ChiNet

Nil Adell **Joseph Cornelius** **Alexander Nedergaard** **Lama Saouma**

1 Introduction

The Story Cloze Task (SCT) (Mostafazadeh et al. [2016]) is a recent dataset for machine comprehension. The test data consists of 5-sentence short stories with two different possible endings, and a label indicating which ending is the correct one. Stories are deliberately constructed such that determining the correct ending requires understanding the semantic content, a very challenging task for machines that humans can nonetheless complete with perfect accuracy. Another challenge of the dataset is that training data only includes correct endings, so it is non-trivial to train a supervised model to solve the task.

2 Methodology

We attempted to reproduce results from a recent paper (Wang et al. [2017]) that uses a conditional generative adversarial network (CGAN) (Mirza and Osindero [2014]) to solve the SCT. CGANs extend generative adversarial networks (GANs) (Goodfellow et al. [2014]) to generate data conditioned on a context. The idea applied to SCT is to use a generative neural network (the "generator") to generate fake story endings conditioned on previous sentences in the story, and a discriminative neural network (the "discriminator") to discriminate between real and fake story endings. Trained together in a competitive and unsupervised manner using only stories with real endings, the generator and discriminator become increasingly good at generating and discriminating story endings. After training, the discriminator can then be used by itself to determine the real ending to a story. Thus, our approach is technically discriminative, but has a generative nature as it relies on a generative model during training.

3 Model

Our CGAN model consists of a sentence recurrent neural network (RNN), a document RNN, a generator RNN and a discriminator. The sentence RNN takes an embedded sentence as input and outputs a distilled representation of the sentence. Distilled sentences can be fed to the document RNN to get a representation of the document context. The generator RNN takes in a document context and generates an embedded sentence. The discriminator takes as inputs a document context and a distilled sentence and outputs the probability that the distilled sentence is the real ending given the document context. Thus, the sentence and document RNNs are shared by the discriminator and generator, but for training we consider the document and sentence RNNs as part of the discriminator. We define our sentence RNN as

$$h_i^s = GRU(s_i; h_{i-1}^s)$$

where s denotes the embedded sentence and GRU denotes a gated recurrent unit (Cho et al. [2014]), a variant of long short-term memory (LSTM) (Hochreiter and Schmidhuber [1997]). We denote the final hidden state of the sentence RNN as r^s.

We then define our document RNN as

$$h_i^d = GRU(r_i^s; h_{i-1}^d)$$

and similarly denote the final hidden state of the document RNN as r^d.

Now, we define our generator RNN as

$$h_i^g = GRU(y_i; h_{i-1}^g)$$

where y_i is the embedded word generated at the previous time step. We set y_0 to the embedded <BOS> token. Unlike the sentence and document RNN, where the initial hidden states h_0^s and h_0^d are set to 0, we initialize our generator hidden state as

$$h_0^g = r^d + z$$

$$z \sim \mathcal{N}(0, 1)$$

Determining the most the most likely word y_i from our hidden state h_i^g would usually involve an $argmax$ operation. However, as the $argmax$ operation is not continuous and we require end-to-end differentiability for backpropagation, we use the Gumbel-Softmax trick (Jang et al. [2016]) to perform reparameterisation with continuous relaxation. Specifically, we determine the generated word from the generator hidden state as follows:

$$\pi_i = softmax(h_i^g W_{d\to e} W_e^T)$$

$$g_i = -log(-log(u))$$

$$u \sim Uniform(0, 1)$$

$$t_i = ReLu(h_i^g W_t) + \epsilon$$

$$p_i = softmax(\frac{log(\pi_i) + g_i}{t_i})$$

$$y_i = p_i W_e$$

where $W_{d\to e}$ is a transformation matrix from document space to embedding space, W_e is an embedding matrix, W_t is a matrix used to determine the temperature t_i, ϵ is a small number to ensure that t_i is positive and $ReLu$ is the rectifier function. With large t_i we get an average embedding over the entire vocabulary, and as t_i approaches 0 we get an embedding corresponding to a specific word. We continue to generate words until the <EOS> token is generated (we threshold π_i at 0.5 to check) or the maximum sentence length is reached. We then stack the words to obtain the generated embedded sentence $\bar{s}$.

We use attention (Wang et al. [2016]) to weigh the inputs of the document RNN, based on their similarity to the ending sentence. The weighted sentences are determined as

$$\tilde{r}_i^s = r_i^s \cdot a_i$$

$$a_i = \sigma(r_t^s W_A (r_i^s)^T)$$

where r_t^s denotes the ending sentence, W_A is an attention matrix, $\cdot$ denotes the scalar product and σ denotes the sigmoid function. We do not use attention when determining the document context for sentence generation, as this would provide information about the ground truth ending to the generator.

The discriminator score is defined as

$$D = \sigma(r^d W_{d\to s} (r_t^s)^T)$$

where $W_{d\to s}$ denotes a transformation matrix from document space to sentence space. The discriminator assigns the score 1 to endings r_t^s that are the most likely given the document context r_d and 0 to the least likely.

Now, given a story without the ending (distilled to a single document representation using the sentence and document RNNs) and two endings (distilled to sentence representations using the sentence RNN), we can determine the most likely ending as having the highest discriminator score D.

4 Training

We train our model by generating fake endings using the generator and feeding them to the discriminator. The discriminator tries to discriminate between real and fake endings, and the generator tries to trick the discriminator. The generator and discriminator are trained together, and we additionally pretrain the generator to improve performance.

The discriminator loss is given by

$$L_D = -log(D) - log(1 - \bar{D})$$

where D denotes the discriminator score of the ground truth ending and $\bar{D}$ denotes the discriminator score of the generated ending. The discriminator loss is minimized when the discriminator assigns a score of 1 to the ground truth ending and a score of 0 to the generated ending. We add noise to the generated sentences during discriminator training after 20 training epochs. It has been proposed that adding noise to the discriminator input can improve GAN training by increasing the support intersection of the true data distribution and the generator distribution (Arjovsky and Bottou [2017]).

The generator loss is given by

$$L_G = -log(\bar{D}) + similarity(s, \bar{s})$$

$$similarity(s, \bar{s}) = 1 - \frac{s \cdot \bar{s}}{\|s\|\|\bar{s}\|}$$

where $\bar{D}$ denotes the discriminator score of the generated sentence, s denotes the ground truth ending and $\bar{s}$ denotes the generated ending. The second term in the loss is not traditionally used in GANs, but was added to improve the performance of the generator. The generator loss is minimized when the discriminator assigns a score of 1 to the generated ending and the generated ending is maximally similar to the ground truth ending.

For each training epoch, the discriminator is trained with n_D batches and the generator with n_G batches, and the number of training batches are updated according to the ratio of the discriminator and generator losses:

$$n_D = \frac{L_D}{L_G}$$

$$n_G = \frac{L_G}{L_D}$$

The losses are averaged over the batches and the numbers of training batches are clipped to the range $[1, 40]$. The initial numbers of training batches are set to 1.

Pretraining of the generator is performed to improve performance. During pretraining, the generator is not conditioned on a document context and instead receives only random noise, much like a traditional GAN. Specifically, during pretraining the generator hidden state is initialized as

$$h_0^g = z$$

$$z \sim \mathcal{N}(0, 1)$$

and the pretraining generator loss is defined as

$$L_G^{pretrain} = similarity(s, \bar{s})$$

We use the AdaDelta optimizer with $p = 0.999$ and $\epsilon = 10^{-5}$. Learning rate is set to 10^{-3} and batch size to 32. For embedding, we use pretrained word2vec weights with embedding size 300 and do not update these during training. During data preprocessing, we add <BOS> and <EOS> tokens to the beginning and ending of sentences, pad sentences shorter than the max sentence length with <PAD> and replace words not in our vocabulary with <UNK>. We use a vocabulary size of 20000 and a max sentence length of 50. All trainable parameters are initialized using Xavier initialization. We use hidden state sizes of 128, 150 and 256 for the sentence, document and generator RNNs respectively.

5 Experiments

We were not able to reproduce the results of the original paper. The accuracy of our model on the SCT validation dataset was significantly lower than what was reported in the paper. It was also significantly lower than the Deep Structured Semantic Model (DSSM), which is the best performing model in the SCT dataset paper. Our model performs worse than a random model that always determines the correct ending as the first one, which is disappointing.

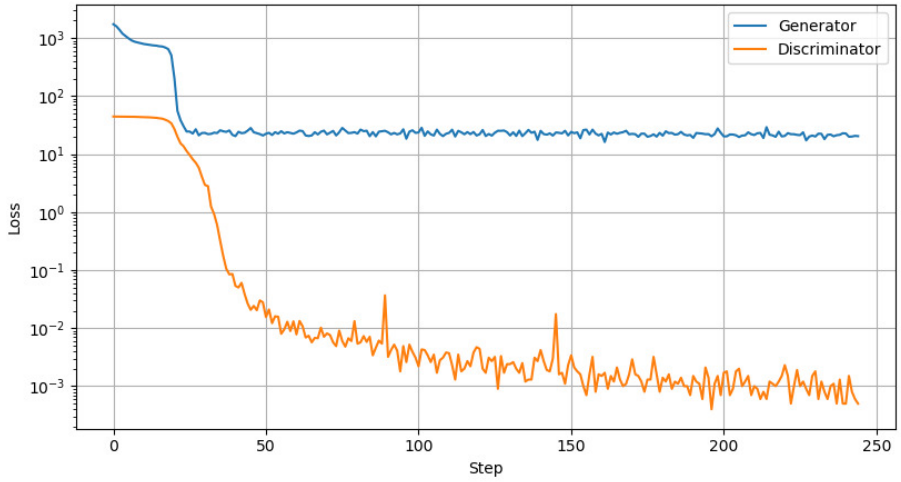
Table 1: Validation accuracy model comparison

Random	DSSM	CGAN (Wang et al.)	CGAN (Ours)	Human
0.514	0.604	0.625	**0.490**	1.0

Comparison of accuracy on the SCT validation dataset for different models.

It is not exactly clear why, but we were unable to train the CGAN properly. Training GANs is known to be challenging, and we would expect extending GANs to conditional text generation to exacerbate the issue. We observed that during training, the discriminator loss would eventually decrease significantly and the generator loss would stop decreasing, which is a common issue in GAN training where the discriminator outcompetes the generator. The behavior persisted despite adding noise to the discriminator inputs around the time when the discriminator loss started to drop.

Figure 1: Generator and discriminator loss



Plot of generator and discriminator loss during training. After around 20 training steps, the discriminator loss drops significantly and the generator loss no longer decreases.

We mostly used the same hyperparameters as the original paper, with exceptions of an embedding size of 300 instead of 100, clipping the number of discriminator and generator training batches at 40 instead of 20 and using Xavier initialization for trainable variables instead of fixing their largest singular values to 1.0. It was unclear what loss they used for pretraining the generator, but we decided that the negative cosine similarity was a sensible candidate. We observed a significant drop and convergence to a minimum in the pretraining generator loss, so we assumed that pretraining was not the issue.

6 Conclusion

We implemented and trained the CGAN architecture described in the original paper. While we were unable to reproduce the results of the paper, it is likely that hyperparameters, training procedure or minor model differences were the issue. CGANs are difficult to train, especially for text generation. The Story Cloze Task is a very challenging dataset when a neural machine comprehension approach is taken. Despite promising advances in deep learning and natural language understanding, machine comprehension remains an unsolved and incredibly challenging problem.

References

N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. *ArXiv e-prints*, April 2016.

B. Wang, K. Liu, and J. Zhao. Conditional generative adversarial networks for commonsense machine comprehension. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017. doi: 10.24963/ijcai.2017/576. URL `https://doi.org/10.24963/ijcai.2017/576`.

M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *ArXiv e-prints*, November 2014.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

K. Cho, B. van Merrienboer, C. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. *ArXiv e-prints*, November 2016.

B. Wang, K. Liu, and J. Zhao. Inner attention based recurrent neural networks for answer selection. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016. doi: 10.18653/v1/P16-1122. URL `http://www.aclweb.org/anthology/P16-1122`.

M. Arjovsky and L. Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *ArXiv e-prints*, January 2017.