Data Driven Interactive Story Generation

Amanul Haque, Shrey Anand
North Carolina State Department
Department of Computer Science

Abstract

Automated story generation finds application in both the gaming and entertainment industry. Most of the prior work in this domain has focused on using planning or simulation-based approach to generate stories. We propose a data-driven text story generation that uses a Recurrent Neural Network (RNN) to generate novel and coherent short stories. We have used skip-thought vectorization to represent sentences in vector space and GRUs to model the RNN. We also used entity resolution to identify characters in the story and coreference resolution to relate character references to the appropriate characters. Our model shows good performance (evaluated on test set through rank frequency) in generating short five sentence commonsense stories that are both coherent and interesting.

Keywords: Interactive story generation, Recurrent neural network, Context representation, Entity extraction

Introduction

Stories form the basis of human conversation. We, as humans, recall events from our past as events that unfold in a story. Advancements in narrative theory can improve human-computer interface (Laurel and Mountford 1990), and make the interface with machines more natural and convenient.

An agent that is aware of the narrative can form a basis for agents with improved social conversation skills. Such an agent system can potentially be applied to personalized education and assisting tools for human authors. A model that can generate syntactically correct and coherent text can be applied to commercial blogs, news, and stories. Generative text that is cognizant of the context can be also be used in question answering systems. From the perspective of games, generative texts can be used for adding randomness in the mission stories and also player preferences. Advancements in this domain can find useful applications in popular personal assistant AI such as Siri, Alexa and Cortana.

Simple stories that seem relatively easy to understand or generate has many obscure challenges. Story generation as a task involves several components and conflicts that need to be resolved. One major challenge in a data driven story generation is resolving pronominal (pronoun) character references. This relatively easy task for humans is highly challenging for a machine with limited understanding of text. Coreference resolution is critical to mapping each action or event to corresponding character it relates to. Another challenge which is a prerequisite for coreference resolution is entity resolution which involves identifying characters in a plot.

Stories usually unfold as a sequence of events that revolve around a central character and has other characters to support the narrative. For most stories, the story proceeds to build up a conflict for the central character. The central character's objective is to resolve this conflict before the story ends. Building a system that generates stories that incorporate these intricate features needs to take the context of the narration into account. This has to be modelled appropriately in any story generation system for it to generate interesting and meaningful stories.

In this work we focus on generating short five sentence stories that are coherent and grammatically correct. We use a data driven approach to model a sentence level RNN to generate new stories. Our story generation system is interactive and takes user inputs for generating stories. The system generates the first sentence to begin the story following which each new sentence in the story is either a system generated text or an input from the user. At each step (each sentence generated for the story) the user has a choice to either choose from a list of system generated sentences or provide a new sentence. Combining the user input and system generated stories yields a final story that can be considered a content generation with human author in the loop.

Related Work

AI researchers in the 1970s and early 80s, started exploring the problem of automated story and narrative generation. Work in this domain lead to the field of narrative intelligence and interactive storytelling. Planning has been used in many previous works of story generation (J. Porteous 2010)(F. Charles 2003). Riedl et al used a variation of partial ordered planning called IPOCL for intent driven story generation (M. O. Riedl 2010). Szilas et al used simulation based intelligent virtual agents to generate narrative in an interactive drama setup (Szilas 2001). Theune et al developed a similar story generator that uses an intelligent director agent in the simulation to have more control over the simulation based narratives (M. Theune and Heylen 2003).

In this work we shift our focus on a more data intensive approach and propose to use neural networks to generate interactive stories. Recurrent Neural Networks (RNNs) have been widely successful in generating text for various natural language tasks (Sutskever, Vinyals, and Le 2014). For the purpose of narrative texts (sequences of events revolving around entities), a naive dumping of characters or words into LSTMs may not be the best approach. The overarching theme of recent models is to convert the raw text in a higher representation before feeding them to deep learning models. The intuition is that such a representation is more general than characters and less unique than sentences. We explore and plan to build on an event based neural language model that uses a similar approach (Martin et al. 2017).

In recent research there has been an active interest in models revolving around entities. Ji. et al. design a model that identifies entity (PERSON, ORGANISATION etc. see NER) mentions in the raw text and use them for their model (Ji et al. 2017). Clark et al., building on the work further, incorporate entities as vectors and update them as story unfolds (Clark, Ji, and Smith 2018). At any instant in the story, the current representation of entities and the previous context helps to predict what happens next. We plan to recognize entities and use them as features, thus using them with a more static approach.

Event based models, being slightly different, represent the raw text in terms of ”Events”. Pichotta et al. define an Event as 5 tuple: (v, p, es, eo, ep) verb, preposition, nouns representing the subject, direction object, and prepositional object, respectively (Pichotta and Mooney 2016). Martin et al. built on the work further by adding a mechanism for turning the event back to natural language text (Martin et al. 2017) In our work, we do not train the network on events as training on events and then turning them back to sentences through another network is computationally expensive and may be an over complicated approach.

Dataset

Data driven approach for story generation depends on the training data. This dependence on data mandates using a dataset of stories that has simple and correct grammatical structure to extract useful information from. In this work, we have used ROCStories dataset (Mostafazadeh et al. 2016) to train our model. ROCStories is a collection of 50,000, five sentence long common sense stories. These stories capture a rich set of causal and temporal common sense relations between events that can be effectively used for story generation. The sentence structure in the stories are suitable for efficient entities and co-references resolution.

Implementation

Entity and Coreference Resolution

We applied entity and coreference resolution as major preprocessing steps on the dataset. We used NER tagging to tag all entities and extract characters from each story. In order to have consistency in data for the model to train, we changed all character names to a restricted list of names. At the end of entity tagging each of our stories had same character names (John, Peter or Jake for male characters, and Emily, Rachel or Sarah for female characters). Same character names across all story ensure that the model can find relevant events in other stories to link with the current narrative. This also provides our model a vast pool of actions and events to train from for a specific character. All stories that have the central character as *John* provide a pool of actions or events that John can be part of, for instance, *John went to the market, John enjoyed the game, John is a good swimmer.* As the entity resolution is critical for our approach we excluded stories in which entity resolution could not be effectively applied.

After entity resolution and replacing all character references with a restricted list of character names, we applied coreference resolution to resolve pronoun character references. We replaced all references with actual names to enable our model to differentiate between character references efficiently. This ensures that the model can effectively relate each action or event directly to the characters and not his/her pronominal reference which would be difficult for our model to relate to a character. We have excluded all stories in which coreference resolution did not apply effectively and their were ambiguity in resolving references programmatically.

As an example, consider a short story of two characters Bobby and Bill:

Bobby thought Bill should buy a trailer and haul it with his car. Bill thought a truck would be better for what he needed. Bobby pointed out two vehicles were much more expensive. Bill was set in his ways with conventional thinking. He ended up buying the truck he wanted despite Bobby's advice.

The challenge of coreference resolution in this example is to identify the character to which the pronominal references like *he, his* and *him* are referring to at each occurrence. For this example after applying entity resolution and coreference resolution our story looks something like this:

John thought Peter should buy a trailer and haul it with Peter's car. Peter thought a truck would be better for what Peter needed. John pointed out two vehicles were much more expensive. Peter was set in Peter's ways with conventional thinking. Peter ended up buying the truck Peter wanted despite John's advice.

Each entity in the narrative has been identified and each character name has been replaced with the predefined set of names for all stories. In this case we replaced the first characters name *Bobby* with *John* and *Bill* with *Peter*. Further, each pronominal reference has been replaced with the actual character name, for instance *he* in the second sentence has been replaced with *Peter* and *his* in the third sentence with *Peters*. In the resulting story we have each action or event attached to actual character names, and no pronominal references.

At the end of the preprocessing steps, we were left with 25000 short stories that have characters limited to our short list of character names and no pronominal references. We proceeded with model training on this preprocessed dataset.

Skip Thought Vectorization

Neural networks work with numerical input and hence the processed text needs to be changed into a suitable vector form before feeding into the model for training. A common choice for vectorization is word2vec or doc2vec but they do not capture the context of the sentences. In this work, we have used the skip thought vectorization instead. Kiros at al proposed skip thought vectors that maps each sentence to a numerical vector representation (Kiros et al. 2015). Beyond statistical measures and word counts, skip thought vectors also captures the semantic and syntactic properties of a sentence, which is vital in story generation.

Skip thought vector is a generic, distributed sentence encoder that uses an unsupervised approach. It internally relies on a neural network (RNN and GRU) to map each sentence to a numerical vector representation. Each sentence text is mapped to a 4800 dimensional space representation. Similarity or dissimilarity between text is captured by the distance of each vector representation in this vector space.

For our purpose of making the story generator interactive and take user inputs, we applied skip thought vectors at two different points. First, for data preprocessing before training our model and second for converting user inputs into vectorized form at run-time before feeding the input to our model. Using skip thought vectorization to process user input adds some delay in processing.

Recurrent Neural Networks

Both entity based (Clark, Ji, and Smith 2018) and event based story generation (Martin et al. 2017) have shown some success in generating stories. However, there is scope for creating models that generate completely coherent text. We take inspiration from both these approaches and propose a representation of text that involves entities and sentence generalization. We train a sentence-level RNN to generate text based on context inspired from the work of Srinivasan et al. (Srinivasan, Arora, and Riedl 2018). The architecture of our model is demonstrated in fig 2.

Gated Recurrent Unit (GRU)

Kyunghyun Cho et al [Kyunghyun] proposed a grating mechanism for Recurrent Neural Networks (RNN) called GRU. A competing gating mechanism that also uses RNNs is Long-short Term Memory (LSTM). We used GRUs as gating mechanism in our system over LSTM, as they exhibit better performance than LSTM on smaller datasets (in terms of accuracy and computational resources) (Cho 2014). Fig 1 demonstrates a simple GRU.

Following equations describe the mathematical structure of the network.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z \tag{1}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r \tag{2}$$

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \sigma_h(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_h) \tag{3}$$

Here $\mathrm{x_t}$ and $\mathrm{h_t}$ represent the input and output vectors respectively. $\mathrm{z_t}$ and $\mathrm{r_t}$ denote the update gate vector and reset gate vector respectively. And W, U and b are parameter matrices and vectors. Activation functions used are σ_{g}, which is the sigmoid function and σ_{h} which is a hyperbolic tangent.

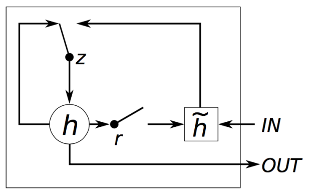


Figure 1: Gated Recurrent Unit (GRU) (Cho 2014)

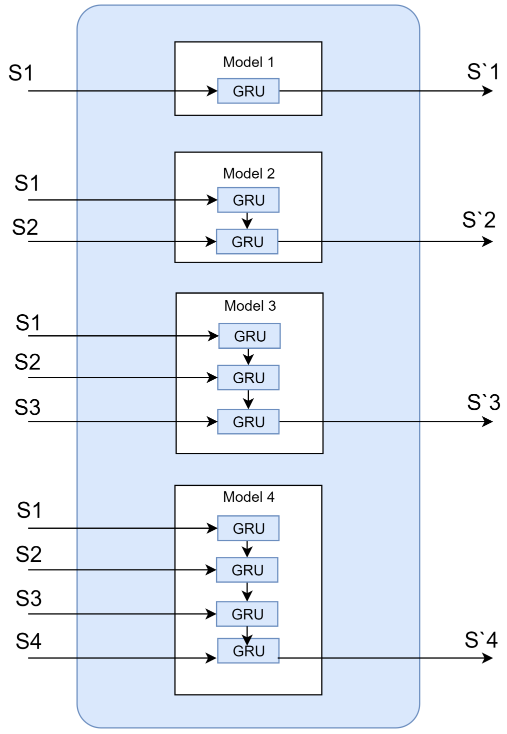
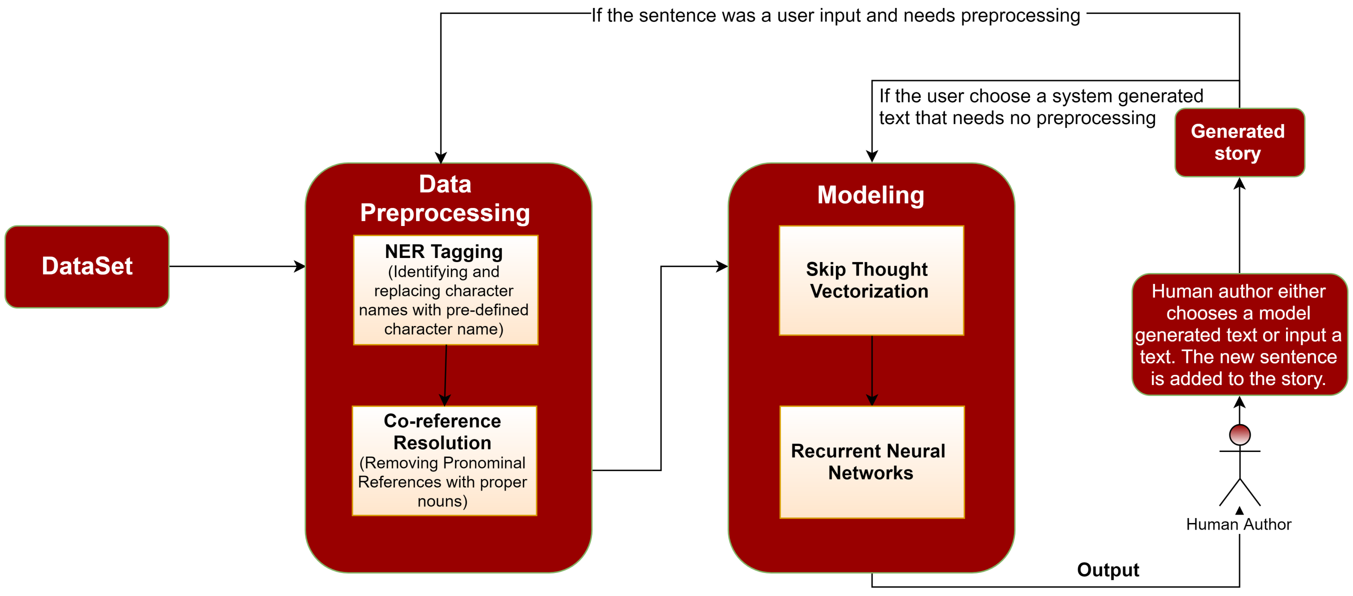


Figure 2: Story generation system architecture: The system consists of 4 RNNs, one for each sentence of the story starting from second to fifth sentence. S1-S4 are sentence inputs while S‘1-S‘2 are suggestions generated by the model. The system starts with a random sentence picked up from our corpus to initiate the story. This becomes the first input sentence S1. The model takes in vector input of sentences and provides suggestions for the next sentence in the story. Sentence 2 (S‘1) for the story is generated based on first sentence, Sentence 3 (S‘2) is generated using both S1 and S2 and so on. The suggested outputs of this model, i.e. S‘1-S‘4, form the final story.



If the sentence was a user input and needs preprocessing
If the user choose a system generated text that needs no preprocessing
DataSet
Data Preprocessing
NER Tagging (Identifying and replacing character names with pre-defined character name)
Co-reference Resolution (Removing Pronominal References with proper nouns)
Modeling
Skip Thought Vectorization
Recurrent Neural Networks
Generated story
Human author either chooses a model generated text or input a text. The new sentence is added to the story.
Human Author
Output

Figure 3: Overall System Pipeline: The data preprocessing involves NER tagging for entity resolution and coreference resolution to resolve character's pronominal reference. The model is trained on the skip thought vector representation of this preprocessed story corpus. The system starts with one random sentence to begin the story and for each new sentence prompts the user to either input a new sentence or choose from system generated suggestions. The input loops back to either the preprocessing step, if its a user input and needs to be preprocessed to maintain character coherence in the story. Or it is directly fed back to the model if the user chooses a system generated suggestion that is already preprocessed.

Results

Experiments

Table 1 lists the experimental settings of our model. We train four different RNN models with these settings. The model takes input in the form of skip thought vector of each sentence. The first model takes in first sentence and learns to predict the second sentence. The second model, takes as input, the first two sentences and predicts the third sentence. Similarly, the third model takes in three sentences and outputs fourth sentence. Finally, the fourth model takes in four sentences and predicts the last line of the story. We train on 90 % of the data and test on 10 % of the data for evaluation.

Evaluation

Figures 4, 5, 6, 7 show the results of the four models respectively. The figures plot the frequency of the ranks predicted by the model. These ranks denote the number of sentences the model thinks are more suitable instead of the right answer. For instance, in figure 1, for 700 data points, the model gives a relatively low rank. That means the model arrives at the right solution in less number of attempts.

Hyperparameters	**Values**
Epochs	15
Batch size	32
Learning rate	1e-4
Dropout	0
Biderectional	True
Layers	1
Hidden Size	4800
Loss	MSE
Update	RMSprop
Train Size	23188
Test Size	2577

Table 1: Experiment Settings

Discussion

Our system effectively generates stories that are coherent and syntactically correct. Low ranks reported in the result section's figures indicate that the model learns the sequential relationship between the sentences of the story.

We describe the entire pipeline of the architecture in detail. As our system is interactive it takes user input before finalizing any new sentence that become part of the narration. Although all the system generated suggestions do not fit well in the context of the narration but some of the top suggestions successfully incorporate the context. Our manual verification of these suggestions over some test runs suggests that our model takes into account the semantic and syntactic meaning of the story and makes suggestions accordingly.

The context of the story at any point is taken into consideration by our model and the generated suggestions are in accordance of that. We found in most cases at least 3-4 out of 10 suggestions that fit well with the context of the narration and can be picked for the story. Choosing from a list of suggestions empowers the human author to take the story in any desired direction.

Rendering interactivity to our system came at a cost of run-time delay. We noticed a constant time delay of approximately 1 minute for any user input to be preprocessed and vectroized by skip thought vectorization. This time delay is further extended by RNN by another 2 mins. This adds upto approximately 3 minutes between any two consecutive sentences that are generated for the story that takes user input.

Limitations and Future work

First limitation of the existing system is that it generates fixed length stories, i.e. five sentence short story. The story ends at the fifth sentence even if it is an absurd ending. This is because the model has been trained on ROC story dataset (Nasrin Mostafazadeh 2016) which is a collection of stories that are each five sentences. Also the architecture of our current model prevents us from extending the current system beyond five sentences. In our future work we would like to modify our architecture to train on a corpus of variable length stories, and generate stories that resolve all conflicts before concluding.

Another limitation in the current system is that it does not revert back the the pronominal references in the final story. In the existing system, all characters are referred by their names in each sentence of the story. We plan to include reverse coreference resolution in our future work on this project.

Making our system interactive not only added complexity to the problem but added delays that we could not avoid. Entity and coreference resolution along with skip thought vectorization adds an overhead of approximately a minute for each user input which is further extended by RNNs in making its suggestion. The overall time between each new sentence adds up to approximately 3 minutes which is a considerable time delay and needs to be improved. Given the limited hardware resources we have used for this work the time delays seemed unavoidable. A better vectorizaton method and hardware resources for running the system could improve the runtime.

Our system currently generates stories that are based on events in daily life and have a relatively simple language and construction. Being limited to five sentences further restricts the story against evolving into something which can include more events and conflicts. The current system generated stories do not have many characters and character conflicts to resolve. As a future work, we can train our model on a different story corpus to see how it performs when there are multiple characters and conflicts to resolve.

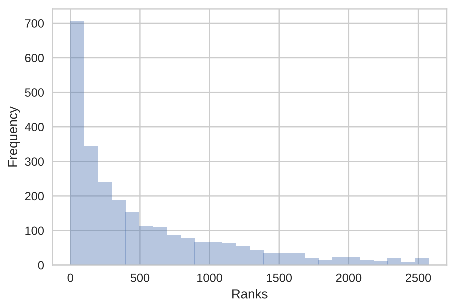


Figure 4: Ranks for model 1

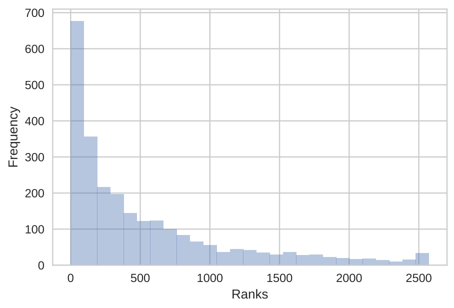


Figure 5: Ranks for model 2

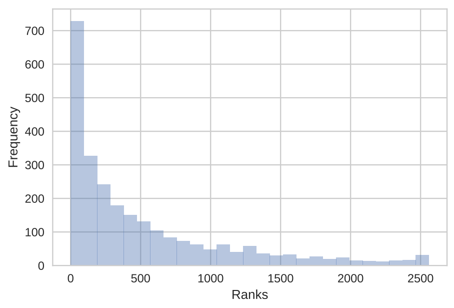


Figure 6: Ranks for model 3

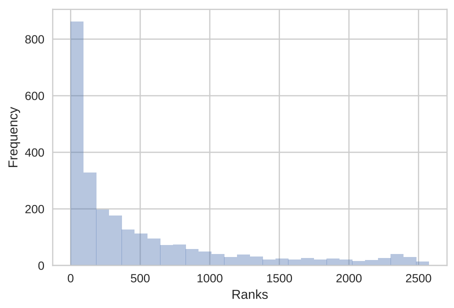
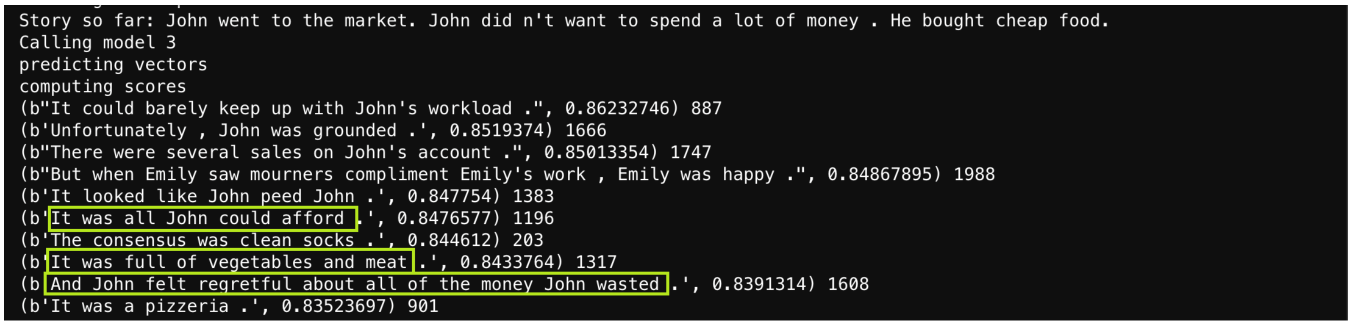


Figure 7: Ranks for model 4



```
Story so far: John went to the market. John did n't want to spend a lot of money . He bought cheap food.
Calling model 3
predicting vectors
computing scores
(b"It could barely keep up with John's workload .", 0.86232746) 887
(b'Unfortunately , John was grounded .', 0.8519374) 1666
(b"There were several sales on John's account .", 0.85013354) 1747
(b"But when Emily saw mourners compliment Emily's work , Emily was happy .", 0.84867895) 1988
(b'It looked like John peed John .', 0.847754) 1383
(b'It was all John could afford .', 0.8476577) 1196
(b'The consensus was clean socks .', 0.844612) 203
(b'It was full of vegetables and meat .', 0.8433764) 1317
(b'And John felt regretful about all of the money John wasted .', 0.8391314) 1608
(b'It was a pizzeria .', 0.83523697) 901
```

Figure 8: Screen shot of the output: This is an intermediate output for a story in generation that contains the story generated so far and a few suggestions from the system to choose from. Though some suggestions do not go with the context, there are a few suggestions (highlighted with green outline box) that fit well with the context and could take the story forward in an interesting direction.

Conclusion

We used a data driven story generation approach to generate short five sentence common sense stories. Our RNN model showed good performance in generating suggestions for story generation one sentence at a time. We also take user inputs and process it in real time. The generated stories are coherent and meaningful if we entirely rely on the system generated suggestions. The claim is validated thorugh the evaluation of the model and the reported rank frequencies.

References

Cho, Kyunghyun; van Merrienboer, B. G. C. B. D. B. F. S. H. B. Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*.

Clark, E.; Ji, Y.; and Smith, N. A. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, 2250–2260.

F. Charles, M. Lozano, S. J. M. A. F. B. M. C. 2003. Planning formalisms and authoring in interactive storytelling.

J. Porteous, M. Cavazza, F. C. 2010. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology (TIST)* 1.

Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic entity representations in neural language models. *arXiv preprint arXiv:1708.00781*.

Kiros, R.; Zhu, Y.; Salakhutdinov, R.; Zemel, R. S.; Torralba, A.; Urtasun, R.; and Fidler, S. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, 3294–3302. Cambridge, MA, USA: MIT Press.

Laurel, B., and Mountford, S. J. 1990. *The art of human-computer interface design*. Addison-Wesley Longman Publishing Co., Inc.

M. O. Riedl, R. M. Y. 2010. Narrative planning: Balancing plot and character. *Journal of Articial Intelligence Research* 39.

M. Theune, E. Faas, A. N., and Heylen, D. 2003. The virtual storyteller: Story creation by intelligent agents.

Martin, L. J.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2017. Event representations for automated story generation with deep neural nets. *arXiv preprint arXiv:1706.01331*.

Mostafazadeh, N.; Chambers, N.; He, X.; Parikh, D.; Batra, D.; Vanderwende, L.; Kohli, P.; and Allen, J. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Nasrin Mostafazadeh, Nathanael Chambers, X. H. D. P. D. B. L. V. P. K. J. A. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *In Proceedings of the 2016 North American Chapter of the ACL (NAACL HLT)*.

Pichotta, K., and Mooney, R. J. 2016. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*, 2800–2806.

Srinivasan, S.; Arora, R.; and Riedl, M. 2018. A simple and effective approach to the story cloze test. *arXiv preprint arXiv:1803.05547*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Szilas, N. 2001. A new approach to interactive drama: From intelligent characters to an intelligent virtual narrator. *Association for the Advancement of Artificial Intelligence (AAAI)*.