# Learning Action Conditions from Instructional Manuals for Instruction Understanding

## Anonymous ACL submission

## Abstract

The ability to infer pre- and postconditions of an action is vital for comprehending complex instructions, and is essential for applications such as autonomous instruction-guided agents and assistive AI that supports humans to perform physical tasks. In this work, we propose a task dubbed action condition inference, which extracts mentions of preconditions and postconditions of actions in instructional manuals. We propose a weakly supervised approach utilizing automatically constructed large-scale training instances from online instructions, and curate a densely human-annotated and validated dataset to study how well the current NLP models do on the proposed task. We design two types of models differ by whether contextualized and global information is leveraged, as well as various combinations of heuristics to construct the weak supervisions. Our experiments show a >20% F1-score improvement with considering the entire instruction contexts and a > 6% F1-score benefit with the proposed heuristics. However, the best performing model is still well-behind human performance.[1]

## 1 Introduction

When performing complex tasks (*e.g. making a gourmet dish*), instructional manuals are often referred to as useful guidelines. To follow the instructed actions, it is crucial to understand the *preconditions*, *i.e.* prerequisites before taking a particular action, and the *postconditions*, *i.e.* the status supposed to be reached after performing the action.

For autonomous agents or assistant AI that aids humans to accomplish tasks, understanding the conditions provides a structured view of a task (Linden, 1994; Aeronautiques et al., 1998; Branavan et al., 2012a; Sharma and Kroemer, 2020) to help the agent correctly judge whether to *proceed* to the next action and *evaluate* the action completions. Knowledge of action-condition dependencies is
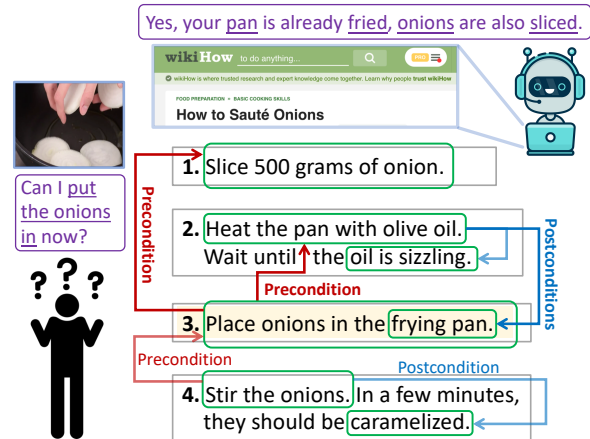


Figure 1: **The Action Condition Inference Task:** We propose a task that probes models' ability to infer both *preconditions* and *postconditions* of an *action* from instructional manuals. It has wide applications to *e.g.* assistive AI and task-solving robots. *This instruction is simplified for illustration.

prevalent and inferable in many instructional texts. For example, in Figure 1, before performing the action "*place onions*" in step 3, both *preconditions*: "*heat the pan*" (in step 2) and "*slice onions*" (in step 1) have to be successfully accomplished. Likewise, executing "*stir onions*" (in step 4), leads to its *postcondition*, "*caramelized*" (also in step 4).

However, no prior work has systematically studied automatically extracting pre- and postconditions from prevalent data resources. To this end, we propose the *action condition inference task* on **real-world instructional manuals**, where a ***dense dependency graph*** is produced, as in Figure 1, to denote the pre- and postconditions of actions. Such a dependency graph provides a systematic task execution plan that agents can closely follow.

We consider two online instruction resources, *WikiHow* (Hadley et al.) and *Instructables.com* (Instructables), to study the current NLP models' capabilities of performing the proposed task. As there is no densely annotated dataset on the desired action-condition-dependencies from real-world instructions, and annotating a comprehensive dependency structure of actions for long instruction con-
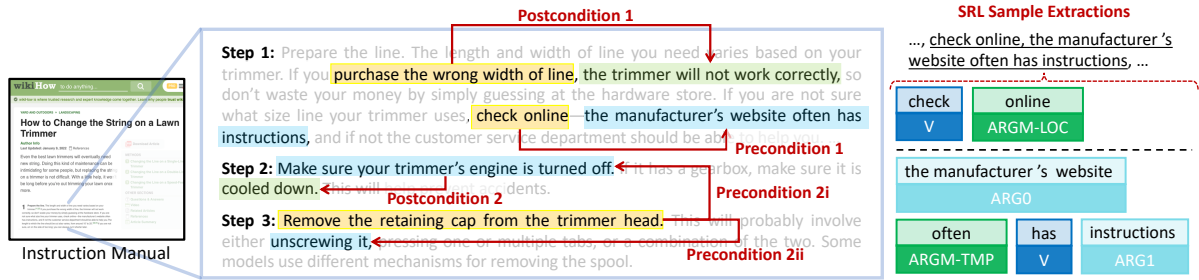
---

[1]We will release dataset and codes upon paper publication.

Figure 2: **Terminologies: (Left)** shows a few exemplar actionables with their associated preconditions and postconditions. Notice that an actionable can have multiple pre- or postconditions and they can span across different instruction steps (for simplicity we do not show an exhausted set of text segments, and the actual instruction contexts are much longer). **(Right)** SRL is used to postulate the text segments (actionables and conditions). We show a sample SRL extraction corresponding to one of the dependency linkages on the left. The SRL `ARG` labels also provide useful information for designing our heuristics (Section 4).

texts can be extremely expensive and laborious, we collect human annotations on a subset of totally 650 samples and benchmark models in either a **zero-shot** setting where no annotated data is used for training, or a **low-resource/shot** setting with limited amount of annotated training data.

We also design the following heuristics and show that they can effectively construct large-scale *weak supervisions*: (1) **Key entity tracing:** Key repetitive entity mentions (including **co-references**) across different instruction descriptions likely suggest a dependency. (2) **Keywords:** Certain keywords (*e.g.* the before in "*do X before doing Y*") can often imply the condition dependencies. (3) **Temporal reasoning:** We adopt a temporal relation module (Han et al., 2021b) to alleviate the potential inconsistencies between the narrated orders of conditional events and their actual temporal orders to better utilize their temporally grounded nature (*e.g.* preconditions are *prior to* an action).

We benchmark two strong baselines based on pretrained language models with or without instruction contexts on our annotated held-out test-set, where the models are asked to make predictions *exhaustively* on **every possible dependency**. We observe that contextualized information is essential (> 20% F1-score gain over non-contextualized counterparts), and that our proposed heuristics are able to augment an effective weakly-supervised training data to further improve the performance (> 6% F1-score gain) on the low-resource setting. However, the best results are still well below human performance (> 20% F1-score difference).

Our key contributions are three-fold: (1) We propose an action-condition inference task and create a densely human-annotated *evaluation dataset* to spur research on structural instruction comprehensions. (2) We design linguistic-centric heuristics utilizing entity tracing, keywords, and temporal

reasoning to construct effective large-scale weak supervisions. (3) We benchmark models on the proposed task to shed lights on future research.

## 2 Terminologies and Problem Definition

Our goal is to learn to infer action-condition dependencies in real-world instructional manuals. We first describe essential terminologies in details:

**Actionable** refers to a phrase that a person can follow and execute *in the real world* (yellow colored phrases in Figure 2). We also consider negated actions (*e.g. do not ...*) or actions warned to avoid (*e.g. if you purchase the wrong...*) as they likely also carry useful knowledge regarding the tasks.[2]

**Precondition** concerns the *prerequisites* to be met for an actionable to be executable, which can be a status, a condition, and/or another prior actionable (blue colored phrases in Figure 2). It is worth noting that humans can omit explicitly writing out certain condition statements because of their triviality as long as the actions inducing them are mentioned (*e.g.* heat the pan → pan is heated, the latter can often be omitted). We thus generalize the conventional precondition formulation, *i.e.* sets of statements evaluated to true/false (Fikes and Nilsson, 1971), to a phrase that is either a passive condition statement or an *actionable that induces* the prerequisite conditions, as inspired by Linden (1994).

**Postcondition** is defined as the outcome caused by the execution of an actionable, which often involves status changes of certain objects (or the actor itself) or certain effects emerged to the surroundings or world state (green colored phrases in Figure 2).

**Text segment** in this paper refers to a textual segment of interest, which can be one of: {actionable, precondition, postcondition}, in an article.

---

[2] We ask workers to single out the actual *actionable* phrases, *e.g. purchase the wrong line* → *trimmer will not work.*

In reality, a valid actionable should have both *pre-* and *postcondition* dependencies, however, we do not enforce this in this work as conditions can occasionally be omitted by human authors.

**Problem Formulation.** Given an input instructional manual and some text segments of interest extracted from it, a model is asked to predict the *directed* relation between a pair of segments, where the relation should be one of the followings: NULL (no relation), *precondition*, or *postcondition*.

## 3 Datasets and Human Annotations

As the condition-dependency knowledge we are interested in is prevalent in real-world instructions, we consider two popular online resources, **Wiki-How** and **Instructables.com**, both consist of detailed multi-step task instructions, to support our investigation. For WikiHow, we use the provided dataset from Wu et al. (2022); for Instructables, we scrape the contents directly from their website.

Since densely annotating large-scale instruction sources for the desired dependencies is extremely expensive and laborious, we mainly annotate a *test-set* and propose to train the models via weakly or self-supervised methods. We hence provide a small subset of the human-annotated data to adapt models to the problem domain. To this end, we collect comprehensive human annotations on a selected subset in each dataset to serve as our **annotated-set**, and particularly the subsets used to evaluate the models as the **annotated-test-set**.[3] In total, our densely annotated-set has 500 samples in WikiHow and 150 samples in Instructables, spanning 7,191 distinct actions (defined by main predicate-object phrases) for diversity. In Section 6.2, we will describe how the annotated-set is split to facilitate the low-resource training. We also collect the human performance on the annotated-test-set to gauge the human upper bound of our proposed task. More dataset details are in Append. Sec. A.

### 3.1 Annotations and Task Specifications
**Dataset Structure.** The desired structure of the constructed data, as in Figure 2, features two main components: (1) **text segment** of interest (see Section 2), and (2) **condition linkage**, a *directed* and *relational* link connecting a pair of text segments.

**Annotation Process.** We conduct the annotated-set construction via Amazon Mechanical Turk (MTurk). Each worker is asked to carefully **read**

over thoroughly a prompted complex multi-step instructional manual, where the annotation process consists of three main steps: **(1) Text segments highlighting:** To facilitate this step (and postulating the text segments for constructing weak-supervisions in Section 4), we *pre-highlight* several text segments extracted by *semantic role labelling* (SRL) for workers to choose from.[4] They can also freely annotate (highlight by cursor) their more desirable segments. **(2) Linking:** We encourage the workers to annotate all the possible segments of interest, and then they are asked to connect certain pairs of segments that are likely to have dependencies with a directed edge. **(3) Labelling:** Finally, each directed edge drawn will need to be labelled as either a *pre-* or *postcondition* (NULL relations do not need to be explicitly annotated).

In general, for each article a worker is required to consider on average $>500$ pairwise relations with all associated article contexts ($>300$ tokens), which is a **decently laborious task**. Comparisons on the linkage annotations from different workers are as well made on *every* pair of *their respective annotated* text segments with the ***actual* candidate-consideration** from the **entire** rest of article.

Since the agreements among workers on both text segments and condition linkages are sufficiently high[5] given the complexity of the annotation task, our final human annotated-set retains the *majority voted* segments and linkages.

**Variants of Tasks.** Although proper machine extraction of the text segments of interest as a span-based prediction can be a valid and interesting task, we find that our automatic SRL extraction is already sufficiently reliable.[6] In this paper, we thus mainly focus on the more essential linkage prediction (and their labels) task assuming that these text segments are given, and leave the possible end-to-end system with the (refined) text segment extraction, as the future work. Our proposed task and the associated annotated-set can be approached by a **zero-shot** or **low-resource** setting: the former involves no train-

---

[3] Following Wu et al. (2022), we first choose from physical categories and then sample a manually inspected subset.

[4] SRL V and ARGs are connected alongside intermediate words to form contiguous segments (see Append. Sec. C.1.1).

[5] The mean inter-annotator agreements (IAAs) per Fleiss Kappa for (segments, linkages) are (0.90, 0.57) and (0.88, 0.58) for WikiHow and Instructables. Note that the Kappa agreement measures the extent to which the observed amount of agreement among raters exceeds what would be expected if all raters made their ratings completely randomly, so the agreement is high. See Append. Sec. B.1 for more details.

[6] ~58% of the time SRL-proposed segments were directly used, with others mostly being few-word-span refinements.

| Heuristics | Examples | Descriptions |
|---|---|---|
| Entity-Tracing & Coref. | … Heat the pan with olive oil. …… Slice 500 grams of onions. … ↑ Precondition 1 … Place them in the frying pan. … Precondition 2 | The shared entities are pan and onions (linked via co-references to them). |
| Keywords | Precondition … Make sure everything is dry *before* you fill your flowerpot with dirt. … … *If* you're using a machine punch, stick the rivet through the hole. … Precondition | Keywords are used to link the segments they separate. If the keyword is at the beginning (2nd example), the (1st) comma is used to segment the sentences. |
| Postcondition | Postcondition … Warm a pan with oil over medium heat… … the oil *is sizzling*. … Postcondition … Do not pour water into your lock …. … the water *will be frozen solid* … SRL Tags: ARGM-MOD V ARG2 | Certain linguistic hints (*e.g.* SRL tags) are utilized to propose plausible (and likely) postcondition text segments. |
| Temporal | … *Step* down hard on the rubber part of the tire … AFTER …*pry* off the back side of the tire first … Precondition | The action prying should occur prior to stepping, but these two segments are reversely narrated in the contexts. |

Table 1: **Heuristics** used for determining condition linkages between text segments, with sample use-cases and descriptions.

ing on any of the annotated data and a heuristically constructed training set can be utilized (Section 4), while the latter allows models to be finetuned on a limited annotated-subset (Section 5.3).

## 4 Training With Weak Supervision

As mentioned in Section 3, our proposed task can be approached via a zero-shot setting, where the vast amount of **un-annotated instruction data** can be transformed into useful training resources (same dataset structure as described in Section 3.1). Moreover, it is proven that in many low-resource NLP tasks, constructing a much larger heuristic-based weakly supervised data can be beneficial (Plank and Agić, 2018; Nidhi et al., 2018).

### 4.1 Linking Heuristics

The goal of designing certain heuristics is to perform a rule-based determination of the linkage (its direction and the condition label). Our design intuition is to harness dependency knowledge by exploiting relations between actions and entities (*entity-level*), certain linguistic patterns (*phrase-level*), and *event-level* information, which should be widely applicable to all kinds of instructional data. Concretely, we design four types of heuristics: (1) **Keywords:** certain keywords are hypothesized to show strong implication of conditions such as *if*, *before*, *after*; (2) **Key entity tracing:** text segments that share the same key entities are likely indicating dependencies; (3) **Co-reference** resolution is adopted to supplement (2); (4) **Event temporal relation resolution** technique is incorporated to handle the inconsistencies between narrative order and the *actual* temporal order of the events.

Without access to human refinements (Section 3.1), we leverage SRL to postulate all the segments of interests to construct the weakly-supervised set.

### 4.1.1 Keywords

Table 2 lists the major keywords that are considered in this work. Denote a text segment as $a_i$, keywords are utilized so as the text segments separated with respect to them, *i.e.* $a_1$ and $a_2$, can be properly linked. Different keywords and their positions within sentences can lead to different *directions* of the linkages, *i.e.* $a_1 \rightleftarrows a_2$ (see second row of Table 1, note that here condition labels are not yet determined). For example, keywords before and after intuitively can lead to different directions if they are placed at non-beginning positions. We follow the rules listed in Table 2 to decide the directions.

### 4.1.2 Key Entity Tracing

It is intuitive to assume that if the two text segments mention the same entity, a dependency between them likely exists, and hence a *trace* of the same mentioned entity can postulate potential linkages. As exemplified in the first row of Table 1, that *heating the pan* being a necessary precondition to *placing onions in the pan* can be inferred by the shared mention "pan". We adopt two ways to propose the candidate entities: (1) We extract all the *noun phrases* within the SRL segments (mostly ARG-tags), (2) Inspired by (Bosselut et al., 2018), a model is learned to predict potential entities involved that are not explicitly mentioned (*e.g.* *fry the chicken* may imply a *pan* is involved) in the context (more details see Append. Sec. C.1.4).

**Co-References.** Humans often use pronouns to refer to the same entity to alternate the mentions in articles, as exemplified by the mentions onions and them, in the first row of Table 1. Therefore, a straightforward augmentation to the aforementioned entity tracing is incorporating co-references of certain entities. We utilize a co-reference reso-

| Keywords | Begin. | Within Sent. |
|---|---|---|
| before, until, in order to, so | $a_1 \longrightarrow a_2$ | $a_1 \longleftarrow a_2$ |
| requires | — | $a_1 \longrightarrow a_2$ |
| after, once, if | $a_1 \longleftarrow a_2$ | $a_1 \longrightarrow a_2$ |

Table 2: **Keywords for deciding a potential linkage:** If a keyword is at the beginning of a sentence, we use the (first) comma of that sentence to separate it to two segments and link them accordingly, while the keyword itself is used as the separator otherwise. The segments are then either refined with SRL or kept as they are if SRL does not detect a valid verb.

lution model (Lee et al., 2018) to propose possible co-referred terms of extracted entities of each segment within the same step description (we do not consider cross-step co-references for simplicity).

### 4.2 Linking Algorithm

After applying the aforementioned linking heuristics, each text segment $a_i$, can have $M$ linked segments: $\{a_1^{l_i}, ..., a_M^{l_i}\}$. For linkages that are *traced* by entity mentions (and co-references), their directions always start from priorly narrated segments to the later ones, while linkages determined by the keywords follow Table 2 for deciding their directions. However, the text segments that are narrated too much distant away from $a_i$ are less likely to have direct dependencies. We therefore *truncate* the linked segments by ensuring any $a_j^{l_i}$ is narrated **no more than** "$S$ step" ahead of $a_i$, where $S$ is empirically chosen to be 2 in this work.

#### 4.2.1 Incorporating Temporal Relations

As hinted in Section 2, the conditions with respect to an actionable imply their temporal relations. The direction of an entity-trace-induced linkage is naively determined by the narrated order of text segments within contexts, however, in some circumstances (*e.g.* fourth row in Table 1), the narrative order can be inconsistent with the actual temporal order of the events. To alleviate such inconsistency, we apply an event temporal relation prediction model (Han et al., 2021b) to fix the linkage directions.[7] The utilized model predicts temporal relations (*before/after/vague*) of each pair of event triggers (extracted by SRL, *i.e.* verbs/predicates), and then we invert the direction of an entity-trace-induced linkage, $a_j^{l_i} \rightarrow a_i$, if their predicted temporal relation is opposite to their narrated order.

#### 4.2.2 Labelling The Linkages

It is rather straightforward to label precondition linkages as a simple heuristic can be used: for a given segment, *any segments that linked to the*

---

[7]These do not include linkages decided by the *keywords*.

current one that are either narrated or temporally prior to it are plausible candidates for being preconditions. For determining postconditions, where they are mostly descriptions of status (changes), we therefore make use of certain linguistic cues that likely indicate human written status, *e.g. the water will be frozen* and *the oil is sizzling*. Specifically, we consider: (1) *be-verbs* followed by present-progressive tenses if the subject **is an entity**, and (2) segments whose SRL tags start with ARGM as exemplified in Table 1.

## 5 Models

Our proposed heuristics do not assume specific model architecture to be applicable, and to benchmark the proposed task, we mainly consider two types of **base models**: (1) **Non-contextualized** model takes only the *two text segments* of interest at a time and make the *pairwise* trinary (directed) relation predictions, *i.e.* NULL, *precondition*, and *postcondition*; (2) **Contextualized** model also makes the relation predictions for every pair of input segments, but the inputs include the whole instruction article so the contexts are preserved. The two models are both based off pretrained language models (the non-contextualized model is essentially a standard transformer-based language model finetuned for classification tasks), and the relation prediction modules are multi-layer perceptrons (MLPs) added on top of the language models' outputs. Cross-entropy loss is used for training.

### 5.1 Non-Contextualized Model

The non-contextualized model takes two separately extracted text segments, $a_i$ and $a_j$, as inputs and is trained similarly to the next sentence prediction in BERT (Devlin et al., 2019) (*i.e.* the order of the segments matters, which will be considered in determining their relations), as shown in Figure 3a.

### 5.2 Contextualized Model

The architecture of the contextualized model is as depicted in Figure 3b. Denote the tokens of the instruction text as $\{t_i\}$ and the tokens of $i$-th text segment of interest (either automatically extracted by SRL or annotated by humans) as $\{a_{ij}\}$. A special start and end of segment token, <a> and </a>, is wrapped around each text segment and hence the input tokens become: "$t_1, ..., t_k,$ <a> $a_{i1}, a_{i2}, ..., a_{iK}$ </a>$, ...$". The contextualized segment representation is then obtained by applying a mean pooling over the language model output representations of each of its tokens, *i.e.* denote the output representation of $a_{ij}$
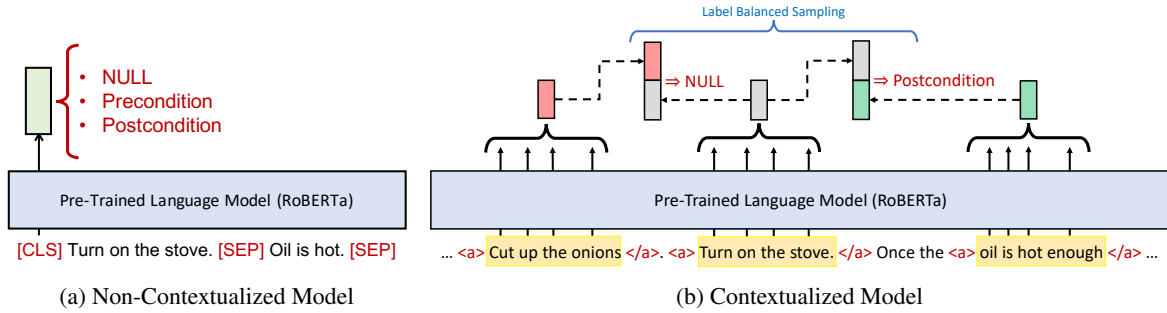
5

(a) Non-Contextualized Model           (b) Contextualized Model

Figure 3: **Model architectures: (a) Non-contextualized model:** The model only considers a pair of given text segments. **(b) Contextualized model:** The model takes the whole instruction paragraphs (*i.e.* contexts) and wrap each text segment with our special tokens (`<a>`), where each segment representation is obtained by taking an average over its token representations. The *ordered* concatenated segment representations will then be fed into an MLP to make the final predictions.

as $\mathbf{o}(a_{ij})$, the segment representation of $\mathbf{o}(a_i)$ is $AvgPool(\sum_{j=1}^{K} \mathbf{o}(a_{ij}))$. To determine the relation between segment $i$ and $j$, we feed their *ordered* concatenated representation, $concat(\mathbf{o}(a_i), \mathbf{o}(a_j))$, to an MLP for the relation prediction.

### 5.3 Learning

**Multi-Staged Training.** For different variants of our task (Section 3.1), we can utilize different combinations of the heuristically constructed dataset and the annotated-train-set. For the low-resource setting, our models can thus be firstly trained on the constructed training set, and then finetuned on the annotated-set. Furthermore, following the **self-training** paradigm (Xie et al., 2020; Du et al., 2021), the previously obtained model predictions can be utilized to either *augment* (*i.e.* adding linkages) or *correct* (*i.e.* revising linkages) the original heuristically constructed data. And hence a second-stage finetuning can be conducted on this model-self-annotated data for improved performance.

**Label Balancing.** It is obvious that most of the relations between randomly sampled text segment pairs will be NULL, and therefore the training labels are imbalanced. To alleviate this, we downsample the negative samples when training the models. Specifically, we fill each training mini-batch with equal amount of positive (relations are not NULL) and negative pairs, where the negatives are constructed by either *inverting* the positive pairs or *replacing* one of the segment with another randomly sampled *unrelated* segment within the same article.

## 6 Experiments and Analysis

Our experiments seek to answer these questions: (1) How well can the models and humans perform on the proposed task? (2) Is instructional context information useful? (3) Are the proposed heuristics and the second-stage self-training effective?

### 6.1 Training and Implementation Details

For both non-contextualized and contextualized models, we adopt the pretrained RoBERTa (-large) language model (Liu et al., 2019) as the base model. All the linguistic features, *i.e.* SRL (Shi and Lin, 2019), co-references, POS-tags, are extracted using models implemented by AllenNLP (Gardner et al., 2017). We truncate the input texts at maximum length of 500 while ensuring all the text segments within this length is preserved completely.

### 6.2 Experimental Setups

**Data Splits.** The primary benchmark of WikiHow annotated-set is partitioned into **train (30%)**, **development (10%)**, and **test (60%)** set, respectively, resulting in 150, 50, and 300 data samples, for low-resource setting. We mainly consider the Instructables annotated-set in a **zero-shot setting** where we hypothesize the models trained on WikiHow can be well-transferred to it. For training conducted on the heuristically constructed data, including the second-stage self-training, we use respective held-out development sets to select the checkpoints around performance convergence for finetuning.

**Evaluation Metrics.** We ask the models to predict the relations on *every* pair of text segments in a given instruction, and compute the average precision (Prec.), recall, and F-1 scores separately with respect to each (pre/post) condition labels.

**Baselines.** There is no immediate baseline we are aware of for the proposed action condition inference task. However, we note that Dalvi et al. (2019)'s dependency graph prediction on scientific procedures (Mishra et al., 2018) shares high-level similarities to specifically our precondition inference task. Our non-contextualized model (without the second-stage self-training) with *only* the noun-phrase-based entity tracing heuristic resembles the KB-induced *prior dependency likelihood*, $g_{kb}$, in

| Model | Heus. | Finetuned/Self | WikiHow Annotated-Test-Set | | | | | | Instructables Annotated-Test-Set | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Precondition | | | Postcondition | | | Precondition | | | Postcondition | | |
| | | | Prec. | Recall | F-1 | Prec. | Recall | F-1 | Prec. | Recall | F-1 | Prec. | Recall | F-1 |
| Prob. Random | — | N/N | 3.55 | 4.42 | 3.54 | 0.61 | 0.86 | 0.68 | 2.94 | 3.88 | 3.04 | 0.46 | 0.46 | 0.42 |
| Prompt. GPT-3 | — | N/N | 3.87 | 73.46 | 7.35 | 4.90 | **77.08** | 9.21 | 3.14 | 64.25 | 5.99 | 1.37 | 34.33 | 2.65 |
| Adapt.-XPAD | — | Y/N | 6.21 | 58.38 | 10.64 | 9.47 | 13.83 | 10.45 | 5.11 | 57.53 | 8.92 | 7.74 | 9.00 | 7.89 |
| Non-Context. | Y | Y/N | 8.21 | 79.52 | 14.32 | 15.43 | 44.99 | 20.56 | 6.49 | 65.05 | 11.31 | 13.64 | 43.50 | 18.65 |
| | Y | Y/Y | 8.56 | **81.19** | 14.91 | 26.53 | 65.95 | 34.31 | 6.64 | **67.13** | 11.54 | 24.53 | **61.93** | 31.78 |
| Context. | N | Y/N | 34.01 | 58.33 | 39.27 | 34.44 | 43.15 | 36.79 | 26.93 | 53.43 | 32.92 | 32.16 | 41.39 | 34.42 |
| | N | Y/Y | 42.26 | 58.45 | 45.41 | 40.99 | 46.51 | 42.32 | 38.16 | 55.77 | 42.23 | 42.57 | 48.00 | 44.07 |
| | Y | N/N | 10.69 | 34.79 | 15.05 | 10.34 | 11.88 | 10.49 | 10.34 | 16.17 | 11.42 | 4.52 | 4.15 | 4.15 |
| | Y | Y/N | 47.92 | 64.63 | 51.38 | 51.15 | 57.64 | 52.59 | 40.70 | 58.97 | 45.17 | 47.92 | 56.51 | 50.06 |
| | Y | Y/Y | **49.42** | 68.40 | **53.51** | **52.39** | 57.35 | 53.42 | **43.81** | 62.71 | **48.34** | 53.41 | 60.51 | **55.17** |
| Human | — | — | 83.91 | 83.86 | 83.55 | 77.39 | 84.81 | 78.81 | 84.74 | 81.32 | 82.78 | 71.90 | 82.51 | 75.53 |

Table 3: **Annotated-test-set performance:** The best performance is achieved by applying all of the proposed **heuristics (heus.)** and undergoing the two-stage training: **finetuned** on the annotated-train-set first and then perform the **self**-training. Note that for the Instructables, both *Finetuned* and *Self* are done on the WikiHow training sets and a zero-shot transfer is performed.

| Heuristics. | WikiHow Annotated-Test-Set | | | | | | Instructables Annotated-Test-Set | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Precondition | | | Postcondition | | | Precondition | | | Postcondition | | |
| | Prec. | Recall | F-1 | Prec. | Recall | F-1 | Prec. | Recall | F-1 | Prec. | Recall | F-1 |
| – temporal – coref. - keywords | 45.60 | 61.22 | 48.59 | 43.71 | 47.56 | 44.35 | 39.35 | 57.03 | 43.49 | 38.45 | 42.96 | 39.39 |
| – temporal – coref. | 43.43 | 64.43 | 48.04 | 46.27 | 51.27 | 47.22 | 37.06 | 59.95 | 42.56 | 38.41 | 44.54 | 39.83 |
| – temporal | 45.83 | 62.48 | 49.17 | 47.72 | 52.70 | 48.81 | 39.39 | 59.53 | 44.23 | 46.81 | 52.15 | 48.23 |

Table 4: **Heuristics ablations:** The models used here are **contextualized** models without the second-stage self-training for both datasets, and "–" indicates exclusion (from using all). In general, each of the designed heuristics give incremental performance gain to both datasets, where the temporal component is particularly effective in postcondition predictions (compare to Table 3).

| Train | Precondition | | | Postcondition | | |
|---|---|---|---|---|---|---|
| | Prec. | Recall | F-1 | Prec. | Recall | F-1 |
| 10% | 41.34 | 61.71 | 46.06 | 45.24 | 55.56 | 47.95 |
| 20% | 45.60 | 67.55 | 50.78 | 49.30 | 58.02 | 51.62 |
| 30% | 57.38 | 64.46 | 57.53 | 50.49 | 54.57 | 51.09 |
| 40% | 49.61 | 73.09 | 55.14 | 50.45 | 57.77 | 52.27 |
| 50% | 54.27 | 70.89 | 57.84 | 51.35 | 55.85 | 52.23 |
| 60% | 53.21 | 69.36 | 56.42 | 53.68 | 58.09 | 54.46 |

Table 5: **Varying annotated-train-set size:** on WikiHow (test-set size is fixed at 30%). We use the (best) model trained with all the proposed heuristics and the self-training paradigm.

their proposed XPAD framework.[8]

Beside this *adapted*-XPAD, we also evaluate our task with (1) **probabilistic random-guess baseline** (random guesses proportional to the training-set label ratio), and (2) **zero-shot GPT-3** (Brown et al., 2020) where the prompts feature exemplar data instances as the task definition (**contextualized**, see Append. Sec. C.2 for prompts used), to setup and justify challenges our task poses.

### 6.3 Experimental Results

Table 3 left half summarizes both the human and model performance on our standard split (30% train, 60% test) of WikiHow annotated-set. Contextualized model obviously outperforms the non-contextualized counterpart greatly, and all learned models perform well-above random baseline. Significant improvements on both pre- and postcondi-

tion inferences can be noticed when heuristically constructed data is utilized, especially when no second-stage self-training is involved. The best performance is achieved by **applying all the heuristics** we design, where further improvements are made by augmenting with second-stage pseudo supervisions. Similar performance trends can be observed in Table 3 right half where a zero-shot transfer from models trained on WikiHow data to Instructables is conducted.

Notice that the zero-shot GPT-3 performs quite poorly compared to our *best low-resource training setting*, and generally worse than our zero-shot contextualized model utilizing only the heuristically constructed data, justifying the effectiveness of our proposed training paradigm and the difficulty of our task. Nevertheless, there are still **large rooms** for improvement as the best model falls well-behind human performance (>20% F1-score gap).

**Heuristics Ablations.** Table 4 features ablation studies on the designed heuristics. One can observe that keywords are mostly effective on inferring the postconditions, and co-references are significantly beneficial in the Instructables data, which can hypothetically be attributed to the writing style of the datasets (*i.e.* authors of Instructables might use co-referred terms more). Temporal relation resolution is consistently helpful across pre- and postconditions as well as datasets, suggesting only relying on

---

[8]With all entity-state-related components excluded (irrelevant to our task) and encoder replaced by RoBERTa model.

| Type | Example | Description |
|------|---------|-------------|
| Heus. Overfit | ... use a sharp <u>blade</u> to cut ...   Precondition ✓ <br> 🤖 Precondition ... look for a <u>blade</u> ... | Overfits on entity trace heuristic. |
| Lacking Causal Reason | ... body start leaning   Precondition ✓ <br> 🤖 NULL ... decrease pedal resistance ... <br> ... can't completely dry ...   Postcondition ✓ <br> 🤖 NULL ... bacteria could form ... | Knowledge-enhanced causal reasoning can be helpful. |

Table 6: **Sample model errors.** The second row are from distant segments not link-able even via the keyword heuristic.

narrated orders could degenerate the performance.

**Error Analysis.** Our (best) models perform well on linkages that exhibit similar concepts to the designed heuristics and generalize beyond their surface forms. However, they can sometimes **overfit to certain heuristic** concepts (Table 6 first row, a food preparation context). Another improvement the models can enjoy is **better causal understanding**, which is currently not explicitly handled by our heuristics and can be an interesting future work (Table 6 second row, biking and cleaning contexts).

### 6.3.1 The Effect of Training Set Size

Table 3 shows that with a little amount of data for training, our models can perform significantly better than the zero-shot setting. This arouses a question – how would the performance change with respect to the training set size, *i.e.* do we just need more data? To quantify the effect of training size on model performance, we conduct an experiment where we vary the sample size in the training set while fixing the development (10%) and test (30%) set for consistency consideration. We use the best settings in Table 3, *i.e.* with all the heuristics and self-training paradigm, for this study. We can observe, from Table 5, a plateau in performance when the training set size is approaching 60%, implying that simply keep adding more training samples does not necessarily yield significant improvements, and hypothesize that the discussed potential improvements are the keys to further effectively exploit the rich knowledge in large-scale instructional data.

### 7 Related Works

**Procedural Text Understanding.** Uncovering knowledge in texts that specifically features *procedural structure* has drawn many attentions, including aspects of tracking entity state changes (Branavan et al., 2012b; Bosselut et al., 2018; Mishra et al., 2018; Tandon et al., 2020), incorporating common sense or constraints (Tandon et al., 2018; Du et al., 2019), procedure-centric question answering (QA) (Tandon et al., 2019), and structural parsing or generations (Malmaud et al., 2014; Zellers et al., 2021). (Clark et al., 2018) leverages VerbNet (Schuler, 2005) with *if-then* constructed rules, one of the keywords we also utilize, to determine object-state postconditions for answering state-related reading comprehension questions. In addition, some prior works also specifically formulate precondition understanding as multiple choice QA for event triggers (verbs) (Kwon et al., 2020) and common sense phrases (Qasemi et al., 2021). We hope our work on inferring action-condition dependencies, an essential knowledge especially for understanding task-procedures, from long instruction texts, can help advancing the goal of more comprehensive procedural text understanding.

Drawing dependencies among procedure steps has been explored in (Dalvi et al., 2019; Sakaguchi et al., 2021), however, their procedures come from manually synthesized short paragraphs. Our work, on the other hand, aims at inferring diverse dependency knowledge directly from more complex real-world and task-solving-oriented instructional manuals, enabling the condition dependencies to go beyond inter-step and narrative order boundaries.

**Event Relation Extraction.** Our work is also inspired by document-level event relation extraction (Han et al., 2019, 2021a; Huang et al., 2021; Ma et al., 2021). Specifically, certain works also adopt weak supervisions to learn event temporal relations (Zhou et al., 2020, 2021; Han et al., 2021b).

### 8 Conclusions

In this work we propose a task on inferring action and (pre/post)condition dependencies on real-world online instructional manuals. We formulate the problem in both zero-shot and low-resource settings, where several heuristics are designed to construct an effective large-scale weakly supervised data. While the proposed heuristics and the two-staged training leads to significant performance improvements, the results still highlight significant gaps below human performance ($> 20\%$ F1-score).

We hope our studies and the collected resources can spur relevant research, and suggest two main future directions: (1) End-to-end propose (refined) actionables, conditions, and their dependencies, by fully exploiting our featured span-annotations of the text segments. (2) Inferred world states from the text descriptions as well as external knowledge of the entities and causal common sense can be factored into the heuristics for weak-supervisions.

## 9 Limitations

We hereby discuss the current limitations of our work: **(1)** As mentioned in Section 3.1, although our annotated dataset enables the possibility of learning an extractive model that can be trained to predict the span of the text segments of interest from scratch, we focus on the more essential action-condition dependency linkage inference task as we find that the SRL extraction heuristic currently applied sufficiently reliable. In the future, we look forward to actualizing such an extractive module and other relevant works that can either further refine the SRL-spans or directly propose the text segments we require. More specifically, the extractive module can be supervised and/or evaluated against with our human annotations on the text segment start-end positions of an article. **(2)** The current system is only trained on unimodal (text-only) and English instruction resources. Multilingual and multimodal versions of our work could be as well an interesting future endeavors to make. **(3)** In this work, we mostly consider instructions from physical works. While certain conditions and actions can still be defined within more social domain of data (*e.g.* a precondition to *being a good person* might be *cultivating good habits*). As a result, we do not really guarantee the performance of our models when applied to data from these less physical-oriented domains.

## 10 Ethics and Broader Impacts

We hereby acknowledge that all of the co-authors of this work are aware of the provided *ACL Code of Ethics* and honor the code of conduct. This work is mainly about inferring pre- and postconditions of a given action item in an instructional manual. The followings give the aspects of both our ethical considerations and our potential impacts to the community.

**Dataset.** We collect the human annotation of the ground truth condition-action dependencies via Amazon Mechanical Turk (MTurk) and ensure that all the personal information of the workers involved (e.g., usernames, emails, urls, demographic information, etc.) is discarded in our dataset. Although we aim at providing a test set that is agreed upon from various people examining the instructions, there might still be unintended biases within the judgements, we make efforts on reducing these biases by collecting diverse set of instructions in order to arrive at a better general consensus on our task.

This research has been reviewed by the **IRB board** and granted the status of an **IRB exempt**. The detailed annotation process (pay per amount of work, guidelines) is included in the appendix; and overall, we ensure our pay per task is above the the annotator's local minimum wage (approximately $15 USD / Hour). We primarily consider English speaking regions for our annotations as the task requires certain level of English proficiency.

**Techniques.** We benchmark the proposed condition-inferring task with the state-of-the-art large-scale pretrained language models and our proposed training paradigms. As commonsense and task procedure understanding are of our main focus, we do not anticipate production of harmful outputs, especially towards vulnerable populations, after training (and evaluating) models on our proposed task.

## References

Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. 1998. Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations (ICLR)*.

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012a. Learning high-level planning from text. In *Association for Computational Linguistics (ACL)*.

S.R.K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012b. Learning high-level planning from text. In *Association for Computational Linguistics (ACL)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901.

Peter Clark, Bhavana Dalvi, and Niket Tandon. 2018. What happened? leveraging verbnet to predict the effects of actions in procedural text. *arXiv preprint arXiv:1804.05435*.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 4496–4505.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 4171–4186.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Xinya Du, Bhavana Dalvi Mishra, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire Cardie. 2019. Be consistent! improving procedural text comprehension using label consistency. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Richard E Fikes and Nils J Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. In *Artificial intelligence*, volume 2, pages 189–208. Elsevier.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Chris Hadley, Katiana Uyemura, Kyle Hall, Kira Jan, Sean Volavong, and Natalie Harrington. Wikihow.

Rujun Han, I-Hung Hsu, Jiao Sun, Julia Baylon, Qiang Ning, Dan Roth, and Nanyun Peng. 2021a. Ester: A machine reading comprehension dataset for event semantic relation reasoning. In *The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rujun Han, Qiang Ning, and Nanyun Peng. 2019. Joint event and temporal relation extraction with shared representations and structured prediction. In *2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rujun Han, Xiang Ren, and Nanyun Peng. 2021b. Econet: Effective continual pretraining of language models for event temporal reasoning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. Document-level entity-based extraction as template generation. In *The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Instructables. instructables.com. [Online; accessed 24-June-2022].

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

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 *Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton Lee, Luheng He, and L. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Keith Vander Linden. 1994. Generating precondition expressions in instructional text. In *Association for Computational Linguistics (ACL)*.

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*.

Mingyu Derek Ma, Jiao Sun, Mu Yang, Kung-Hsiang Huang, Nuan Wen, Shikhar Singh, Rujun Han, and Nanyun Peng. 2021. Eventplus: A temporal event understanding pipeline. In *2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Demonstrations Track*.

Jonathan Malmaud, Earl Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Aldrian Obaja Muis Naoki Otani Nidhi, Vyas Ruochen Xu, and Yiming Yang Teruko Mitamura Eduard Hovy. 2018. Low-resource cross-lingual event type detection in documents via distant supervision with minimal effort. In *International Conference on Computational Linguistics (COLING)*.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Association for Computational Linguistics (ACL)*.

Barbara Plank and Željko Agić. 2018. Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Empirical Methods in Natural Language Processing (EMNLP)*.

10

Ehsan Qasemi, Filip Ilievski, Muhao Chen, and Pedro Szekely. 2021. Corequisite: Circumstantial preconditions of common sense knowledge. In *West Coast NLP Summit (WeCNLP)*.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proScript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.

Mohit Sharma and Oliver Kroemer. 2020. Relational learning for skill preconditions. In *Conference on Robot Learning (CoRL)*.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255.

Niket Tandon, Bhavana Dalvi Mishra, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Niket Tandon, Bhavana Dalvi Mishra, Keisuke Sakaguchi, Antoine Bosselut, and Peter Clark. 2019. Wiqa: A dataset for" what if..." reasoning over procedural text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417.

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.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2022. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals. In *Association for Computational Linguistics (ACL)*.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. Piglet: Language grounding through neuro-symbolic interaction in a 3d world. In *Association for Computational Linguistics (ACL)*.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639.

Ben Zhou, Qiang Ning, Daniel Khashabi, and Dan Roth. 2020. Temporal common sense acquisition with minimal supervision. In *Association for Computational Linguistics (ACL)*.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2021. Temporal reasoning on implicit events from distant supervision. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Yilun Zhou, Julie Shah, and Steven Schockaert. 2019. Learning household task knowledge from WikiHow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56, Macau, China. Association for Computational Linguistics.

## A  Details of The Datasets

Resource-wise our work utilizes online instructional manuals (*e.g.* WikiHow) following many existing works (Zhou et al., 2019; Zhang et al., 2020; Wu et al., 2022), specifically, the large-scale WikiHow training data is provided by (Wu et al., 2022), while we scrape the Instructables.com data on our own. Since Instructables.com dataset tend to have noisier and more free-formed texts, we thus manually sub-sample a smaller (as compared to the test-set of WikiHow) high quality subset.

We report the essential statistics of the annotated-sets in Table 7. Although our definition of actionable is **any** textual phrase that can be actually **acted** in the real world, every unique phrase in our dataset is basically a distinct actionable. We compute the number of distinct actions by extracting the main verb-noun phrases (with lemmatization applied) in a text segment as a *valid-action*, and report their counts in Table 7 as well. Each unique action in this way can lead to roughly only 1-to-3 pairwise relation instance in our annotated dataset. Both this and the aforementioned unique action count justifies the diversity of our collected annotated-set.

Each unique URL of WikiHow can have different multi-step *sections*, and we denote each unique section as a *unique article* in our dataset; while for Instructables.com, each URL only maps to a single section. As a result, for WikiHow we firstly manually select a set of URLs that are judged featuring high quality (*i.e.* articles consisting clear instructed actions, and contain not so much non-meaningful or unhelpful monologues from the writer) instructions and then sample one or two sections from each of the URLs to construct our annotated-set. The statistics of the datasets used to construct the large-scale weakly supervised WikiHow training set can be found in Section 3 of (Wu et al., 2022), where we use their provided WikiHow training samples that are mostly from physical categories.

*Our densely annotated datasets and relevant tools will be made public upon paper acceptance.

### A.1  Dataset Splits

The whole annotated Instructables.com data samples are used as an evaluating set so we do not need to explicitly split them. For WikiHow, we split mainly with respect to the URLs to ensure that no articles (*i.e.* sections) from the same URL are put into different data splits, so as to prevent model exploiting the writing style and knowledge from the

| Type | Counts | | | |
|---|---|---|---|---|
| Total Unique Articles | 500 | | | |
| Total Unique URLs | 326 | | | |
| Annot.-Train / Annot.-Test | 200 / 300 | | | |
| Type-Token Ratio | 9799 / 173920 = 0.06 | | | |
| Pre-/Postcondition Ratio | 16457 / 2839 = 5.80 | | | |
| Distinct Actions | 5205 | | | |
| Avg. Instance per Unique Action | 3.33 | | | |
| Avg. Possible Text Segment Pairs | 717.49 | | | |
| **Type** | **Mean** | **Std** | **Min** | **Max** |
| Tokens in a Step Text | 67.67 | 23.77 | 2 | 161 |
| Sentences in a Step Text | 4.20 | 1.00 | 1 | 6 |
| Tokens in an article | 319.12 | 91.71 | 96 | 631 |
| Sentences in an article | 19.81 | 4.03 | 11 | 28 |

(a) WikiHow

| Type | Counts | | | |
|---|---|---|---|---|
| Total Unique Articles | 150 | | | |
| Total Unique URLs | 150 | | | |
| Annot.-Train / Annot.-Test | 0 / 150 | | | |
| Type-Token Ratio | 5580 / 60150 = 0.09 | | | |
| Pre-/Postcondition Ratio | 5157 / 698 = 7.39 | | | |
| Distinct Actions | 1986 | | | |
| Avg. Instance per Unique Action | 1.11 | | | |
| Avg. Possible Text Segment Pairs | 633.75 | | | |
| **Type** | **Mean** | **Std** | **Min** | **Max** |
| Tokens in a Step Text | 64.75 | 42.57 | 2 | 234 |
| Sentences in a Step Text | 4.27 | 2.73 | 1 | 17 |
| Tokens in an article | 333.3 | 143.22 | 124 | 877 |
| Sentences in an article | 21.98 | 9.47 | 10 | 50 |

(b) Instructables.com

Table 7: **General statistics of the two annotated-sets**: We provide the detailed component counts of the annotated-sets used in this work, including the statistics of tokens and sentences from the instruction steps (lower halves).

same URL of articles on WikiHow. The splitting on the URL-level is as well a random split.

## B  Details of Human Annotations

### B.1  Inter-Annotator Agreements (IAAs)

There are two types of inter-annotator agreements (IAAs) we compute: (1) **IAA on text segments** and (2) **IAA on linkages**, and we describe the details of their computations in this section.

**IAA on Text Segments.**  For each worker-highlighted text segment, either coming from directly clicking the pre-highlighted segments or their own creations, we compute the percentage of the overlapping of the tokens between segments annotated by different workers. If this percentage is > 60% of each segment in comparison, we denote these two segments are *aligned*. Concretely, for all the unique segments of the same article, annotated by different workers, we can postulate a segment dictionary where the *aligned* segments from different worker annotations are combined into the

same ones. And hence each worker's annotation can be viewed as a binary existence of each of the items in such a segment dictionary, where we can compute the Cohen's Kappa inter-annotator agreement scores on every pair of annotators to derive the averaged IAA scores.

**IAA on Linkages.** Similar to the construction of a segment dictionary, we also construct a *linkage dictionary* where every link has a *head segment* pointing to the *tail segment*, with both of the segments coming from an item in the segment dictionary. We thus can also treat the annotation of the linkages across different worker annotations as a binary existence and perform similar inter-annotator agreement computations.

The resulting IAAs for each dataset and annotation types are reported in Section 3.1.

**Majority Vote.** To obtain the final multi-annotator-judged refined data, with our collection budget allowance, we ensure that the number of annotators per data instance (instruction article) is at least 2 (mostly 3), where *consensus* (strict agreement) is used for instances with 2 annotators, and *majority vote* is adopted for 3 annotators.

## B.2 Annotation Process

We adopt Amazon Mechanical Turk (MTurk) to publish and collect our annotations, where each of the annotation in the MTurk is called a Human Intelligence Task (HIT). As shown in Figure 4a, on the top of each HIT we have a detailed description of the task's introduction, terminologies, and instructions. For the terms we define, such as actionables and pre-/postconditions, we also illustrate them with detailed examples. To make it easier for workers to quickly understand our tasks, we provide a video version explaining important concepts and the basic operations. We also set up a Frequently Asked Question (FAQ) section and constantly update such section with some questions gathered from the workers.

Figure 4b shows the layout of the annotation panel. A few statements are pre-highlighted in grey and each of them is clickable. These statements are automatically pre-selected using the SRL heuristics described in Section 3.1, which are supposed to cover as much potential actionables and pre-/postconditions as possible. Workers can either simply click the pre-highlighted statements or *redo* the selection to get their more desired segments. The clicked or selected statements will pop up to

| Confidence Level | WikiHow | Instructables.com |
|---|---|---|
| 5 (Very) | 27.27 | 16.33 |
| 4 (Fairly) | 27.11 | 23.47 |
| 3 (Moderately) | 28.25 | 22.95 |
| 2 (Somewhat) | 16.23 | 29.10 |
| 1 (Not-At-All) | 1.14 | 8.16 |

Table 8: **Confidence-Level Statistics (%):** In WikiHow, majority (> 80%) of the annotators indicate at least > 3 (Moderately) confidence level. As for Instructables.com, it has lower confidence level as the articles tend to be more free-formed and noisy, however, there are still more than 60% of the time workers report confidence levels at least moderately.

the right panel as the text-blocks. For the convenience to manage the page layout, each text-block is *dragable* and can be moved anywhere within the panel. The workers then should examine with their intelligence and common sense to connect text-blocks (two at a time) by right clicking one of them to *start* a directed linkage (which ends at another text-block) and choose a proper dependency label for that particular drawn linkage.

Since our annotation task can be rather complicated, we would like our workers to fully understand the requirements before proceeding to the actual annotation. All annotators are expected to pass three qualification rounds, each consisting of 5 HITs, before being selected as an official annotator. 15 HITs are annotated internally in advance as the standard answers to be used to judge the qualification round qualities.

We calculate the IAAs of each annotator against our standard answers to measure their performance in our task. In each round, only the best performers move on to the next. At the end of each round, we email annotators to explain the questions they asked or some of the more commonly made mistakes shared across multiple workers. In total, over 60 workers participated in our task, and 10 of them passed the qualification rounds.

We estimate the time required to complete each of our HITs to be 10-15 minutes, and adjust our pay rate to $2.5 and $3 USD for the qualification and the actual production rounds, respectively. This roughly equates to a $15 to $18 USD per hour wage, which is above the local minimum wage for the workers. We also ensure that each of our data samples in the official rounds is annotated by at least two different *good workers*.

**Confidence Levels.** We compute the averaged percentage of confidence levels reported by the workers in Table 8. Note that majority of the workers

13

indicate a *moderately* or *fairly* confidence levels, implying they are sufficiently confident about their annotations. We also see feedback from workers that some of them rarely use strong words such as *very* to indicate their confidence levels, and hence the resulted statistics of their confidences could be a bit biased towards the medium.

**Human Performance.** We randomly select 100 samples from the WikiHow annotated-test-set and 50 samples from the Instructables.com annotated-test-set for computing the human performance. The allowed inputs are exactly the same as what models take, *i.e.* given all the instruction paragraph as context and highlighted (postulated text segment boxes) text segments of interests, workers are asked to predict the relations among such segments so as to induce a complete dependency graph. For each sample, we collect inputs from two different workers, and ensure that the workers are not the ones that give the original annotations of the action-condition dependencies. The human performance is then computed by taking the averaged metrics similar to the models on the given samples.

## C   Modelling Details

### C.1   More on Heuristics

#### C.1.1   SRL Extraction

As SRL can detect multiple plausible ways to form the ARG frames to the same *central* verb, we need to determine which one is the most likely to be desirable. When such multiple argument patterns exist for the same central verb, we simply determine the most desirable formation of segments by maximizing both the number of plausible segments (where they do not overlap above certain threshold, which is set to be 60% in this work) *within a sentence* and the number of ARGs in each segment.

#### C.1.2   Linking Algorithm

In Section 4.2 we mention that a maximum distance of 2 steps between linked segments is imposed to filter out possible non-dependent conditions. While this still can potentially include many not-so-much depended text segments, our goal is to exploit the generalization ability of large-scale pretrained language models to *recognize* segments that are most probable conditions by including as much as heuristically proposed linkages as possible, which is empirically proven effective. A better strategy on making such a design choice of maximum allowed step-wise distance is left as a future

work.

#### C.1.3   Keywords

About 3% of the entire un-annotated data have sentences containing the keywords we use in this work (Table 2). Despite the relatively small amount compared to other heuristics, they are quite effective judging from the results reported in Table 3.

#### C.1.4   Key Entity Tracing

For the key entity tracing heuristic described in Section 4.1.2, as long as two segments share at least one mentioned entity, they can be linked (*i.e. traced* by the shared entity). We do not constraint the number of key entities within a segment, so there can be more than one being used to conduct the tracing.

**Constructing Entity Prediction Datasets.** As mentioned in Section 4.1.2, one way to postulate the key entities is via constructing a predictive model for outputting potentially involved entities. To do so, we firstly construct an *entity vocabulary* by extracting all the noun phrases within each SRL extracted segments of the entire un-annotated-set articles. To prevent from obtaining a too much large vocabulary as well as improbable entities, we only retain entities (without lemmatization) that appear with $> 5$ occurrences in at least one article.

We then train a language model (based on RoBERTa-large as well) where the output is the multi-label multi-class classification results on the predicted entities. When predicting the key entities for a given segment, we further constraint the predictions to be within the local vocabulary (more than 5 occurrences) within the article such segment belongs to. This model is inspired by the entity selector module proposed in (Bosselut et al., 2018) while we only consider single step statements. We verify the performance of the learned model on the dataset provided by (Bosselut et al., 2018) (the entity selection task), where our model can achieve roughly 60% on F-1 metric, indicating the trained model is sufficiently reliable.

#### C.1.5   Temporal Relations

We use the temporal relation resolution model from (Han et al., 2021b) that is trained on various temporal relation datasets such as *MATRES* (Ning et al., 2018). We train the model on three different random seeds and make them produce a *consensus* prediction, *i.e.* unless all of the models jointly predict a specific relation (BEFORE or AFTER), otherwise the relation will be regarded as VAGUE.

## C.2 GPT-3 Baseline

We use the most powerful version of GPT-3 (Davinci)[9] provided by the OpenAI GPT-3 API (zero-shot prompted version) with the following prompt:

*Extract the preconditions and postconditions from this text:*

*Text: "Slice 500 grams of onion. Heat the pan with olive oil. Wait until the oil is sizzling. Place onions in the frying pan. Stir the onions. In a few minutes, they should be caramelized."*

*Segment 1: "Heat the pan with olive oil."*

*Segment 2: "oil is sizzling."*

*Label: post-condition*

*Text: "Slice 500 grams of onion. Heat the pan with olive oil. Wait until the oil is sizzling. Place onions in the frying pan. Stir the onions. In a few minutes, they should be caramelized."*

*Segment 1: "Slice 500 grams of onion."*

*Segment 2: "Place the onions in the frying pan."*

*Label: pre-condition*

*Text: "Slice 500 grams of onion. Heat the pan with olive oil. Wait until the oil is sizzling. Place onions in the frying pan. Stir the onions. In a few minutes, they should be caramelized."*

*Segment 1: "Slice 500 grams of onion."*

*Segment 2: "Heat the pan with olive oil."*

*Label: no relation*

*Text:* `"Fill-In an Article"`

*Segment 1:* `"Fill-In Text Segment 1"`

*Segment 2:* `"Fill-In Text Segment 2"`

*Label:* `GPT-3 Prediction`

In other words, we provide an exemplar simplified instance to instruct what pre- and postconditions should be like to the model with the article context and a pair of text segments of interest. And then, the GPT-3 model should *generate* the text description-based prediction label (non-case-sensitive). For preconditions we allow verbalized label to be within {*precondition, pre-condition*}, and postconditions within {*postcondition, post-condition*}. For the `NULL` relation, we allow {*no relation, unrelated, null, none*}.

## C.3 Development Set Performance

We select the model checkpoints to be evaluated using the held-out development split (annotated-dev-set). We also report the performance on this annotated-dev-set in Table 9.

## C.4 More Results on Train-Set Size Varying

Table 10 is a similar experiment as Table 5 but here we conduct the experiments with the models that do not utilize the weakly supervised data constructed with the proposed heuristics at all. One can observe that similar trends hold that a plateau can be noticed when the training set size is approaching 60%. Compared to Table 5, we can also observe that the smaller the train-set size is, the larger gaps shown between the models with and without utilizing the heuristically constructed data. This can further imply the effectiveness of our heuristics to construct meaningful data for the action-condition dependency inferring task. The models with heuristics, if compared at the same train-set size respectively, significantly outperforms every model counterparts that do not utilize the heuristics.

Table 11 reports similar experiments but in the Instructables.com annotated-test-set. Note that we perform a direct zero-shot transfer from the Wiki-How annotated-train-set, so the test-set size is always 100% for the Instructables.

Finally, both Tables 12 and 13 report the same experiments, however, this time the second-stage self-training is not applied. It is worth noting that the self-training is indeed effective throughout all the train-set-size and across different datasets and model variants, however, the trends of model performance hitting a saturation point when the train-set size increases still hold.

## C.5 Training & Implementation Details

**Training Details.** The maximum of 500 token length described in Section 6.1 is sufficient for most of the data in the annotated-test-sets, as evident in Table 7. All the models in this work are trained on a single Nvidia A100 GPU[10] on a Ubuntu 20.04.2 operating system. The hyperparameters for each model are manually tuned against different datasets, and the checkpoints used for testing are selected by the best performing ones on the held-out development sets in their respective datasets.

**Implementation Details.** The implementations of the transformer-based models are extended from the HuggingFace[11] code base (Wolf et al., 2020), and our entire code-base is implemented in PyTorch.[12]

---

[9]https://openai.com/api/pricing/

[10]https://www.nvidia.com/en-us/data-center/a100/

[11]https://github.com/huggingface/transformers

[12]https://pytorch.org/

| Model | WikiHow Annotated-Dev-Set Heuristics | Finetuned | Self | Precondition Prec. | Recall | F-1 | Postcondition Prec. | Recall | F-1 |
|---|---|---|---|---|---|---|---|---|---|
| Non-Context. | All | Y | Y | 8.22 | 74.77 | 14.00 | 19.70 | 69.94 | 28.36 |
| Context. | No Heuristics | Y | N | 29.96 | 56.91 | 35.41 | 30.28 | 39.10 | 32.03 |
| | No Heuristics | Y | Y | 40.09 | 57.60 | 43.20 | 41.10 | 48.59 | 42.53 |
| | All | N | N | 9.59 | 32.69 | 13.35 | 7.48 | 9.26 | 7.81 |
| | – temporal – coref. - keywords | Y | N | 43.59 | 58.74 | 45.95 | 39.33 | 44.45 | 40.64 |
| | – temporal – coref. | Y | N | 38.43 | 60.48 | 42.83 | 39.72 | 47.80 | 41.92 |
| | – temporal | Y | N | 41.19 | 57.06 | 43.92 | 47.63 | 54.69 | 48.91 |
| | All | Y | N | 45.05 | 59.59 | 47.35 | 45.65 | 50.35 | 46.42 |
| | All | Y | Y | 44.93 | 65.25 | 49.12 | 46.06 | 52.04 | 47.21 |

Table 9: **Annotated-dev-set performance on WikiHow:** Similar to Table 3, we report the development set performance on the WikiHow dataset (Instructables.com does not have the development set as we are conducting a zero-shot transfer).

| Train | Precondition Prec. | Recall | F-1 | Postcondition Prec. | Recall | F-1 |
|---|---|---|---|---|---|---|
| 10% | 33.44 | 56.41 | 38.69 | 42.37 | 53.86 | 45.25 |
| 20% | 35.05 | 60.97 | 40.86 | 40.76 | 51.35 | 43.19 |
| 30% | 44.57 | 60.19 | 47.68 | 43.00 | 47.26 | 43.83 |
| 40% | 39.38 | 72.23 | 46.63 | 45.51 | 54.27 | 47.57 |
| 50% | 40.97 | 69.70 | 47.24 | 49.15 | 59.04 | 51.76 |
| 60% | 46.99 | 71.14 | 52.27 | 48.80 | 56.51 | 50.74 |

Table 10: **Varying annotated-train-set size without weakly supervised training:** on WikiHow (test-set size is fixed at 30%). The model used in this experiment is without training on any of the heuristically constructed dataset, but we apply the self-training paradigm.

| Train | Precondition Prec. | Recall | F-1 | Postcondition Prec. | Recall | F-1 |
|---|---|---|---|---|---|---|
| 10% | 32.25 | 50.50 | 36.36 | 41.37 | 51.37 | 44.03 |
| 20% | 35.95 | 56.99 | 40.89 | 48.77 | 60.10 | 51.86 |
| 40% | 39.62 | 64.19 | 45.77 | 48.83 | 60.30 | 52.08 |
| 50% | 57.38 | 64.46 | 57.53 | 50.49 | 54.57 | 51.09 |
| 60% | 45.62 | 61.02 | 49.06 | 55.00 | 65.04 | 57.54 |
| 10% | 27.50 | 50.32 | 32.74 | 34.99 | 47.66 | 38.18 |
| 20% | 26.86 | 51.73 | 32.34 | 40.31 | 52.89 | 43.43 |
| 40% | 30.58 | 64.38 | 38.16 | 44.78 | 60.86 | 49.28 |
| 50% | 39.65 | 63.28 | 45.41 | 50.96 | 59.98 | 53.54 |
| 60% | 39.90 | 65.68 | 45.95 | 49.64 | 58.83 | 51.97 |

Table 11: **Varying annotated-train-set size:** on Instructables.com (test-set size is fixed at 100%). Note that here the train-set size is from WikiHow annotated-set, and the 30% is basically Table 3. The upper half is with models that utilize both the heuristically constructed dataset and the self-training paradigm, while the lower half is with models that do not use any weak supervisions.

| Train | Precondition Prec. | Recall | F-1 | Postcondition Prec. | Recall | F-1 |
|---|---|---|---|---|---|---|
| 10% | 39.77 | 61.58 | 44.65 | 45.76 | 53.42 | 47.57 |
| 20% | 42.75 | 64.32 | 47.40 | 47.97 | 56.99 | 50.21 |
| 30% | 52.37 | 64.59 | 54.43 | 50.70 | 55.93 | 51.87 |
| 40% | 43.77 | 68.58 | 49.28 | 45.47 | 53.78 | 47.48 |
| 50% | 51.98 | 67.29 | 54.94 | 50.45 | 54.84 | 51.21 |
| 60% | 47.96 | 69.77 | 52.61 | 47.81 | 52.27 | 48.77 |
| 10% | 26.37 | 51.61 | 31.80 | 31.52 | 47.68 | 35.33 |
| 20% | 28.62 | 56.40 | 34.53 | 33.68 | 48.10 | 37.30 |
| 30% | 37.20 | 60.09 | 42.32 | 37.44 | 45.52 | 39.39 |
| 40% | 32.74 | 68.97 | 40.57 | 36.33 | 47.00 | 39.00 |
| 50% | 40.30 | 65.62 | 45.94 | 44.86 | 53.36 | 46.85 |
| 60% | 38.80 | 68.16 | 45.27 | 42.03 | 51.96 | 44.43 |

Table 12: **Varying annotated-train-set size:** on WikiHow (test-set size is fixed at 30%). The upper half is with models that utilize the heuristically constructed dataset, while the lower half is with models that do not use any weak supervisions. Both upper and lower halves do **not** undergo any second-stage self-training.

| Train | Precondition Prec. | Recall | F-1 | Postcondition Prec. | Recall | F-1 |
|---|---|---|---|---|---|---|
| 10% | 29.59 | 52.25 | 34.76 | 40.31 | 50.26 | 42.92 |
| 20% | 31.46 | 53.34 | 36.37 | 44.11 | 55.32 | 46.94 |
| 40% | 34.02 | 60.66 | 40.20 | 43.62 | 51.56 | 45.43 |
| 50% | 42.57 | 59.24 | 46.38 | 49.83 | 57.26 | 51.77 |
| 60% | 37.69 | 61.36 | 43.34 | 48.49 | 54.29 | 49.70 |
| 10% | 18.44 | 41.85 | 23.20 | 21.97 | 39.08 | 26.02 |
| 20% | 20.91 | 48.63 | 26.52 | 28.93 | 44.85 | 32.98 |
| 40% | 23.89 | 61.51 | 31.59 | 36.43 | 51.98 | 40.50 |
| 50% | 30.56 | 58.10 | 36.90 | 41.35 | 54.48 | 44.95 |
| 60% | 28.59 | 60.24 | 35.52 | 40.06 | 53.41 | 43.20 |

Table 13: **Varying annotated-train-set size:** on Instructables.com (test-set size is fixed at 100%). The structure of this table is similar to that of Table 12, *i.e.* no self-training is conducted.

## C.6 Hyperparameters

We train our models until performance convergence is observed on the heuristically constructed dataset. The training time for the weakly supervised learning is roughly 6-8 hours. For all the finetuning that involves our annotated-sets, we train the models for roughly 10-15 epochs for all the model variants, where the training time varies from 1-2 hours. We list all the hyperparameters used in Table 14. The basic hyperparameters such as learning rate, batch size, and gradient accumulation steps are kept consistent for all kinds of training in this work, including training on the weakly supervised data, finetuning on the annotated-sets, as well as during the second-stage self-training. We also include the search bounds and number of trials in Table 15, that all of our models adopt the same search bounds and

| Models | Batch Size | Initial LR | # Training Epochs | Gradient Accumulation Steps | # Params |
|---|---|---|---|---|---|
| Non-contextualized | 8 | $1 \times 10^{-5}$ | 15 | 1 | 355M |
| Contextualized | 4 | $1 \times 10^{-5}$ | 15 | 1 | 372M |

Table 14: **Hyperparameters in this work:** *Initial LR* denotes the initial learning rate. All the models are trained with Adam optimizers (Kingma and Ba, 2015). We include number of learnable parameters of each model in the column of *# params*.

| Type | Batch Size | Initial LR | # Training Epochs | Gradient Accumulation Steps |
|---|---|---|---|---|
| **Bound (lower–upper)** | 2–8 | $1 \times 10^{-5}$–$1 \times 10^{-6}$ | 5–15 | 1 |
| **Number of Trials** | 2–4 | 2–3 | 2–4 | 1 |

Table 15: **Search bounds** for the hyperparameters of all the models.

ranges of trials.

# D   Releases & Codes

The comprehensive human-annotated datasets, including both on WikiHow and Instructables.com will be released upon acceptance, along with a clearly stated documentation for usages. We plan to also release the codes (a snippet of our codes are included as a `.zip` file during the reviewing period) for processing the datasets as well as the implementation of our models and proposed training methods. We hope that by sharing the essential resources, our work can incentivize more interests into research on procedural understanding that specifically targets condition and action dependencies and their applications to autonomous task-solving agents and assistant AI that guides humans throughout accomplishing complex tasks.

Hello, about us, and thank you for your help!

Introduction and Terminologies

Instructions and Annotation Flow

FAQ (Optional but VERY HELPFUL)

**\* Please DO NOT refresh the page or press the go back button of your browser. Otherwise, some results may be lost!**

**Tips:**

- If you hover your mouse cursor on a connected edge, the text blocks will change colors to indicate their types for your references.
- Colors used to identify each type of the blocks: Pre-condition color   Actionable color   Post-condition color

Read above for detailed instructions and examples!

(a) Human Annotation Instruction

(b) Sample Annotation Interface

Figure 4: **MTurk Annotation User Interface: (a)** We ask workers to follow the indicated instruction. All the blue-colored text bars on the top of the page are expandable. Workers can click to expand them for detailed instructions of the annotation task. **(b)** The annotation task is designed for an intuitive *click/select-then-link* usage, followed by a few additional questions such as confidence level and feedback (this example is obtained from WikiHow dataset). The grey-color-highlighted text segments are postulated by the SRL, where the color of a segment will turn yellow if either being selected or cursor highlighted. Notice that for better illustration, the directions of the links in our paper are opposite to those in the annotation process.