HorrifAI: Using AI to Generate Two-Sentence Horror

Adam Keppler
Department of Computer Science
Stanford University
akeppler@stanford.edu

Jennie Chen
Department of Computer Science
Stanford University
jchen437@stanford.edu

Abstract

Natural language generation is a rapidly evolving field that has already been applied to various domains, from the practical - like summarization - to the more whimsical - like recipe creation; however, one area in which research has been fairly limited is that of horror fiction. In this project, we address the task of generating two-sentence horror, a form of flash fiction in which a self-contained horror story is told in only two sentences. Using models from the OpenNMT toolkit [1], we demonstrate the efficacy of current text generation approaches in a new domain with significant constraints, such as a distinctive style as well as a full narrative arc in a limited scope. In addition, we explore a two-pronged approach to the evaluation of our generated text by addressing language fluency and story cohesiveness separately. We find that standard Seq2Seq models used as retrieval models manage to achieve high-levels of readability in terms of grammar and spelling but often struggle with generating endings that are relevant to the beginning of the story.

1 Introduction

The vast strides that have been made in fiction generation tasks, from story continuation to image-based story generation are remarkable. However, there has been little work done on the specific subdomain of horror fiction. Perhaps the only attempt so far is Shelley, the AI built by the MIT Media Lab to create horror snippets as a continuation of a prompt or random seed [2]; however, no work has been published on the models behind Shelley and little is known about its design. In this report, we document an approach to generating two-sentence horror. Two-sentence horror is a form of flash fiction in which a self-contained horror story is told in only two sentences. This medium provides several challenges. The generated output must maintain coherency and complete a full narrative arc with an impactful and often unexpected conclusion in the limited scope of two sentences.

We use a supervised learning approach and define our task as the following: given an initial sentence, train a model to generate a second sentence that completes the two-sentence horror story coherently - in a fluent fashion - and cohesively - in such a way that the topic and content of the input and generated sentence are related. We approach this task by using an NMT style Seq2Seq approach using the OpenNMT toolkit [1], thus leveraging the techniques used in the better-defined field of NMT. In keeping with standard seq2seq practice, we initially aimed to train a sequence model, with high language modeling capabilities. We found that due to our dataset's limited size, our model is unable to reach a reasonable level of coherence under conventional methods; however, when allowed to overfit extensively on the training set, our output becomes much more readable. We therefore train our models to behave in a manner more akin to retrieval models than to purely generative models. Despite this, we find that our models are able to generate new creative sentences with previously unseen n-grams by combining existing n-grams. Our work provides useful insight into the effectiveness of an NMT-style approach for the relatively new domain of horror text generation.

2 Related Work

Little work has been done on horror-specific fiction generation, or even horror-specific content generation of any media type. The task of horror-related image generation was tackled by the MIT Media Lab in 2016 with the Nightmare Machine [3], one of the first applications of AI to the horror subdomain. More recently, the MIT Media Lab created *Shelley* [2], described as the "first collaborative AI Horror Writer". Trained on stories from the subreddit r/nosleep [4], Shelley takes inspiration from a random seed or from a prompt (often written by various Twitter users) and works to generate a horror story. Unfortunately, the MIT Media Lab has not released or published many details about the approaches behind either of their projects.

While the horror domain itself has not been well explored, more work has been done in the field of fictional story generation. The work of Fan et al. in story generation demonstrates an approach to generating multi-sentence output given only a simple prompt [5]. Noting that standard seq2seq models have trouble with creating stories relevant to the prompt, Fan et al. introduce a fusion mechanism by training a new seq2seq model on top of a second pre-trained model, with the idea that the first model focuses on language modeling while the second model links generated stories to the prompt. Due to the length of their stories, Fan et. al use a convolutional architecture to encode their text; however, as we focus on text of only two sentences, we don't believe this is necessary for our work. Fan et Al also highlight the challenges associated with open-ended text evaluation, acknowledging that many common metrics such as BLEU and ROUGE are not useful for such an open-ended task. Instead, they pursued an approach using model perplexity to measure language model quality and prompt ranking accuracy to measure the relevance of the output to the input. While these metrics proved effective for their task, they do not transfer particularly well to our work.

3 Dataset

We created a script utilizing the API for PushShift.io - a service dedicated to archiving Reddit [6] - to scrape posts from the subreddit /r/TwoSentenceHorror [7]. We preprocessed our data by replacing Unicode characters with ASCII, filtering out deleted posts, non-story posts, and stories with two many or too few sentences. Examples were then split such that the first and second sentences were in separate files. After preprocessing, our dataset contains roughly 22,000 examples, which we randomly divided into training, validation, and test sets with a split of 85/7.5/7.5 respectively.

An additional dataset was generated to train a neural classifier for the evaluation of two-sentence horror story cohesion. We define a sentence pair as cohesive if the second sentence is a valid continuation of the story arc created by the first sentence; "validity" includes consideration of the subject, the entities in the story, and the relevance of the second sentence to the first sentence. The "cohesive" class contains the original 22,000 examples. For the "not cohesive" class, the second sentences were permuted across examples, creating texts that were grammatically sound but theoretically not cohesive. This allowed us to produce a balanced dataset with both classes using the same vocabulary, forcing our classifier to pick up on context rather than word usage. This dataset contains roughly 44,000 examples and is split like the other dataset.

As a note, we make the assumption that all human-generated two-sentence horror stories are cohesive. Although this may not be true for all examples, we feel it is a fair assumption to make given that we lack the resources necessary to filter and hand-label our data.

4 Approach

Our original approach involved the use of a NMT-style seq2seq approach, trained with teacher-forcing, to learn a language model from our training data and use it to generate output. Using early stopping, we used the model that gave the lowest perplexity on our validation set. However, upon inspecting our output, we found that it was largely incoherent, with the model often just generating the same short phrases over and over within a sentence or generating the same sentence for nearly all examples. We suspect that these results were due to our small dataset, which only had 22,000 examples, whereas most NMT-style approaches are trained on the order of at least hundreds of thousands of examples, if not millions of examples [8].

Given that our dataset was not sufficiently large to train a fluent language model that had readable and diverse output, we decided to allow the model to overfit to the training set and behave more like a retrieval model. By fitting so closely, our seq2seq models learn the exact phrases used in the training data; at generation, they create output that is a combination of these phrases, sometimes so effectively as to exactly match a training example. While this approach is likely not as powerful as traditional generation approaches, it demonstrates another technique for generating readable and cohesive output with a limited dataset and is overall effective for this specific task. We explored a variety of architectures and decoding algorithms to identify the optimal configuration for our task.

4.1 Baseline

We created a non-neural retrieval model baseline using Jaccard similarity. Jaccard similarity is calculated as the ratio of unique overlapping words between the two sentences to the number of unique words across the two sentences; in more formulaic terms, if A is the set of words in the first sentence and B is the set of words in the generated sentence, the Jaccard similarity is calculated as

$$Jaccard = \frac{|A \cap B|}{|A \cup B|}$$

where $|S|$ represents the number of elements in set S. Since cohesive stories tend to share entities across sentences, we believe that good stories should have a high Jaccard similarity. Therefore, the baseline generates output as follows: for each input sentence, output the second sentence of a two-sentence horror story in the training data that has the highest Jaccard similarity with the input.

4.2 LSTM

Our first experimental model uses the default configuration of the OpenNMT model [10], a 2-layer unidirectional LSTM encoder and decoder with multiplicative attention, where both the encoder and the decoder have 500 hidden units. The model uses 500-dimensional word embeddings, and the LSTM stack uses dropout 0.3.

4.3 Bidirectional LSTM

Our second experimental model is the same as the previous model, but uses a bidirectional encoder.

4.4 Transformer

Our third model is the OpenNMT implementation [10] of Vaswani et al.'s Google's transformer model for English-German and English-French translation [9]. The model uses six-layer transformer encoder and decoders, each with 512 hidden units. The transformers use eight attention heads with scaled dot-product self attention as well as a position-wise feed-forward network layer with 2048 units and a dropout probability of 0.1. The model also uses 512-dimensional word embeddings, with a sinusoidal position encoding added to the word embeddings to ensure that positional information is available to our non-RNN architecture. All model parameters use Xavier initialization.

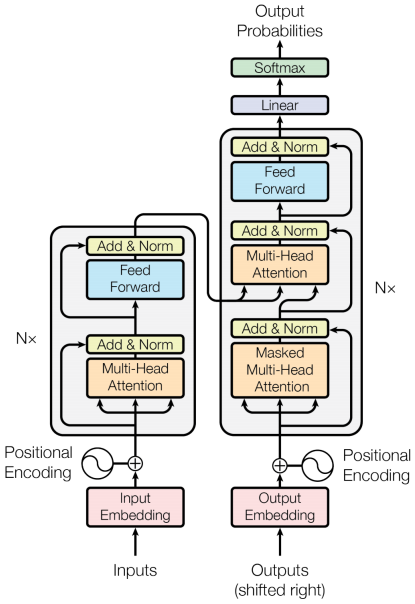


Figure 1: *Transformer model architecture; taken from Vaswani et al. [9]*

4.5 Decoding Algorithms

We experimented with several different decoding algorithms for each of our models. Our first approach was random top-k sampling, where at each generation step we perform random sampling from the k most-likely next tokens; this was performed for k = 3, 5, 8, and 10. With random top-k sampling, we also experimented with the random sampling temperature, which was used to divide logits before computing softmax to find token probabilities. Higher temperatures led to smaller logits, which allowed the decoding algorithm to have more diversity in sampling, but were also likely to

produce less coherent outputs with more mistakes. We experimented with temperatures of 0.5, 0.75, and 1.0. In addition, we experimented with using beam search with beam sizes 3, 5, 8, and 10. Beam search explores the k most probable translations throughout decoding, where k is the beam size. Higher beam sizes keep track of more partial translations, and thus tend to produce more coherent and output at the cost of being more general and likely less relevant.

5 Experiments

5.1 Model Training

For the LSTM and the BiLSTM, the learning rate dynamically adapts over 100,000 training steps; initially set to 1.0, it begins to decay after the first 50,000 steps at a rate of 0.5 every 10,000 steps. The models were trained using stochastic gradient descent and a negative log likelihood loss function with batch size 64.

For the transformer model, the learning rate dynamically adapts over 30,000 training steps. Initially set to 2.0, the model uses the Noam decay method with a warmup of 8,000 steps, which adapts the learning rate according to the following equation

$$lr = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

where d_{model} is the embedding dimension. The learning rate is increased linearly for the first 8,000 steps; it is then decreased proportional to the inverse square root of the step number. This model was trained using a negative log likelihood loss function and an Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.998$. We use dynamic batching with batch size 4096; gradients are computed over 2 batches.

5.2 Quantitative Metrics

5.2.1 Non-neural Metrics for Output Evaluation

For evaluating the coherency of our output, we explored a variety of metrics used in other text generation research [11]. We calculated the average length of the generated sentence, used two rule-based grammar checkers - Language Tool [12] and Proselint [13] - to analyze the number of grammatical errors in our generated output, and explored lexical diversity - calculated as the number of unique words relative to the total number of words across all generated sentences - to evaluate writing quality; for all of these metrics we considered only the second sentence of each story. However, we determined after careful analysis that none of these metrics were especially useful. Sorting the generated sentences by their score for each metric, we examined the examples with the best and worst scores and found little correlation between score and either fluency of language or story cohesion. As a result, we did not utilize any of these metrics in our final evaluation.

One metric that we did find to be useful was Jaccard similarity; on our generated output, we found that examples with a higher Jaccard similarity were generally more cohesive overall than stories with a low Jaccard similarity. Given that Jaccard similarity increases when common words are used across the two sentences, we believe that this makes sense; sentences with the same pronouns, proper nouns, or nouns are more likely to relate to the same topic or subject.

5.2.2 Neural Cohesion Classifier

We explored two neural classifiers to determine if our generated output was cohesive. An important element of our domain is ensuring that the first and second sentence are highly cohesive, and while Jaccard similarity provided us with some insight, we wanted a more powerful and comprehensive evaluation system. We feel that cohesion has a high correlation with the overall quality of the story and that a model able to accurately detect cohesion may serve as a good evaluator of generation quality. To evaluate the success of our classifiers, we used accuracy as our metric.

We began by exploring a CNN model based on Denny Britz's implementation [14] of Yoon Kim's architecture for sentence classification [15], with modifications to allow for the usage of pre-trained word embeddings [16]. We trained a CNN classifier on our dataset of "cohesive" and "not cohesive" examples. The input is a sentence pair which is passed through an embedding layer with pre-trained 300-dimensional GloVe embeddings trained on the Wikipedia 2014 data [17]. The

architecture then uses 3 convolutional layers with filter size 3 and ReLU activation; each one is followed by a max-pooling layer that condenses the feature map down to a single feature vector. Finally, the outputs of each max-pooling layer are combined to form a single set of features used in the softmax layer to predict whether the pair is considered "cohesive" or "not cohesive". To train this classifier, we used an Adam optimizer with a learning rate of $1e{-}3$, a cross-entropy loss function, and L2-regularization with $\lambda = 0.1$. Unfortunately, our CNN classifier was unable to achieve particularly effective performance. Although it was able to overfit to the training data, even with regularization the model only achieved a peak validation accuracy of about 51%, barely better than random guessing.

Due to these results, we pivoted away from this model; considering the similarity between our task of cohesion detection and the task of natural language inference (NLI), we decided to instead adapt Facebook's InferSent model [18] as our classifier. InferSent uses a self-attentive bidirectional LSTM sentence encoder with max-pooling to generate sentence embeddings [18]. Relations are then extracted from these sentence embeddings and fed through 3 fully connected layers, producing a predicted class. We began by training the InferSent model on our classification dataset using SGD, which was able to achieve a peak validation accuracy of about 58%, a noticeable improvement of our CNN classifier. To further improve the model performance, we built on our assumption that cohesion evaluation was closely related to NLI and pre-trained the InferSent model on the SNLI dataset [19]. We then modified the model architecture by interleaving two dropout layers between the fully connected layers and introduced L2 regularization with $\lambda = .001$; our modified architecture is shown in Figure 2. With these modifications, we achieved a training accuracy of 78.4% and a validation accuracy of 70.6%. This final model, which we use as our cohesion classifier for generated output, achieved 70.1% accuracy on our classification test set.

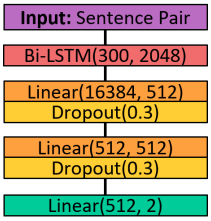


Figure 2: *Modified architecture of our InferSent model*

5.3 Qualitative Metrics

Due to the limitations of our automatic metrics, we created a survey to provide us with further insight from human evaluation. In our survey, we included 25 samples of two-sentence horror, with 5 samples selected from each of our five "models": human-written, Jaccard-based, LSTM, BiLSTM, and Transformer. These samples were generated (or in the case of human-written stories, selected) from our test set. For each category, we chose three samples through random selection. The remaining two were selected by choosing 10 random samples and selecting what we considered to be the best one, to demonstrate the higher end of what the models could generate. For each sample, we asked respondents whether they believed the story to be written by a human or an AI, with a confidence level from 1 to 5. We also asked respondents to rate the overall readability, cohesion, similarity to horror, scariness, and general quality of each story on a scale of 1 to 10, with 1 as strongly negative and 10 as strongly positive. For specific questions asked in the survey, please see Appendix A.

6 Results and Analysis

6.1 Model and Algorithm Selection

6.1.1 Retrieval Model Evaluation

To evaluate our retrieval models, we explored exact match occurrence and n-gram occurrence. We define exact match occurrence as the number of generated outputs that exactly match a sentence in the training dataset. Similarly, we define n-gram occurrence to be the percentage of n-grams in the generated output that appear in the training dataset.

As we deliberately overfit our models to create retrieval models, we used these metrics to find the stopping point for model training. For each of our models, we used each of our decoding algorithm variations to generate output. This was done every 2500 steps for the LSTM and BiLSTM models, and every 500 steps for the transformer model. We then calculated the exact match occurrence and n-gram occurrence for $n \in \{2, 3, 4, 5, 6\}$ every generated output set. We then chose the combination with the highest overall exact match occurrence and n-gram occurrences, as we believe that this is a good indication that the model is doing well at effectively retrieving phrases from the training data.

6.1.2 Decoding Algorithm Selection

From our model evaluation, we found that random top-k sampling was often significantly less readable but had more diverse output, especially with higher temperatures; in comparison, beam search across all beam sizes gave us output that was both readable and reasonably diverse. We did not observe any particular improvement in cohesiveness when using random top-k sampling instead of beam search, but such an improvement may have been lost with the general decrease in readability. Given our findings that beam search generates significantly more readable output, we opted to use beam search as our decoding algorithm.

When examining the differences between beam search sizes, we noticed that as the beam size increased, n-gram and exact match occurrences also increased. In keeping with our belief that an effective retrieval model would produce the most human-like and readable output, we selected beam search with size 10 as our standard decoding algorithm as it reliably provided what we considered to be the most effective retrieval performance.

6.2 Quantitative Results and Analysis

Model	Avg. Jaccard	2-Gram Occ. %	3-Gram Occ. %	4-Gram Occ. %	5-Gram Occ. %	6-Gram Occ. %	# Exact Match	% Cohesive
Human	0.068	*76.92*[1]	*39.79*	*15.55*	*6.47*	*3.31*	*30*	78.76
Jaccard	0.271	*100*	*100*	*100*	*100*	*100*	*1864*	85.78
LSTM	0.052	94.12	71.42	42.78	25.34	16.13	166	61.16
BiLSTM	0.055	95.03	74.71	46.46	27.55	17.46	236	61.00
Trans.	0.046	99.68	96.80	92.94	90.72	89.82	1331	47.16

Table 1: *Model performance for automated metrics on test set*

Each model was run on the test set to generate output, which was then evaluated by our metrics. The results indicate that the transformer model has the best performance as a retrieval model; nearly every 2-gram in its generated output is also present in the training data, and the percentage only drops slightly for 6-grams, in contrast to the LSTM and BiLSTM model. We can also see that out of 1864 sentences generated by the transformer, 1331 are an exact match to sentences from the training set. The BiLSTM is slightly better than the LSTM model as seen in Table 1; however, both are far behind the performance of the transformer. We ignore the occurrence metrics for the human-written text and the baseline, as by construction all text generated by the Jaccard model comes straight from the training data, and we use these metrics only to compare the performance of neural generation.

While our neural models heavily overfit to the training dataset, achieving perplexity levels less than 2 during training, the output from these models still contain previously unseen n-grams. We explored this behaviour by first verifying that the models achieved a 100% 1-gram occurrence rate and as such were not creating new words. Upon further exploration, we found that the source of these new n-grams was the models' merging of retrieved n-gram phrases in ways not seen in the training data. The combination of phrases allowed the model to produce previously unseen n-grams while still retrieving the core parts of its composition from content it was exposed to during training.

Although our transformer model does very well at retrieving text, it performs poorly in terms of generating second sentences that are cohesive with the input sentences. We can see that only about 47% of the transformer's generated output was considered cohesive by our InferSent classifier, far below what we consider to be the standard of nearly 79% on our human-written text. Interestingly, the Jaccard model outperforms even the human-written stories according to our cohesion classifier, with just below 86% of generated output considered to be cohesive. The LSTM and BiLSTM hover in the middle; although they are deemed to generate more cohesive output than the transformer, they do not reach the cohesion of Jaccard and human-written stories.

These trends are reflected in the average Jaccard similarity across our models; ignoring the Jaccard model which of course has the highest similarity at 0.271, we can see that the transformer model has the lowest Jaccard similarity at 0.046, compared to the LSTM and BiLSTM models at 0.052 and 0.055 respectively. We believe that our InferSent classifier may be considering factors

[1] Italicized figures included for completeness, but not meaningful

similar to Jaccard similarity when attempting to detect cohesion; this may account for why the Jaccard model achieved a cohesion percentage that exceeded even the human-level standard. It is also possible that the results of our InferSent classifier are mildly skewed, as the model only achieved a peak test accuracy of approximately 70% on our classification dataset.

6.3 Qualitative Results and Analysis

Model	Avg. Readability	Avg. Cohesion	Avg. Horror Rating	Avg. Scariness	Avg. Quality	% AI vs. Human Acc.
Human	8.54	8.59	7.77	7.03	7.69	84.44
Jaccard	7.14	5.73	5.97	4.86	5.71	61.48
LSTM	6.74	5.21	5.10	4.36	5.28	68.15
BiLSTM	6.63	4.43	4.39	3.63	4.63	81.48
Trans.	6.93	5.11	5.10	4.36	5.06	68.88

Table 2: *Survey results aggregated over 27 respondents; categories rated 1-10*

From the survey results, we can see that unsurprisingly, respondents rated human-written as the highest in all categories. Jaccard-generated stories follow closely behind, again in all categories; although we expected this for readability (as the second sentence was taken straight from training data) and therefore quality (which is of course heavily impacted by whether or not stories are understandable), it is interesting to see that the simple metric was so successful in generating stories that readers considered both cohesive and (somewhat) scary. In fact, output from the Jaccard model was most often confused for human-written text; over the 5 samples included in the survey for the model, respondents were only 61.48% accurate in guessing that the text was not written by a human, meaning that nearly 40% of the time the generated output was indistinguishable from human-level writing.

In contrast, output from the BiLSTM was rated the worst in all categories, including overall quality, and was easily distinguished as AI-written over 81% of the time. The transformer model and the LSTM performed similarly in most categories; however, output from the transformer was generally considered more readable, while output from the LSTM was generally rated to have slightly higher cohesion and to be of higher overall quality. However, we can see similar trends to what we saw in our automated metrics, with the Jaccard model excelling, the LSTM giving middle-ground results, and the transformer model struggling to generate cohesive text.

Although we achieved good results with our models, the gap between model performance and human performance (as seen in Table 2) indicates that there is still a long way to go, especially in the categories of cohesion and scariness. Further work with our models may help close to this gap.

For a few examples of our most-effective generated output, please see Table 3. For more examples, see Appendix B.

7 Conclusion

Text generation is a complex and difficult task; this difficulty is compounded by the lack of a standard automatic metric. Our work demonstrates that two-sentence horror generation at a near-human level is possible and provides an avenue for such text generation given a limited dataset. In the case of two-sentence horror, vocabulary and subject matter is generally limited to a few overall themes, and as such a retrieval model can often find text with reasonable similarity to the input.

Cohesion between sentences was the most challenging element for all of our models. Although our neural models had generally high readability, all three struggled to generate a cohesive second sentence. We believe that if better automatic metrics for cohesion could be developed, our models could account for this factor while training.

Overall, we were able to successfully generate the second sentence of a two sentence horror story at a level reasonably similar to human level quality, as demonstrated by the results of our survey. We also developed a moderately successful classifier for story cohesion using Facebook's InferSent model. We believe that our approach of applying an NLI model towards the evaluation of a generative model provides useful insight into possible future approaches for the evaluation of text generation.

Model	Example
Jaccard	Other than the Craigslist ad that I answered a few months ago, I've never actually seen or spoke directly to my landlord. But for the last few days, a man I've never seen before has shown up everywhere I go.
LSTM	'Good night beautiful,' I happily told my wife. The last thing I heard was the high-pitched scream and a sickening crunch.
BiLSTM	My attempts at astral projection finally worked. It's been over 24 hours now and I don't know how to get back into my body.
Trans.	She had waited her whole life for her parents to look at her with pride in their eyes. That pride quickly turned to terror as I saw the oncoming headlights.

Table 3: *Examples of generated output*

8 Future Work

8.1 Improving InferSent Evaluation

While our modified InferSent Architecture has achieved reasonable performance, we believe that its accuracy could be improved. An increase in the amount of cohesive two-sentence horror stories would be extremely useful to improving the classifier's performance. However, it is unlikely that a larger dataset for this topic exists, so outside of data augmentation or the use of similar data, further transfer learning could prove to be useful. Without hyperparameter tuning, transfer learning accounted for an approximately 8% increase in performance; it is possible that training the model further on other similar tasks could result in an additional boost.

8.2 Data Modifications

Our biggest challenge for this task was the limited size of our data; to that end, it might be helpful to explore approaches for expanding our dataset. Subreddits such as /r/nosleep [4] and /r/creepypasta [20] are also focused on horror fiction, so their posts could be useful as additional horror-related content pairs. Although these pairs would not be constrained by the limited scope of two-sentence horror, our models may still benefit from the additional examples. By feeding these additional pairs into our models, they may better learn what constitutes horror and how different sentences should be related. Another possibility is to better use our limited data by exploiting the upvote (or popularity) scores of our examples. By pruning out the lowest-scored examples and oversampling our highly-scored examples to effectively re-weight our data, our models may be able to better learn what makes a good story and in turn generate better output.

8.3 Model Modifications

It would be interesting to perform further model modifications to explore their effect on generated output; for example, experimenting with using GRU gates instead of LSTM gates or investigating alternate attention types could possibly lead to output that is more cohesive.

Additionally, we believe that pre-training could improve the quality of our models. OpenNMT offers several pre-trained models; however, when trying to run them we ran into issues regarding setting mismatch due to models being generated with an earlier version of OpenNMT. We believe that training a model on a language modeling task before training it on our dataset would improve the quality of our generative models, as transfer learning may allow the model to compensate for the lack of data.

8.4 Human Evaluation

While we are very happy to have received many responses to our survey, we were only able to collect responses for three days due to time constraints; it would be useful to poll an even broader range of people. Amazon Mechanical Turk or a larger survey time window would be very useful for expanding the survey response count and giving us more data for analyzing the quality of our models.

Mentor

Our mentor for this project is Abigail See. Thanks Abi!

References

[1] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. URL: https://doi.org/10.18653/v1/P17-4012, doi:10.18653/v1/P17-4012.

[2] MIT Media Lab. Shelley. [Online; accessed 5-March-2019]. URL: http://shelley.ai/.

[3] MIT Media Lab. Nightmare machine. [Online; accessed 5-March-2019]. URL: http://nightmare.mit.edu/.

[4] Reddit. nosleep. [Online; accessed 5-March-2019]. URL: https://www.reddit.com/r/nosleep/.

[5] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *CoRR*, abs/1805.04833, 2018. URL: http://arxiv.org/abs/1805.04833, arXiv:1805.04833.

[6] Jason Baumgartner. Reddit statistics - pushshift.io. [Online; accessed 1-March-2019]. URL: https://pushshift.io/.

[7] Reddit. Two-sentence horror stories: Bite-sized scares. [Online; accessed 3-March-2019]. URL: https://www.reddit.com/r/TwoSentenceHorror/.

[8] Stanford NLP. The Stanford Natural Language Processing group. [Online; accessed 16-March-2019]. URL: https://nlp.stanford.edu/projects/nmt/.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. arXiv:1706.03762.

[10] OpenNMT. Opennmt-py: Open source neural machine translation in pytorch, 2019. URL: https://github.com/OpenNMT/OpenNMT-py.

[11] Melissa Roemmele, Andrew S. Gordon, and Reid Swanson. Evaluating story generation systems using automated linguistic analyses. ACM, 2017. URL: http://people.ict.usc.edu/~roemmele/publications/fiction_generation.pdf.

[12] Steven Myint. language-check: Python wrapper for language tool grammar checker, 2019. URL: https://github.com/myint/language-check.

[13] Amperser Labs. proselint: A linter for prose, 2018. URL: https://github.com/amperser/proselint.

[14] Denny Britz. cnn-text-classification-tf: Convolutional neural network for text classification in tensorflow, 2018. URL: https://github.com/dennybritz/cnn-text-classification-tf.

[15] Yoon Kim. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. URL: http://dx.doi.org/10.3115/v1/D14-1181, doi:10.3115/v1/d14-1181.

[16] Cahya Wirawan. Multiclass classification and pre-trained word embedding (word2vec glove) support and it's comparison, 2017. URL: https://github.com/dennybritz/cnn-text-classification-tf/issues/69.

[17] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL: http://www.aclweb.org/anthology/D14-1162.

[18] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017. URL: http://arxiv.org/abs/1705.02364, arXiv:1705.02364.

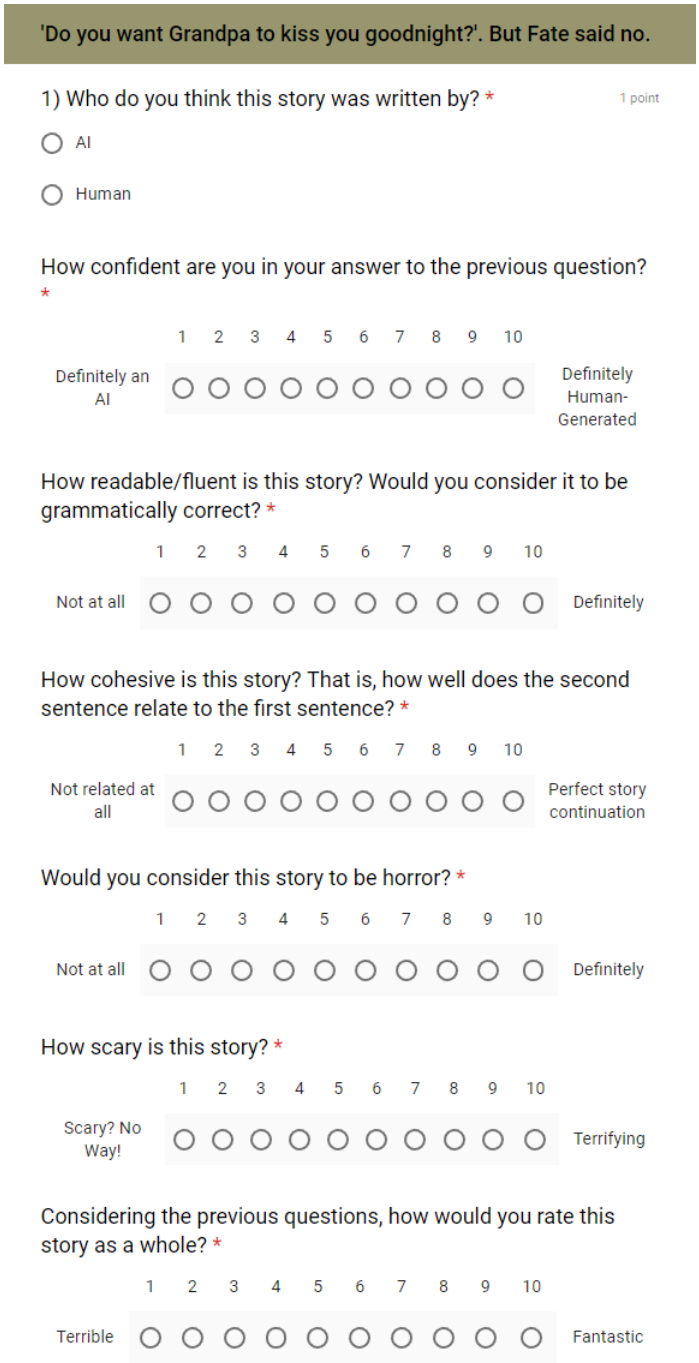
[19] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[20] Reddit. A community for those who love scary stories and the paranormal. [Online; accessed 5-March-2019]. URL: `https://www.reddit.com/r/creepypasta/`.

Appendix

Appendix A

For our survey, we asked respondents the same set of 7 questions for each of the 25 samples. Below is a sample page of our survey that shows 1 sample and its associated questions.



'Do you want Grandpa to kiss you goodnight?'. But Fate said no.

1) Who do you think this story was written by? * 1 point

- AI
- Human

How confident are you in your answer to the previous question? *

	1	2	3	4	5	6	7	8	9	10	
Definitely an AI	○	○	○	○	○	○	○	○	○	○	Definitely Human-Generated

How readable/fluent is this story? Would you consider it to be grammatically correct? *

	1	2	3	4	5	6	7	8	9	10	
Not at all	○	○	○	○	○	○	○	○	○	○	Definitely

How cohesive is this story? That is, how well does the second sentence relate to the first sentence? *

	1	2	3	4	5	6	7	8	9	10	
Not related at all	○	○	○	○	○	○	○	○	○	○	Perfect story continuation

Would you consider this story to be horror? *

	1	2	3	4	5	6	7	8	9	10	
Not at all	○	○	○	○	○	○	○	○	○	○	Definitely

How scary is this story? *

	1	2	3	4	5	6	7	8	9	10	
Scary? No Way!	○	○	○	○	○	○	○	○	○	○	Terrifying

Considering the previous questions, how would you rate this story as a whole? *

	1	2	3	4	5	6	7	8	9	10	
Terrible	○	○	○	○	○	○	○	○	○	○	Fantastic

Appendix B

Here are some additional samples of generated output for each of our models.

Jaccard Model
They told me it was an accident," I reminded myself as i lay bleeding in the bed of the pick up truck, miles of dark country roads passing us by. Now I sit, alone in the dark, barely aware of another eternity passing me by.
My wife says we're going to have a baby tonight, I asked the gender. I would rather have asked to be buried in a coffin, if I knew I would feel the burning.
What can I say, when my life has been a series of farewells, to my family, friends, and all I hold dear? This is all I can remember of the last moments of my life and I'm damned to re-live them over and over and over.
As I fell asleep I felt a cool breeze on my face. That's when I felt a hand on my back.

Table 4: *Output generated with the Jaccard model and beam search with beam size 10*

LSTM Model
He was a smart boy, he gloated that he knew everything. But he was the last person he ever heard.
The voice in my head has finally stopped screaming. I knew the owner of my breathing.
I woke up this morning with a hangover and the quilt my mom made for me covering me. I realized I've been dead.
My dog barks every single night. He's been dead for 5 years.

Table 5: *Output generated with the LSTM model and beam search with beam size 10*

BiLSTM Model
I drove as fast as I could to get away from the thing in the woods. I forgot to get out of the straitjacket, but it wouldn't budge.
Squiggles turns ten today! I just wish he would quit crying about his wife and kids.
I was just minding my own business and playing my game I got just a week ago. Then I woke up from the coma.
I was home alone when lightning struck and the power went out. The moaning and banging didn't stop.

Table 6: *Output generated with the BiLSTM model and beam search with beam size 10*

Transformer Model
It had been hours since I had lost radio contact and even longer that since I had been sent flying away from the shuttle. It might have been happier if I hadn't seen their dead bodies in orbit 11 months ago.
Those bugs didn't stop chewing away. I tried to scream.
When I was a lad, my father told me that if a man consumes the beating heart of an infant upon the eve of every solstice, he shall remain young for all eternity. It seems to be a legend that has dissipated with time, as I have never heard another person speak of it for the past three centuries.
I thought it odd when he said "Got your nose!". "And your teeth.Your eyeballs, your liver, your intestines."

Table 7: *Output generated with the Transformer model and beam search with beam size 10*