients** that you'll need for your cake (coconut oil, powdered sugar, eggs, etc.) 3. preheat oven to 350 4. pour on flour and mix by hand 5. pour in egg and mix by hand 6. add in powdered sugar and mix by hand 7. add in coconut oil and mix by hand 8. let it all sit for a minute 9. add in your flour and mix by hand 10. add in your sugar and mix by hand 11. add in your flour and mix by hand 12. add in your flour and mix by hand 13. add in your flour and mix by hand 14. add in your flour and mix by hand 15. add in your egg and mix by hand 16. add in your salt
8	**describe baking a cake in small sequences of short sentences: 1. get a cake mix 2. gather together other ingredients** 3. start to bake 4. bake a cake 5. take it to the oven 6. the cake is done 7. is it done? 8. is it still hot? 9. what is that crust? 10. what does the cake look like? If we think of a sequence of a few hundred or so words, we can take the basic idea of this example and say that we could write an equivalent, yet shorter, sentence: 1. Get a cake mix 2. Get together the ingredients 3. Bake a cake 4. Take it to the oven 5. The cake is done 6. Is it done? 7. is it hot? 8. what is that crust? 9. what does the cake look like

Table 1: Scripts generated from GPT2-L for BAKING A CAKE scenario with **bold-faced** prompts.

SEQUENCE	here is a sequence of events that happen while baking a cake: 1. e_1 2. e_2
EXPECT	these are the things that happen when you bake a cake: 1. e_1 2. e_2
ORDERED	here is an ordered sequence of events that occur when you bake a cake: 1. e_1 2. e_2
DESCRIBE	describe baking a cake in small sequences of short sentences: 1. e_1 2. e_2
DIRECT	baking a cake: 1. e_1 2. e_2
TOKENS	⟨SCR⟩ baking a cake ⟨ESCR⟩: 1. e_1 2. e_2
ALLTOKENS	⟨SCR⟩ baking a cake ⟨ESCR⟩: ⟨BEVENT⟩ e_1 ⟨EEVENT⟩ ⟨BEVENT⟩ e_2 ⟨EEVENT⟩

Table 2: Different prompt formulations for BAKING A CAKE scenario with two events (e_1 and e_2).

stage. In the second stage, ESDs generated using the fine-tuned LM are passed through a sequence of RoBERTa-based classifiers (Liu et al., 2019) to identify relevant events, remove redundant events, and predict pair-wise temporal ordering between the events. These pair-wise orderings are then used to create a full event ordering using topological sorting on a directed graph created from the predicted orderings.

4.1 Stage I: Fine-tuning PLMs

PLMs fine-tuned on commonsense datasets like ATOMIC (Sap et al., 2019) can generalize beyond the scenarios observed during fine-tuning (Bosselut et al., 2019). Hence, we investigate the learning capability of LMs when a small number of script examples are available. We fine-tune LMs on ESDs using different natural language and pseudo-natural language prompt formulations for encoding ESDs (Table 2) to study the effect of prompt formulations on this task as observed during the probing experiments. We fine-tune LMs using negative log-likelihood objective.

4.2 Stage II: Post-processing Generated ESDs

We sample ESDs for an unseen scenario using the fine-tuned LMs and employ a 3-step post-processing method to correct them for relevance, repetitions, and ordering.

4.2.1 Step 1: Irrelevant Events Removal

The first post-processing step is to remove non-scenario-relevant events from an ESD. An event is not relevant for a scenario if it is not a part of the scenario (e.g., 'tipping a waiter' is not a part of BAKING A CAKE scenario). For irrelevant events removal, we first need to identify irrelevant events for a scenario. We pose this identification problem as a binary classification task to predict if a given event belongs to a given scenario. For training purpose, a positive example is constructed by pairing a scenario with an event belonging to that scenario; negative samples are drawn from another scenario in the training data. Using this data, we train a RoBERTa-L-based (Liu et al., 2019) classifier and remove those events from an ESD which are predicted as irrelevant by this classifier.

4.2.2 Step 2: Event De-duplication

The second step involves the identification and removal of repeated events. Repetition of events can occur by an exact copy of an event or by a paraphrase of an event (e.g., '6. You have your batter prepared' and '7. You have your batter made' in 5 of Table 1). To identify such de-duplications, we train a RoBERTa-L-based paraphrase identification system using MRPC (Dolan and Brockett, 2005) dataset. However, we observe many false-positives (e.g., 'open a faucet' and 'close a faucet' were identified as paraphrases) with this system. Since false-positives can lead to unnecessary removal of events, we employ a conservative approach of only identifying repeated events. We find edit distance between each pair of events in an ESD and remove multiple occurrences of an event from the ESD, as identified by the edit distance score of 0.

4.2.3 Step 3: Temporal Order Correction

The final step is to correct the order of events in an ESD. We correct the ESDs for ordering by first obtaining pair-wise event orderings and then using a graph-based approach to get the final overall ordering. We pose the problem of pair-wise event ordering as a binary classification task to predict if the order of a given pair of events is correct with respect to the given scenario. We sample event pairs from gold ESDs to construct positive (sequence order) and negative (reverse order) examples to train a RoBERTa-L-based classifier. Topological sort is then used to get the final ESD for a scenario from the ordering predictions for all the $\binom{N}{2}$ pairs of events in an ESD. We construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of events in a scenario with events as nodes ($\mathcal{V}$) of the graph and a directed edge from node $v_1 \in \mathcal{V}$ to $v_2 \in \mathcal{V}$ if event represented by v_2 is predicted to occur after the event represented by v_1. We keep the original ordering of events in case the constructed graph is cyclic[3] due to incorrect predictions from the classifiers.

4.3 Implementation Details

4.3.1 Dataset pre-proccessing

We fine-tune LMs on ESDs from DeScript (Wanzare et al., 2016) dataset which consists of 100 ESDs each for 40 scenarios, collected via crowdsourcing. The scenarios are randomly partitioned into 8 folds with each fold consisting of ESDs from 5 scenarios to perform 8-fold cross-validation of `SIF` for each of the prompt formulation. We lowercase and enclose each ESD within a begin of scenario $\langle$`BOS`$\rangle$ and an end of scenario $\langle$`EOS`$\rangle$ token for fine-tuning. The input to the relevance classifier is: `scenario` $\langle$`/s`$\rangle$ e and to the temporal classifier is `scenario name` $\langle$`/s`$\rangle$ e_1 $\langle$`/s`$\rangle$ e_2, where $\langle$`/s`$\rangle$ is a separator token and e, e_1, e_2 are events.

4.3.2 Training details

We use huggingface's transformers library (Wolf et al., 2020) to fine-tune LMs on each of the 7 prompt formulations, leading to 7 variations for each LM, for 1 epoch with a batch size of 1, gradient accumulation per 16 steps, and block size of 150. At inference time, 5 ESDs are sampled for each of the given scenarios with top 50 probable tokens, nucleus sampling (Holtzman et al., 2019) probability of 0.9, and maximum length set at 150. We use RoBERTa-L architecture from the transformers library for relevance and temporal order classifiers. Relevance (Temporal) classifier is trained for 10 (5) epochs with average validation accuracy of 84.50% (83.87%) across the folds. The model with the best accuracy on the valid split is used in the post-processing stage. We use python's *editdistance* library to compute edit distance for the de-duplication step. We use Adam optimizer with an initial learning rate of $2e^{-5}$, warm-up steps set at 0.06 of total steps, batch size of 16, and maximum input length 150 for both the classifiers. All the models are trained and tested on NVIDIA Tesla V100 SXM2 16GB GPU machine.

5 Evaluation

We use `SIF` to induce script knowledge in GPT2, BART, and T5, and evaluate full ESDs generated for a given unseen scenario using **BLEU** metric (Papineni et al., 2002), following Pichotta and Mooney (2016b) who use BLEU to score individual LM-generated events. As BLEU is a precision-based metric, we measure n-gram overlap of the sampled ESDs against multiple gold-reference ESDs[4] for each scenario in the test fold.

Additionally, for deeper analysis of the generated ESDs, two of the authors evaluate a subset of the generated ESDs (blinded to the identity of the models and prompt variants) on three levels –

[3] $66\pm15\%$ (averaged across all the input variants and folds) of the complete graphs are acyclic for GPT2.

[4] We use NLTK python library to calculate BLEU score with add-1 smoothing function and n-grams upto $n = 4$. We convert the outputs of different variants & gold references into numbered form, 1. e_1 2. $e_2 \ldots n.\ e_n$ for a fair comparison.

Models	TOKENS	EXPECT	SEQUENCE	ALLTOKENS	DESCRIBE	DIRECT	ORDERED
(1) Zero-shot	03.1 (5.2)	03.6 (5.5)	05.4 (2.8)	03.1 (5.2)	03.2 (3.6)	03.9 (5.1)	06.2 (6.6)
(2) GPT2-L$_{\text{SCRATCH}}$	17.2 (3.1)	19.3 (3.7)	16.8 (2.9)	18.6 (4.5)	17.6 (2.6)	14.4 (3.9)	17.7 (3.2)
(3) BART-FT	15.5 (6.0)	20.8 (3.5)	19.6 (3.5)	19.7 (9.2)	19.2 (3.9)	18.0 (6.6)	11.7 (4.8)
(4) GPT2-FT	30.7 (5.1)	31.3 (5.5)	32.4 (6.3)	30.7 (6.6)	32.3 (5.9)	31.4 (5.8)	31.0 (4.8)
(5) BART-**SIF**	16.8 (5.1)	21.1 (4.2)	19.9 (3.7)	20.5 (11.1)	20.0 (3.8)	19.6 (7.2)	13.7 (5.0)
(6) GPT2-**SIF**	**33.6** (5.4)	**33.9** (5.6)	**35.2** (6.9)	**32.5** (6.9)	**34.2** (5.3)	**33.6** (5.7)	**33.2** (5.5)

Table 3: Automatic evaluation results: Mean BLEU scores (and std. dev.) over 8 folds of held-out scenarios are reported. (1) is pre-trained GPT2 (no fine-tuning or post-processing); (2) is randomly initialized GPT2 with fine-tuning; (3-4) are fine-tuned BART and GPT2; (5-6) are **SIF** applied to BART and GPT2.

Models	TOKENS	EXPECT	SEQUENCE	ALLTOKENS	DESCRIBE	DIRECT	ORDERED
(1) GPT2-FT	30.7 (5.1)	31.3 (5.5)	32.4 (6.3)	30.7 (6.6)	32.3 (5.9)	31.4 (5.8)	31.0 (4.8)
(2) GPT2-FT+Relevance (R)	33.1 (5.1)	33.1 (4.9)	34.7 (6.9)	31.9 (6.7)	33.7 (5.0)	32.6 (5.8)	33.2 (5.2)
(3) GPT2-FT+R+De-duplicate (D)	33.5 (5.2)	33.6 (5.2)	35.1 (6.9)	32.1 (6.7)	**34.3** (5.0)	32.9 (5.7)	**33.6** (5.5)
(4) GPT2-FT+R+D+Reorder (GPT2-SIF)	**33.6** (5.4)	**33.9** (5.6)	**35.2** (6.9)	**32.5** (6.9)	34.2 (5.3)	**33.6** (5.7)	33.2 (5.5)

Table 4: Ablation analysis of each step in the proposed pipeline for GPT2. Mean BLEU scores (and std. dev.) over 8 folds of held-out scenarios are reported. (1) fine-tuned GPT2; (2-4) are fine-tuned GPT2 with successive post-processing steps.

individual events (**Relevance (R)**), pairwise events (**Order (O)**), and the overall sequence (**Missing (M)**). **R** measures the % of generated events relevant to a scenario; **O** measures the % of consecutive event pairs correctly ordered given a scenario; and **M** measures the degree to which important events are missing on a 4-point Likert scale defined as (1) no or almost no missing events, (2) some insignificant missing events, (3) notable missing events, and (4) severe missing events. As scripts are complex structures and require an understanding of scenarios, we chose not to resort to a crowdsourcing platform for manual analysis. We manually analyze the outputs to evaluate SIF as well as perform an error analysis to identify opportunities for future research directions.

We evaluate our framework on scenarios in each of the eight folds as well as novel scenarios from Regneri et al. (2010), and day-to-day activities. As we do not have access to gold-reference ESDs for the novel scenarios, we demonstrate our framework's performance only using manual evaluation.

6 Results and Analysis

6.1 Automatic Evaluation

We present the automatic evaluation results on held-out scenarios in Table 3. As baselines, we report scores from non-fine-tuned GPT2-L (Zero-shot), a randomly-initialized GPT2-L$_{\text{SCRATCH}}$ model fine-tuned on DeScript ESDs, and BART-FT and GPT2-FT models which are fine-tuned in the first stage of SIF. We do not report any results for T5 as it was even struggling to learn the input ESD formulations during fine-tuning. We explain the findings from automatic evaluation below.

SIF significantly outperforms fine-tuning baselines. Both GPT2-SIF and BART-SIF have higher BLEU scores as compared to their corresponding fine-tuned (GPT2-FT and BART-FT) models across all the prompt variants. This clearly reflects the advantage of the post-processing stage in SIF framework. Improvement across different LMs reinforces the LM-agnostic nature of our framework. Variation in the extent of induction across prompt variants indicates the sensitivity of LMs to prompt formulations.

Script knowledge is best accessible through GPT2 than other LMs. As previously mentioned in probing experiments, BART and T5 were not able to output anything useful in the zero-shot setting while GPT2 could produce ESDs, although erroneous and of poor quality. We observe same trends even after fine-tuning these LMs or using SIF to induce script knowledge in these LMs. Interestingly, a randomly initialized and fine-tuned GPT2 (GPT2-L$_{\text{SCRATCH}}$) is able to perform comparable to a pre-trained BART fined-tuned using DeScript (BART-FT), and even better for TOKENS and ORDERED variants. Overall, GPT2 is found to be better than BART in terms of the presence and accessibility of script knowledge through them. One possible explanation for this is that GPT2 is a generative language model while BART and T5 are encoder-decoder-based language models making it challenging to encode complete script knowledge within a scenario name.

Performance across LMs is sensitive to prompt formulation and scenario. We consistently ob-

Variants	BLEU↑	Manual Evaluation		
		R↑	O↑	M↓
TOKENS	19.2/**22.8**	77.2/**84.3**	72.3/**89.3**	**2.6**/**2.6**
EXPECT	22.8/**26.0**	81.9/**82.7**	74.5/**86.5**	**3.0**/**3.0**
SEQUENCE	27.8/**33.4**	73.3/**83.2**	74.0/**87.5**	**2.5**/**2.5**
ALLTOKENS	33.5/**35.0**	83.5/**85.7**	82.7/**89.5**	**2.6**/**2.6**
DESCRIBE	27.1/**28.6**	80.7/**86.3**	83.9/**85.9**	**2.8**/**2.8**
DIRECT	30.9/**34.1**	81.2/**84.2**	**88.5**/86.1	**2.6**/**2.6**
ORDERED	**31.9**/31.5	84.9/**86.2**	78.6/**86.8**	**2.6**/**2.6**

Table 5: Manual and BLEU scores on fine-tuned GPT2 (GPT2-FT) SIF applied to GPT2 (FT/SIF), computed for a stratified sample of outputs (one ESD per scenario across two folds). Mean scores across two annotators are reported. Annotator agreement is measured with Cohen's Kappa (Cohen, 1960) (κ=0.61 for **O**, κ=0.56 for **R**) and Spearman's correlation (ρ=0.64 for **M**). Underline and **bold** denotes the best across variants, and between FT and Ours, respectively. O scores are calculated only when both the events are marked as relevant by the two annotators.

Scenario	R↑	O↑	M↓
Order fastfood online	81.5	84.6	2.6
Cook in a microwave	89.5	92.0	2.4
Answer telephone	65.5	91.7	2.0
Buy from vending machine	77.1	81.3	3.4
Tie shoe laces	65.8	66.7	3.6
Brush teeth	75.9	71.4	2.6
Make ginger paste	41.5	85.7	3.4
Attend a wedding	71.9	100.0	2.4
Wash a car	85.7	90.0	3.0
Take out trash	88.5	92.3	2.2
Take a taxi	85.7	76.2	2.0
Surf the internet	73.3	62.5	2.8
Watch television	77.4	73.7	3.0
Go to a club to dance	100.0	93.5	1.4
Average Score	77.1	83.0	2.6

Table 6: Manual evaluation of ESDs for novel scenarios. Averaged across 5 sampled ESDs per scenario generated using the best performing SEQUENCE variant of GPT2-SIF as per automatic measure.

serve variation in performance across prompt variants. Moreover, this variation is also observed across LMs. For BART, EXPECT outperforms other prompt variants while SEQUENCE performs the best for GPT2. High variance across folds also shows that different prompts perform differently depending upon a scenario. This indicates the sensitivity of LMs to prompt formulations and thus justifies our experiments with different prompt formulations to study the extent of script knowledge that can be accessed through PLMs.

6.2 Ablation Analysis of SIF

We next analyze the contribution of each the stage of SIF and each step of stage II leading to improvement in the performance via an ablation study, on GPT2, in Table 4. As expected stage I contributes maximum to the performance boost. and There is a consistent improvement in BLEU after each of the post-processing steps except in the case of DESCRIBE and ORDERED wherein, reordering leads to a slight decrease in BLEU as the trained classifiers are not perfectly accurate. We present qualitative outputs when SIF is used to induce script knowledge in GPT2 in Table 7.

6.3 Manual Evaluation and Error Analysis

We manually evaluate a total of 140 ESDs (for M) comprising 652 individual events (for R) and 582 consecutive pair of events (for O) generated from GPT2-FT and GPT2 SIF across all the prompt variants (Table 5). BLEU scores are also reported for the same set of ESDs to study the correlation between manual and automatic metrics. We find that outputs from SIF have higher BLEU, R, and O scores than FT across all prompt variants (except O for DIRECT and BLEU for ORDERED). M scores do not change, which shows that significantly important events are not dropped during the irrelevant events removal step. Different prompts perform well in different aspects. DESCRIBE generates most relevant events, ALLTOKENS has the best temporal ordering knowledge, and SEQUENCE leads to least severe missing events after Stage II of SIF. To our surprise, we find no statistically significant correlation between BLEU and any of the manual evaluation metrics (pearson correlation between BLEU and R, O and M was $r = 0.23, -0.06, -0.49$ with p$>$ 0.1, respectively), emphasizing a need for more sophisticated automatic metrics than BLEU for evaluating full ESDs, having a complex structure. The best performing variant as per BLEU score differs from the best one in Table 3 due to variance in performance across scenarios as well as different sampled ESDs of the same scenario in Table 5.

Manual evaluation revealed that a model can miss significant events, even though it can generate many relevant ones. As we only de-duplicate multiple occurrences of exactly the same events in a scenario, we observe repeated paraphrases (4.6% across all prompt variants) of the same event, such as 'pour some milk in the pot' and 'pour the milk into the coffee pot' (MAKING COFFEE scenario). 23.9% of the irrelevant events (13.5% across all prompt variants) are incoherent ('take the flat to the bathroom' for CLEANING A FLAT), 11.4% mixed ('sit in front of coffee shop' for MAKING COF-

		Good quality generations
GOING ON A TRAIN	FT	1. get dressed 2. go to station 3. buy ticket 4. get on train 5. wait for train 6. get on train 7. sit in seat 8. read newspaper 9. wait for train 10. get off train 11. get dressed 12. leave station
	SIF	1. go to station 2. buy ticket 3. wait for train 4. get on train 5. sit in seat 6. get off train 7. leave station
FLYING IN AN AIRPLANE	FT	1. go to airport 2. get boarding pass 3. go to the airline counter 4. present boarding pass and boarding pass id to attendant 5. present ticket and id to attendant 6. sit down in seat 7. make sure you are in seat 8. board airplane
	SIF	1. go to airport 2. go to the airline counter 3. present ticket and id to attendant 4. present boarding pass and boarding pass id to attendant 5. get boarding pass 6. board airplane 7. make sure you are in seat 8. sit down in seat
		Poor quality generations
CLEANING UP A FLAT	FT	1. get the flat 2. take the flat to the bathroom 3. take some water 4. take the flat to the sink 5. turn on the water 6. let it wash the flat 7. then put in some dryer sheet 8. get a sponge 9. scrub the flat 10. put the flat in the dryer 11. when done dry the flat with a towel 12. put away the flat
	SIF	1. get the flat 2. take the flat to the bathroom 3. take some water 4. take the flat to the sink 5. turn on the water 6. let it wash the flat 7. then put in some dryer sheet 8. get a sponge 9. scrub the flat 10. put the flat in the dryer 11. when done dry the flat with a towel 12. put away the flat
TAKING THE UNDER-GROUND	FT	1. arrive at the designated location 2. get in car 3. park in car 4. walk into the designated area 5. enter the room 6. find a chair 7. sit on a chair 8. watch movie
	SIF	1. walk into the designated area

Table 7: Scripts generated using SEQUENCE variant of GPT2 for held-out scenarios. FT denotes output from the fine-tuned model and SIF refers to outputs from our framework applied to GPT2.

		Good quality generations
BRUSHING TEETH	FT	1. go into bathroom 2. turn on faucet 3. remove any dirt or debris 4. grab a brush 5. scrub and floss the teeth 6. leave the bathroom
	SIF	1. go into bathroom 2. grab a brush 3. scrub and floss the teeth 4. leave the bathroom
GOING TO A CLUB TO DANCE	FT	1. choose which club to attend. 2. drive or park your car. 3. get in your car. 4. go to the club. 5. enter the club. 6. get up and dance.
	SIF	1. choose which club to attend. 2. get in your car. 3. go to the club. 4. drive or park your car. 5. enter the club. 6. get up and dance.
TAKING A TAXI	FT	1. get in car 2. get into car 3. wait for taxi 4. enter the car 5. pay the fare 6. get out the driver 7. get out the door 8. exit car
	SIF	1. get into car 2. get in car 3. wait for taxi 4. enter the car 5. pay the fare 6. get out the driver 7. exit car 8. get out the door
		Poor quality generations
MAKING GINGER PASTE	FT	1. get your hot water 2. get your bowl 3. turn on the hot water 4. whisk a bowl of sugar into a paste 5. put the bowl on the stove 6. turn on the hot water 7. boil the paste 8. add salt to the paste 9. turn off the water 10. put the bowl on a rack 11. pour the hot water into a saucepan 12. put some salt and sugar in the saucepan 13. turn the heat on 14. pour the sauce on to the bowl 15. eat the paste
	SIF	1. get your hot water 2. get your bowl 3. turn on the hot water 4. whisk a bowl of sugar into a paste 5. put the bowl on the stove 6. boil the paste 7. add salt to the paste 8. put the bowl on a rack 9. pour the hot water into a saucepan 10. put some salt and sugar in the saucepan 11. turn the heat on 12. pour the sauce on to the bowl 13. eat the paste
SURFING THE INTERNET	FT	1. get in the car. 2. drive to the beach. 3. rent a surfboard. 4. go to the beach. 5. put on my sunglasses. 6. get in my car. 7. drive to my beach. 8. go to the surfing.
	SIF	1. 1. rent a surfboard. 2. get in the car. 3. go to the surfing.
ATTENDING A WEDDING	FT	1. schedule an appointment for the reception 2. drive to the venue 3. show your guest information 4. present your wedding ceremony/ reception to the guests 5. invite the friends/ family/ coworkers/ etc. to the wedding 7. present your wedding present to the guests 8. make the guest list 9. take the invitation 10. go to the reception 11. pay the fee/ gratuity 12. give your guest his/her ticket 13. leave the venue
	SIF	1. schedule an appointment for the reception 2. drive to the venue 3. show your guest information 4. present your wedding ceremony/ reception to the guests 5. invite the friends/ family/ coworkers/ etc. to the wedding 6. present your wedding present to the guests 7. make the guest list 8. take the invitation 9. go to the reception 10. leave the venue
BUYING FROM A VENDING MACHINE	FT	1. enter the shop 2. look for the item you wish to purchase 3. go into the register to purchase the item 4. pay for the item 5. take your item to your car 6. take the item to the car and remove the card 7. leave the shop 8. return the card 9. leave the shop
	SIF	1. enter the shop 2. look for the item you wish to purchase 3. go into the register to purchase the item 4. pay for the item 5. take your item to your car 6. take the item to the car and remove the card 7. leave the shop 8. return the card

Table 8: Scripts generated using SEQUENCE variant of GPT2 for novel scenarios. FT denotes output from the fine-tuned model and SIF refers to outputs from our framework applied to GPT2[5].

FEE), 61.4% unrelated ('add shampoo' for WASHING DISHES), and rest ungrammatical.

We present a manual evaluation of novel scenarios to gauge the generalizability of our framework in Table 6. The framework generalizes to most of the novel scenarios except for those which involve very granular events like MAKING GINGER PASTE or TYING SHOE LACES. Although GPT2 is a contextualized model, it confuses BUYING FROM VENDING MACHINE with buying from a store, SURFING THE INTERNET with the 'surfing' activity, or ATTENDING A WEDDING with 'getting married'. Additionally, we provide a few good and bad quality outputs from GPT2 models for held-out (Table 7) and novel (Table 8) scenarios to identify the avenues for improving script induction in LMs.

7 Limitations

De-duplication of Events. As mentioned previously, SIF cannot de-deuplicate paraphrased version of an event. Therefore, more sophisticated paraphrase identification systems could be used to de-duplicate such events. There could be scenarios where multiple occurrence of same event is required. For instance, WASHING DISHES wherein faucet needs to be opened and closed once at the starting before applying soap and secondly after applying soap (when washed by hands). Hence, it is required to differentiate between desirable and undesirable repetition of events.

Full vs Partial Temporal Ordering. While we consider the task of generating full event sequence descriptions for a scenario, we acknowledge that many scenarios may not have strict ordering of events (e.g., either wet ingredients can be mixed first or dry ones in a BAKING A CAKE scenario) or there can be overlapping events (e.g., while oven is pre-heating, batter can be prepared). Instead of considering partial ordering of events (Sakaguchi et al., 2021), we focus on generating multiple possible full sequence of events for a scenario and report the averaged scores.

8 Conclusion and Future Work

We investigate whether pre-trained language models are capable of generating full event sequence descriptions with minimal prompting and find that pre-trained GPT2 has an incomplete understanding of scripts, while BART and T5 did not even produce anything useful through zero-shot probing experiments. We propose `SIF`, an LM-agnostic script induction framework, that is shown to produce meaningful ESDs for unseen scenarios and mitigate errors (such as scenario-irrelevant, repeated, and misordered events) that were observed during probing experiments, as measured by automatic and manual evaluation. We also provide evidence for the generalization capability of our framework to novel scenarios. However, there is great room for improvement which is evident from manual error analysis and qualitative outputs. Future work may focus on developing more sophisticated automatic metrics as well as an end-to-end system for script induction which might help in mitigating cascading of errors, due to each component, common to any pipeline-based approaches.

Acknowledgements

We would like to thank Benjamin Van Durme, Sweta Agrawal, and all the anonymous reviewers for their valuable feedback ad suggestions.

References

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7456–7463.

Nathanael Chambers. 2017. Behind the scenes of an evolving event cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 41–45.

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

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

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Joshua Feldman, Joe Davison, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. *arXiv preprint arXiv:1909.00505*.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 25–30.

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

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models. *arXiv preprint arXiv:2005.00683*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with wikihow. In *Proceedings of The 2020 Conference on Empirical Methods In Natural Language Proceedings (EMNLP)*.

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2021. Goal-oriented script construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 184–200, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Risto Miikkulainen. 1995. Script-based inference and memory retrieval in subsymbolic story processing. *Applied Intelligence*, 5(2):137–163.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2017. Inscript: Narrative texts annotated with script information. *arXiv preprint arXiv:1703.05260*.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Erik T Mueller. 2004. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research*, 5(4):307–340.

Simon Ostermann. 2020. Script knowledge for natural language understanding.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge. *arXiv preprint arXiv:1803.05223*.

Simon Ostermann, Michael Roth, and Manfred Pinkal. 2019. Mcscript2. 0: A machine comprehension corpus focused on script events and participants. *arXiv preprint arXiv:1905.09531*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *arXiv preprint arXiv:2005.04611*.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229.

Karl Pichotta and Raymond J Mooney. 2016a. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*, pages 2800–2806.

Karl Pichotta and Raymond J Mooney. 2016b. Using sentence-level lstm language models for script inference. *arXiv preprint arXiv:1604.02993*.

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

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988, Uppsala, Sweden. Association for Computational Linguistics.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686, Lisbon, Portugal. Association for Computational Linguistics.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proscript: Partially ordered scripts generation via pre-trained language models. *arXiv preprint arXiv:2104.08251*.

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.

Roger C Schank and Robert P Abelson. 1975. Scripts, plans, and knowledge. In *IJCAI*, volume 75, pages 151–157.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2017a. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2017b. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering. In *Proceedings of the 2nd*

Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics, pages 1–11, Valencia, Spain. Association for Computational Linguistics.

Lilian DA Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge.

Nathaniel Weir, Adam Poliak, and Benjamin Van Durme. 2020. Probing neural language models for human tacit assumptions. CogSci.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020. Analogous process structure induction for sub-event sequence prediction. *arXiv preprint arXiv:2010.08525*.

Ben Zhou, Qiang Ning, Daniel Khashabi, and Dan Roth. 2020. Temporal common sense acquisition with minimal supervision. *arXiv preprint arXiv:2005.04304*.

Yilun Zhou, Julie A Shah, and Steven Schockaert. 2019. Learning household task knowledge from wikihow descriptions. *arXiv preprint arXiv:1909.06414*.