Nathan Dass

Story Generation with Deep Reinforcement Learning

Fall 2017

Undergraduate Thesis

Reader Signatures:

Dhruv Batra

Digitally signed by Dhruv Batra
DN: cn=Dhruv Batra, o=Georgia Institute of Technology, ou=School of Interactive Computing, email=dbatra@gatech.edu, c=US
Date: 2017.12.14 11:31:22 -08'00'

Abstract

Storytelling has applications in areas ranging from creating books and novels to engrossing movie scripts to the gaming industry. It would be useful to create interactive plot lines for games, motivate mission generation, and ordering of events for different characters. In the military, we can make the system learn from thousands of written real-world events, incidents, and missions to create interactive simulations to train army personnel. We introduce a deep reinforcement learning approach to story generation trained on a textual story corpus. Unlike other neural network based approaches to story generation, a reward function allows a human user to control the direction that the story follows.

Introduction

Automated story generation is the problem of automatically selecting a sequence of events, actions, or words that can be told as a story. Story generation is inherently a creative task. It is heavily reliant on two innately human characteristics, i.e. inventiveness and imagination. There are no clearly defined constraints or parameters that we can optimize to make a good story, it is wholly at the discretion of the readers interpretation. In this paper, we introduce a deep reinforcement learning approach to automated story generation in which the generator learns how to tell stories from large text-based corpora such as book and movie plot summaries mined from Wikipedia (Bamman, O'Connor, & Smith, 2013).

One way to teach a system how to tell stories with a corpus is to use Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). LSTM networks have recently been trained on story corpora in order to generate novel stories (Roemmele, Kobayashi, Inoue, & Gordon, 2017; Martin et al., 2017;

Khalifa, Barros, & Togelius, 2017). LSTMs can be thought of as learning a distribution over successor tokens, whether that be characters, words, or sentences. Thus, story generation with RNNs is a process of iteratively sampling from a learned distribution. However, this presents two challenges. First, LSTMs do not have any incentive to maintain context. Any sequence of more than a few sentences will tend to lose narrative coherence. Second, language models are not goal-driven. It is unlikely that they will produce a story with a given ending or structure.

Reinforcement learning algorithms learn to complete a task by receiving rewards and punishments from their environments. In the case of story generation, we reward the system for generating stories that meet a desired authorial goal while also creating stories that look like those from the training corpus. For example, we might reward the generator for generating a story in which characters meet, fall in love, and marry. Our deep reinforcement learning approach uses policy gradients to iteratively train a neural network, so that the distribution it learns over successor events increases the likelihood that the target events that occur follow the direction dictated by the user.

Background

Story creation is a planning process---it is driven by the author's goals as a storyteller (Sharples, 1999). Most story generation systems have used symbolic planning algorithms (Meehan, 1977; Lebowitz, 1987; Pérez y Pérez & Sharples, 2001; Riedl & Young, 2010) or case-based planning (Gervás, Díaz-Agudo, Peinado, & Hervás, 2005) in well-defined micro-worlds. Other approaches learn from crowdsourced corpora (Li, Lee-Urban, Johnston, & Riedl, 2013), retrieve sentences of stories from blogs (Swanson & Gordon, 2012), or use language modeling (Roemmele et al., 2017; Martin et al., 2017; Khalifa et al., 2017). Reinforcement learning has been

used to generate interactive narrative, where a player assumes the role of a character in a virtual world and can influence the state of the world through his or her actions (Wang, Rowe, Min, Mott, & Lester, 2017; Roberts, Nelson, Isbell, Mateas, & Littman, 2006). Harrison, Purdy, and Riedl (2017) uses Markov Chain Monte Carlo search trained on stories.

Our work follows Martin et al. (2017) in reducing the dimensionality of the training corpus of stories by transforming sentences into *events*, where an event is a tuple <*s*, *v*, *o*, *m*> such that *s* is the subject of the verb, *v* is a verb, *o* is the direct object, and *m* is a modifier that provides additional context, such as an indirect object, propositional object, causal complement, or unknown dependency. Martin et al. (2017 show that replacing named entities with generic placeholders, using WordNet (Fellbaum, 1998) synsets for the subject, direct object, and modifier, and using VerbNet (Schuler, 2005) frames for the verb further improves generation. They also provide a technique for translating events back into natural language. Events are also useful for user interactivity. A user can change the state of the story by introducing a new sentence. Once this sentence is translated to an event, it is more likely to correlate to known events that the system has been trained on because there is less variability in events.

Methods

Our technique shares similarities with policy-gradient reinforcement learning approaches to dialogue generation (Li et al., 2016). Whereas reinforcement for dialogue generation uses the reward function to correct for consistency, coherence, and repetition during neural net decoding, our approach incorporates external authorial goals into the reward function.

Our technique starts with a sequence-to-sequence network (Sutskever, Vinyals, & Le, 2014) with LSTM cells that has been pre-trained on our story corpus. This model approximates

the distribution over potential next events given a current event. The set of actions in our domain space is the set of all possible events, with no legality constraints on what sentences can follow each other. In other words, this sequence-to-sequence network generates the next event that is most likely to happen based on the training data without directly accounting for syntax and semantics. We use a variation of policy-gradient search developed by Sutton, McAllester, Singh, and Mansour (2000) because of the size of our action space. See Figure 1 for an illustration of our training process.

The training process can be thought of as iteratively shifting the distribution over successor events from the pre-trained network to one that increases the likelihood of generating rewarding events. We reward the system for generating stories that contain a given target verb, such as "meet", "fall in love" (which turns into "admire" in VerbNet), or "marry". The user can input any number of target verbs, which gives the user overarching control over the direction of the stories that are generated.

To train the policy network, we iteratively choose events from the training corpus and sample from the distribution of potential next events which is approximated by the network. We calculate a reward for the chosen event and multiply it by the probability of that event being the next event. This value is backpropagated through the policy network.

Rewards are sparse in our domain and our technique will only learn if it sees rewarding events often. Early in training, we periodically replace the expected output with a target event (which meets the user's intent) so that it yields a positive reward. This has the effect of rapidly shifting the model early on and allowing it to slowly revert. The resulting model balances following the original pre-trained distribution and generating events that drive the story towards a target

event. As training continues, we decrease the frequency of which we replace the expected output which decreases the amount of artificially produced positive rewards.

Reward is calculated as follows. Unlike prior policy gradient methods that use the likelihood of outputs sampled from an RNN, we estimate the Bayesian probabilities from our training data and utilize these probabilities. The objective of our system is thus to maximize Equation 1.

$$\frac{1}{n}\sum_i R_i \frac{p(y_i|X_i, X_{i-1}, X_{i-2}) \prod_j p(y_{i,k}|y_{i,j})}{p(y_i|X_i, X_{i-1}, X_{i-2}) + \prod_j p(y_{i,k}|y_{i,j})} \quad (1)$$

In this function, n is batch size, which was 10 in our experiments. i iterates through each element in the batch. For each event y_i, we generate an authorial reward of R_i where R_i is set to an arbitrarily high value (currently 10,000) if the verb in y_i equals the current target verb and is set to 1 if the verb in y_i is not equal to the target verb. X_i, X_{i-1}, X_{i-2} denotes the three events of the story that occur before event y_i, so $p(y_i|X_i, X_{i-1}, X_{i-2})$ calculates the probability of the current event occurring given the previous three events in the training data. We call $p(y_i|X_i, X_{i-1}, X_{i-2})$ the vertical probability because it calculates the probability of the current event with respect to previous events. On the other hand, $\prod_j p(y_{i,k}|y_{i,j})$ estimates the probability of the k^{th} token in the current event given the previous j tokens in the same event where k and j are bound by the size of the event tuple. We call $\prod_j p(y_{i,k}|y_{i,j})$ the horizontal probability because it calculates the probability of specific elements of the event with respect to previous elements of the event. For example, the horizontal probability calculates the probability of all direct objects given a pair of nouns and verbs. By substituting $p(y_i|X_i, X_{i-1}, X_{i-2})$ for V and $\prod_j p(y_{i,k}|y_{i,j})$ for H, we can simplify our objective to Equation 2.

$$\frac{1}{n}\sum_i R_i \frac{V*H}{V+H} \quad (2)$$

The horizontal and vertical probabilities create two separate distributions that encapsulate coherence within the same event and across multiple events when combined. By multiplying the two vertical and horizontal in the numerator and adding the vertical and horizontal probabilities in the denominator, we are able to reduce the fraction to 0 if either of the probabilities are 0. This strongly pushes the model towards the target verb if an event contains the target verb and has a relatively high ratio of the product to the sum of vertical and horizontal probability. The use of Bayesian probabilities gives us the ability to estimate the probability of the event with respect to the story corpus instead of just the previous event. When put together, this function allows us to utilize more context than the previous event while still being able to drive the story towards a target verb. In cases where the reward falls short of a predefined threshold, the model calculates the loss with respect to the output of the pre-trained sequence-to-sequence model.

Results

Tables 1-4 show outputs that are generated by our model. In Table 1, the user had no control over the direction of the story, but in Tables 2-4, the target verbs (meet, admire, and marry) were chosen by the user. In tables 2-4, events generated by the reinforcement learner are also converted to natural language using the technique described in Martin et al. (2017).

To generate the story, the user enters the first line and then the next line of the story is generated. The user can either use the outputted line as the input for the next line or input a new line. This gives the user control over the story and is what makes our approach interactive.

Our interpretation of Table 1 is as follows. There is a group of assassins that is hard to believe exists. One of these assassins sends a death threat to Jack in the form of written communication. Jack is appalled that the assassin would act in such a way. The assassin gives a

speech to the rest of the assassins and tells them about the demands that Jack must meet. Jack offers to fulfill the demands and asks the assassins to not kill him in return. The head of the assassins accepts Jack's conditions and decides not to kill Jack.

It should also be noted that the models used to generate the stories in Tables 2-4 were trained on one third of the full corpus while the story in Table 1 was trained on the full corpus. In the interest of time, the models used for Tables 2-4 were also trained for half the number of epochs as the model used for Table 1. Since the models are not as developed, the generated stories appear less coherent and do not make as much logical sense.

We note that our model does not always follow proper grammar and does not always generate tuples that make sense. For example, the first tuple in Table 1, <hard, assassins, believe, Synset('group.n.01')>, does not make sense in the order that it is generated. However, even if the tuple was <assassins, believe, hard, Synset('group.n.01')> instead, it would make syntactical sense and continues the flow of the rest of the generated story. Our model is not always accurate with the details, but it accomplishes our goal of creating stories that make sense on a high level and also allow the user to control the direction of the story.

Discussion

An important distinction between the method we proposed and the vanilla policy gradients technique is the process of incorporating the likelihoods of the events that are being rewarded. Instead of directly using the likelihoods outputted by the RNN, we chose to estimate Bayesian probabilities from our training data and utilize these probabilities. Unlike in the case of a dialogue generation system, where this method has been employed previously, an important attribute of stories is that they need to be contextually coherent. The use of Bayesian probabilities affords us

the ability to estimate the probability of the event with respect to the whole story instead of just the previous line.

We note that the generated natural language stories in Tables 2-4 do not make syntactical sense, that there can be other interpretations of the events in Table 1, and that our interpretation of the events in Table 1 can also originate from other combinations of events. We provide the natural language stories and the interpretation to give insight into what current generated stories look like and into what future generated stories might look like with further improvements on the model. Before we train our models, we transform our corpus from sentences to events. This makes it easier for our model to learn a good distribution as the input and output lengths are fixed and we were able to incorporate vertical and horizontal statistics while training the model. At the same time, we are losing information. Using events also makes it easier to train models on datasets that are not in English as we just need to convert the input into fixed-length events. While events are beneficial, they make it much harder to maintain this information for reuse when using the method developed in Martin et al. (2017) to convert events back to natural language.

We are also considering various evaluation methods for our model. Initially, we focused on creating a working architecture with a basic reinforcement learner before implementing the learner that we currently have. To evaluate the learner, we thought of using automatic and manual methods. We initially thought of three metrics: lengths of stories, vocabulary sizes, and cohesion. Comparing lengths of stories allows us to check that our model is not ending stories abruptly or sooner than actual stories. Comparing vocabulary size between the computer-generated stories and human-written stories to check that the computer-generated stories has the same level of word complexity. Comparing cohesion would ensure that our model is actually making grammatical sense. Ensuring we have high cohesion is necessary because we are not trying to solve the simpler

problem of generating arbitrary stories, but we want to solve the problem of generating stories that make contextual sense.

While we were originally planning to implement these methods for automatic evaluation, we decided to create a more exhaustive list of methods first and then implement the evaluation methods of the larger list. So far, our automatic metrics include reincorporation, avoiding objectionable content, maintaining an appropriate number of characters and developing character profiles throughout the story, adhering to syntax, incorporating drama into the story where the level is adjusted based on the genre of the story, proper logical ordering, and local contextuality. Reincorporation is the reuse of old content or characters, which can be incorporated into our reward function and measured by keeping a counter for characters, verbs, or even all words. By avoiding objectionable content, we ensure readers do not get offended when reading a computer-generated list. For this, we simply plan to have a blacklist of words not to use. The number of characters as well as the amount of drama in a story can vary depending on the genre and we can enforce this by learning the distributions based on the genre and use that distribution as part of the reward while training. Adhering to syntax can be evaluated through existing syntax parsers and keeping track of the score. Logical ordering can be tracked by creating a graph from the training and testing stories and checking if the generated story follows that graph. Local contextuality is already part of our reward function and can be calculated by finding the probability of an event occurring given a window of size n around it.

For manual evaluation, we plan to have people read a computer-generated story and a human-written story and try to distinguish the computer-generated story from the human-written story. Human evaluations provide the best insight into whether or not a generated story is indistinguishable from a human-written story, but they are very expensive and time-intensive.

Using the quantitative and qualitative results collected, we can modify our model until the computer-generated stories cannot be reliably distinguished from human-written stories and achieve our goal of generating interesting, cohesive stories.

Conclusion

While this is very early work, we present a deep reinforcement learner that shows signs of maintaining a good high-level flow following the direction that the user dictates through target verbs and authorial reward based on our initial results. Before we can generate stories that are indistinguishable from human-generated stories, we have a lot to improve in terms of what the deep reinforcement learner learns from and encapsulating more information in events so that converting events back to natural language is more coherent. We see this deep reinforcement learner as the basis of more complicated deep reinforcement learners that better incorporate previous sentences to further improve the flow of generated stories.

Future Work

While our model is capable of producing creative stories in the form of events, there is still a lot of room for improvement and further development. Since we convert stories from plain English to the event format that was mentioned, the model outputs stories in the same event format. A lot of information is lost through this conversion, but it also makes it significantly easier for the model to create long-term dependencies and generate better stories on a higher level. We are working with Martin et al. (2017) to improve the conversion of events back into natural language sentences. This integration would create a full end-to-end system that takes in an English sentence and produces the following English sentence in a story.

Additionally, we trained our policy model on a reduced version of the CMU Movie Plot Summary corpus (Bamman et al., 2013) to save time. This likely came at the cost of coherence and grammar. As seen in Tables 2-4, the generated sentences often do not follow all grammar rules. We expect that training on the full corpus will allow for more diversity in the generated events as well as better overall coherence.

We are also exploring ways of improving our deep reinforcement learner. Currently, we only take the previous three events into account when training, but we would like to be able to extend this to a dynamic number of previous events. With this, our stories would have better narrative arcs as the story is not abruptly pushed towards the target verb, but is rather pushed in the right direction.

Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. W911NF-15-C-0246. This work was pursued in collaboration with graduate students Pradyumna Tambwekar, Murtaza Dhuliawala, and Animesh Mehta under the guidance of Professor Mark Riedl.

References

D. Bamman, B. O'Connor, and N. A. Smith, "Learning latent personas of film characters," in *Proceedings ACL2013*, 2013.

C. Fellbaum, *WordNet*. Wiley Online Library, 1998.

P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás, "Story plot generation based on cbr," *Knowledge-Based Systems*, vol. 18, no. 4, pp. 235–242, 2005.

B. Harrison, C. Purdy, and M. O. Riedl, "Toward automated story generation with markov chain monte carlo methods and deep neural networks.," in *Proceedings of the 2017 AAAI Workshop on Intelligent Narrative Technologies*, 2017.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

A. Khalifa, G. A. Barros, and J. Togelius, "Deeptingle," *arXiv preprint arXiv:1705.03557*, 2017.

M. Lebowitz, "Planning stories," in *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, pp. 234–242, 1987.

B. Li, S. Lee-Urban, G. Johnston, and M. O. Riedl, "Story generation with crowdsourced plot graphs," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, (Bellevue, Washington), July 2013.

J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *arXiv:1606.01541*, 29 Sep 2016.

L. J. Martin, P. Ammanabrolu, W. Hancock, S. Singh, B. Harrison, and M. O. Riedl, "Event representations for automated story generation with deep neural nets," *CoRR*, vol. abs/1706.01331, 2017.

J. R. Meehan, "TALE-SPIN: An interactive program that writes stories," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pp. 91–98, 1977.

R. Pérez y Pérez and M. Sharples, "MEXICA: A computer model of a cognitive account of creative writing," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 13, pp. 119–139, 2001.

M. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *Journal of Artificial Intelligence Research*, vol. 39, pp. 217–268, 2010.

D. L. Roberts, M. Nelson, C. Isbell, M. Mateas, and M. Littman, "Targeting specific distributions of trajectories in MDPs," in *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

M. Roemmele, S. Kobayashi, N. Inoue, and A. M. Gordon, "An rnn-based binary classifier for the story cloze test," *LSDSem 2017*, p. 74, 2017.

K. K. Schuler, "Verbnet: A broad-coverage, comprehensive verb lexicon," 2005.

M. Sharples, *How We Write: Writing as Creative Design*. London: Routledge, 1999.

I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in Neural Information Processing Systems*, vol. 12, p. 1057–1063, 2000.

R. Swanson and A. Gordon, “Say Anything: Using textual case-based reasoning to enable open-domain interactive storytelling,” *ACM Transactions on Interactive Intelligent Systems*, vol. 2, no. 3, pp. 16:1–16:35, 2012.

P. Wang, J. Rowe, W. Min, B. Mott, and J. Lester, “Interactive narrative personalization with deep reinforcement learning,” *The Twenty-Sixth International Joint Conference on Artificial Intelligence*, vol. 3852-3858, 2017.

Appendix

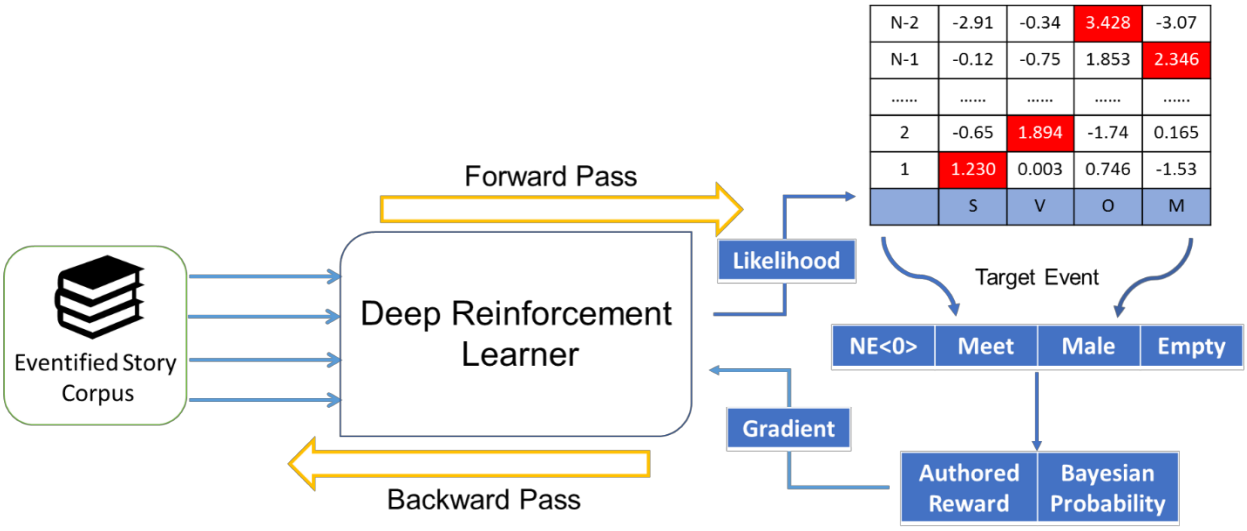

Figure 1. Deep Reinforcement learning pipeline. A corpus containing sequential events from stories is used to train the model. The model generates a likelihood over all words seen during training. A target event is sampled. Reward is computed on the sample with respect to the input event and combined with the loss calculated with respect to the target event. Gradient is computed on this loss and backpropagated.

<start_of_story>
hard, assassins, believe, Synset('group.n.01')
Synset('person.n.01'), send, Synset('written_communication.n.01'), <NE>0
<NE>0, appall, Synset('entity.n.01'), Synset('act.n.02')
Synset('person.n.01'), give, EmptyParameter, Synset('social_group.n.01')
<NE>0, offers, return, which
Synset('male.n.02'), accept, <NE>0, Synset('event.n.01')
<end_of_story>

Table 1. An example output from our system. Each row after the third row shows the model generating events by using the previous row as the input.

Input	Target Verb	Raw events (above) and Generated sentence (below)
Jack and Jill went up the hill.	Meet	1. <NE0>, meander-47.7, EmptyModifier, Synset('geological-formation.n.01) 2. <NE>1, meet-36.3-1, meander-47.7, EmptyModifier
		Jack and Arlyn go to the hill.That time Jack and Arretta meet up with the Remorseful.
That time Jack and Arretta meet up with the Remorseful.	Admire	1. EmptyObject, EmptyObject, EmptyObject, <NE>1 2. <NE>0, admire-31.2, EmptyObject, Synset('subjugation.n.01')
		One duration actual and Jack cries as Nachman Elana Whom Jack believes to be a surrender.
Jack believes to be a surrender.	Marry	1. Synset('person.n.01'), forbid-67-1, EmptyObject, Synset('genitor.n.01') 2. <NE>0, marry-36.2, Synset('odd-toed-ungulate.n.01'), EmptyModifier
		The two I are unable to stop Jack from his parent. Jack and Dimple marry a rhinoceros in the pianism.

Table 2. Input and story generated using our system. Each row shows the model generating events, which leads to an event containing the target verb and ends on that sentence. We have the sentences translated from the event representations along with the corresponding event representation shown here.

Input	Target Verb	Generated sentence
John rode his horse to the bank.	Meet	John theseus the horse off the bank. The horse proves himself as a lawn chair entrepreneur. The give phase the reassigns cat. Oilman possesses John Kelley s lover Loneta Denise. Together bank slices the kidnaps uniform.
Together bank slices the kidnaps uniform.	Admire	John is confronts to owes the main bank of the improvised snuffler in the stain.
John is confronts to owes the main bank of the improvised snuffler in the stain.	Marry	The phony faller then arrives back to the that. The that marry a great anger in hate.

Table 3. Input and story generated using our system. Each row shows the model generating events which leads to an event containing the target verb and ends on that sentence. The last column shows the natural language translations for the events that are generated.

Input	Target Verb	Generated sentence
Ryan walked to the beach with his guitar.	Meet	The craftsman and Ryan Monte are traveling to the secure beach. The critic will be able to quit his own rings beverly out. Ryan and Wade born metempsychosis the city s pickup on space. Ryan klutzy serious the admiring voice. Ryan and Fernando decide to be set. Where she meets she with Plotting.
she meets she with Plotting.	Admire	When he catches murky the acquaintance. Ryan turns to display the euthanasia.
Ryan turns to display the euthanasia.	Marry	The that wins Ryan with Ryan. When Ryan became directed concentration marital status to Antone.

Table 4. Input and story generated using our system. Each row shows the model generating events which leads to an event containing the target verb and ends on that sentence. The last column shows the natural language translations for the events that are generated.