NLU Report - Project 2: Story Cloze Task

Martina Forster **Adrian Meier** **Christina Puthenkalam** **Virginia Ramp**

1 Introduction

The Story Cloze Task [5] is a problem that humans can solve with 100% accuracy, while it remains incredibly difficult for machines to achieve comparable results. The challenge is, given a context of four sentences and two proposed endings, to decide which is the correct ending to the story. Given are a training set, consisting of the context sentences and right endings, and a validation set, which additionally contains wrong ending sentences.

There are several challenges connected with this task: First of all, the given wrong endings are grammatically correct and relate to the theme of the story. Therefore, the detection of the right ending requires a certain common sense knowledge. Secondly, the training set does not contain wrong endings. Hence, machine learning algorithms that require negative examples can only be trained on the significantly smaller validation set. This may lead to bad performance of systems that need more data to train. Furthermore, these systems do not make use of the information contained in the training data.

In order to tackle the issue of missing wrong endings in the training set, we try to generate our own. And to introduce common sense knowledge, we use Skip-Thought embedding vectors [2]. Our final model reached a validation accuracy of 76.96%.

2 Methodology & Model

We followed the approach of Srinivasan et al. [6] and used a discriminative method to predict which of the two endings has a higher probability of being correct. We did this by making use of 4800-dimensional Skip-Thought embeddings [2] in combination with a three-layer feed-forward neural network (FFNN). To embed our sentences we use a pre-trained Skip-Thought encoder [1]. The feed-forward neural network takes an input vector and outputs the probability of the current ending being correct. After obtaining these probabilities for both possible endings, the one with the higher probability of being correct is chosen.

As in [6], we implemented the following different modes: "No Context" (NC), where the input is only the Skip-Thought embedded fifth sentence, "Last Sentence" (LS), where the input to the neural network is the sum of the Skip-Thought embeddings of the fourth and the fifth sentence, and "Full Context" (FC), where the whole context, consisting of the first four sentences, is encoded and then added to the Skip-Thought embedded ending, before getting passed as an input to the neural network. While Srinivasan et al. [6] encode the four context sentences by feeding their Skip-Thought embeddings into a Gated Recurrent Unit (GRU), we decided to simply get one Skip-Thought embedding of all four sentences, instead of embedding them separately. In addition, our model differs from the original implementation by including dropout layers which were added in order to improve accuracy. An overview of our adapted system can be found in Figure 1.

[1]https://github.com/ryankiros/skip-thoughts

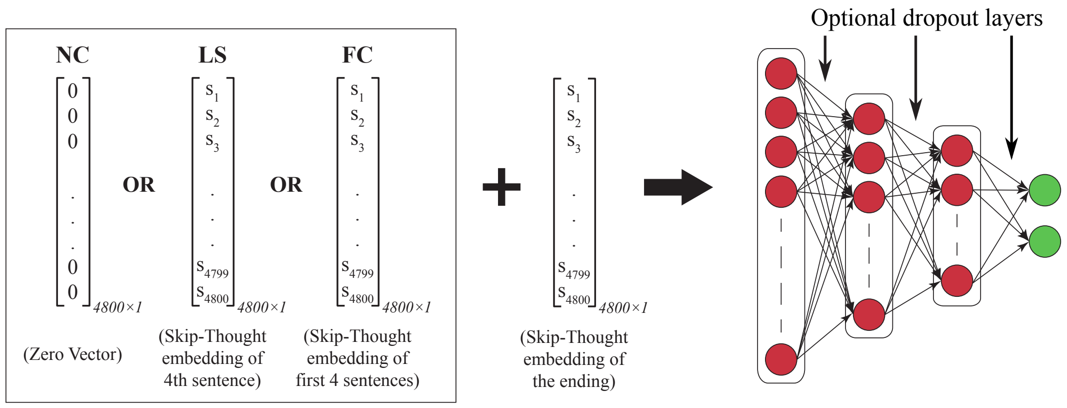
NC
LS
FC
OR
OR
(Zero Vector)
(Skip-Thought embedding of 4th sentence)
(Skip-Thought embedding of first 4 sentences)
(Skip-Thought embedding of the ending)
4800×1
Optional dropout layers

Figure 1: Overview of our system, adapted from [6]

2.1 Data generation

Since our model needs both right and wrong endings to train, our training data was limited to the given validation set. But this set is significantly smaller than the given training set. Thus, we decided to augment the stories of the training set by generating wrong endings. To do so, we pursued two different strategies: The first one was using a Long Short-Term Memory (LSTM). The second and more complex method was using a Generative Adversarial Network (GAN).

2.1.1 Long Short-Term Memory (LSTM)

This approach uses a Recurrent Neural Network (RNN) with LSTM cells to build a language model which generates sentences based on a given context.
Our RNN consists of three layers: an embedding layer which is initialized with a pre-trained 100-dimensional Word2Vec [4] embedding, an LSTM layer and a dense layer with softmax activation.
As training input, we take n-grams of the full training stories. To generate new sentences, we use the last n words of the first four sentences, appending the newly generated word at each time step, while removing the first word to keep the context size invariant. Since the performance of LSTMs decline with longer input sequences, we consider the choice of n=20 a good compromise between enough context and short sequences.

2.1.2 Generative Adversarial Network (GAN)

To improve the data generation, we aimed to implement a Generative Adversarial Network [1], which has an LSTM as generator and a CNN as discriminator. Zhang et al. [7] propose a model which trains an LSTM to generate single sentences, then feeds them into the CNN to discriminate whether the sentences are generated or real. We used the implementation from [8], which only processes single sentences, and adapted it in order to take the context of four sentences into consideration. Also, to further improve the model, we used a 100-dimensional Word2Vec [4] embedding instead of the proposed uniform random embedding.

3 Training

3.1 Feed-Forward Neural Network (FFNN)

The training of the FFNN was done using the sparse categorical cross-entropy loss function, optimizing with Stochastic Gradient Descent (SGD) and a learning rate of 0.01. In the setup with dropout layers, we used a dropout rate of 0.5.
Our training was done on two data sets, the first one being only the validation set, and the second one being the validation set augmented with 5000 stories from the original training set with wrong endings generated by our LSTM.
We chose to only include 5000 augmented stories for two reasons. Firstly, the Skip-Thought em-

bedding [2] of the sentences takes a lot of time. Secondly, when comparing the results of 5000 and 10000 additional sentences, we noticed that the prediction accuracy declined for the bigger dataset.

3.2 Long Short-Term Memory (LSTM)

Our LSTM was trained with the sparse categorical cross-entropy loss function and optimized with the Adam optimizer, using a learning rate of 0.001.
We found that the generated sentences were better and more complex when training the LSTM for more epochs. The choice of 10 epochs yields good sentences at a fairly low training time.

3.3 Generative Adversarial Network (GAN)

We pre-trained our generator G with the training data, and the discriminator D with the validation data. Both G and D were trained with the full story context. In each adversarial training step, G aimed to generate story endings which D could not distinguish from the correct ending. On the other hand, D minimized its discrimination loss. For the formal definition of the objective functions, we refer to [7].

Our GAN suffered from mode collapse [3]. We experimented with different training-step-to-prediction-step ratios per adversarial step, aiming to find a balance, where both G and D learn at a similar rate. In addition, we adapted the number of training steps per epoch, to see if a shorter or longer feedback-loop between G and D had an impact on the generated story endings. In the end, we were unable to tune the GAN such that it produces higher quality sentences than the LSTM alone. Hence we did not conduct any experiments using the data generated by G.

4 Experiments

Experiment Mode	Max. Validation Accuracy	Number of Epochs
VAL NC no dropout	0.7364973262032086	100
LSTM-END NC no dropout	0.6951871657754012	80
VAL NC dropout	0.735450563204005	15
LSTM-END NC dropout	0.6987522281639929	10
VAL LS no dropout	0.74560814654682	100
LSTM-END LS no dropout	0.7023172905525846	70
VAL LS dropout	0.7664353168733644	80
LSTM-END LS dropout	0.7040998217468806	40
VAL FC no dropout	0.7455882352941177	95
LSTM-END FC no dropout	0.7379679144385026	75
VAL FC dropout	**0.7696381840937536**	65
LSTM-END FC dropout	0.7147950089126559	100

Table 1: Experimental results. For each mode, shows the maximum validation accuracy that was achieved and at which number of epochs it was achieved.

All our experiments were run on Leonhard. We performed two types of experiments: Training only on the validation set (VAL) as suggested by [6], and training on 90% of the validation set combined with 5000 samples of the training set enhanced with LSTM-generated wrong endings (LSTM-END).

For both VAL and LSTM-END we performed experiments for "No Context" (NC), "Last Sentence" (LS) and "Full Context" (FC) modes. For each of them, we did the experiment with and without dropout layers. When dropout was applied, we also normalized the data. The results of these experiments can be found in Table 1. We trained each mode for different epoch numbers ranging from 5 to 100 epochs, with a step size of 5. For the VAL experiments, we did 10-fold cross-validation, i.e. we trained on 90% of the validation data, used the remaining 10% for validation, and then took the average of the resulting 10 accuracies. For the LSTM-END experiments, we performed the experiment 3 times, each with a different random 10% of the validation data as 'real' validation data, and averaged these three accuracies.

We observed the best validation accuracy for VAL FC dropout, achieving an accuracy of 0.7696 (see Table 1). For all modes, the highest measured validation accuracies of the LSTM-END experiments were lower than the ones with VAL. This indicates that it might not be beneficial to augment the training data with wrong endings generated by our LSTM.

While Srinivasan et al. [6] achieved an accuracy of 0.738 in the VAL FC experiment, we observed an accuracy of 0.7456 without dropout and 0.7696 with dropout with our approach (as shown in Table 1).

Also, comparing the 'LSTM-END' experiments with the 'trn' experiments from [6] (where they trained on the whole training set, using ending sentences from different samples as wrong endings), shows that 'LSTM-END' performs better in terms of accuracy. This might be due to various reasons: Firstly, we did not use random ending sentences from different samples, but generated the wrong endings with an LSTM. Secondly, Srinivasan et al. trained on the full training set, while we trained on 90% of the validation set and 5000 samples from the enhanced training data.

Based on our experiments, we decided to choose the 'VAL FC dropout' mode to process the given test set. Table 2 shows the accuracy on the original test set for a selection of modes and 65 epochs of training. We performed three repetitions per experiment and computed the average.

Experiment Mode	**Test Accuracy**	**Standard Deviation**	**Number of Epochs**
VAL NC dropout	0.721361125957598	0.0072674427416	65
VAL LS dropout	0.748441118831284	0.0017636727127	65
VAL FC dropout	0.76607874576875	0.0031569651072	65

Table 2: Experimental results. For each mode, shows the average test accuracy out of 3 repetitions that was achieved after training for 65 epochs, and the standard deviation.

5 Future work

Our generative approach aims to produce endings which are indistinguishable from the original good endings. Ultimately, this is not what we want. It would be interesting to generate data that resembles wrong endings instead. The sentences would need to be of a quality such that it should not be possible to determine the correct ending without the story context. However, they are still required to match the context of the stories. The differentiating factor needs to be the logical connection to the story. Maybe one could "negate" the right endings cleverly to achieve this.

6 Conclusion

To build our model we followed the approach of Srinivasan et al. [6] with some alterations. Our best validation accuracy of 76.96% was obtained using our VAL FC dropout model. This model only trains on the validation set and uses our "Full Context" mode, which embeds all four context sentences at once with the Skip-Thought encoder. The results indicate that our version of the FC mode yields better results than the original paper's. Additionally, our final model has a dropout layer which prevents overfitting. This is especially beneficial in our case since we only learn on the rather small validation set.
As for augmenting the given data with data generation, our experiment results indicate that our LSTM-generated endings do not lead to a better performance. Thus, we conclude, that the quality of the generated sentences must be very good for the augmented data to be useful.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[2] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[3] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[5] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. A corpus and evaluation framework for deeper understanding of commonsense stories. *CoRR*, abs/1604.01696, 2016.

[6] Siddarth Srinivasan, Richa Arora, and Mark Riedl. A simple and effective approach to the story cloze test. *arXiv preprint arXiv:1803.05547*, 2018.

[7] Kai Fan Zhi Chen Ricardo Henao Dinghan Shen Lawrence Carin Yizhe Zhang, Zhe Gan. Adversarial feature matching for text generation. 2017.

[8] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *SIGIR*, 2018.