Goal-Oriented Script Construction

Qing Lyu[*] **Li Zhang**[*] **Chris Callison-Burch**
University of Pennsylvania
{lyuqing,zharry,ccb}@seas.upenn.edu

Abstract

The knowledge of scripts, common chains of events in stereotypical scenarios, is a valuable asset for task-oriented natural language understanding systems. We propose the Goal-Oriented Script Construction task, where a model produces a sequence of steps to accomplish a given goal. We pilot our task on the first multilingual script learning dataset supporting 18 languages collected from wikiHow, a website containing half a million how-to articles. For baselines, we consider both a generation-based approach using a language model and a retrieval-based approach by first retrieving the relevant steps from a large candidate pool and then ordering them. We show that our task is practical, feasible but challenging for state-of-the-art Transformer models, and that our methods can be readily deployed for various other datasets and domains with decent zero-shot performance[1].

1 Introduction

A *script* is a standardized sequence of events about stereotypical activities (Feigenbaum et al., 1981). For example, "*go to a restaurant*" typically involves "*order food*", "*eat*", "*pay the bill*", etc. Such script knowledge has long been proposed as a way to enhance AI systems (Abelson and Schank, 1977). Specifically, task-oriented dialog agents may greatly benefit from the understanding of *goal-oriented scripts*[2]. However, the evaluation of script knowledge remains an open question (Chambers, 2017). Moreover, it is unclear whether current models can generate complete scripts. Such an ability is in high demand for recent efforts to reason about complex events (Li et al., 2020; Wen et al., 2021)[3].

Input: (How to) Obtain Travel Documents

Output:

1. Authority grants approval for travel documents.
2. Applicant takes a photograph if that is required to receive travel document.
3. Travel documents approved and granted to traveler.
4. Get and/or fill out all documents that must be submitted to apply for the travel document.
5. Traveler pays fees and submits application documents.
6. Applicant receives letter, passport, visa stamp, or whatever travel documents were requested.
7. Item is delivered to the destination.
8. Prepare required application documents.
9. Pay any required application fees.

Figure 1: An example script constructed by our Step-Inference-Ordering pipeline in a zero-shot manner. The input is a *goal*, and the output is an ordered list of steps.

We propose the task of *Goal-Oriented Script Construction* (GOSC) to *holistically* evaluate a model's understanding of scripts. Given a *goal* (or the name of a script), we ask the model to construct the sequence of *steps* (or events in a script) to achieve the goal. This task targets a model's ability to narrate an entire script, subsuming most existing evaluation tasks. Our rationale is that a model that *understands* some scripts (e.g. how to "*travel abroad*" and "*go to college*") should be able to produce *new* ones (e.g. how to "*study abroad*") using the absorbed knowledge, close to how humans learn.

While almost all prior script learning work has focused on English, we introduce a novel multilingual corpus. Our corpus is collected from wikiHow (`wikihow.com`), a website of how-to articles in 18 languages. The articles span a wide range of domains, from commonplace activities like going to a restaurant to more specific ones like protecting

[*] Equal contribution.

[1] Our models and data are be available at `https://github.com/veronica320/wikihow-GOSC`.

[2] `https://developer.amazon.com/alexaprize`

[3] `www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas`

oneself from the coronavirus.

We train and evaluate several baseline systems on our GOSC task. First, we consider a generation-based approach where a pretrained language model, multilingual T5, is finetuned to produce scripts from scratch. As an alternative, observing that most desired steps can be drawn from the training scripts due to their magnitude and high coverage, we also propose a retrieval-based approach. Concretely, we develop a Step-Inference-Ordering pipeline using existing models to retrieve relevant steps and order them. We also improve the pipeline with techniques such as multitask learning. From the experiments, the GOSC task proves challenging but feasible for state-of-the-art Transformers. Furthermore, we show that our pipeline trained on wikiHow can generalize to other datasets and domains (see an example in Figure 1). On three classic script corpora, OMICS, SMILE, and DeScript, it achieves strong zero-shot performance. It can also be directly deployed to construct scripts in distant domains (e.g. military/political).

In this paper, we make several contributions:
1) We propose the GOSC task targeting the comprehensive understanding of scripts.
2) We introduce the first multilingual script learning dataset available in 18 languages.
3) We compare generation-based and retrieval-based approaches using both automatic and human judgments, which demonstrate the feasibility but also the difficulty of GOSC.
4) We show that our approach can be readily applied to other datasets or other domains.

2 Related Work

The notion of *scripts* (Abelson and Schank, 1977), or *schemas* (Rumelhart, 1975), encodes the knowledge of standardized event sequences. We dissect previous work on script learning into two lines, *narrative* and *procedural*.

One line of work focuses on *narrative* scripts, where *declarative*, or *descriptive* knowledge is distilled from narrative texts like news or stories (Mujtaba and Mahapatra, 2019). Such scripts are not goal-oriented, but descriptions of sequential events (e.g. a traffic accident involves a collision, injuries, police intervention, etc.). Chambers and Jurafsky (2008) introduced the classic Narrative Cloze Test, where a model is asked to fill in the blank given a script with one missing event. Following the task, a few papers made extensions on representation (Chambers and Jurafsky, 2009; Pichotta and Mooney, 2014) or modeling (Jans et al., 2012; Pichotta and Mooney, 2016a,b,c), achieving better performance on Narrative Cloze. Meanwhile, other work re-formalized Narrative Cloze as language modeling (LM) (Rudinger et al., 2015) or multiple-choice (Granroth-Wilding and Clark, 2016) tasks. However, the evolving evaluation datasets contain more spurious scripts, with many uninformative events such as "say" or "be", and the LMs tend to capture such cues (Chambers, 2017).

The other line of work focuses on *procedural* scripts, where events happen in a scenario, usually in order to achieve a goal. For example, to "visit a doctor", one should "make an appointment", "go to the hospital", etc. To obtain data, Event Sequence Descriptions (ESD) are collected usually by crowdsourcing, and are cleaned to produce scripts. Thus, most such datasets are small-scale, including OMICS (Singh et al., 2002), SMILE (Regneri et al., 2010), the Li et al. (2012) corpus, and DeScript (Wanzare et al., 2016). The evaluation tasks are diverse, ranging from event clustering, event ordering (Regneri et al., 2010), text-script alignment (Ostermann et al., 2017) and next event prediction (Nguyen et al., 2017). There are also efforts on domain extensions (Yagcioglu et al., 2018; Berant et al., 2014) and modeling improvements (Frermann et al., 2014; Modi and Titov, 2014).

In both lines, it still remains an open problem what kind of automatic task most accurately evaluates a system's understanding of scripts. Most prior work has designed tasks focusing on various fragmented pieces of such understanding. For example, Narrative Cloze assesses a model's knowledge for completing a close-to-finished script. The ESD line of work, on the other hand, evaluates script learning systems with the aforementioned variety of tasks, each touching upon a specific piece of script knowledge nonetheless. Recent work has also brought forth generation-based tasks, but mostly within an open-ended/specialized domain like story or recipe generation (Fan et al., 2018; Xu et al., 2020).

Regarding data source, wikiHow has been used in multiple NLP efforts, including knowledge base construction (Jung et al., 2010; Chu et al., 2017), household activity prediction (Nguyen et al., 2017), summarization (Koupaee and Wang, 2018; Ladhak et al., 2020), event relation classification (Park and Motahari Nezhad, 2018), and next passage completion (Zellers et al., 2019). A few recent papers

(Zhou et al., 2019; Zhang et al., 2020b) explored a set of separate goal-step inference tasks, mostly in binary-classification/multiple-choice formats, with few negative candidates. Our task is more holistic and realistic, simulating an open-ended scenario with retrieval/generation settings. We combine two of our existing modules from Zhang et al. (2020b) into a baseline, but a successful GOSC system can certainly include other functionalities (e.g. paraphrase detection). Also similar is Zhang et al. (2020a), which doesn't include an extrinsic evaluation on other datasets/domains though.

In summary, our work has the following important differences with previous papers:
1) Existing tasks mostly evaluate fragmented pieces of script knowledge, while GOSC is higher-level, targeting the ability to invent *new, complete* scripts.
2) We are the first to study *multilingual* script learning. We evaluate several baselines and make improvements with techniques like multitask learning.
3) Our dataset improves upon the previous ones in multiple ways, with higher quality than the mined narrative scripts, lower cost and larger scale than the crowdsourced ESDs.
4) The knowledge learned from our dataset allows models to construct scripts in other datasets/domains without training.

3 Goal Oriented Script Construction

We propose the Goal-Oriented Script Construction (GOSC) task. Given a *goal* g, a system constructs a complete script as an ordered list of *steps* S, with a ground-truth reference T. As a hint of the desired level of granularity, we also provide an expected number of steps (or length of the script), l, as input. Depending on whether the set of possible candidate steps are given in advance, GOSC can happen in two settings: Generation or Retrieval.

In the **Generation setting**, the model must generate the entire script from scratch.

In the **Retrieval setting**, a large set of candidate steps C is given. The model must predict a subset of steps S from C, and provide their ordering.

4 Multilingual WikiHow Corpus

Our previously wikiHow corpus (Zhang et al., 2020b) is a collection of how-to articles in English (en). We extend this corpus by crawling wikiHow in 17 other languages, including Spanish (es), Portuguese (pt), Chinese (zh), German (de), French (fr), Russian (ru), Italian (it), Indonesian (id), Dutch (nl), Arabic (ar), Vietnamese (vn), Thai (th), Japanese (jp), Korean (ko), Czech (cz), Hindi (hi), and Turkish (tr). The resulting multilingual wikiHow corpus may be used in various tasks in NLP and other fields.

Figure 2: An abridged example script extracted from the English wikiHow article "How to Eat at a Sit Down Restaurant".

For script learning, we extract from each wikiHow article the following critical components to form a *goal-oriented script*.
Goal: the title stripped of "How to";
Section: the header of a "method" or a "part" which contains multiple steps;[4]
Steps: the headlines of step paragraphs;
Category: the top-level wikiHow category.
An example wikiHow script is shown in Figure 2.

Our previous corpus provides labels of whether each English article is ordered, predicted by a high-precision classifier. We project these labels to other languages using the cross-language links in each wikiHow article. For articles without a match to English, it defaults to unordered. In our task setup, we only require the model to order the steps if an article is ordered.

For all experiments below, we randomly hold out 10% articles in each language as the test set, and use the remaining 90% for training and development.[5]

We use the corpus to construct a dataset for multilingual GOSC. For the Retrieval setting, the set of candidate steps C are all the steps present in the test set. However, we observe that not only the large number of steps may render the evaluation intractable, but most steps are also evidently distant from the given goal. To conserve computing power, we restrict C as all the steps from articles within the same wikiHow category for each script.

5 Models

We develop two systems based on state-of-the-art Transformers for the GOSC task.[6]

5.1 Generation Approach: Multilingual T5

For the Generation setting, we finetune mT5 (Xue et al., 2020), a pretrained generation model that is

[4]We ignore this hierarchical relation and flatten all steps in all Sections as the Steps of the script.
[5]See Appendix A for our corpus statistics.
[6]Reproducibility details can be found in Appendix C.

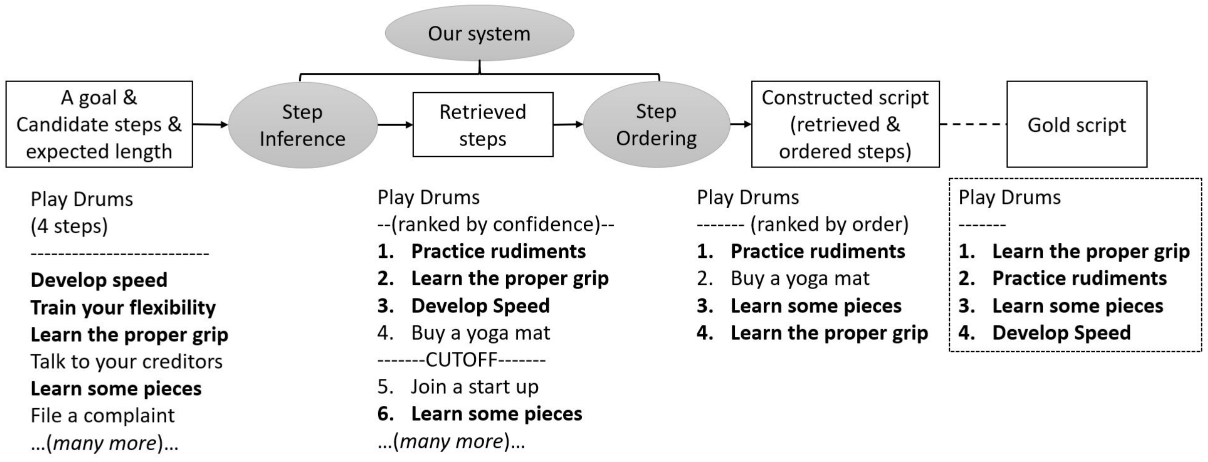


Figure 3: Our Step-Inference-Ordering pipeline for the GOSC Retrieval task. An example *ordered* script is shown with example steps in the input and output. Those that appear in the ground-truth script is in bold.

not only state-of-the-art on many tasks but also the only available massively multilingual one to date.

During finetuning, we provide the goal of each article in the training set as a prompt, and train the model to generate the sequence of all the steps conditioned on the goal. Therefore, the model's behavior is similar to completing the task of inferring relevant steps and sorting them at once. At inference time, the model generates a list of steps given a goal in the test set.

5.2 Retrieval Approach: Step-Inference-Ordering Pipeline

We then implement a *Step-Inference-Ordering pipeline* for the Retrieval setting. Our pipeline contains a Step Inference model to first gather the set of desired steps, and a Step Ordering model to order the steps in the set. These models are based on our previous work (Zhang et al., 2020b). Under the hood, the models are pretrained XLM-RoBERTa (Conneau et al., 2020) or mBERT (Devlin et al., 2019) for binary classification, both state-of-the-art multilingual representations.

Our Step Inference model takes a goal and a candidate step as input, and outputs whether the candidate is indeed a step toward the goal with a confidence score. During training, for every script, its goal forms a positive example along with each of its steps. We then randomly sample 50 steps from other scripts within the same wikiHow category and pair them with the goal as negative examples. The model predicts a label for each goal-step pair with a cross-entropy loss. During evaluation, for each script in the test set, every candidate step is paired with the given goal as the model input. We then rank all candidate steps based on the model confidence scores decreasingly. Finally, the top l steps are retained, where l is the required length.

Our Step Ordering model takes a goal and two steps as input, and outputs which step happens first. During training, we sample every pair of steps in each ordered script as input to the model with a cross-entropy loss. During evaluation, we give every pair of retrieved steps as input, and count the total number of times that a step is ranked before others. We then sort all steps by this count to approximate their complete ordering.

An illustration of our Step-Inference-Ordering pipeline is shown in Figure 3. We also consider two additional variations.

Multitask Learning (MTL): The Step Inference and the Step Ordering models share the encoder layer, but have separate classifier layers. During training, the MTL system is then presented with a batch of examples from each task in an alternating fashion. During evaluation, the corresponding classifier is used.

Cross-Lingual Zero-Shot Transfer (C0): While there are abundant English training scripts, data in some other languages are scarce. Hence, we also attempt to directly evaluate the English-trained models on non-English data.

6 In-Domain Evaluation

To demonstrate the performance of models on the GOSC task, we evaluate them on our multilingual wikiHow dataset using both automatic metrics and human judgments. The ultimate utility for this task is the extent to which a human can follow the constructed steps to accomplish the given goal.

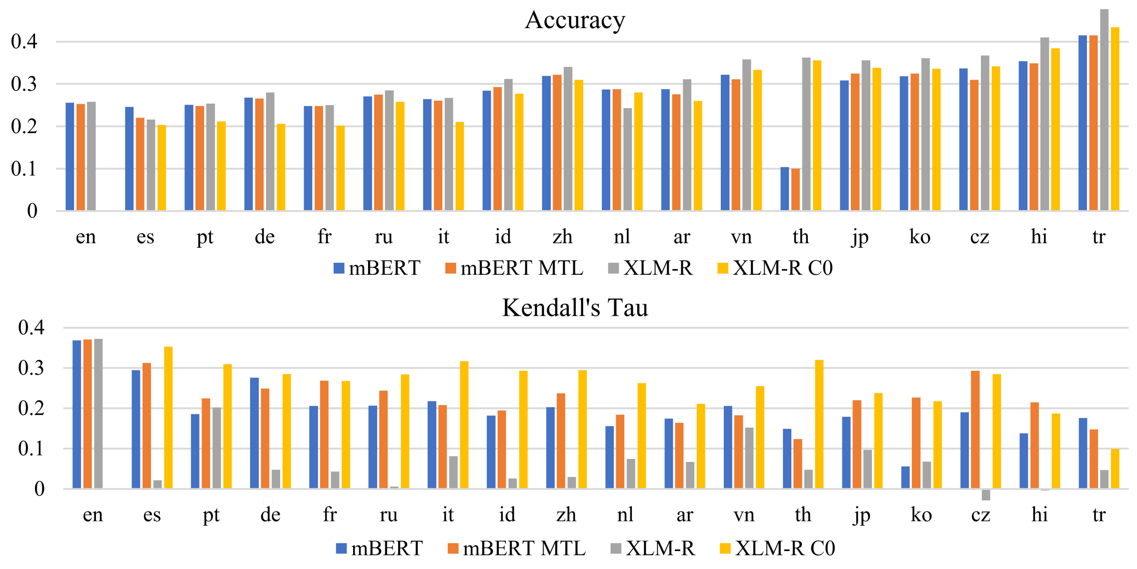


Figure 4: Detailed performance on each language from Table 2.

Lang.	en	es	pt	de	fr	ru
Perp.	17	11	24	97	46	79
Bert.	.823	.702	.682	.677	.718	.682
Lang.	it	id	zh	nl	ar	vn
Perp.	116	269	13,249	955	746	97
Bert.	.653	.692	.667	.690	.701	.695
Lang.	th	jp	ko	cz	hi	tr
Perp.	29,538	73,952	2,357	1,823	2,033	36,848
Bert.	.701	.679	.692	.682	.704	.665

Table 1: Auto evaluation results for the Generation setting (Perplexity and BERTScore F1 measure). The performance of multilingual T5 is reported.

Model	English only		Avg. all lang.s	
	Acc.	Kendall's τ	Acc.	Kendall's τ
mBERT	.256	.369	.286	.198
mBERT MTL	.253	.371	.283	.226
XLM-R	.258	.372	.317	.075
XLM-R C0	-	-	.291	.264

Table 2: Auto evaluation results for the Retrieval setting (Accuracy and Kendall's Tau). The performance of mBERT and XLM-RoBERTa, along with their multitask (MTL) and crosslingual zero-shot transfer (C0) variations, are reported [7].

As direct user studies might be costly and hard to standardize, we carefully choose measures that adhere to this utility. By default, all models are trained and evaluated on the same language.

6.1 Auto Evaluation for Generation Setting

To automatically evaluate models in the Generation Setting, we report **perplexity** and **BERTScore** (Zhang et al., 2019), as two frequently used metrics for evaluating text generation.

The mean perplexity of mT5 on the test set of each language is shown in Table 1. The results show a large range of variation. To see if perplexity correlates with the data size, we conduct a Spearman's rank correlation two-tailed test. We find a Spearman's ρ of -0.856 and a p-value of $1e-5$ between the perplexity and the number of articles in each language in our dataset; we find a Spearman's ρ of -0.669 and a p-value of $2e-4$ between the perplexity and the number of tokens in each language in the mC4 corpus where mT5 is pretrained on. These statistics suggest a significant correlation between perplexity and data size, while other typological factors are open to investigation.

Table 1 also shows the BERTScore F1 measure of the generated scripts compared against the gold scripts. Except for English (.82), the performance across different languages varies within a relatively small margin (.65 - .72). However, we notice that as a metric based on the token-level pairwise similarity, BERTScore may not be the most suitable metric to evaluate scripts. It is best designed for *aligned* texts (e.g. a machine-translated sentence and a human-translated one), whereas in scripts, certain candidate steps might not have aligned reference steps. Moreover, BERTScore does not measure whether the ordering among steps is correct.

[7]Multitask XLM-R and cross-lingual zero-shot mBERT are found to perform a lot worse and thus omitted.

To address these flaws, we further perform human evaluation in Section 6.3.

6.2 Auto Evaluation for Retrieval Setting

To automatically evaluate models in the Retrieval Setting, we first calculate **accuracy**, i.e. the percentage of predicted steps that exist in the ground-truth steps. To account for the ordering of steps, we also compute **Kendall's** τ between the overlapping steps in the prediction and the ground-truth.

The performance of our Step Inference-Ordering pipeline using mBERT and XLM-RoBERTa[8] on all 18 languages are shown in Figure 4. Complete results can be found in Appendix D. Across languages, the results are generally similar with a large room for improvement. On average, our best system constructs scripts with around 30% accuracy and around 0.2 Kendall's τ compared to the ground-truth. Compared to the baseline, our multi-task and cross-lingual zero-shot variations demonstrate significant improvement on ordering. This is especially notable in low-resource languages. For example, MTL on Korean and C0 on Thai both outperform their baseline by 0.17 on Kendall's τ.

6.3 Human Evaluation

To complement automatic evaluation, we ask 6 annotators[9] to each *edit* 30 output scripts by the Step-Inference-Ordering pipeline and mT5 in English, French, Chinese, Japanese, Korean and Hindi, respectively. The *edit* process consists of a sequence of two possible actions: either 1) delete a generated step entirely if it is irrelevant, nonsensical or not a reasonable step of the given goal, or 2) move a step somewhere else, if the order is incorrect. Then, the generated script is evaluated against the edited script in 3 aspects:
Correctness, approximated by the length (number of steps) of the edited script over that of the originally constructed script (c.f. precision);
Completeness, approximated by the length of the edited script over that of the ground-truth script (c.f. recall);
Orderliness, approximated by Kendall's τ between overlapping steps in the edited script and the generated script.[10]

[8]XLM-RoBERTa is not able to converge on the training data for Step Ordering for all but 3 languages using a large set of hyperparameter combinations.

[9]The annotators are graduate students and native or proficient speakers of the language assigned.

[10]In this formulation, the correctness and completeness of a retrieval-based model are equal, since the length of its constructed script is equal to that of the ground truth script by definition.

Retrieval: Step-Inference-Ordering pipeline						
Language	en	fr	zh	jp	ko	hi
Correctness	.70	.39	.50	.49	.45	.82
Completeness	.70	.39	.50	.49	.45	.82
Orderliness	.45	.38	.16	.12	.10	.75

Generation: mT5						
Language	en	fr	zh	jp	ko	hi
Correctness	.39	.51	.46	.40	.37	.49
Completeness	.35	.40	.46	.30	.36	.41
Orderliness	.82	.46	.60	.81	.69	.88

Table 3: Human judgments of correctness, completeness and orderliness of the output of the Step-Inference-Order pipeline and the mT5 model for the same set of 30 gold scripts, in six languages.

The results are shown in Table 3. While the constructed scripts in the Retrieval setting contain more correct steps, their ordering is significantly worse than those in the Generation setting. This suggests that the generation model is better at producing *fluent* texts, but can easily suffer from hallucination.

6.4 Qualitative Examples

To understand models' behavior, we present two representative scripts produced by the mBERT Retrieval model and the mT5 Generation model side by side, accompanied by the ground-truth script, shown in Figure 5.

The retrieved "Draw Santa Claus" script has a high step accuracy (85%), with a reasonable ordering of drawing first the outline and then details. The generation output is more off-track, hallucinating irrelevant details like "singing" and "scorpion", despite being on the general topic of drawing. It also generates more repetitive steps (e.g. the head is drawn twice), most of which are abridged.

As for "Make a Quotebook", the retrieved script has a 50% step accuracy. The third step, though not in the gold reference, is similar enough to "find some quotes", suggesting that our exact match evaluation isn't perfect. In the generated script, all steps are also generally plausible, but some essential steps are missing (e.g. find a book, find quotes). This suggests that the generation model dwells too much on the details, ignoring the big picture.

These patterns in the two scripts are common in the model outputs, a larger sample of which is included in the Supplementary Materials.

Script Name: Draw Santa Claus

Retrieval Output

1. Sketch in Santa's clothes.
2. Go back over and outline Santa's body.
3. Outline Santa's head.
4. Draw Santa's legs.
5. Draw Santa's beard.
6. Draw Santa's hat.
7. Color in your Santa.
8. Draw Santa's eyes.
9. Add Santa.
10. Give Santa some eyebrows.
11. Begin with the nose.
12. Make the nose.
13. Draw the mustache.
14. Add the arms and hands.

Generation Output (abridged)

1. Practice singing with children and ask them to draw in bells and an ear.
2. Draw the hands using slightly curved lines.
3. On the face, sketch a medium circle for Santa's head.
4. On the body for the legs, draw a slightly slanted oval for the body.
5. Draw a similar length triangle for the face, above the lines and to the right side of the base of the body.
6. Draw smaller ovals for hands in depicting a trapezoid for a larger scorpion.
7. Trace with a pen and erase unnecessary lines.
8. Color to your liking!
9. Draw another Santa Claus head.
10. …(repeated)

Gold Reference

1. Outline Santa's head.
2. Draw two large circles for the body.
3. Add the arms and hands.
4. Draw Santa's legs.
5. Begin with the nose.
6. Draw the mustache.
7. Draw Santa's eyes.
8. Give Santa some eyebrows.
9. Draw Santa's beard.
10. Draw Santa's hat.
11. Draw the mouth.
12. Go back over and outline Santa's body.
13. Sketch in Santa's clothes.
14. Color in your Santa.

Script Name: Make a Quote Book

Retrieval Output

1. Find a blank book to record your quotes in.
2. Decide how you want to write out quotes.
3. Look up quotations from popular shows, movies, and songs.
4. Unpack unusual or poetic phrases one at a time.

Generation Output (abridged)

1. Insert a quote when you need to include a word or clause in it. …
2. Add a right emphasis to highlight important words in a quote. …
3. Show your quote to others. …
4. Make sure you spell your quote correctly. …
5. Cite sourced texts in nonfiction. …

Gold Reference

1. Find a blank book to record your quotes in.
2. Find some quotes.
3. Decide how you want to write out quotes.
4. Decorate it!

Figure 5: Two example scripts constructed by our Retrieval and Generation approaches.

7 Zero-shot Transfer Learning

To show the potential of our model for transfer learning, we use the retrieval-based Step-Inference-Ordering pipeline finetuned on wikiHow to construct scripts for other datasets and domains. We quantitatively evaluate our model on 4 other script learning corpora, and qualitatively analyze some constructed scripts in a case study.

7.1 Quantitative Evaluation

Since no multilingual script data are available yet, we perform transfer learning experiments on 4 other English script corpora, OMICS (Singh et al., 2002), SMILE (Regneri et al., 2010), DeScript (Wanzare et al., 2016) [11], and the KAIROS Schema Learning Corpus (LDC2020E25). The first 3 pertain to human activities, while the last is in the military and political domain. They are all in the format of different *scenarios* (e.g. "eat in a restaurant", similar to our *goal*) each with a number of *event sequence descriptions* (ESDs, similar to our *steps*). Statistics for each corpus are in Table 4.

For each dataset, we select the ESD with the most steps for every scenario as a representative script to avoid duplication, thus converting the dataset to a GOSC evaluation set under the Retrieval setting. We then use the XLM-RoBERTa-based Step-Inference-Ordering pipeline trained on our English wikiHow dataset to directly construct scripts on each target set, and report its zero-shot performance in Table 4. We see that $30\% - 60\%$ steps are accurately retrieved, and around 40% are correctly ordered. This is close to or even better than the in-domain results on our English test set. As a comparison, a random baseline would have only 0.013 Accuracy and 0.004 τ on average. Both facts indicate that the script knowledge learned from our dataset is clearly non-trivial.

Corpus	Corpus Stats.		Results	
	Scenarios	ESDs	Acc.	Kendall's τ
SMILE	22	386	.435	.391
OMICS	175	9044	.346	.443
DeScript	40	4000	.414	.418
KAIROS	28	28	.589	.381

Table 4: The zero-shot GOSC Retrieval performance of XLM-RoBERTa finetuned on wikiHow on 4 target corpora.

7.2 Case Study: The *Bombing Attack* Scripts

To explore if the knowledge about *procedural* scripts learned from our data can also facilitate the zero-shot learning of *narrative* scripts, we present a case study in the context of the DARPA KAIROS program[12]. One objective of KAIROS is to automatically induce scripts from large-scale narrative texts, especially in the military and political domain. We show that models trained on our data

[11] The above 3 corpora are all obtained from `http://www.coli.uni-saarland.de/projects/smile/`

[12] `www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas`

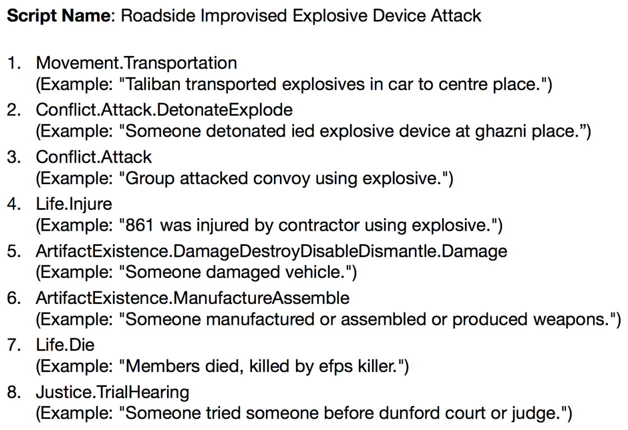
of commonplace events can effectively transfer to vastly different domains.

With the retrieval-based script construction model finetuned on wikiHow, we construct five scripts with different granularity levels under the *Improvised Explosive Device (IED) attack* scenario: “Roadside IED attack”, “Backpack IED attack”, “Drone-brone IED attack”, “Car bombing IED attack”, “IED attack”. We take the name of each script as the input *goal*, and a collection of related documents retrieved from Wikipedia and Voice of America news as data sources for extracting step candidates.

Our script construction approach has two components. First, we extract all events according to the KAIROS Event Ontology from the documents using OneIE (Lin et al., 2020). The ontology defines 68 event primitives, each represented by an *event type* and multiple *argument types*, e.g. a Damage-type event has arguments including Damager, Artifact, Place, etc. OneIE extracts all event instances of the predefined primitives from our source documents. Each event instance contains a *trigger* and several *arguments* (e.g. Trigger: “destroy”, Damager: “a bomber”, Artifact: “the building”, ...). All event instances form the candidate pool of steps for our target script.

Since the events are represented as trigger-arguments tuples, a conversion to the raw textual form is needed before inputting them into our model. This is done by automatically instantiating the corresponding event type template in the ontology with the extracted arguments. If an argument is present in the extracted instance, we directly fill it in the template; else, we fill in a placeholder word (e.g.“some”, “someone”, depending on the argument type). For example, the template of Damage-type events is “$\langle arg1 \rangle$ damaged $\langle arg2 \rangle$ using $\langle arg3 \rangle$ instrument”, which can be instantiated as “A bomber damaged the building using some instrument”). Next, we run the Step Inference-Ordering Pipeline in Section 5.2 on the candidate pool given the “goal”. The only modification is that since we don’t have a gold reference script length in this case, all retrieved steps with a confidence score higher than a threshold (default=0.95) are retained in the final script.

We manually evaluate the constructed scripts with the metrics defined in Section 6.3, except *Completeness* as we don’t have gold references. The 5 constructed scripts have an average *Correctness* of


Script Name: Roadside Improvised Explosive Device Attack

1. Movement.Transportation
   (Example: "Taliban transported explosives in car to centre place.")
2. Conflict.Attack.DetonateExplode
   (Example: "Someone detonated ied explosive device at ghazni place.")
3. Conflict.Attack
   (Example: "Group attacked convoy using explosive.")
4. Life.Injure
   (Example: "861 was injured by contractor using explosive.")
5. ArtifactExistence.DamageDestroyDisableDismantle.Damage
   (Example: "Someone damaged vehicle.")
6. ArtifactExistence.ManufactureAssemble
   (Example: "Someone manufactured or assembled or produced weapons.")
7. Life.Die
   (Example: "Members died, killed by efps killer.")
8. Justice.TrialHearing
   (Example: "Someone tried someone before dunford court or judge.")

Figure 6: An example narrative script produced by our retrieval-based pipeline trained on wikiHow. Each event is represented by its Event Type and an example sentence.

0.735 and *Orderliness* of 0.404. Despite the drastic domain shift from wikiHow to KAIROS, our model can still exploit its script knowledge to construct scripts decently. An example script, “Roadside IED attack”, is shown in Figure 6. All the steps retrieved are sensible, and most are ordered with a few exceptions (e.g. the ManufactureAssemble event should precede all others).[13]

8 Limitations

Event representation: Our representation of goals and steps as natural language sentences, though containing richer information, brings the extra difficulty in handling steps with similar meanings. For example, “change strings frequently” and “put on new strings regularly” have nearly identical meanings and both are correct steps for the goal “maintain a guitar”. Hence, both could be included by a retrieval-based model, which is not desired.
Modeling: Since GOSC is a new task, there is no previously established SOTA to compare with. We build a strong baseline for each setting, but they are clearly not the necessary or sufficient means to do the task. For example, our Step-Inference-Ordering pipeline would benefit from a paraphrasing module that eliminates semantic duplicates in retrieved steps. It also currently suffers from long run-time especially with a large pool of candidates, since it requires pairwise goal-step inference. An alternative is to filter out most irrelevant steps using similarity-based heuristics in advance.

[13]More details on the format of the script, all five constructed scripts, the event ontology, and a list of news documents used can be found in the Supplementary Materials.

Evaluation: Under the retrieval-based setting, our automatic evaluation metrics do not give credit to inexact matches as discussed above, which can also be addressed by a paraphrasing module. Meanwhile, for the generation-based setting, BERTScore, or other comparison-based metrics like BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014), may not be the most suitable metric to evaluate scripts. They are best designed for *aligned* texts like translation pairs, and do not measure whether the ordering among steps is correct. While we complement it with manual evaluation, only one human annotator is recruited for each language, resulting in potential subjectivity. Alternatively, crowdsourcing-based evaluation is costly and hard to standardize. Due to the complexity of the GOSC task and its evaluation, we suggest that future work investigate better means of evaluation.

9 Conclusion and Future Work

We propose the first multilingual script learning dataset and the first task to evaluate the holistic understanding of scripts. By comprehensively evaluating model performances automatically and manually, we show that state-of-the-art models can produce complete scripts both in- and out-of-domain, with a large room for improvement. Future work should investigate additional aspects of scripts, such as usefulness, granularity, etc., as well as their utility for downstream tasks that require automated reasoning.

Acknowledgments

This research is based upon work supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), and the IARPA BETTER Program (contract 2019-19051600004). Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, IARPA, or the U.S. Government.

Special thanks go to our annotators: Artemis Panagopoulou, Daniel Joongwon Kim, Simmi Mourya, and Liam Dugan. We also thank Daphne Ippolito for the help on mT5; Ying Lin, Sha Li, Zhenhailong Wang, Pengfei Yu, Tuan Lai, Haoyang Wen, and Heng Ji for providing the source documents and the OneIE results for our case study. Finally, we appreciate the valuable feedback from the anonymous reviewers.

References

Robert Abelson and Roger C Schank. 1977. Scripts, plans, goals and understanding. *An inquiry into human knowledge structures New Jersey*, 10.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

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.

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. 2017. Distilling task knowledge from how-to communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 805–814.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Edward A Feigenbaum, Avron Barr, and Paul R Cohen. 1981. The handbook of artificial intelligence.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–57.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344.

Yuchul Jung, Jihee Ryu, Kyung-min Kim, and Sung-Hyon Myaeng. 2010. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2-3):110–124.

Mahnaz Koupaee and William Yang Wang. 2018. WikiHow: A large scale text summarization dataset. *ArXiv*, abs/1810.09305.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. 2020. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048, Online. Association for Computational Linguistics.

Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O Riedl. 2012. Crowdsourcing narrative intelligence. *Advances in Cognitive systems*, 2(1).

Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009.

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.

Dena Mujtaba and Nihar Mahapatra. 2019. Recent trends in natural language understanding for procedural knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424. IEEE.

Dai Quoc Nguyen, Dat Quoc Nguyen, Cuong Xuan Chu, Stefan Thater, and Manfred Pinkal. 2017. Sequence to sequence learning for event prediction. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 37–42, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Simon Ostermann, Michael Roth, Stefan Thater, and Manfred Pinkal. 2017. Aligning script events with narrative texts. *arXiv preprint arXiv:1710.05709*.

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.

Hogun Park and Hamid Reza Motahari Nezhad. 2018. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *Companion Proceedings of the The Web Conference 2018*, pages 351–358.

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 Mooney. 2016a. Statistical script learning with recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 11–16.

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

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

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.

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.

David E Rumelhart. 1975. Notes on a schema for stories. In *Representation and understanding*, pages 211–236. Elsevier.

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 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. In *Proceedings of the tenth international conference on language resources and evaluation (LREC'16)*, pages 3494–3501.

Haoyang Wen, Ying Lin, Tuan Lai, Xiaoman Pan, Sha Li, Xudong Lin, Ben Zhou, Manling Li, Haoyu Wang, Hongming Zhang, Xiaodong Yu, Alexander Dong, Zhenhailong Wang, Yi Fung, Piyush Mishra, Qing Lyu, Dídac Surís, Brian Chen, Susan Windisch Brown, Martha Palmer, Chris Callison-Burch, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, and Heng Ji. 2021. RESIN: A dockerized schema-guided cross-document cross-lingual cross-media information extraction and event tracking system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 133–143, Online. Association for Computational Linguistics.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: Controllable story generation with external knowledge using large-scale language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mT5: A massively multilingual pre-trained text-to-text transformer.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. *arXiv preprint arXiv:1809.00812*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020a. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1541–1550, Online. Association for Computational Linguistics.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020b. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

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 Corpus Statistics

Table 5 shows the statistics of our multilingual wikiHow script corpus.

B Evaluation Details

In Section 3, we formalize the Goal-Oriented Script Construction (GOSC) task as follows: Given a *goal* g, the model is asked to construct a complete script as an ordered list of *steps* S, with a ground-truth reference T. As a hint of the desired level of granularity, we also provide an expected number of steps (or length of the script), l, as input.

In the **Retrieval setting**, a set of candidate steps C is also available. We evaluate an output script from two angles: content and ordering.

First, we calculate the accuracy, namely the percentage of predicted steps that exist in the ground-truth. Denote s_i as the i-th step in S.

$$\text{acc} = (\sum_{i}^{l}[s_i \in T])/l$$

If the gold script is ordered, we further evaluate the ordering of the constructed script by calculating Kendall's τ between the intersection of the predicted steps and the ground-truth steps.

$$\tau = \frac{NC(S \cap T, T \cap S) - ND(S \cap T, T \cap S)}{\binom{l}{2}}$$

Language	en	es	pt	de	fr	ru	it	id	zh
Num. articles	112,111	64,725	34,194	31,541	26,309	26,222	22,553	21,014	14,725
Num. ordered articles	54,852	26,620	7,408	10,681	6,834	5,335	4,308	5,536	4,784
Avg. num. of sections / article	2.5	2.6	3.1	2.7	2.9	3.1	3.1	2.9	2.8
Avg. num. of steps / article	13.7	14.4	16.1	15.1	15.7	16.4	16.2	16.0	15.4
Num. of articles for train/dev	100,900	58,253	30,775	28,387	23,679	23,600	20,298	18,913	13,253
Num. of articles for test	11,211	6,472	3,419	3,154	2,630	2,622	2,255	2,101	1,472

Language	nl	ar	vn	th	jp	ko	cz	hi	tr
Num. articles	13,343	12,157	6,949	5,821	5,567	5,179	5,043	3,104	1,434
Num. ordered articles	2,113	2,567	1,157	1,244	1,209	917	920	888	520
Avg. num. of sections / article	3.1	3.0	3.2	3.1	3.1	3.2	3.1	3.0	3.2
Avg. num. of steps / article	16.2	16.4	17.1	17.7	16.8	17.5	16.4	16.8	19.2
Num. of articles for train/dev	12,009	10,942	6,255	5,239	5,011	4,662	4,539	2,794	1,291
Num. of articles for test	1,334	1,215	694	582	556	517	504	310	143

Table 5: Statistics of our multilingual wikiHow corpus by language, ordered by the number of articles in each language. Each article is converted to a script, including all steps from all sections.

where NC is the number of concordant pairs, ND the number of discordant pairs; $A \cap B$ is used as a special notation for the intersection of ordered lists, denoting elements that appear in both A and B, in the order of A.

It is likely that a model includes two modules: a retrieval module and an ordering module. In this case, it is sensible to separately evaluate these two modules.

To evaluate the retrieval module independently, assume that the model retrieves a large set of steps R ranked by their *relevance* to the goal g. Denote r_i as the i-th step in R. We calculate recall and normalized discounted cumulative gain[14] at position k. Assume $k > l$.

$$recall_k = (\sum_i^k [r_i \in T])/k$$

$$NDCG_k = \frac{\sum_{i=1}^{k} \frac{2^{[r_i \in T]}-1}{\log_2(i+1)}}{\sum_{i=1}^{l} \frac{2^1-1}{\log_2(i+1)} + \sum_{i=l+1}^{k} \frac{2^0-1}{\log_2(i+1)}}$$

To evaluate the ordering module independently, we directly give the model the set of ground-truth steps to predict an ordering. We again use Kendall's τ to evaluate the ordered steps.

$$\tau = \frac{NC(T', T) - ND(T', T)}{\binom{l}{2}}$$

where T' is the set ground-truth steps ordered by the model.

In the **Generation** setting, a model is evaluated using perplexity on the test set, following standard practice.

$$perplexity(S) = \exp\left(-\frac{L(S)}{\text{count of tokens in } S}\right)$$

where $L(S)$ is the log-likelihood of the sequence of steps assigned by the model.

When evaluating a model on multiple scripts, all aforementioned metrics are averaged.

C Modeling Details

All our models are implemented using the HuggingFace Transformer service[15]. For all experiments, we hold out 5% of the training data for development.

The pretrained models we use include: the `bert-base-multilingual-uncased` checkpoint (168M parameters) for mBERT, the `xlm-roberta-base` checkpoint (270M parameters) for XLM-RoBERTa, the `roberta-base` checkpoint (125M parameters) for RoBERTa[16], and the `mT5-Large` checkpoint (1B parameters) for mT5[17].

For mBERT, XLM-RoBERTa and RoBERTa, we finetune the pretrained models on our dataset using the standard `SequenceClassification` pipeline on HuggingFace[18]. For mT5, we refer to the offi-

[14]We set the true relevance of each predicted step as 1 if it exists in the ground-truth steps, and 0 otherwise.

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

[16]The above 3 models are available at https://huggingface.co/transformers/pretrained_models.html

[17]https://github.com/google-research/multilingual-t5

[18]https://huggingface.co/transformers/model_doc/auto.html?highlight=sequence%20classification#transformers.AutoModelForSequenceClassification

Lang.	en	es	pt	de	fr	ru
Acc.	.253	.220	.248	.266	.248	**.275**
τ	**.371**	**.313**	**.225**	**.249**	**.269**	.244
Lang.	it	id	zh	nl	ar	vn
Acc.	.261	**.293**	**.322**	**.288**	.276	.311
τ	.208	**.195**	**.237**	**.184**	.164	.183
Lang.	th	jp	ko	cz	hi	tr
Acc.	.100	**.325**	**.325**	.310	.349	.415
τ	.124	**.220**	**.227**	**.293**	**.215**	.148

Table 6: The GOSC Retrieval performance of multitask learning mBERT. Results higher than those produced by the single-task mBERT are in bold.

cial finetuning scripts[19] from the project's Github repository.

For each in-domain evaluation experiment, we perform grid search on learning rate from $1e-5$ to $5e-8$, batch size from 16 to 128 whenever possible, and the number of epochs from 3 to 10. As mBERT and XLM-RoBERTa have a large number of hyperparameters, most of which remain default, we do not list them here. Instead, the hyperparameter values and pretrained models will be available publicly via HuggingFace model sharing. We choose the model with the highest validation performance to be evaluated on the test set. For the Retrieval setting, we consider the accuracy of contracted scripts; for the Generation setting, we consider perplexity.

We run our experiments on an NVIDIA GeForce RTX 2080 Ti GPU, with half-precision floating point format (FP16) with O1 optimization. The experiments in the Retrieval setting take 3 hours to 5 days in the worst case for all languages. The experiments in the Generation setting take 2 hours to 1 day in the worst case for all languages.

D Additional Results

Our complete in-domain evaluation results can be found in Table 6, 7, and 8.

E More Qualitative Examples

Aside from the examples shown in Section 6.4, we show 2 more example scripts constructed by the mBERT baseline under the Retrieval setting in Section 5.2 vs. those by the mT5 baseline under the Generation setting in Section 5.1. For each script name, the Retrieval output and the Generation output are shown side by side. Please see Figure 7 and 8 for English examples, and Figure 9 and 10 for Chinese ones.

For more examples, please see the Supplementary Materials. We include 20 examples for each language for the in-domain evaluation, and all 5 examples for the out-of-domain case study on the *Bombing Attack* scenario.

[19] https://colab.research.google.com/github/google-research/text-to-text-transfer-transformer/blob/master/notebooks/t5-trivia.ipynb

Lang.	Step Retrieval				Ordering	Script Construction	
	Recall@25	Recall@50	NDCG@25	NDCG@50	Kendall's τ	Accuracy	Kendall's τ
en	.337 / .342	.424 / .429	.660 / .660	.648 / .648	.368 / .375	.256 / .258	.369 / .372
es	.319 / .397	.403 / .786	.653 / .532	.642 / .571	.321 / .022	.246 / .216	.295 / .022
pt	.313 / .319	.401 / .412	.679 / .672	.664 / .659	.207 / .212	.251 / .254	.186 / .202
de	.337 / .350	.421 / .438	.687 / .707	.676 / .692	.260 / .026	.268 / .280	.276 / .048
fr	.315 / .320	.405 / .411	.673 / .672	.661 / .659	.244 / .020	.248 / .250	.206 / .043
ru	.336 / .353	.423 / .446	.701 / .715	.688 / .701	.181 / .042	.271 / .285	.207 / .006
it	.332 / .333	.424 / .431	.700 / .705	.686 / .687	.184 / .035	.264 / .267	.218 / .081
id	.351 / .383	.435 / .480	.712 / .744	.699 / .725	.190 / .011	.284 / .312	.182 / .026
zh	.401 / .429	.498 / .536	.750 / .753	.732 / .737	.260 / .027	.319 / .340	.203 / .030
nl	.354 / .382	.447 / .758	.721 / .546	.708 / .597	.179 / .011	.287 / .243	.156 / .075
ar	.351 / .381	.447 / .485	.710 / .735	.694 / .717	.161 / .055	.288 / .311	.175 / .067
vn	.381 / .436	.464 / .544	.769 / .784	.753 / .766	.170 / .171	.322 / .358	.206 / .152
th	.146 / .448	.273 / .566	.330 / .784	.369 / .764	.106 / .056	.104 / .362	.149 / .048
jp	.383 / .447	.487 / .579	.754 / .766	.732 / .751	.170 / .107	.308 / .356	.179 / .097
ko	.381 / .435	.474 / .553	.762 / .780	.744 / .766	.154 / .044	.318 / .361	.056 / .068
cz	.416 / .456	.532 / .582	.772 / .776	.751 / .758	.211 / -.007	.337 / .367	.190 / -.028
hi	.421 / .484	.530 / .610	.782 / .814	.763 / .798	.156 / .029	.354 / .410	.138 / -.004
tr	.509 / .577	.676 / .718	.859 / .881	.829 / .854	.154 / .014	.415 / .477	.176 / .047
Mean	.355 / .404	.454 / .542	.704 / .724	.691 / .714	.204 / .069	.286 / .317	.198 / .075

Table 7: The GOSC Retrieval performance of mBERT and XLM-RoBERTa, divided by a slash in each cell. Both the performance of individual modules and that of script construction are reported.

Lang.	Step Retrieval				Ordering	Script Construction	
	Recall@25	Recall@50	NDCG@25	NDCG@50	Kendall's τ	Accuracy	Kendall's τ
es	0.270	0.338	0.567	0.564	0.360	0.203	0.353
pt	0.265	0.339	0.595	0.590	0.276	0.212	0.310
de	0.271	0.346	0.556	0.558	0.264	0.206	0.285
fr	0.253	0.319	0.585	0.580	0.283	0.202	0.268
ru	0.313	0.390	0.672	0.661	0.252	0.258	0.284
it	0.264	0.338	0.575	0.574	0.268	0.210	0.317
id	0.338	0.424	0.681	0.670	0.321	0.277	0.293
zh	0.379	0.471	0.718	0.706	0.318	0.310	0.295
nl	0.340	0.424	0.684	0.673	0.280	0.280	0.263
ar	0.319	0.400	0.643	0.635	0.235	0.260	0.211
vn	0.392	0.480	0.748	0.733	0.249	0.333	0.255
th	0.418	0.520	0.771	0.753	0.307	0.356	0.320
jp	0.403	0.512	0.751	0.733	0.232	0.338	0.238
ko	0.391	0.485	0.767	0.749	0.182	0.336	0.218
cz	0.406	0.519	0.749	0.732	0.300	0.342	0.285
hi	0.449	0.557	0.770	0.754	0.205	0.384	0.187
tr	0.505	0.646	0.805	0.787	0.167	0.434	0.099
Mean	0.357	0.448	0.692	0.681	0.259	0.296	0.258

Table 8: The GOSC Retrieval performance of XLM-RoBERTa using cross-lingual zero-shot transfer trained on the English data. Both the performance of individual modules and that of script construction are reported.

Script Name: Ice an Injured Ankle

Retrieval Output

1. Place an ice pack on the injured ankle for 15–20 minutes.
2. Rest the injured ankle as much as possible for 48 hours.
3. Compress the injured ankle for 48 hours to prevent swelling.
4. Elevate the injured ankle above your heart to decrease any swelling.
5. Go to the emergency room if the injured ankle cannot bear weight.
6. Apply a compression bandage to a sprained ankle.
7. Consult with a doctor if the ankle does not improve after 2–3 days.
8. Ask a friend to grab your ankles from the bottom.

Generation Output

1. Hang the injured ankle up in a large, open space.
2. Wrap the injured ankle up in a heated towel. Pad your ankle with a bandage.
3. Put the ice pack against your leg for 15-20 minutes.
4. Elevate your leg above the level of irritation.
5. Take an over-the-counter (OTC) pain reliever.
6. See a doctor if 2-3 days persist or worsen.
7. Keep your foot elevated to avoid further swelling.
8. Use an orthopedic shoes to trim your ankle.
9. Wear the right shoes and footwear.

Gold Reference

1. Rest the injured ankle as much as possible for 48 hours.
2. Place an ice pack on the injured ankle for 15–20 minutes.
3. Apply ice 4–8 times daily for the first 48 hours following the injury.
4. Compress the injured ankle for 48 hours to prevent swelling.
5. Elevate the injured ankle above your heart to decrease any swelling.
6. Consult with a doctor if the ankle does not improve after 2–3 days.
7. Take NSAID pain relievers to help with swelling and manage the pain.
8. Go to the emergency room if the injured ankle cannot bear weight.

Figure 7: The "Ice an injured Ankle" script.

Script Name: Solve a Simple Linear Inequality

Retrieval Output

1. Understand the inequality signs.
2. Move the variable to one side of the inequality.
3. Approach the inequality as you would an equation.
4. Combine like terms, or otherwise simplify the inequality.
5. Solve this inequality: 3x+2<–3x+6+5x.
6. Reverse the inequality sign whenever you take the reciprocal of both sides.
7. Solve this inequality in which you need to reverse the sign: –6x–18>12.
8. Solve this compound inequality: 14<2x+4<22.
9. Reverse the inequality sign whenever you multiply or divide by a negative number.
10. Write, then, equation 3 as (see the image above)

Generation Output

1. Recognize that the line is not in the standard form of two numbers plus one.
2. Identify the product of the linear inequality.
3. Find the product of the linear inequality.
4. Do this for all the products of the linear inequality.
5. Substitute in our terms for a product that is in the standard form of two.
6. Choose a more economic alternative to a simple inequality and make them distasteful for your own values.
7. Read the problem and consider solving it for truth.
8. Check your work.
9. Take a break if you made a mistake.

Gold Reference

1. Understand the inequality signs.
2. Combine like terms, or otherwise simplify the inequality.
3. Move the variable to one side of the inequality.
4. Isolate the variable.
5. Approach the inequality as you would an equation.
6. Reverse the inequality sign whenever you multiply or divide by a negative number.
7. Reverse the inequality sign whenever you take the reciprocal of both sides.
8. Solve this inequality: 3x+2<–3x+6+5x.
9. Solve this inequality in which you need to reverse the sign: –6x–18>12.
10. Solve this compound inequality: 14<2x+4<22.

Figure 8: The "Solve a Simple Linear Inequality" script.

Script Name: 竞选美国总统 (Run for the US President)

Retrieval Output

1. 总统必须满足在美国本土出生、至少年满三十五岁、在美国至少已居住十四年以上的条件。
2. 设立总统竞选考察委员会。
3. 让自己出现在候选人列表上。
4. 注册成为正式候选人。
5. 开始竞选。
6. 创建竞选纲领。
7. 建立自己的竞选平台。
8. 你必须提高你的实力。
9. 排序。
10. 考虑组织的愿景。
11. 教英语。
12. 用“领导”代替“管理”。
13. 接受教育。
14. 签订合约。

Generation Output

1. 证明自己是合法出生于美国的公民。
2. 年满35岁。
3. 参加竞选前必须至少在美国连续居住14年。
4. 学习美国总统史。
5. 与家人及支持者仔细商讨。
6. 筹备竞选研究委员会。
7. 到联邦选举委员会(FDC)登记。
8. 公开声明参选。
9. 募款。
10. 走近平民百姓。 参加初选及党团会议并胜出, 或获得多数州代表支持。 参加你所属政党的全国代表大会。
11. 参加大选。
12. 赢得大选。

Gold Reference

1. 总统必须满足在美国本土出生、至少年满三十五岁、在美国至少已居住十四年以上的条件。
2. 打造一个好形象。
3. 明确肢体语言。
4. 丰富自己的经历。
5. 广泛交友。
6. 注册成为正式候选人。
7. 让自己出现在候选人列表上。
8. 设立总统竞选考察委员会。
9. 建立自己的竞选平台。
10. 对时事有一个清晰的认识。
11. 创建竞选纲领。
12. 开始竞选。
13. 磨练辩论技巧。
14. 做好一切准备。

Retrieval Output (Translated)

1. The president must be born in the United States, be at least 35 years old, and have lived in the United States for at least 14 years.
2. Establish a presidential campaign inspection committee.
3. Let yourself appear on the candidate list.
4. Register as an official candidate.
5. Start the election.
6. Create a manifesto.
7. Build your own campaign platform.
8. You must improve your capabilities.
9. Sort.
10. Consider the organization's vision.
11. Teach English.
12. Replace "management" with "leadership".
13. Get an education.
14. Sign a contract.

Generation Output (Translated)

1. Prove that you are a citizen legally born in the United States.
2. You must be at least 35 years old.
3. You must have lived in the United States for at least 14 consecutive years before participating in the election.
4. Learn about the history of American presidents.
5. Discuss carefully with family and supporters.
6. Prepare the campaign research committee.
7. Register with the Federal Election Commission (FDC).
8. Publicly declare to participate in the election.
9. Fundraising.
10. Approach the common people. Participate in primary elections and party caucus meetings and win, or get the support of most state representatives. Attend the national convention of your political party.
11. Participate in the general election.
12. Win the election.

Gold Reference (Translated)

1. The president must be born in the United States, be at least 35 years old, and have lived in the United States for at least 14 years.
2. Create a good image.
3. Use clear body language.
4. Enrich your experience.
5. Make friends widely.
6. Register as an official candidate.
7. Let yourself appear on the candidate list.
8. Establish a presidential campaign inspection committee.
9. Build your own campaign platform.
10. Have a clear understanding of current events.
11. Create a manifesto.
12. Start the election.
13. Hone your debate skills.
14. Be prepared for everything.

Figure 9: The “Run for the US President” script.

Script Name: 弹奏钢琴 (Play the Piano)

Retrieval Output

1. 前面我们讨论的都是C调音阶，它由9个音符组成，从C开始，全部对应的是钢琴的白键。
2. 认识琴的品位。
3. 可以考虑从钢琴入手。
4. 学习键盘。
5. 练习手移动弹奏。
6. 学习和弦。
7. 钢琴或键盘乐器定弦。
8. 学习弹奏乐谱。
9. 获得一架钢琴。
10. 调音管定弦。
11. 使用参照设备定弦。
12. 弹奏一些音符。
13. 学会使用踏板。

Generation Output

1. 你自己要懂琴谱,并且会弹钢琴。 计划一个星期教几节课,每节课时长多少,怎么收费。 决定上课地点。
2. 找学生。
3. 计划上课内容。
4. 上第一节课。
5. 多鼓励你的学生。
6. 加入当地、省际或国际的音乐教师组织。
7. 参加培训。
8. 对于年龄小的学生,你可以建立一个奖励机制。

Gold Reference

1. 获得一架钢琴。
2. 购买配件。
3. 准备正确的姿势。
4. 再次检查你的手部姿势。
5. 学习键盘。
6. 弹奏一些音符。
7. 指法练习。
8. 学习和弦。
9. 学习弹奏乐谱。
10. 单手练习。
11. 练习手移动弹奏。
12. 学会使用踏板。
13. 多练习。

Retrieval Output (Translated)

1. What we discussed earlier is the C scale, which consists of 9 notes. Starting from C, all correspond to the white keys of the piano.
2. Know the capos.
3. Consider starting with the piano.
4. Learn the keyboard.
5. Practice hand movement.
6. Learn chords.
7. Piano or keyboard instrument tuning.
8. Learn to play sheet music.
9. Get a piano.
10. Tuning tube tuning.
11. Use reference equipment to tune.
12. Play some notes.
13. Learn to use the pedal.

Generation Output (Translated)

1. You need to know sheet music yourself and be able to play the piano. Plan how many classes to teach each week, how long each class will be, and how much will be charged. Decide where the class will take place.
2. Find students.
3. Plan class content.
4. Give the first class.
5. Encourage your students.
6. Join a local, inter-provincial or international music teacher organization.
7. Get some training.
8. For younger students, you can establish a reward mechanism.

Gold Reference (Translated)

1. Get a piano.
2. Purchase accessories.
3. Prepare in the correct posture.
4. Check your hand posture again.
5. Learn the keyboard.
6. Play some notes.
7. Fingering practice.
8. Learn chords.
9. Learn to play sheet music.
10. Practice with one hand.
11. Practice hand movement.
12. Learn to use the pedal.
13. practice more.

Figure 10: The “Play the Piano” script.