A Case Study on the Story Cloze Task

Anagnostidis S. **Sachinoglou I.** **Solomou A.** **Vasilakopoulos G.**

1 Introduction

In recent years, deep learning models have demonstrated state-of-the-art performance on a wide range of Natural Language Understanding (NLU) applications [1, 2, 3]. Yet, narrative story understanding remains an extremely challenging task within the research community. The recently proposed Story Cloze Task (SCT) [4] introduces an evaluation framework for story understanding. Given a four-sentence short story, one has to identify the correct ending of the story from two candidate ending sentences. The associated training corpus (referred to here as ROCStories) consists of crowdsourced five-sentence stories without ending alternatives. In contrast, test stories are in line with the SCT structure and consist of a four-sentence story context followed by two ending alternatives.

The lack of incorrect story endings during training hinders the direct application of binary classification models. Moreover, training solely on the validation set [5, 6] might fail to generalize to unseen stories since it greatly reduces the size of the training corpus. Various methods have been proposed to augment the ROCStories with negative training samples. Wang et al. [7] propose a Conditional GAN architecture to generate fake endings conditioned on the story context. Furthermore, Roemmele et al. [8] propose an RNN-based Language Model (LM) for conditional ending generation as well as techniques for sampling negative endings from the training corpus.

In our work, we first experiment with training on the ROCStories by employing two augmentation techniques, namely random sampling and conditional generation, reproducing the results of several papers [5, 6, 9]. Thereinafter, we demonstrate the effectiveness of fine-tuning pretrained language models on the validation set [10, 11]. Doing so, we achieve an absolute improvement of 5% compared to the current top-performing approach proposed by Chen et al. [9].

2 Methodology

In the majority of our approaches we train a binary classifier to distinguish between correct and incorrect endings given the context of the story. We examine different approaches regarding the representation of the data and the way to deal with the lack of incorrect endings in the ROCStories dataset.

2.1 Story Representations

We consider three embedding strategies for representing stories as input to our models. First, we use VADER [12], a sentiment analysis tool, to extract a 3-dimensional **sentiment polarity vector** for every sentence. The entries of this vector correspond to the probability of the sentence being positive, neutral and negative. Secondly, we consider a **word-level representation** for each story based on GloVe embeddings [13]. We use 100-dimensional vectors pretrained on Wikipedia 2014 and Gigaword 5 data[1]. In terms of text pre-processing, we make use of the tokenization module provided by BERT [11]. Thirdly, we consider a **sentence-level representation** scheme based on the Skip-Thought embeddings [14]. These are 4800-dimensional sentence embeddings arising from the concatenation of uni- and bi-skip models trained on the BookCorpus dataset [15]. An immediate advantage of a sentence-based representation is that each story can be represented by means of five vectors, greatly simplifying the structure of many neural classification models.

[1] https://nlp.stanford.edu/data/wordvecs/glove.6B.zip

2.2 Dealing with the Lack of Incorrect Story Endings

We employ two methods to overcome the lack of incorrect story endings in the ROCStories.

Incorrect Ending Generation We examine two techniques: 1) Given a ROCStory, we replace its ending with the ending of a randomly selected story from the training set and 2) Following the approach proposed by Roemmele et al. [8] we use an RNN-based LM to generate an incorrect ending conditioned on the story context. When training the LM we provide the context of the ROCStories but also the context of the validation sentences followed by the incorrect ending. The motivation behind this approach is to encourage the model to capture the semantics of incorrect endings during training. During generation, we provide to the LM the ROCStory context along with the beginning of the correct ending (from one to three tokens; chosen randomly) and produce an ending by greedily selecting words one by one according to the predicted probability distribution.

In this framework, we then use the augmented ROCStories dataset to train a binary classifier to learn the conditional probability of an ending sentence being plausible given the story context. During inference, we provide both ending alternatives to the model and select the one that has the higher probability of being true.

Correct Ending Generation The second approach is to train the model to generate an ending that is close in terms of cosine similarity to the true story ending for a given representation [9]. Then, during testing we choose the ending whose representation is closer to the one generated by the model. We experimented with this approach for both sentiment polarity and Skip-Thought sentence representation. We should make clear that this approach does not augment the ROCStories dataset but intends to directly predict an ending sentence that is close to the true one.

2.3 Supervised Fine-Tunning of Pretrained Language Models

Recent research has demonstrated that incorporating external commonsense knowledge can provide a significant boost in performance on several NLU tasks [9, 16]. In particular, LM pretraining has proved to be an effective method for incorporating language features [10]. Chen et al. [9] present a neural story ending selection model that leverages information from various sources, including a pretrained LM, to achieve state-of-the-art performance on the SCT. We build upon this method using BERT [11], a language representation model that is conceptually simple but empirically powerful.

The novelty of BERT is that it is trained to represent unlabeled text by jointly conditioning on both left and right context in all layers. Consequently, it captures semantic representations that allow the model to be easily adjusted to a wide range of NLU tasks as demonstrated empirically by Devlin et al. [11].

3 Models

3.1 Sentiment Analysis

We use an LSTM RNN to encode the sentiment of the story context and finally use the output of the RNN encoder to predict the sentiment vector of the story ending [9]. We train the model by maximizing the cosine similarity between the predicted and extracted sentiment vectors of the correct ending sentence. Our approach differs from [9] in the sense that we do not train on the validation set. During inference, we consider both ending alternatives and choose the one whose sentiment vector is closer (in terms of cosine similarity) to the one predicted by the RNN encoder.

3.2 Word-Based Classifier

Following the work in [6], we employ two distinct RNN encoders to encode the story context and story ending respectively. The two encoder outputs are concatenated into a single vector which is then fed as an input to a feedforward network with a single hidden layer and a softmax output layer. In contrast to [6], we found that using a plain Bidirectional RNN with LSTM units (BiLSTM) as the encoding unit is not capable by its own to capture the semantics of the story. We believe that this is due to the large length of the context encoder (100 timesteps) and the vanishing gradient problem [17]. We deal with this issue by adding the attention mechanism proposed by Bahdanau et al. [2] to both encoders. In this setting, the context vector of each encoder is used as its output.

3.3 Sentence-Based Architectures

In the context of sentence-based architectures we reproduce two of the approaches proposed in [5], namely the models "Full Context" (**FC-Skip**) an "Last Sentence" (**LS-Skip**). In the former, we use a 4800-dimensional GRU RNN to encode the story context and add the embedding representation of the ending sentence to the output of the RNN encoder. The result is then fed to a two-hidden layer feedforward network with softmax output unit. In the latter, we replace the RNN context encoder by simply the Skip-Thought embedding representation of the fourth sentence in the story. By concatenating the outputs of the story context encoder and the ending sentence embedding, instead of summing, we were able to achieve better results compared to the original implementation.

Inspired by the encoder-decoder architecture in sequence to sequence learning [18], we employ an LSTM RNN encoder-decoder architecture where the four-step encoder receives the embedded story context and the one-step decoder predicts the Skip-Thought representation of the story ending. We denote the model as "**Predict-Skip**". We train the model by maximizing the cosine similarity between the predicted Skip-Thought vector and the true one. During testing, we consider both ending candidates and choose the one whose Skip-Thought vector is closer (in terms of cosine similarity) to the predicted embedding vector.

3.4 BERT

We rely on two BERT models that are currently available[2], namely **Base** and **Large**. Our first approach is based on the **Pooled-Output** representation. This is the first token of the story which summarizes the representation of the entire story and is subsequently provided as input to a dense layer with two units and softmax activation. Another approach is the following. For each candidate story ending, we derive the transformed representation of the entire five-sentence story. Thereinafter, we provide only the extracted representation of the ending to a BiLSTM followed by a dense layer with two hidden units with softmax activation (**BiLSTM**). An alternative option is to use a Convolutional Neural Network (**Conv**) along the sequence length and hidden state axes. As before, these modules are succeeded by either a feedforward network (**FeedForw**) with softmax output unit or a Highway Network (**Highway**) [19]. The latter enables the optimization of networks with virtually arbitrary depth and allows the network to adaptively copy or transform representations. Finally, we also performed some data augmentation techniques [20], where for each epoch we randomly replace non-stop words in the stories with randomly selected synonyms. The purpose of this technique is to limit over fitting issues that can be caused by the large number of parameters compared to the number of samples.

4 Training and Experiments

We outline the training configuration used for each model in Table 1. When training on the ROCStories we use a 1:3 negative sampling ratio for "Random" and 1:1 for "Conditional Generation". In this case, we evaluate on both the validation and test set. Instead, when training on the validation set, we use the entire set for training purposes and therefore evaluate only on the test set. Further details regarding the training configuration of each model are provided in Tables 3 and 4 in Appendix A. Table 2 lists the accuracy achieved by each model on the corresponding dataset.

[2] https://github.com/google-research/bert

Table 1: Configuration for different architectures

Model	Representation	Negative Sampling	Loss	Optimizer
Sentiment	Polarity Scores	N/A	Neg. Cosine Similarity	Adam
Word-Based	GloVe	Cond. Generation	Cross-Entropy	Adam
Predict-Skip	Skip-Thoughts	N/A	Neg. Cosine Similarity	Adam
FC-Skip	Skip-Thoughts	Random	Cross-Entropy	Adam
LS-Skip	Skip-Thoughts	Random	Cross-Entropy	Adam
BERT Models	Default [11]	N/A	Cross-Entropy	Adam

Table 2: Accuracy score achieved by different models

ID	**Model**	**Validation**	**Test**
Models Trained on ROCStories			
1	Sentiment	0.577	0.587
2	Word-Based	0.603	0.552
3	Predict-Skip	0.615	0.613
4	FC-Skip (train)	**0.650**	**0.640**
5	LS-Skip (train)	0.621	0.631
Models Trained on Validation Set			
6	FC-Skip (validation)	N/A	0.765
7	LS-Skip (validation)	N/A	0.772
8	BERT-Base + Pooled-Output	N/A	0.870
9	BERT-Large + Pooled-Output	N/A	0.889
10	BERT-Large + Augmentation	N/A	0.886
11	BERT-Large + BiLSTM	N/A	0.897
12	BERT-Large + BiLSTM + Highway	N/A	0.900
13	BERT-Large + Augmentation + BiLSTM + Highway	N/A	0.905
14	BERT-Large + Augmentation + BiLSTM + Highway (ensemble)	N/A	0.923
15	BERT-Large + Augmentation + Conv + FeedForw	N/A	0.903
16	BERT-Large + Augmentation + Conv + FeedForw (ensemble)	N/A	0.917
17	Ensemble of 14 and 16	N/A	**0.926**

We notice that the performance of the models when trained solely on the ROCStories dataset is significantly poorer compared to the models trained or fine-tuned on the validation set. We argue that the reason for this is twofold. First, when training a binary classifier on the ROCStories by employing random negative sampling, the sampled ending is not semantically related to the story context. This is in contrast to the SCT where both endings in the test set are related to the story context. This justified empirically by observing the improvement in accuracy of "FC-Skip" when trained solely on the validation set instead of the ROCStories. Secondly, all the models relying on conditional generation are considerably less accurate. We argue that the amount of data provided to these models is not enough to generate good quality endings.

It is evident that fine-tuning approaches based on BERT significantly outperform the rest of our methods. This should not come as a surprise given that BERT has been pretrained on an extensive corpus allowing the model to incorporate external commonsense knowledge that is required to solve the SCT. We observe that the different fine-tuning techniques we use, such as BiLSTM or Augmentation, have an additive effect to the model's performance. It is well known that ensemble models tend to perform better [9] and this is in agreement with our findings. Finally, our results based on BERT are in line Radford et al. and Chen et al. which use OpenAI's pretrained language model to tackle the SCT outperforming the previous top-performing approach.

5 Conclusion

The SCT is an extremely challenging problem given the fact that a successful model must capture common sense knowledge regarding the language. By working in an incremental fashion, we experiment on different data representations, different ways of dealing with the lack of wrong endings and different training methodologies. Finally we come to the conclusion that fine-tuning of pretrained language models, namely the BERT model, can achieve state-of-the-art performance on the SCT.

References

[1] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[3] Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu. Direct speech-to-speech translation with a sequence-to-sequence model. *CoRR*, abs/1904.06037, 2019.

[4] 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.

[5] Siddarth Srinivasan, Richa Arora, and Mark Riedl. A simple and effective approach to the story cloze test. *CoRR*, abs/1803.05547, 2018.

[6] Michael Bugert, Yevgeniy Puzikov, Andreas Rücklé, Judith Eckle-Kohler, Teresa Martin, Eugenio Martínez-Cámara, Daniil Sorokin, Maxime Peyrard, and Iryna Gurevych. LSDSem 2017: Exploring data generation methods for the story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 56–61, Valencia, Spain, April 2017. Association for Computational Linguistics.

[7] Bingning Wang, Kang Liu, and Jun Zhao. Conditional generative adversarial networks for commonsense machine comprehension. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4123–4129, 2017.

[8] Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew Gordon. An RNN-based binary classifier for the story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 74–80, Valencia, Spain, April 2017. Association for Computational Linguistics.

[9] Jiaao Chen, Jianshu Chen, and Zhou Yu. Incorporating structured commonsense knowledge in story completion. *CoRR*, abs/1811.00625, 2018.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[12] Shravan I.V. Sentiment analysis in python using nltk. *OSFY - OpensourceForYou*, 12 2016.

[13] 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.

[14] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., 2015.

[15] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[16] Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Improving question answering by commonsense-based pre-training. *CoRR*, abs/1809.03568, 2018.

[17] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[19] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[20] Jason W. Wei and Kai Zou. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196, 2019.

A Model architectures

Table 3: Description of each model

ID	Description
1	LSTM cell with hidden state 3 followed by softmax layer
2	128-unit BiLSTM encoder with attention for story context (120 timesteps) and story ending (30 timestep). Dropout (0.5) is applied to the concatenation of the outputs. Feedforward network with one hidden layer (64 units, dropout 0.5 relu) and sigmoid output unit.
3	LSTM cell with hidden state 4800 followed by tanh layer
4	GRU cell with hidden state 4800 followed by feed forward network 256-64-2 followed by softmax layer
5	Pass outputs of concatenated input to a feed forward network 2400-1200-600-2 followed by softmax layer
6	Same as 4
7	Same as 5
13	BiLSTM (1024 hidden units) on the outputs corresponding to the last sentence followed by a Highway network of depth 3
15	Two convolutions of kernel sizes [5,5] and [5,5] with 32 and 16 filters, followed by max pooling layers of size [2,2] and stride 2. Finally a Dense network with 512 layers, relu activation and 0.1 dropout

Table 4: Hyperparameters of each model

ID	Learning rate	Decay	Batch size	Epochs
1	1e-3	N/A	64	10
2	5e-3	Exponential	64	8
3	1e-3	N/A	16	10
4	1e-4	Exponential	64	10
5	1e-4	Exponential	64	5
6	1e-3	Exponential	32	20
7	1e-3	Exponential	32	10
13	2e-5	Polynomial	16	5
15	5e-6	Polynomial	16	7