An RNN-Based True-In-Spirit Approach to the Story Cloze Task

Zhao Meng
zhmeng@student.ethz.ch

Meng-Yuan Yang
meyang@student.ethz.ch

Zacharias V. Fisches
zfisches@ethz.ch

Kangning Liu
liuka@student.ethz.ch

Abstract

In this paper, we treat the *story cloze task (SCT)* as a binary classification problem. We analyze shortcomings of the SCT design and then evaluate our own flavor of neural language model applied to the task, relying on methods from [10] for generation of negative training samples. Our approach tries to stay true in spirit to the SCT and slightly improves over current in-kind results with an accuracy of 59.65%, when compared under narrow constraints, while staying far behind the overall state of the art.

1 Introduction

In the context of Natural Language Understanding (NLU) it has been an elusive goal to develop systems that capture the semantics of narratives and demonstrate commonsense understanding. In 2016, Mostafazadeh et al. (2016) [8] published the ROCStories corpus of self-contained five-sentence storys and the *story cloze task* where additional crowd-sourced *right* and *wrong* story endings where commissioned for a subset of the corpus such that humans could always discern the *right* ending as the only plausible conclusion among the two. [8] showed that several advanced NLU systems fared only slightly better than chance on the *story cloze task* (SCT), with DSSM [5] reaching 58.5%, and conjectured that true commonsense understanding was required to perform well on the SCT.

The SCT has since been addressed through various methods inspired heavily by recent advances in neural machine translation [4, 1, 7]. Notably in 2017 by Roemmele et al. [10] who generate their own *wrong* endings for the whole ROCStories corpus in order to train an RNN-based binary classifier achieving 67.2% accuracy and by Cai et al. [3] using an attention-augmented BiLSTM achieving 74.7% accuracy.

Despite the stated goal to prevent superficial features from being sufficient to predict the *right* story endings [8], it was demonstrated convincingly that this is not actually the case. By drawing from years of research in linguistic style analysis (notably research related to *deceptive statements*), Schwartz et al. (2017) [11] were able to prove that the assignment given to authors to write *right* and *wrong* story endings introduces sufficient bias that the endings are so different in kind that a logistic regression over a couple of linguistic features achieves 72.4% accuracy without taking the stories themselves into consideration at all. This result is corroborated empirically by Cai et al. [3] who still achieve 72.5% accuracy by learning features exclusively on the endings.

As a consoling fact though, both [3] and [11] achieve slightly higher accuracy by combining these features with features extracted from the stories. Schwartz et al. [11] combine above mentioned features with those learned by a standard RNN language model, to achieve state-of-the-art accuracy of 75.2%.

In the spirit of the *story cloze task (SCT)* this work forgoes the easier accuracy gains possible by exploiting weaknesses in the dataset design and instead follows the approach of [10] by formulating the SCT problem as a supervised binary classification problem. Although the state of the art [3, 11]

ETH NLU 2018 project 2.

disregards the ROCStories training set of $\sim 5 \cdot 10^4$ stories entirely and only trains on the validation set of less than 2000 stories with *right* and *wrong* endings, we will again choose the more principled approach of only training on the training set by ad-hoc generation of negative story endings for the purpose of training the classifier.

For the generation of negative sentences, we again follow [10] and mix three different sampling methods: (1) *Random:* uniformly sampling an ending from another story, which in most cases gives us semantically unrelated endings. (2) *Backward*: sampling one of the first four sentences from the same story, which gives us semantically related but implausible negative endings. (3) *Nearest-Ending* (abbreviated as "near" later): sampling uniformly from the endings of the six most similar stories in the dataset, where similarity is defined as the *cosine similarity* between the *tf-idf* transformations of the stories represented in a *bag of words* document model. This provides us with semantically similar but still incoherent endings. These methods together provide us with negative endings exhibiting different properties and degrees of semantical relation to the original endings.

An obvious choice in light of results from [3] and recent successes in language modeling with LSTM-based recurrent neural networks (RNN) by [2, 4] and others is to base a *narrative model* on these methods as well. In short, our approach in this project could be characterized as exploring the efficacy of a best-practice neural language model when applied in a principled and true-in-spirit fashion to the *story cloze task*. The model we investigate achieves an accuracy of 59.65% on the original story cloze test set and 59.59% on the test set distributed with this project, thereby slightly beating the best result of 57.9% that Roemelle et al. [10] achieve when relying on word-level embeddings like we do here.

2 Model

Our model is made up of a sentence encoder, an attention-based feature aggregation and a non-linear regression through a feed-forward neural network (FFNN). As input the sentence encoder takes pre-trained 300-dimensional GloVe [9] word-vectors that are mapped to context vectors, aggregated and then used as features for scoring the two candidate endings. Figure 1 illustrates our model.

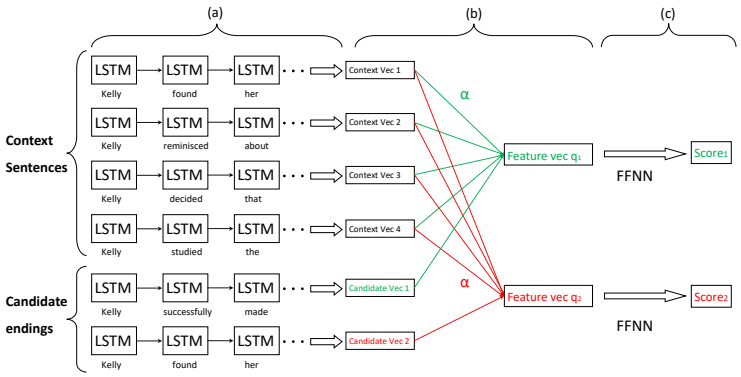

Figure 1: The model consists of an LSTM-based sentence encoder (a) for feature extraction (attention mechanism ommitted), feature aggregation via an attention mechanism (b), where the candidate vectors *attend* to the context vectors and a non-linear regression (c) for scoring of the candidate endings.

2.1 Sentence Encoder

We use a single (i.e. sharing weights) 1-layer recurrent neural network (RNN) with 512 LSTM units to encode the four context sentences and the two ending sentences into *context vectors*, where a context vector $\boldsymbol{a} = [\boldsymbol{v}; \boldsymbol{o}_t]$ is a concatenation of the last output $\boldsymbol{o}_t$ of the LSTM and the convex combination $\boldsymbol{v} = \sum_{i=1}^{t-1} \beta_i \cdot \boldsymbol{o}_i$ of the outputs o_i at each time-step of the LSTM. Learning the weights of a convex combination of some target vectors, we can in principle gather information from those target vectors that matter most and disregard others. Combining outputs from different time-steps like this has proved very effective in preserving important information in the aggregate [1]. The specific form of this *attention mechanism* was first introduced in [1] and determines the weights β_i: Let $o_1, ..., o_t$ be the outputs of the LSTM for a sentence of length t. We use o_t to *attend* to all its previous outputs $o_1, ..., o_{t-1}$ by defining β_i as

$$\beta_i = \frac{\boldsymbol{u}^\intercal \tanh(\boldsymbol{W}[\boldsymbol{o}_t; \boldsymbol{o}_i])}{\sum_{i=1}^{t-1} \boldsymbol{u}^\intercal \tanh(\boldsymbol{W}[\boldsymbol{o}_t; \boldsymbol{o}_i])} \,. \tag{1}$$

We choose the dimensions of the learned parameters as $\boldsymbol{u} \in \mathbb{R}^{512}$ and $\boldsymbol{W} \in \mathbb{R}^{512\times1024}$.

2.2 Feature Aggregation

Since *attention mechanisms* have proved effective in many NLP applications, including machine translation [7], relation classification [12] and others, we also leverage a similar mechanism during feature aggregation (Figure 1 (b)) to obtain the feature vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. We denote the sentence vectors as $\boldsymbol{a}_1, ..., \boldsymbol{a}_6$, where $\boldsymbol{a}_1, ..., \boldsymbol{a}_4$ are sentence vectors of the context sentences, $\boldsymbol{a}_5, \boldsymbol{a}_6$ are sentence vectors of the two candidate ending sentences. Then

$$\alpha_{ji} = \frac{\boldsymbol{u'}^{\intercal} \tanh(\boldsymbol{W'}[\boldsymbol{a}_j; \boldsymbol{a}_i])}{\sum_{i=1}^{4} \boldsymbol{u'}^{\intercal} \tanh(\boldsymbol{W'}[\boldsymbol{a}_j; \boldsymbol{a}_i]} \,, \qquad \text{parametrized by } \boldsymbol{u'} \in \mathbb{R}^{512}, \boldsymbol{W'} \in \mathbb{R}^{512\times1024} \tag{2}$$

$$\boldsymbol{q}_{j-4} = \left[\sum_{i=1}^{4} \alpha_{ji} \cdot a_i; a_j \right] \,, \qquad (i \in \{1,2,3,4\},\, j \in \{5,6\}) \,. \tag{3}$$

The candidate ending sentence vector $\boldsymbol{a}_5$ ($\boldsymbol{a}_6$) attends to context sentence vectors $\boldsymbol{a}_1, ..., \boldsymbol{a}_4$, yielding an vector, which is concatenated with $\boldsymbol{a}_5$ ($\boldsymbol{a}_6$) to obtain the final feature vector $\boldsymbol{q}_1$ ($\boldsymbol{q}_2$). Weights of the aggregations are shared across the two candidate sentences.

To the best of our knowledge, the context vectors have not been aggregated like this in previous work, where the aggregation is sometimes performed through a RNN as well.

2.3 Scoring

After obtaining the feature vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, we feed them to a fully connected feed forward neural network (FFNN) to get the final score s_1 and s_2, which are two scalars expressing how coherent an ending to the story the two candidate sentences are. We use ReLU activation for the two 300-dimensional hidden layers of the FFNN. Figure 1 (c) illustrates the scoring mechanism of our model. We then choose the candidate sentence with the higher score as the predicted ending sentence. Formally, $s_i = \text{FFNN}(\boldsymbol{q}_i)$, for $i = 1, 2$.

3 Evaluation

We trained the proposed model in an end-to-end fashion. Each training sample consists of six sentences: four context sentences c_1, c_2, c_3, c_4, one original *right* ending sentence e_a, and one generated (*wrong*) ending sentence e_b. We used a softmax function to normalize the scores s_1 and s_2, and then we maximize the log probability of the correct ending sentence. Formally, $p_i = \exp(s_i)/\sum_{i=1}^{2} \exp(s_i)$ and the loss $L = -\sum_{i=1}^{2} \log p_i \cdot y_i$, where y_i is the label ($i \in \{1, 2\}$), indicating whether candidate ending sentence i is the correct ending sentence.

3.1 Hyperparameters

In all our experiments we used 300 dimensional pretrained GloVe word-level embeddings [9]. The pre-trained embeddings (on a word- and sentence-level) used have been shown to have a large effect on the performance [10], but optimization of this categorical hyperparameter is out of the scope of this project. The hidden units of the RNNs are set to be 512 dimensional. We use 2-layer FFNN with 300 dimensional hidden units in our experiments. To regularize our model, we apply dropout in all our experiments during training. The dropout rate is set to be 0.2. We use Adam [6] with default parameters to train our model.

3.2 Experiments

In addition to our full model introduced in section 2, we experimented with two other variants to better understand different parts of our model, namely we have:

Vanilla: We do not use any attention in this model. $\boldsymbol{a}_i = \boldsymbol{o}_i$ in the encoder and the attention mechanism in the aggregation is replaced by a dense 2-layer NN $f_\theta(\cdot) \in \mathbb{R}^{512}$ with ReLU activation, such that $\boldsymbol{q}_i = (f_\theta(\boldsymbol{a}_1; \boldsymbol{a}_2; \boldsymbol{a}_3; \boldsymbol{a}_4); \boldsymbol{a}_{i+4})$, which takes the context sentence vectors and the candidate sentence vector as input, and outputs the attention vector.

Single attention: In this variant, we remove the attention in the LSTM encoder, but keep the attention for aggregation.

Combination	Backward:1								
	Near:1			Near:3			Near:5		
	Rand:1	Rand:3	Rand:5	Rand:1	Rand:3	Rand:5	Rand:1	Rand:3	Rand:5
Accuracy%	59.75	58.63	**59.65**	57.78	57.88	59.01	58.20	57.67	58.47

Table 1: Test accuracy of Vanilla model with mixed negative endings. The model reaches its overall highest validation accuracy with 1 backward, 1 near, and 5 rands (corresponding test accuracy bold in table).

Performance[1] Three plots are shown in the following figure to illustrate the performance of different models (i.e. Full attention, Single attention, Vanilla) over the factor of over-sampling of negative endings and ending generation methods.The highest performing model was chosen for the experiments shown in Table 1. It displays the performance of the Vanilla model with different combinations of negative endings.

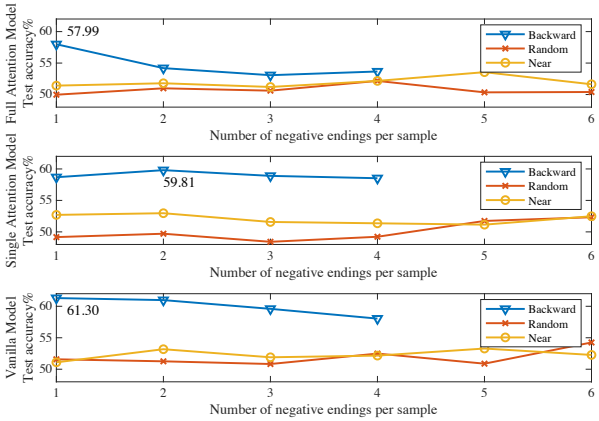

Figure 2: Performance of different models: negative endings are produced from three generation methods with different numbers for oversampling

Analysis As for methods of negative endings generation, it appears that for all the models and different number of negative endings per sample, backwards method achieves the best performance; the "near" method achieves similar performance as random method, relative performance of these two models depends on specific model and number of negative endings. In terms of models, after proper selection of negative ending generation method and number per sample, the Vanilla model achieved the highest test accuracy using one negative ending per sample generated from the "backwards" method. We found in figure 2 that the model reaches its highest performance when using the Vanilla settings and fixing the number of backward sentences to 1. We then used the vanilla model in another set of experiments, shown in table 1. For 9 different combinations, the model achieved best test accuracy with the combination with 1 backward ending, 1 near ending and 1 random ending, which is still lower than only using one negative ending generated from backwards methods. However, when considering all the experimental settings, the highest validation accuracy appears when we set the number of backward sentences, random sentences and nearest ending sentences to 1, 5 and 1, respectively. Hence, we report our model performance on the standard story cloze test test set as 59.65%. The same model is used to evaluate the performance on the test set provided by TA.

4 Conclusion

In this project, we used an LSTM-based classifier for the story cloze test task. We used three ways to generate negative samples for the training set. The model accuracy was 59.65% on the original test set and 59.59% on the test set provided by TA. This result slightly improves on the best *in-kind* result that we know of from [10], as discussed in Sec. 1, while staying far behind results that exploit strong weaknesses in the dataset design.

[1]The accuracy reported in this section is on the standard SCT test set, associated with the highest validation accuracy on the standard SCT val set.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[3] Zheng Cai, Lifu Tu, and Kevin Gimpel. Pay attention to the ending: Strong neural baselines for the roc story cloze task. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 616–622, 2017.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[5] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[8] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[10] 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 LSDSem*, pages 74–80, Jan. 2017.

[11] Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A Smith. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. *arXiv preprint arXiv:1702.01841*, 2017.

[12] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212, 2016.