Benchmarking Hierarchical Script Knowledge

Yonatan Bisk[1] **Jan Buys**[1] **Karl Pichotta**[*] **Yejin Choi**[1,2]
[1]Paul G. Allen School of Computer Science and Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence
{ybisk, jbuys}@cs.washington.edu

Abstract

Understanding procedural language requires reasoning about both hierarchical and temporal relations between events. For example, "boiling pasta" is a *sub-event* of "making a pasta dish", typically happens *before* "draining pasta," and *requires* the use of omitted tools (e.g. a strainer, sink...). While people are able to choose when and how to use abstract versus concrete instructions, the NLP community lacks corpora and tasks for evaluating if our models can do the same. In this paper, we introduce **KIDSCOOK**, a parallel script corpus, as well as a cloze task which matches video captions with missing procedural details. Experimental results show that state-of-the-art models struggle at this task, which requires inducing functional commonsense knowledge not explicitly stated in text.

1 Introduction

The level of detail used in natural language communication varies: descriptive or instructive text for experts may elide over details the reader can seamlessly infer, while text for more novice audiences may be more verbose. A given document typically adheres to a single level of verbosity suited to its presumed audience (Grice, 1975), so learning correspondences between abstract and detailed descriptions of similar concepts from text is a challenging problem.

Commonsense knowledge of how complex events decompose into stereotypical sequences of simpler events is a necessary component of a system that can automatically understand and reason about different types of discourse. Hierarchical correspondences between abstract and detailed representations of concepts and events were an important aspect of the original formulation of scripts for natural language understanding (Schank and Abelson, 1977; DeJong, 1981) but required handwritten data structures encoding world knowledge. However, the automatic induction of such commonsense knowledge from open-domain noisy text corpora remains an open problem (Chambers, 2013; Weber et al., 2018; Zellers et al., 2018). As a step towards solving this problem we consider textual descriptions of actions in a cooking domain.

[*]Author now at Google. Work done while unaffiliated.

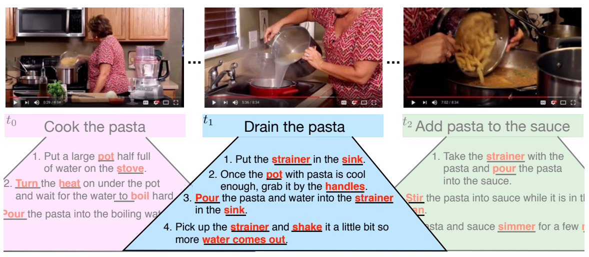

t_0 Cook the pasta
1. Put a large pot half full of water on the stove.
2. Turn the heat on under the pot and wait for the water to boil hard.
Pour the pasta into the boiling wat
t_1 Drain the pasta
1. Put the strainer in the sink.
2. Once the pot with pasta is cool enough, grab it by the handles.
3. Pour the pasta and water into the strainer in the sink.
4. Pick up the strainer and shake it a little bit so more water comes out.
t_2 Add pasta to the sauce
1. Take the strainer with the pasta and pour the pasta into the sauce.
Stir the pasta into sauce while it is in t
an.
sta and sauce simmer for a few

Figure 1: An example **KIDSCOOK** sequence with multiple types of hierarchy and abstraction: the example contains sequences of complex instructions, given both as sentences and sequences of simpler instructions.

We introduce a dataset, **KIDSCOOK**, targeted at exploring the automatic acquisition of correspondences between abstract and concrete descriptions of actions. The dataset consists of higher-level single-sentence imperative descriptions paired with lower-level descriptions with elided details included. Descriptions come from real grounded actions, built on top of the YouCookII video caption dataset (Zhou et al., 2017).

Figure 1 gives an example annotation from the dataset: the phrase "drain the pasta," presented to an annotator with its corresponding video clip, was annotated as corresponding to four constituent steps appropriate as instruction for a child. The constituent steps are "simpler" in the sense that they correspond to more atomic actions, but not necessarily in their linguistic complexity. We identify over 1,500 procedures and tools which **KIDSCOOK** makes explicit but are assumed as commonsense world knowledge by YouCookII.

The **KidsCook** dataset allows us to learn mappings between abstract and concrete descriptions via sequence-to-sequence prediction. We apply several standard neural sequence-to-sequence models; however, since these models do not expose explicit, interpretable correspondences between abstract and concrete descriptions, we also propose the application of neural transduction models which capture correspondences with latent hard alignment variables. We define a cloze-style evaluation to complement our dataset, in which models must predict the values of held-out tokens which target knowledge of tool usage, temporal ordering, and kitchen commonsense. We find that our neural transduction models are able to match the predictive power of traditional neural sequence models while providing interpretable alignments between abstract and concrete subsequences useful for our primary goal of analysis of implicit hierarchical script knowledge.

2 Data & Task

Our approach situates script learning as a case of grounding. For simplicity of exposition, let us assume there are three levels of abstraction to grounding: *abstract* $\rightarrow$ *concrete* $\rightarrow$ *motor control*. Most prior work in grounding treats language monolithically[1] and ignores the issue of audience. In practice, this means the task formulation or exposed API may implicitly bias the language to be more concrete. By viewing the task as purely linguistic, we have no API or robot that constrains our language; instead, we define our audience as children. By eliciting child-directed instructions, we collect concrete language capturing otherwise implicit world knowledge that a child would not know. Because annotators assume a smart and capable but uninformed listener, we posit this language corresponds closely to the most "concrete" form in which language naturally occurs.

2.1 Data Collection

We construct a task on Amazon's Mechanical Turk, where workers are asked to explain a video action caption to a child.[2] Every instruction is paired with the original YouTube video and YouCook caption so the annotator could see how the action was performed, rather than hallucinating additional details. All captions received three simplifications. The instructions ask users to focus on missing information and allow them up to five steps. Finally, we explicitly asked annotators to simplify complex actions (e.g. *dice*) that can be defined by a series of more basic actions (e.g. *cut*).

[1]Notable exceptions include the hierarchical instructions of (Regneri et al., 2013) and (Bisk et al., 2016).

[2]Pay was calculated based on \$15/hr and assuming workers took 1.5x as long to complete a task as the experimenters.

				Avg Len	
	Seqs	Tokens	Vocab	YC	**KC**
Train	8,125	307,573	3,573	10.0	37.9
Valid	1,014	36,830	1,479	8.8	36.3
Test	1,020	37,156	1,489	8.8	36.4

Table 1: **KidsCook** corpus statistics

Our **KidsCook** corpus statistics are shown in Table 1. In total we collected over 10K action sequences (~400K tokens). The average caption is approximately 4x longer than a YouCook caption. Most importantly 1,536 lemmas and 2,316 lexical types from **KidsCook**'s vocabulary do not appear in any of the original captions. This indicates that there are over 1,500 new concepts, tools, and procedures that were assumed by YouCookII but are now explicit in **KidsCook**.

2.2 Cloze Task

To investigate what new knowledge is being introduced and whether a model has captured it, we construct a cloze-style slot-filling task (Chambers, 2017; Hermann et al., 2015). We drop key content words from the concrete realization of an abstract instruction and ask the model to predict them. Several examples from the validation set are shown in Table 2. Correctly predicting the missing words requires knowledge of the manner of executing a task and the tools required.

To choose candidate words to drop, we only allow words that occur primarily in the concrete instructions. Additionally, we do not drop stop words, numbers, or words occurring fewer than five times. We do, however, drop units of measure (*cup*, *minute*, etc.). This ensures we create blanks whose answers are previously omitted concrete details. Relatedly, under this filter the answer to a blank is very rarely an ingredient, as our goal is not to memorize recipes, but to infer the tool knowledge necessary to execute them. In total, we whitelist ~1,000 words that can be dropped to create blanks. We prefer longer blanks when available to give preference to compound nouns (e.g. *wire whisk*). Finally, we do not drop any words

ABS	chop garlic into small pieces .
CON	put garlic on **cutting board**. **press** on back of **knife** with **hand**, **cutting** into small pieces.
ABS	add some parmesan cheese into the bowl and mix them well.
CON	use a **grater** to **grate** some parmesan cheese into the bowl. use a **wire whisk** to **stir** the cheese in.
ABS	add the tofu to the wok.
CON	drain the **water** from the tofu using a **strainer**. add the tofu into the **pan**. use a **spoon** to **stir** the tofu in the mixture.

Table 2: Example abstract/concrete pairs with blanks (**red**) where predictions and surprisal are computed.

from the concrete sentence if they occur in the abstract description. This restriction eliminates any benefits that might have been achieved via models with copy mechanisms. Examples that do not meet our criteria are removed from the corpus.

3 Models

We investigate the utility of sequence-to-sequence models with attention (Bahdanau et al., 2015) to generate concrete realizations of abstract task descriptions. We hypothesize that models that learn explicit alignments are particularly amenable to interpretable analysis on the task. Therefore, in addition to using the global attention model of (Luong et al., 2015), we adapt the transducer model proposed by Yu et al. (2016), which uses learned latent discrete variables to model phrase-to-phrase alignments. In contrast to many standard neural models, this approach enables us to incorporate prior knowledge about the alignment structure, and to extract interpretable alignments between task phrases. Closely related architectures have been proposed for segmental sequence modeling (Wang et al., 2017) and phrase-based neural machine translation (Huang et al., 2018).

We train the transducer models using Viterbi EM (after doing marginal likelihood training for the initial iterations), as we found it gave higher predictive accuracy than marginal likelihood training only. Following Yu et al. (2016) we experiment with both a fixed alignment transition probability model and a transition model with a neural parameterization. Cloze task prediction is performed greedily.[3] At each slot the Viterbi alignment of the prefix of the sequence up to that slot is computed. See appendix 7 for model details.[4] We also evaluate the performance of a language modelling baseline and a seq2seq model without attention (Sutskever et al., 2014), to compare the effect of not modeling alignment at all.

We expect all the models to implicitly capture aspects of world knowledge. However, the discrete latent variable models provide Viterbi alignments over the training data, from which we can compile a look-up table with the extracted knowledge. In neural attention models, this knowledge is only weakly recoverable: extracting information requires hand tuning attention thresholds and there is no direct way to extract contiguous alignments for multi-word phrases.

4 Results

4.1 Evaluation Metrics

During generation, we provide the model with the number of words in each blank to be predicted. We consider two setups for evaluating examples with multiple blanks, both assuming that predictions are made left-to-right: Oracle, where the gold prediction of each blank is fed into the model to condition on for future predictions, and Greedy, where the model prediction is used for future predictions. We compute the proportion of exact word matches over each blank and the precision of the top $k = 5$ predictions for both setups. Additionally we compute the average surprisal of the gold prediction (conditioning on oracle predictions). The *surprisal* of a word (Attneave, 1959; Hale, 2001) is its negative log probability under the model: $-log(P(w_i|w_{1:i-1}))$. The higher the probability of the ground truth, the lower the model's "surprise" at seeing it in that context.

Finally, as a quantitative proxy for interpretability, we report the length of the transducer models' average Viterbi alignment span: our goal is a model which balances low average alignment lengths and high matching or ranking scores.

4.2 Cloze Task Results

We report results on the prediction task in Table 4. First we consider models trained only on our dataset: All the models that incorporate a notion of alignment do substantially better than those who

[3]During preliminary experiments beam search did not improve performance.

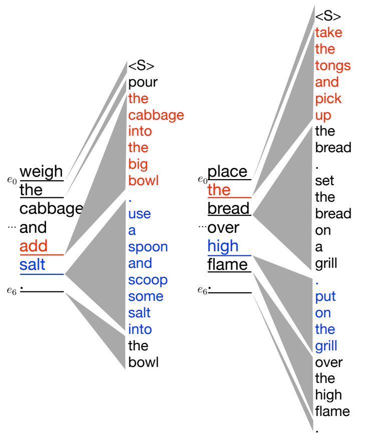
[4]All code and data is available at `https://github.com/janmbuys/ScriptTransduction`.

abstract → concrete		**concrete → abstract**	
parmesan	sprinkle grated, grate, hold a grater, ...	whisk	eggs, mayonnaise, milk, combine, pour, stir, ...
macaroni	stove on high heat, large pot, bowl, ...	spatula	colors, thickens, coated, simmer, grill, ...
egg	place the boiled, gently crack the, crack, ...	tongs	shrimp, bratwurst, turn, bun, marinate, ...
sauce	stir hot, pour gravy, lower setting, find a spoon, ...	cutting board	onions, bell pepper, meat, bok choy, ...
oil	spray cooking, splashing, slowly pour, ...	preheat	oven, broil, medium, degrees, ...

Table 3: Example Viterbi Alignments. For concrete to abstract, we match any phrase containing the word(s).

	Oracle		Greedy		
Model	Match	Top-5	Match	Top-5	Surp
Language Model	21.59	52.32	21.72	43.33	3.970
Seq2seq	23.57	53.38	23.44	45.76	3.755
+Att	24.52	53.98	24.57	47.34	3.780
Transducer	*24.72*	*55.09*	*24.91*	*47.80*	4.780
+ParamTran	23.81	53.98	24.00	46.19	*3.547*
OpenAI GPT	23.19	42.43	15.55	32.86	4.781
+ fine-tuning	**38.01**	**63.69**	**31.05**	**57.05**	**3.151**

Table 4: Results on the Cloze prediction task (Match = Exact Match, Top-5 = Precision of Top 5 predictions, Surp = Surprisal). Transducer results are reported for models with unparameterized and parameterized (+ParamTran) alignment transition models. The **best** and *second best* results are emphasized.



e_0
weigh
the
cabbage
... and
add
salt
e_6 .
<S> pour the cabbage into the big bowl . use a spoon and scoop some salt into the bowl
e_0
place
the
bread
... over
high
flame
e_6 .
<S> take the tongs and pick up the bread . set the bread on a grill . put on the grill over the high flame .

Figure 2: Example Viterbi alignments

do not. We see that our transducer model with fixed alignment transition probabilities performs best in terms of predictive accuracy (exact match and top-5 precision), while the seqseq model with attention is the next best in most comparisons. The model with parameterized transitions has the lowest surprisal though, as it is more confident about the alignment predictions it is making.

Using average alignment length to quantify whether the phrase alignments exhibit desirable structure, we see that the alignments found by the unparameterized transition model (average length 6.18) are significantly shorter than those of the parameterized model (average length 16.61). Investigation shows that the paramaterized model mostly learns degenerate alignments, aligning most of the concrete sequence to either the start or end of the abstract sentence. In contrast, qualitative analysis of the unparameterized transition model show that its alignments learn desirable correspondences (see Figure 2). Therefore among our proposed models (trained on in-domain data only) the transducer with unparameterized transitions satisfies our desiderata of displaying both good predictive power for word generation, and learning interpretable alignments.

Given the recent success of massively pre-trained language models (Peters et al., 2018), we are interested if these approaches transfer to our cloze task. We evaluate the OpenAI GPT transformer language model (Radford et al., 2018) with and without fine-tuning.Without fine-tuning this model does slightly worse than our best domain-specific model. With fine-tuning, its accuracy is substantially higher, but it still suffers from the same fundamental limitations as our other models (see Table 5). The transformer (Vaswani et al., 2017) attention is multi-headed and multi-layered which prohibits direct interpretability.

5 Qualitative Analysis

We visualize alignments of our transduction model over two partial sequences in Fig. 2. This shows which hidden vector of the abstract sentence aligned to every region of the concrete sequence. Specifically, we see how tools like *the big bowl*, *spoon*, and *tongs* are introduced to facilitate the actions. There are also implications, e.g. that *high* indicates *grill*. For further analysis we extract alignments over the training corpus, linking each decoded phrase with the word from the encoding it used during generation. We then aggregate these tuples into a table which we can filter (based on our whitelist) and sort (with PMI). This process is imprecise as it discards the context in which the alignment occurs, but it nonetheless extracts many

Abs	shape each dough ball into a circle and add tomato sauce .
Pred	flatten out your dough into a flat circle using your **hands**. take a **knife** to add tomato sauce to the center of your dough. use the back side of the **knife** to **cut** the sauce out. make **sure** you keep the sauce about **an inch** from the edges.
Gold	flatten out your dough into a flat circle using your **hands**. take a **spoon** to add tomato sauce to the center of your dough. use the back side of the **spoon** to **spread** the sauce out. make **sure** you keep the sauce about **an inch** from the edges.
Abs	place the kale cucumber bell peppers carrots and radishes on the wrapper .
Pred	put the **cut** on a **cutting** . put a **cutting** amount of kale on the **cutting** . add a **cut** amount of cucumber ...
Gold	put the **wrap** on a **plate** . put a **small** amount of kale on the **wrap** . add a **small** amount of cucumber ...
Abs	wrap the pizza .
Pred	find a **large piece** to put the pizza om . place the pizza in the center for it not to **stick** around . grab the plastic wrap and start **wrapping** the **entire thing** and pizza . wrap all around until **completely covered** on all corners . put in freezer on a **cold water** and freeze overnight
Gold	find a **hard surface** to put the pizza om . place the pizza in the center for it not to **slide** around . grab the plastic wrap and start **wrapping** the **hard surface** and pizza . wrap all around until **fully covered** on all corners . put in freezer on a **flat surface** and freeze overnight

Table 5: Above is the output of OpenAI GPT when forced to greedily decode answers to blanks in validation.

of the phenomena we would hope to see (Table 3).

The left-hand side of the table shows words from the abstract YouCook annotations and corresponding phrases in the concrete annotation. For the righthand side we searched for common concrete terms that may be preceded or followed by other terms, and present the abstract terms they were most often generated by.

Finally, Table 5 shows three randomly chosen examples (from the validation set) of greedy decodings for slot filling with GPT fine-tuned on our dataset. These examples demonstrate that, first, there are cases where GPT is successful or produces a semantically valid answer (e.g. *fully* vs *completely*). Second, as is common with greedy decoding, the model can get stuck in a loop (e.g. *cut, cutting, cutting, ...*). Finally, note there are nonsensical cases where the model appears to have discarded the abstract context (e.g. *knife to add tomato sauce* or *freezer on a cold water*).

6 Related Work

Many script learning systems are based on event co-occurrence and language modeling in large text corpora, and can infer implicit events without creating explicit situation-specific frame structures (Chambers and Jurafsky, 2008; Rudinger et al., 2015; Pichotta and Mooney, 2016). Other systems induce situation-specific frames from text (Cheung et al., 2013; Balasubramanian et al., 2013). However, these methods do not explicitly target the commonsense correspondence between differing levels of detail of complex events.

Most relevant to this paper is the pioneering work of Regneri et al. (2013) as extended by Senina et al. (2014) and Rohrbach et al. (2014). These papers present the TACOS corpus, consisting of natural language descriptions of activities in videos paired with low-level activity labels. Senina et al. (2014) collect an additional level of multi-sentence annotations on the corpus, which allowing for video caption generation at multiple levels of detail. Rohrbach et al. (2014) describe a similar corpus of natural descriptions of composite actions, useful for activity recognition in video. These corpora differ in a number of important ways from **KIDSCOOK**; in particular, the language has somewhat limited complexity and "naturalness" when describing complex scenarios, a phenomenon also observed in the robotics literature (Scalise et al., 2018). Our data collection process avoids more formulaic language by eliciting "child-directed" descriptions.

7 Conclusion

We introduce a new hierarchical script learning dataset and cloze task in which models must learn commonsense world knowledge about tools, procedures and even basic physics to perform well. Our aim is to begin a conversation about abstraction in language, how it is modeled, and what is implicitly hidden. Our abstract and concrete instructions are grounded in the same videos yet differ dramatically due to their assumed audiences. We show that a neural transduction model produces interpretable alignments for analyzing these otherwise latent correlations and phenomena.

Acknowledgements

This work was supported in part by NSF (IIS-1524371 & 1703166) and through DARPA's CwC program through ARO (W911NF-15-1-0543).

References

Fred Attneave. 1959. *Applications of information theory to psychology: A summary of basic concepts, methods, and results.* Henry Holt.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*.

Yonatan Bisk, Daniel Marcu, and William Wong. 2016. Towards a dataset for human computer communication via grounded language acquisition. In *Proceedings of the AAAI 2016 Workshop on Symbiotic Cognitive Systems*, pages 729–732, Phoenix, AZ.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *CL*, 19(2):263–311.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*.

Nathanael Chambers. 2017. Behind the scenes of an evolving event cloze test. In *LSDSem*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *NAACL*.

Gerald F. DeJong. 1981. Generalizations based on explanations. In *IJCAI*.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society.*, pages 1–38.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Workshop on Structured Prediction for NLP*.

H. Paul Grice. 1975. Logic and conversation. *Syntax and Semantics 3: Speech Acts*, pages 41–58.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *ICLR*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *AAAI*.

Alex Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Mechaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*, 1.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *GCPR*.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. 2015. Learning to predict script events from domain-specific text. In **Sem*.

Rosario Scalise, Yonatan Bisk, Maxwell Forbes, Daqing Yi, Yejin Choi, and Siddhartha Srinivasa. 2018. Balancing shared autonomy with human-robot communication. *arXiv preprint arXiv:1805.07719*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. LEA.

Anna Senina, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. *arXiv preprint arXiv:1403.6173*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

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

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *ICML*.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Hierarchical quantized representations for script generation. In *EMNLP*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017. The neural noisy channel. In *ICLR*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *EMNLP*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2017. Towards automatic learning of procedures from web instructional videos. *arXiv preprint arXiv:1703.09788*.

A Transducer Model

We briefly describe the model of Yu, Buys, and Blunsom (2016) and our minor modifications thereto.

A.1 Alignment with Latent Variables

We model the conditional probability of a concrete sequence y given abstract sequence x through a latent alignment variable a between x and y, which is a sequence of variables a_j, with $a_j = i$ signifying that y_j is aligned to x_i. The marginal probability of y given x is

$$p(y|x) = \sum_a p(y, a|x). \quad (1)$$

In the following, we use m to denote the length of x and n to denote the length of y. The model formulation restricts alignments to be monotonic, i.e. $a_{j+1} \geq a_j$ for all j.

The model factorizes over timesteps into alignment and word prediction probabilities, such that the word prediction at each timestep is informed by its alignment:

$$p(y, a|x) = \prod_j p(a_j|a_{j-1}, x_{1:a_{j-1}}, y_{1:j-1}) \times p(y_j|a_j, x_{1:a_j}, y_{1:j-1}) \quad (2)$$

The abstract and concrete sequences are both encoded with LSTM Recurrent Neural Networks (Hochreiter and Schmidhuber, 1997). In contrast to standard attention-based models, the aligned encoder representation is not fed into the decoder RNN state, but only used to make next word predictions. Due to the small size of the training data, words in both sequences are embedded using fixed GloVe embeddings (Pennington et al., 2014). The word emission probability is then defined as

$$p(y_j|a_j, x_{1:a_j}, y_{1:j-1}) = \text{softmax}(\text{MLP}(e_{a_j}, d_j)) \quad (3)$$

with e the encoder hidden states and d the decoder hidden states.

The alignment probability factorizes into *shift* and *emit* probabilities, where a *shift* action increments the alignment to the next word in the input sequence, and an *emit* action generates the next output word. We refer to these as *transition* probabilities. This formulation enables us to restrict the hard alignment to be monotonic.

We consider two parameterizations of this distribution. In the first, the probabilities are parameterized by the neural network, using the encoder and decoder hidden state in a similar manner to how the word emission probability was computed. The alignment probability at a given timestep is therefore parameterized as

$$p(a_j|a_{j-1}, x_{1:a_{j-1}}, y_{1:j-1}) = p(\text{emit}|a_j, x_{1:a_j}, y_{1:j-1}) \times \prod_{i=a_{j-1}}^{a_j - 1} p(\text{shift}|i, x_{1:i}, y_{1:j-1}), \quad (4)$$

where

$$p(\text{shift}|i, x_{1:i}, y_{1:j-1}) = \sigma(MLP(e_i, d_j)), \quad (5)$$
$$p(\text{emit}|i, x_{1:i}, y_{1:j-1}) = 1 - p(\text{shift}|i, x_{1:i}, y_{1:j-1}). \quad (6)$$

We also consider using the simpler, fixed alignment parameterization in Yu, Buys, and Blunsom (2016), where the transition probability is conditioned only on sequence length, not on x or y, and can therefore be estimated using the ratio between input and output sentence lengths. The alignment probabilities are not updated during training, and consequently the posterior distribution over the alignments is biased towards this prior, favoring alignments close to the diagonal.

The parameterized alignment model contains as special cases two degenerate solutions: (1) an unconditional language model and (2) a seq2seq model. These occur if the model performs all emits before shifting or all shifts before emitting, respectively. To prevent the creation of a language model we force the last output word to be aligned to the last word in the abstract sequence, similar to Yu et al. (2017). However, the parameterized transition model could still in practice revert to a pure sequence-to-sequence model.

A.2 Marginalization

Next we briefly describe the dynamic program used to marginalize over alignments during training and to find the most likely alignments of a given alignment during inference; we refer the reader to Yu, Buys, and Blunsom (2016) for a more thorough treatment.

The forward variable $\alpha_i(j)$ representing $p(y_{1:j}, a_j = i|x_{1:i})$ is recursively as

$$\alpha_i(j) = p(y_j|i, x_{1:i}, y_{1:j-1}) \times \sum_{k=1}^{i} \alpha_k(j-1) p(a_j = i|k, x_{1:k}, y_{1:j-1}). \quad (7)$$

The marginal likelihood objective is to train the model to optimize $\alpha_m(n) = p(y_{1:n}, a_n =$

$m|x_{1:m})$. The gradients are computed with automatic differentiation; as this is is equivalent to using the forward-backward algorithm to estimate the gradients (Eisner, 2016), only the forward algorithm has to be implemented.

To make the implementation GPU-efficient, we vectorize the computation of α. The computation iterates through decoding steps, each of which can be generated from an alignment to any of the encoder tokens. We can efficiently construct a transition matrix T, corresponding to all possible encoder states performing all possible shifts, and emission matrix E_j which is a gather by word index j.

To compute the forward probabilities at each timestep, the current forward probabilities are first multiplied by all possible transitions. A sum in logspace collapses all paths, and the emission (word generation) probabilities are multiplied to obtain the new forward probabilities. When fixed transition probabilities are used, T is precomputed.

A.3 Viterbi EM Training

Latent variable models can be trained either through directly optimizing the likelihood objective through gradient descent (as described above), or with the Expectation Maximization (EM) algorithm (Dempster et al., 1977), which alternates between calculating expectations over the values of the latent variables given the current parameters, and maximizing the expected complete data log likelihood given those expectations. We consider training our alignment model with Viterbi EM (Brown et al., 1993), also known as "hard" EM, where at each iteration the most likely assignment of the hidden variables (alignments) are found and the parameters are updated to optimize the log likelihood given those alignments. Viterbi EM has been shown to give superior performance to standard EM on unsupervised parsing (Spitkovsky et al., 2010), due to better convergence properties in practice by making the distribution more peaked.

We perform batched Viterbi EM training by computing the Viterbi alignments for a batch, and then performing a gradient step based on treating those alignments as observations.

We follow a two-stage training procedure: we first directly optimize the marginal likelihood with batched SGD to find a reasonable initial distribution over alignments, before switching to Viterbi EM training. Such a strategy has been shown to reduce the chance that the model will get stuck in local optima (Spitkovsky et al., 2011).

A.4 Inference

We apply the trained models to multiple inference problems to evaluate how well they are capturing script knowledge. The first is finding the most likely alignment given a pair of abstract and concrete sequences. We use the standard Viterbi algorithm, in which we replace the sum in equation (7) with $\max$, and keep track of the index corresponding to each value of α during the forward computation. The most likely alignment can then be traced back from $a_n = m$.

The second inference problem is slot-filling, for application to the cloze task. Given an abstract sentence and a partially-filled concrete sequence, we want to use the model to predict words to fill the given blanks. To make the prediction, we sample 5 candidate sequences by predicting words for each slot, in left-to-right order, and then choosing the sequence with the highest overall probability. Words are predicted by sampling with temperature 0.1, in order to peak the distribution while still allowing some diversity in the samples. The motivation for selecting the final output from multiple samples is that the original samples are biased, as they are only conditioned on the left context.

At the start of the prediction for each slot, the Viterbi alignment of the prefix of the sequence up to the start of that slot is re-predicted, independent of previous alignment predictions. Consequently alignment decisions can be revised, and the slot alignments are no longer constrained to be monotonic, which makes the slot prediction model more flexible. For the parameterized transition model, the slot alignment is predicted greedily by incrementing the last predicted alignment while the shift probability is greater than 0.5. The fixed transition model assumes that the alignment of the word preceding the slot is shared across the slot.