Script Parsing with Hierarchical Sequence Modelling

Fangzhou Zhai, Iza Škrjanec and **Alexander Koller**
Dept. of Language Science and Technology
Saarland Informatics Campus
Saarland University
{*fzhai, skrjanec, koller*}*@coli.uni-saarland.de*

Abstract

Scripts (Schank and Abelson, 1977) capture commonsense knowledge about everyday activities and their participants. Script knowledge has been shown to be useful in a number of NLP tasks, such as referent prediction, discourse classification, and story generation. A crucial step for the exploitation of script knowledge is ***script parsing***, the task of tagging a text with the events and participants from a certain activity. This task is challenging: it requires information both about the ways events and participants are usually realized in surface language as well as the order in which they occur in the world. We show how to do accurate script parsing with a hierarchical sequence model. Our model improves the state of the art of event parsing by over 16 points F-score and, for the first time, accurately tags script participants.

1 Introduction

Script knowledge is a category of common sense knowledge that describes how people conduct everyday activities sequentially (Schank and Abelson, 1977). Script knowledge of a specific **scenario**, e.g. GROCERY SHOPPING, includes the **events** that comprise the scenario, the **participants** involved, and the relations between them. Script knowledge is useful for various downstream NLP applications, such as referent prediction (Ahrendt and Demberg, 2016; Modi et al., 2017), discourse sense classification (Lee et al., 2020), story generation (Zhai et al., 2019, 2020).

Script parsing identifies pre-defined sets of script events and participants from surface text (see Figure 1). For a specific scenario, script parsing essentially boils down to determining what each verb and each NP (which we term **candidate**) refers to in the context of that scenario.

Script parsing is an under-investigated, complex task. It is highly contextualized and corresponds to each specific scenario. The task is challenging even for humans: the inter-annotator agreement is quite modest, at 0.64 and 0.77 Fleiss' κ for event and participant parsing, respectively (Modi et al., 2016). Various factors need to be taken into consideration for this task. (1) At the local level, the basic semantics of the candidates. (2) At the discourse level, the sequence of events and participants should sketch a reasonable agenda for the activity. For example, the events must occur in a feasible order; when an NP is a dependent of an adjacent verb, the predicted participant type must be one that participates in the predicted event. In Figure 1, the same verb *found* was assigned different event classes: ***found*** *bike pump* as *get tools* whereas ***found*** *a small glass shard* as *examine tire*. One would have to consider the arguments of the verb, and potentially where it appears in the story to make the decision: what happens after 'check' as *examine tire* is not likely to be *get tools*. Apart from the modelling difficulties, annotated corpora are quite limited in size, given the high cost. Many event / participant classes have less that 10 instances across these corpora, occasionally missing from a validation set generated by random split.

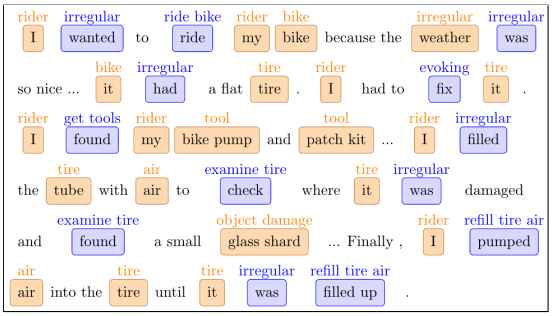

Figure 1: Descriptions of FIXING A FLAT TIRE from InScript. Script parsing identifies events and participants from surface text.

We contribute the following to the area of script

Proceedings of the 10th Conference on Lexical and Computational Semantics, pages 195–201
August 5–6, 2021, Bangkok, Thailand (online)

parsing: methodology-wise, **(1)** we propose a hierarchical sequence model, which learns patterns in different granularity levels with different sequence models. **(2)** We investigate data augmentation approaches to this task. In terms of results, **(3)** we achieve accurate participant parsing results for the first time and **(4)** improve the state of the art in event parsing by over 16 points F1-score.

2 Related Work

Theoretical considerations of scripts in AI (as described by Schank and Abelson, 1977; Barr and Feigenbaum, 1981) were analyzed in a wide coverage empirical study by Regneri et al. (2010), who crowdsourced event descriptions of several everyday activities (scenarios). They applied unsupervised methods to compute a graph representation of the script's temporal structure. As a direct extension, Modi et al. (2016) and Wanzare et al. (2016) collected the **InScript** and **DeScript** corpora of event descriptions and manually annotated the scenario-specific types of events and participants to accommodate aligning surface text with data-driven script knowledge. The goal is to identify spans of the text that refer to the events and participants that are typically involved in a script. For the case of the script about fixing a bike, the typical events include riding a bike, noticing a flat tire, getting tools, repairing the tire and testing it. These events and participants are pre-defined for each activity and the task is to label the tokens with these abstract classes, whereby the surface forms vary.

The model by Ostermann et al. (2017) is the state of the art for event parsing over InScript which was formulated as a sequence tagging task. The authors used a linear CRF to identify the script events. Its features include syntax, FrameNet (Ruppenhofer et al., 2006) features, pre-trained word embeddings and a number of script-related features encoding script-specific aspects like event order.

Our work shares many similarities with Berant et al. (2014). To do question answering in the biological domain, they first build a graph representation of events, participants and their relations given a text about a biological process. Token spans that denote an occurrence of an event are considered event triggers. However, unlike our approach, these events are not based on a pre-defined set. They separately train a model for identifying event triggers and a model for finding plausible argument candidates. The features used rely on syntax, semantic roles and some external domain resources. In contrast, we propose a model that jointly learns how to identify as well as label events and participants.

Much of the previous work on inferring script knowledge from text is focused around completing an event chain by predicting the missing event (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015), the missing text (Bisk et al., 2019) or both (Pichotta and Mooney, 2016). These approaches consider surface forms of event verbs and syntactic relation types of their arguments (*subj*, *obj*), while our task operates on abstract event and participant types.

3 Method

3.1 Data and Pre-processing

Our work is based on two English corpora, InScript and DeScript. InScript (Modi et al., 2016) includes around 100 stories about each of 10 daily activities (scenarios), e.g., GOING GROCERY SHOPPING, TAKING A BATH, and RIDING IN A PUBLIC BUS. The corpus annotates surface text with event and participant classes, and specifies the candidates according to their syntactic dependency (see Figure 1 for an excerpt from InScript). DeScript (Wanzare et al., 2016) includes, among others, 50 process descriptions for each of the 10 scenarios in InScript. These process descriptions are telegram-style short phrases, like '*Find hole in tire. Plug hole. Find tire pump. Insert tire pump into tire. Pump tire until full of air*'. The events in these texts are annotated with the same set of labels as InScript. DeScript has no participant annotations. We mainly perform experiments on InScript, whereas DeScript is used as auxiliary training data.

For each of the 10 scenarios in the InScript corpus, the authors (Modi et al., 2016) designed prototypical scenario-specific event and participant classes. For example, the story about FIXING A FLAT TIRE in Figure 1 shows that typically this activity includes a bike rider, a bike, tools and a tire. These participants are involved in the events of riding a bike, getting tools, fixing the tire, and checking it.

Following Ostermann et al. (2017), we distinguish between **regular events**, events corresponding to the crucial steps of the respective scenario, and **irregular events**, the ones that take place in the course of the story, but are not directly related to the scenario's core event chain. For example,

in Figure 1, *the weather **was** nice* is considered an irregular event as it does not directly relate to the core steps of the scenario, fixing a flat tire. We collapse the classes UNREL, RELNSCR, OTHER and UNKNOWN into a single irregular event class for each scenario. 12,902 (33.5%) events in InScript are regular, whereas 4,185 (89.1%) events in DeScript are regular. We also distinguish **regular participants** from **irregular participants** in a similar manner. The identification of irregular candidates is a crucial component of a script parser, as naturally occurring text very often includes such content, making the text more interesting and personal.

The 10 scenarios vary in complexity (TAKING A BATH VS. FLYING IN AN AIRPLANE) and specificity (RIDING A PUBLIC BUS VS. REPAIRING A FLAT TIRE). This is reflected in the class sizes of regular events and regular participants. On average, each scenario has 19.2 regular event and 18.9 regular participant classes. TAKING A FLIGHT has the largest class sizes for events and participants (29 and 26, respectively), while PLANTING A TREE has the smallest (14 and 15, respectively).

3.2 Model

We train a scenario agnostic model that parses all InScript scenarios. Thus our model implicitly contains a *scenario detection* model, which determines the scenario that a piece of text is about. Our model consists of two sequence models: **(1)** a **word sequence model** that captures how each event and participant is usually realized in surface language, and **(2)** an **event sequence model** that operates on the event level, which models the sequence of events and participants to capture procedural script knowledge.

The Hierarchical Model. Figure 2 shows the model architecture. The model takes as input a story x from corpus d, an ordered set of indices I that specifies the positions of the candidates. It assigns an event / participant label to each of these candidates as output. These labels are pre-defined in InScript and specific to each scenario. The set of candidates consists of all NPs and verbs in the text. We use the InScript tokens as annotated in it; yet they could also be extracted with a syntactic parser.

The **word sequence model** encodes the entire story with pre-trained contextualized word embeddings into a list of vectors (we use *xlnet-base-cased*

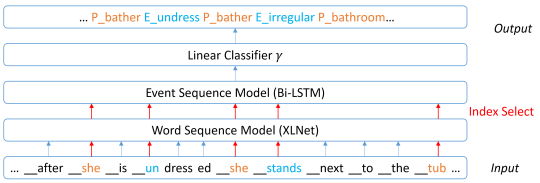

Figure 2: The model architecture. Note that index selection is performed before the Bi-LSTM layer.

(Yang et al., 2019)):

$$\tilde{x} = \text{XLNet}(x) \quad (1)$$

Next, to accommodate sequence modelling at the discourse level, we only keep the representations in $\tilde{x}$ that corresponds to the candidates, namely, the NP heads and the verbs. Their positions in $\tilde{x}$ are specified by the ordered set of indices I. We directly take the vector representation of these tokens and ignore the rest to form a sub-sequence of $\tilde{x}$. This operation is termed **index_select** in the Pytorch library:

$$c = \tilde{x}.index\_select(I) \quad (2)$$

Now, we apply the event sequence model ψ (a Bi-LSTM) on it, to yield the features $\tilde{c}$ for linear classifier γ, to generates a distribution over the labels for each token:

$$p(y|x, I; \theta_\psi, \theta_\gamma) = softmax(\gamma_{\theta_\gamma}(\psi_{\theta_\psi}(c))) \quad (3)$$

The model is trained by optimizing data likelihood:

$$\theta^* = \underset{\theta_\psi, \theta_\gamma}{\text{argmax}} \sum_{x,I} log(p(y|x, I; \theta_\psi, \theta_\gamma)) \quad (4)$$

3.3 Addressing Data Sparsity

The class distribution is skewed. Some classes are quite small: the largest regular event class has 397 instances (e.g., *get groceries* in GROCERY SHOPPING), whereas there are 26 classes with less than 10 instances (e.g., *get receipt* in GROCERY SHOPPING, *scalp massage* in TAKING A BATH), which means they are hard to learn reliably. Our efforts to address this are two-fold.

Domain Adaptation. Firstly, we use DeScript as additional training data. We experiment with two domain adaptation methods. Most straightforwardly, **(1)** *Data concatenates*, which concatenates DeScript with the original InScript train set. Having noticed InScript and DeScript varying greatly in

Index	Model	Train set	Events		Participants	
			Macro-F1	Micro-F1	Macro-F1	Micro-F1
1.	*Ostermann*	In	58.1	66.0	n/a	n/a
2.	*fine-tuned XLNet*	In	62.1^{1}	79.3^{1}	79.7^{345}	77.2
3.	*no_index_select*	In	63.3^{1}	78.3^{1}	74.3	87.1^{2}
4.	*hierarchical*	In	70.1^{123}	83.7^{123}	78.7^{3}	89.3^{23}
5.	*concatenate*	In, De	69.3^{123}	82.5^{123}	79.1^{3}	89.9^{23}
6.	*corpus_embedding*	In, De	74.9^{12345}	82.9^{123}	78.6^{3}	89.4^{23}
7.	*hierarchical*^{BT}	In, BT	$\mathbf{75.1}^{1234568}$	$\mathbf{85.7}^{1234568}$	80.3^{123456}	$\mathbf{90.3}^{1234568}$
8.	*corpus_embedding*^{BT}	In, De, BT	74.3^{12345}	83.8^{123456}	$\mathbf{80.9}^{123456}$	89.5^{12345}

$^{1-8}$: performance improvement over the respective model is significant at $\alpha = 0.05$ according to independent T-test.

Table 1: Results. The highest metrics in each column are displayed in boldface. Thanks to a larger training set, the optimization of **7.** and **8.** are quite stable, thus its performance difference compared to others can be significant despite small margins.

the style of their language, we also perform explicit domain modelling with **(2)** *Corpus embedding*. We follow Stymne et al. (2018) to train a vector representation (the **corpus embedding**) for each corpus to capture corpus-specific patterns. We concatenate the corpus representation with each candidate representation, to substitute the input term ($\psi_{\theta_\psi}(c)$) to the linear classifier γ with

$$\tilde{c} = \psi_{\theta_\psi}(c; \eta_{\theta_\eta}(d)) \tag{5}$$

Here $\eta(\cdot)$ denotes the corpus embeddings.

Data Augmentation. Secondly, we augment InScript via back-translation (Bojar and Tamchyna, 2011; Sennrich et al., 2016; Xie et al., 2020) to paraphrase the original data and help the model generalize better over the surface text. The stories are translated to French and back with Google Translate. The participant and event annotations are mapped to the paraphrases according to heuristics based on word-level semantics and string matching. The new data was concatenated with the original InScript and both were treated as a single domain. See the appendix for more implementation details[1]. In the example below, the event verb takes a different tense and surface form in the back-translation.

O : when I_{rider} was $riding_{ride}$ my_{rider} $bike_{bike}$ this past summer

Fr : l'été dernier, je montais mon vélo

BT : when I_{rider} $rode_{ride}$ my_{rider} $bike_{bike}$ this summer

[1]Our code, data and virtual environment are shared at https://github.com/coli-saar/SSP_sem.

4 Experiments

4.1 Ablations and Baselines

We randomly split (80/10/10) InScript by entire stories to create the train/val/test sets. We have two external baselines: (1) the SotA model from Ostermann et al. (2017); (2) a fine-tuned XLNet, for which we train a linear classifier, apart from tuning its pre-trained parameters. As for our models, *Hierarchical* is the hierarchical model described by formulas (1)-(4). For *no_index_select*, we ablate the index selection (thus $c = x$) to neutralize the event sequence model. Its event sequence model now takes every token of the story as input, and still operates on the token level. The variants *concatenate* and *corpus_embedding* exploit DeScript with respective domain adaptation methods.

4.2 Results

The results are shown in Table 1. A fine-tuned XLNet already outperforms *Ostermann*. Yet our model variants deliver further, substantial improvements. All our models outperform *Ostermann* by a considerable margin, on both event macro and micro F1. We also see that all micro-F1s are noticeably higher than the respective macro-F1s. This difference is due to the data including many small classes that are in general harder to learn.

hierarchical see substantial improvements over *no_index_select*, which fails to perform sequence modelling at the discourse level. We also note that *hierarchical* improved both micro-F1s. Analysis shows that the hierarchical models are generally better at addressing the most frequent yet problematic class, the irregular candidates. These candidates do not participate in the core event chain, a decision better made after taking the structure of

the candidate sequences into consideration. That is exactly the job of our event sequence model.

Participant parsing yields much higher scores than event parsing. The reason is, a large proportion of errors come from the *irregular* candidates (see also Section 5). However, irregular participants (19.6%) are proportionally fewer than irregular events (66.5%). Moreover, a lot of participant candidates refer to the protagonist (31.0%), an easy class usually realized with first person pronouns, making participant parsing generally easier.

For the variants that performs domain adaptation, *corpus_embedding* is clearly a better way to exploit DeScript, due to the apparent difference between the language styles of both corpora. *hierarchical*^{BT} sees a larger improvement over *hierarchical* as it has paraphrased InScript as additional training data, which is larger and a more similar domain than DeScript to InScript. These improvements over *hierarchical* are more prominent in event macro-F1s, which means these models are generally better at tagging smaller event classes, achieving our original goal to alleviate the issues caused by the uneven class sizes. Further addition of DeScript on top of *hierarchical*^{BT} (model 8.) does not yield further significant improvement, but sees, overall, a modest performance drop: with the addition of back-translated data thus a larger, relatively homogeneous training set, the domain difference between DeScript and InScript is beginning to outweigh the benefit of having DeScript.

5 Error Analysis

We manually classified the validation set errors made by our best-performing model, *hierarchical*^{BT}, case by case. A breakdown is presented in Table 2.

Noisy Corpus Labels. The corpus annotations of these instances are possibly incorrect. This is quite common, given the complexity (thus the moderate inter-annotator agreement) of the task. For example, in a story about BORROWING A BOOK FROM A LIBRARY, ... *I had to **get** a library card* ... is a clear match for the event *obtain card*, as is predicted by our model; but in the corpus it was annotated as 'irregular', a mistake probably due to the light verb 'get' seemingly irrelevant to the scenario at first glance by the original annotator.

False Positives of *irregular*. A large proportion of errors feature a wrongly predicted *irregular*. We identified two main sources of such errors: (1) small class sizes; (2) instances that are particularly difficult because pragmatic inference is needed to make the right decision. As an example, ... ***get** materials for the assignment* ... corresponds to the event class *evoking_library*, i.e. it evokes the scenario of BORROWING A BOOK FROM A LIBRARY without explicitly referring to a scenario event. However, without taking the situational context of the scenario into consideration, it cannot be inferred that '***get** materials for the assignment*' actually means '*borrow a book from a library*'.

Wrong Category. A small number of events are tagged as participants and vice versa, e.g. some homonyms of verbs and nouns (*board* or *love*).

Type	Events	Participants
Noisy corpus label	23%	26%
False irregular predictions	49%	37%
Wrong category	2%	4%
Others	26%	33%

Table 2: A breakdown of the error types.

6 Conclusion

We present the first model that achieves high performance on both event and participant parsing. The model adopts a hierarchical design to model both the sequence of tokens and the sequence of script events and participants. Further exploitation of domain adaptation and data augmentation methods yields a substantial performance boost. This work has established methods to accurately parse both script events and participants, in a supervised learning framework. Our next step is approaching this complex task with less supervision, to lift the requirement on finely-annotated data, thus enabling wide-coverage script parsing.

Acknowledgments

We thank Simon Ostermann for providing the data from his experiments and his help all along the way of re-implementing his model. We thank Vera Demberg for the useful comments and suggestions. We also thank the anonymous reviewers for the valuable comments. This research was funded by the German Research Foundation (DFG) as part of SFB 1102 (Project-ID 232722074) "Information Density and Linguistic Encoding".

References

Simon Ahrendt and Vera Demberg. 2016. Improving event prediction by representing script participants. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 546–551.

Avron Barr and Edward Feigenbaum. 1981. *The Handbook of Artificial Intelligence: Volume 2*. William Kaufman Inc, Los Altos, CA.

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, Doha, Qatar. Association for Computational Linguistics.

Yonatan Bisk, Jan Buys, Karl Pichotta, and Yejin Choi. 2019. Benchmarking hierarchical script knowledge. 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 4077–4085.

Ondřej Bojar and Aleš Tamchyna. 2011. Improving translation model by monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 330–336, Edinburgh, Scotland. Association for Computational Linguistics.

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

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, Avignon, France. Association for Computational Linguistics.

I-Ta Lee, Maria Leonor Pacheco, and Dan Goldwasser. 2020. Weakly-supervised modeling of contextualized event embedding for discourse relations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4962–4972, Online. Association for Computational Linguistics.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2016. Inscript: Narrative texts annotated with script information. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3485–3493.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association for Computational Linguistics*, 5:31–44.

Simon Ostermann, Michael Roth, Stefan Thater, and Manfred Pinkal. 2017. Aligning script events with narrative texts. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 128–134.

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, Gothenburg, Sweden. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany. Association for Computational Linguistics.

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, Lisbon, Portugal. Association for Computational Linguistics.

Josef Ruppenhofer, Michael Ellsworth, Myriam Schwarzer-Petruck, Christopher R Johnson, and Jan Scheffczyk. 2006. Framenet ii: Extended theory and practice.

Roger C Schank and Robert P Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia. Association for Computational Linguistics.

Lilian DA Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. Descript: 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.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Fangzhou Zhai, Vera Demberg, and Alexander Koller. 2020. Story generation with rich details. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2346–2351, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Fangzhou Zhai, Vera Demberg, Pavel Shkadzko, Wei Shi, and Asad Sayeed. 2019. A hybrid model for globally coherent story generation. In *Proceedings of the Second Workshop on Storytelling*, pages 34–45, Florence, Italy. Association for Computational Linguistics.