The Thirtieth AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18)

InspireMe: Learning Sequence Models for Stories

Vincent Fortuin
Disney Research Zürich, ETH Zürich
fortuin@inf.ethz.ch

Romann M. Weber
Disney Research Zürich
romann.weber@disneyresearch.com

Sasha Schriber
Disney Research Zürich

Diana Wotruba
Disney Research Zürich

Markus Gross
Disney Research Zürich, ETH Zürich

Abstract

We present a novel approach to modeling stories using *recurrent neural networks*. Different story features are extracted using *natural language processing* techniques and used to encode the stories as sequences. These sequences can be learned by deep neural networks, in order to predict the next story events. The predictions can be used as an inspiration for writers who experience a *writer's block*. We further assist writers in their creative process by generating visualizations of the character interactions in the story. We show that suggestions from our model are rated as highly as the real scenes from a set of films and that our visualizations can help people in gaining deeper story understanding.

Introduction

Many avenues in natural language processing (NLP) have recently benefitted from the application of deep neural networks (Goldberg 2016). Recurrent neural networks (RNNs), such as long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber 1997) have been especially successful when used to learn language models (Graves 2013) and have yielded impressive results on generating different styles of text (Karpathy 2015). However, RNNs have not yet been applied to the specific field of story generation and narrative artificial intelligence, where the main task is not to predict single words or characters but rather consistent sequences of story scenes or events.

In the field of automatic story generation, early approaches have been based on predefined formal rules and grammars (Klein, Aeschlimann, and Balsiger 1973). Recently there have also been approaches to learning certain story rules from data in an unsupervised fashion (Chambers and Jurafsky 2009) and learning to generate rule-based story graphs in certain domains (Li et al. 2013). However, these systems were always designed to generate stories from scratch, without regarding potential preexisting story parts. Moreover, they sample from a highly constrained space of potential storylines, thereby ensuring consistency in story logic but arguably restricting creativity.

A common problem in the creative-writing-community is the so-called *writer's block* (Clark 2011). This is a psychological condition in which the writer experiences an inhibition in creativity and lack of new ideas while writing a story. Although some computational methods have been developed to assist writers in this condition by suggesting what to write next (Roemmele and Gordon 2015), the use of deep learning for this purpose has only been proposed but never successfully tested (Roemmele 2016).

Previous work that constitutes the current state of the art only regards the very last written sentence as a context for its suggestion and only suggests whole sentences that already exist in its database of stories (Roemmele and Gordon 2015). Our work improves on this method by conditioning on the whole written story and sampling the proposed continuations from a learned generative model. It is therefore able in principle to extract more relevant information from the input and capable of producing more creative and interesting novel outputs.

The lack of inspiration in writer's block can often be caused by a perceived mental restriction to certain ideas and storylines, which is why common techniques to overcome the condition include free association methods and brainstorming (Clark 2011). In order to support this process with a software tool, it is therefore critical to generate suggestions that are open and vague enough to not increase the feeling of restriction. We allow for this by not generating whole storylines, but rather only suggesting loose connections between characters, places and actions. Furthermore, we generate statistical summaries of the existing story in a visual form in order to facilitate contemplation on the current state of the work.

Our main contributions are the following:

- We develop a novel representation for stories, especially movies, as sequences of encoded events and scenes using natural language processing techniques.
- We train a neural sequence model on these representations and generate suggestions for the next scenes, conditioned on a whole or partially written story, with the results rated by users as highly as real scenes from actual films.
- We visualize character interactions in a novel way in order to help people gain a deeper story understanding.

Model

We developed a novel way to represent stories as sequences of encoded scenes or events and trained a neural sequence model on the following representations.

Story representation

Stories are represented as sequences on different time scales. The *event sequence* representation focuses on a more fine-grained encoding of the story, where the relationship between subsequent character actions is to be modeled. The *scene sequence* representation focuses on more coarse-grained transitions between story scenes and their respective locations and sets of acting characters. Both representations are complementary descriptions of the story and are used to train separate models.

Event sequence In order to understand event-sequence encoding, the notion of a *story event* has to be properly defined. We define an event as an action that is performed by some character in the story, potentially with respect to some object. The event is then encoded as the vector $e = [c, a, o]^T$, where $c \in \mathbb{R}^n$ is a one-hot vector encoding which of the characters in the story is acting and n is the number of characters considered in the encoding. The vectors a and o are word embeddings for the performed action and object, respectively.

We trained the word embeddings on our corpus using skip-gram *word2vec* (Mikolov et al. 2013) as implemented in the Python package `gensim` (Řehůřek and Sojka 2010). Our trained embeddings are 300-dimensional. If an event does not contain any explicit object, we set $o = [0, \ldots, 0]^T \in \mathbb{R}^{300}$.

Our event vectors are therefore $e \in \mathbb{R}^{600+n}$. In our experiments, we set $n = 10$; that is, we only consider actions performed by the ten most prominent characters in the story. For parsing the characters, actions, and objects from the stories, we used the `spaCy` package in Python (Honnibal and Johnson 2015).

Scene sequence For the scene-sequence encoding, we first segmented the story into scenes, which we define as a story part with a constant set of characters at a constant location. A scene is encoded as the vector $s = [l, c, k]^T$, where l is a one-hot vector over all the possible locations in the corpus, $c \in \{0, 1\}^n$ a binary vector of the characters present in the scene and k a word embedding of a keyword describing the scene. The location and characters are identified using the `spaCy` package (Honnibal and Johnson 2015), while the keywords for the scenes are extracted with the *TextRank* algorithm (Mihalcea and Tarau 2004) as implemented in the Python package `textrank`. The character vector is again $c \in \{0, 1\}^n$, where n is the number of main characters considered.

Sequence model

To model our sequence encodings and predict the next element in the sequence, we use a recurrent neural network, particularly a long short-term memory (LSTM) network (Hochreiter and Schmidhuber 1997). We use batch normalization (Ioffe and Szegedy 2015) and dropout (Srivastava et al. 2014) on the LSTM output features and train the network with the Adam optimizer (Kingma and Ba 2015). Moreover, we use gradient norm clipping (Pascanu, Mikolov, and Bengio 2012) and a cyclic learning rate scheme (Smith 2017). The parameters are initialized using the Glorot method (Glorot and Bengio 2010). During training we also use teacher forcing (Williams and Zipser 1989).

The hyperparameters of the network were optimized manually. We used one LSTM layer with 1,024 units, followed by a batch normalization layer and a dropout layer with a keep probability of 0.7. We did not observe any benefit of using more than one LSTM layer.

Story visualization

We visualize character appearances and interactions in the existing story input in the form of a graph in order to provide a rich but still tangible representation of different story statistics.

Different characters appear in the story and interact with each other in varying amounts. Moreover, they can convey different sentiments in their appearances and interactions. We visualize statistics regarding these features in a summary graph.

First, we calculate a sentiment value for every scene using the *AFINN* lexicon (Nielsen 2011) as implemented in the Python package `afinn`. We then use the sentiment information for the different scenes to compute average sentiments for each character. These average sentiments are defined as the average over the sentiments of each scene featuring a particular character, i.e.

$$S_c(c_i) = \frac{\sum_{j=1}^{N} S_s(s_j)\mathbb{1}_{C_j}(c_i)}{\sum_{j=1}^{N} \mathbb{1}_{C_j}(c_i)},$$

where $S_c(\cdot)$ is the character sentiment function, c_i is the ith character, s_j is the jth scene, N is the number of scenes, $S_s(\cdot)$ is the scene sentiment funtion, C_j is the set of characters present in scene s_j and $\mathbb{1}_A(\cdot)$ is the indicator function for membership in set A.

Analogously, the interaction sentiment is defined as the average sentiment over all scenes in which two characters both appear, i.e.

$$S_i(c_i, c_j) = \frac{\sum_{k=1}^{N} S_s(s_k)\mathbb{1}_{C_k}(c_i)\mathbb{1}_{C_k}(c_j)}{\sum_{k=1}^{N} \mathbb{1}_{C_k}(c_i)\mathbb{1}_{C_k}(c_j)}.$$

The graph is then constructed by depicting the characters as nodes and their interactions as edges. The nodes are colored according to their respective character sentiment and the edges according to their interaction sentiment. Both nodes and edges are scaled in size according to the number of scenes in which they are present.

The graph is constructed using the Python package `networkx` and plotted using `matplotlib` (Hunter 2007) (see Fig. 1).

Table 1: Generated sentences for the predictions from the scene-sequence model for the film *Frozen*. The sentences were generated using rule-based logic dependent on the keywords' part-of-speech tags and the number of characters in the scene. τ is a temperature parameter regulating the entropy of the output distribution and therefore the "adventurousness" of the predictions.

τ	Generated sentence
0.3	Anna and Elsa are at the hallway discussing about the night.
	Anna and Olaf are at the street whispering about the night.
	Anna is at the kitchen monologuizing about the night.
0.6	Anna and Olaf are at the kitchen debating about the night.
	Anna and Kristoff and Olaf and Elsa are at the bedroom arguing about the night.
	Anna and Olaf and Elsa are yellow at the car.
1.0	Anna and Hans are at the laundromat talking about the night.
	Anna and Kristoff are at the street chatting about the woman.
	Olaf is beautiful at the dock/waterfront.
1.5	Olaf is at the armadillo monologuizing about the bounce.
	Anna and Sven are happy at the tank turret.
	Kai and Sven are at the bank conversing about the apache helicopter.
2.0	Anna and Kristoff are at the makeshift lab conversing about the pledge.
	Anna is at the labyrinth singing about the afterthought.
	Anna and Kristoff are at the castle debating about the floorboard.
3.0	Anna is about to talk at the downtown theater.
	Anna and Kristoff are at the lola's house arguing about the shoulder.
	Anna and Sven are holstered pistol at the hood house.

Experiments

We downloaded 1,054 movie scripts from the Internet Movie Script Database (www.imsdb.com) and 8,459 novels from the Project Gutenberg (www.gutenberg.org). All of these stories were written in English. We randomly withheld 64 stories as a validation set and 80 stories as a test set and trained our models on the rest.

The number of possible characters to sample was fixed to 10 main characters by design. We noticed that the 10 main characters contribute more than 90 percent of the actions in our stories, such that the behavior of any other characters would likely be hard to learn due to a lack of observations. There were 2,928 actions, 7,997 objects, 47,577 locations and 49,247 keywords to sample from, which were extracted from all the stories in our training set.

Our sequence models have separate softmax output layers for the prediction of every scene or event feature (character, location, action, object, keyword). The softmax distribution we used is defined as

$$P(y=i|x;\tau) = \frac{\exp(\frac{x^T w_i}{\tau})}{\sum_{k=1}^{C} \exp(\frac{x^T w_k}{\tau})}$$

where x is the output of the penultimate layer, w_i is the weight vector of the last layer associated with class i, C is the number of classes and τ is a temperature parameter we introduce to regulate the entropy of the distribution.

With a value of $\tau = 1$, one retrieves the standard softmax. Higher values lead to more diversity in the samples and a more uniformly distributed prediction, while lower values lead to the opposite. With $\tau \to 0$, the distribution approaches the $\arg\max(x^T w)$ function and with $\tau \to \infty$, the uniform distribution.

We trained the model until our validation set error no longer decreased and evaluated it on the test set using different values for the temperature parameter. It became apparent that the "adventurousness" of the predictions increases with temperature. This parameter can be adjusted at prediction time and can therefore be left for the user to choose.

In order to make the scene predictions more human-readable, we generated sentences from them using a rule-based logic. The sentences describe the acting characters being at some location performing some action. What they do is dependent on the part-of-speech tag of the keyword.

If the keyword is an adjective, the characters simply gain that attribute. If it is a verb, the characters are about to perform that action. If it is a noun, the characters are conversing about it, where we sample the conversational verb from a precompiled list at random. Examples of generated sentences from the scene-sequence prediction model are presented in Table 1.

The predictions exhibit features of real stories to some degree, e.g. the appearance of the same characters throughout multiple subsequent scenes as well as the fact that characters usually do not move unreasonably far between their appearances in different scenes.

User study

We recruited 22 people (11 with story writing experience and 11 naïve users) to test our tool. They were asked to read the plot summaries of three films that were not included in our training set, in which we changed the characters' names to make recognizing the stories more difficult. Using the partial summaries as a writing prompt, the users were asked to write the next scene, with the option of using our software

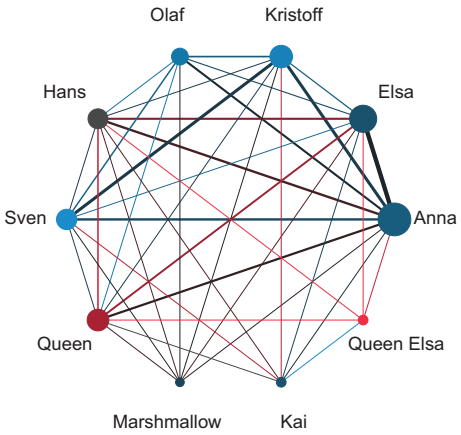

Figure 1: Character interaction graph for the film *Frozen*. The nodes represent the main characters and the edges their interactions. The nodes and edges are scaled according to the number of appearances in the film. The character nodes are colored according to their average sentiment over the whole story, the edges with respect to the sentiment of the represented interaction. The most positive sentiments are depicted in a light blue color, the most negative ones are a bright red. Neutral sentiments are black. It is apparent that some characters are biased in their sentiment, while others are more nuanced. It can moreover be seen that the entity parser has not recognized *Elsa* and *Queen Elsa* as the same character and that there is a large difference in sentiment between the two personalities. (Best viewed in color.)

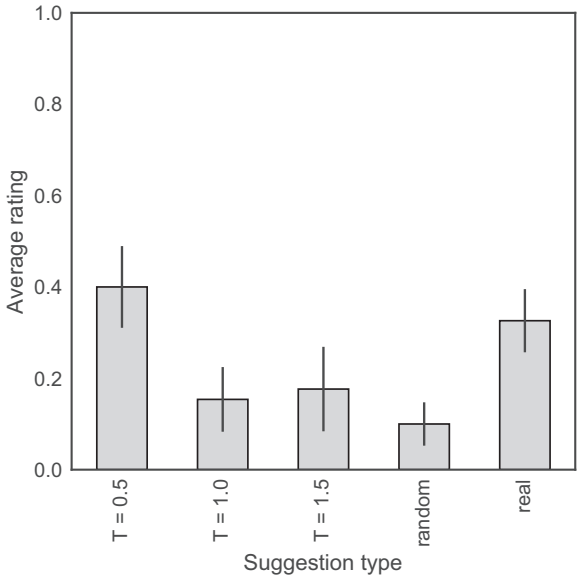

Figure 2: Average rating of suggestions generated by different models in the writing task of the user study. The rating is computed as the percentage of suggestions from a given model that were liked by the users. The suggestions were sampled from our model at different softmax temperatures, from a random model and from the real next scenes in the film. Error bars indicate standard errors.

for inspiration.

Apart from predictions from our neural network at different softmax temperatures, we also included the actual next scenes from the story prompts as well as predictions that were generated by sampling randomly from the characters, locations and keywords. In this user study, we only worked with the scene predictions from the network and not the event predictions.

After the writing tasks, the users were asked for their opinion about the tool along with a few questions about some additional example stories using our character interaction graph visualizations.

For every suggestion that the users got in the writing task—whether model-generated, random, or "ground truth"—they could indicate whether they liked it. It can be seen that the predictions generated at higher temperatures ($\tau = 1.0$ and $\tau = 1.5$) were not significantly preferred to random suggestions. This result is consistent with the fact that the random model is identical to the softmax with $\tau \to \infty$.

Interestingly, the suggestions generated at $\tau = 0.5$ were slightly (although not significantly) preferred to the *real* next scenes in the films (Fig. 2). This suggests that our sequence model is producing next-scene predictions at least as plausible and coherent as those produced by the professional screenwriters who wrote the actual films. However, it should be noted that both the model-produced suggestions and the real next scenes were liked by the users less than half of the time, indicating that providing satisfactory inspiration for writers is a hard task, even when those "suggestions" come from professionally produced stories.

The users were asked to rate their experience along different dimensions on a five-point *Likert scale* (Likert 1932). As mentioned above, on average users were not particularly fond of *any* suggestions they got, even if those suggestions came directly from the professionally produced stories they were tasked with continuing. This could be due to the fact that many users did not seem to find the general concept of getting suggestions helpful (Likert score 2.36 ± 0.29). This is consistent with the observation that many users reported their inspirational flow to be interrupted by consulting the suggestions (Likert score 2.73 ± 0.26) as well as being confused by them (Likert score 3.91 ± 0.23), which possibly could have been a side-effect of the timed writing component of the study. Nevertheless, the suggestions did spark mental images (35.5 ± 6.5 percent of the suggestions). The tool reached a score of 65.5 ± 4.3 on the System Usability Scale, which is considered to be near average to slightly below average (Brooke 1996).

In the character interaction task, 94.3 percent of the users managed to successfully identify more- and less-prominent characters as well as "nice" and "less nice" characters, and 95.5 percent could correctly spot pairs of characters that had or had not interacted in positive or negative ways. The users found the visualization to be helpful in answering these questions (Likert score 4.55 ± 0.22).

Discussion

Our results suggest that given the right representation of stories, certain features and correlations over time can be learned by deep sequence models. Most of these features are rather generic and apply to most stories, such as the conservation of characters over scenes and the geographical anchorage of certain characters. It is likely that our data set is too small to learn more nuanced story features, since roughly 75 percent of the combinations of actions and locations only appear once in our whole story corpus.

One could possibly say that most stories share a set of a few simple rules and otherwise contain a multitude of very complex and specific rules that are common only to a handful of stories. A larger amount of data would also allow for the use of a more powerful model, such as a sequence-to-sequence variational autoencoder (Bowman et al. 2015) or a sequence generative adversarial network (Yu et al. 2016). Such models may be better able to abstract rules in the story world and generate plausible samples.

Our main contribution is the development of different sequence encodings of stories. The quality of these encodings depends largely on the performance of current NLP techniques, which are in turn vital for the quality of the learned models. Since the field of NLP is arguably still far from achieving the goal of automated natural language understanding (NLU), advances in these areas have the potential to facilitate better story encodings and hence better models. It will be exciting to see where research will lead the fields of NLP and narrative artificial intelligence in the years to come.

The character interaction graph discards temporal information in favor of information about single characters aggregated over the whole story. Different methods have been proposed to generate and analyze character interaction graphs (Kaminski et al. 2012; Bonato et al. 2016), but they do not contain sentiment information. Our method, on the other hand, explicitly incorporates sentiment analysis on the level of the single characters as well as their interactions.

We believe that this additional information can facilitate identifying characters that are so biased in their overall sentiment as to become "one-dimensional" and uninteresting. It can also help in spawning ideas for new interactions between characters, which may not have interacted in the story yet.

Regarding the sentiment analysis used in this work, it should be noted that it relied on the use of lexicons, making it impossible to distinguish between homonyms or the use of a word in different contexts. There are more accurate and involved ways to analyze sentiment using contextual information (Wilson, Wiebe, and Hoffmann 2005), deep learning (Glorot, Bordes, and Bengio 2011), and parse trees (Socher et al. 2013), but these are computationally more expensive and were not used in our case.

We saw in our user study that it is quite hard to satisfy people with generated suggestions in a creative writing task. However, as our model's predictions fared no worse than the real next scenes taken from the respective films, we consider our results to be a successful proof of concept. A wider range of use cases—including open-ended, untimed writing sessions—will be considered in future work.

We also noted that the standard errors for the ratings in our user study were quite high, indicating a large variance in the acceptance of suggestions between different users. Indeed, some users will be more receptive to inspirational suggestions in their creative process than others. We hypothesize that those more receptive users could benefit from using our tool, at least as much as they would from getting suggestions produced by screenwriters.

Our users seemed to find our visualizations helpful in gaining a deeper understanding of the stories and found that they facilitated their answering questions about them. This suggests that the visualizations could also be helpful in the creative process, when it comes to analyzing the story in order to spot flaws or weaknesses.

Conclusions

We developed a novel way to encode stories from natural language into sequences of different features. We have shown that it is possible to use sequence models, especially recurrent neural networks, to learn from these story-feature sequences and predict new story events.

We moreover visualize the stories in novel ways, depicting interactions between characters and their respective sentiments. These visualizations can give writers valuable insights into their stories and may help in optimizing the dramatic development.

We performed a user study in which participants had to continue fragments of film stories in a creative writing task. We showed that samples from our model at low softmax temperatures are as appealing to those people as the real next scenes from the respective films. This suggests that we succeeded at modeling some aspects of story sequences that maintained a certain "story logic". Furthermore, we have shown that our visualizations helped users in gaining a deeper understanding of different story features.

Our tool provides many features that are currently not available in any digital story writing assistant and shows considerable advances over the previous state of the art in computer-aided creative writing. Our method therefore has the potential to constitute a valuable addition to the story writer's toolbox, especially in cases in which the writer's well of ideas has temporarily run dry.

Acknowledgments The authors would like to thank Brian McWilliams, Bob Sumner, Steven Poulakos, and Jessica Falk for many helpful discussions; Daniel Inversini for providing invaluable help in preparing the front end for our study; and Rick Sanchez for constant inspiration.

References

Bonato, A.; D'Angelo, D. R.; Elenberg, E. R.; Gleich, D. F.; and Hou, Y. 2016. Mining and modeling character networks. *arXiv Preprints*.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Jozefowicz, R.; and Bengio, S. 2015. Generating Sentences from a Continuous Space. *arXiv Preprints*.

Brooke, J. 1996. System Usability Scale (SUS): A Quick-and-Dirty Method of System Evaluation User Information. In *Usability Evaluation In Industry*, volume 189. 4–7.

Chambers, N., and Jurafsky, D. 2009. Unsupervised Learning of Narrative Schemas and their Participants. In *Proceedings of the Annual Meeting of the ACL*, 602–610.

Clark, I. L. 2011. *Concepts in Composition*. Abingdon: Routledge, 2nd edition.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 249–256.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the International Conference on Machine Learning*, 513–520.

Goldberg, Y. 2016. A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research* 57:1–75.

Graves, A. 2013. Generating Sequences With Recurrent Neural Networks. *arXiv Preprints*.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Honnibal, M., and Johnson, M. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, 1373–1378.

Hunter, J. D. 2007. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering* 9(3):99–104.

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv Preprints*.

Kaminski, J.; Schober, M.; Albaladejo, R.; and Zastupailo, O. 2012. Moviegalaxies - social networks in movies. *http://www.moviegalaxies.com/*.

Karpathy, A. 2015. The Unreasonable Effectiveness of Recurrent Neural Networks. *https://karpathy.github.io/2015/05/21/rnn-effectiveness/*.

Kingma, D. P., and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*.

Klein, S.; Aeschlimann, J.; and Balsiger, D. 1973. Automatic novel writing: A status report. *Wisconsin University*.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. O. 2013. Story Generation with Crowdsourced Plot Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 598–604.

Likert, R. 1932. A Technique for the Measurement of Attitudes. *Archives of Psychology* (140):1–55.

Mihalcea, R., and Tarau, P. 2004. TextRank: Bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 85, 404–411.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv Preprints*.

Nielsen, F. Å. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *CEUR Workshop Proceedings*, volume 718, 93–98.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2012. On the difficulty of training recurrent neural networks. In *Proceedings of The International Conference on Machine Learning*, number 2, 1310–1318.

Roemmele, M., and Gordon, A. S. 2015. Creative help: A story writing assistant. In *Proceedings of the International Conference on Interactive Digital Storytelling*, 81–92.

Roemmele, M. 2016. Writing Stories with Help from Recurrent Neural Networks. In *Proceedings of the Conference on Artificial Intelligence*, 2011–2012.

Smith, L. N. 2017. Cyclical Learning Rates for Training Neural Networks. *arXiv Preprints*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1631–1642.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Williams, R. J., and Zipser, D. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation* 1(2):270–280.

Wilson, T.; Wiebe, J.; and Hoffmann, P. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 347–354.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2016. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv Preprints*.

Řehůřek, R., and Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50.