# The Story Cloze Task with the ROCStories Dataset

**Joe Gelsomini**
**Scott Gordon**

## Abstract

The Story Cloze Task is a natural language understanding evaluation framework for evaluating story understanding and commonsense reasoning capabilities. Mostafazedeh et al have introduced their ROCStories dataset, with the intent to provide a framework for testing NLP models on the Story Cloze task. We aimed to study the approaches others have taken in regards to solving this task, as well as fine tuning our own adaptations inspired by what we have seen in the literature. Our main goal was to have a system that could generate a distinction between coherent and incoherent stories with moderate success, but we encountered problems that we believed to be attributed to the nature of our approach, as well as with data shortages and data quality. Even with a high quality pretrained model such as BERT, we struggled to make strong distinctions between sensical and nonsensical stories. Our reported accuracy from using Bert was actually worse then our final models we had been using before swapping to Bert. Our greatest success with our experiments were confirming findings made by Cai et al in their paper, *Pay Attention to the Ending: Strong Neural Baselines for the ROC Story Cloze Task* which showed that a system could evaluate the correct ending for a story by processing the ending itself, while discarding the context. Their system had an accuracy of 72.5%, which is close to the state of the art for the ROCStories dataset. In our paper we will explain the approaches others have taken as well as show our experiments and findings, followed by our discussion and insights into the results.

## Introduction

As stated before, the story cloze task is a relatively new commonsense reasoning framework for evaluating story understanding, script learning, and story generation. The problem that NLP systems have to tackle is conceptually simple; given the first four sentences of an “incomplete” story, as well as two possible candidate endings, a system needs to pick the ending that makes the most sense for a “complete” five sentence story. For humans, completion of this task is trivial, just looking at a few samples from the dataset, it is easy to see why human accuracy on this task  is 100%. For NLP models however, not so much. The ROCStories team have a paper published on their website comparing results of other scholars within the NLP community, with the current state of the art giving 75% accuracy for this task, more on that approach later. A high level view of our approach is straightforward, we wanted to see if we could train a model to make a distinction between coherent and incoherent stories by simplifying the task to a classification problem. Our first attempt was with a Gated Recurrent Unit (GRU) to create a classification between the two story types, doing this with the original training set was not possible, since it only consisted of complete coherent stories. So we had to modify the training corpus so we could have a system train on coherent and incoherent stories. We hypothesized that modifying the train corpus to have equal amounts of coherent and incoherent stories was flawed. The resulting training corpus we thought was too dissimilar to the testing and validation set. Afterwards, we adopted an approach taken by Cai et al, by discarding the original training set of 50k examples, and using the validation set for training. The motivation for doing this was to have a training and testing set that were more similar to each other in terms of appearance and story quality. We also began to conduct our experiments on the independent story endings as well as for full stories, to confirm that there were possible biases present in the story endings for the dataset. We found an increase in performance for classifying endings over classifying full stories, and this stayed consistent as we scaled up our models. The first attempt to scale up the model was to  initialize the embedding matrix of the GRU with the embedding vectors of the 300 dimensional pretrained GLOVE embeddings. The thought was that the pretrained vectors would give our model the greater ability to pick out important word usage and have a greater ability to differentiate coherent and incoherent stories. Our final attempt at scaling up the model was swapping our GRU layer of our model with an LSTM layer, while experimenting with and without bidirectional capabilities. Each new experiment did provide slight improvements over the previous, but we thought maybe the models we were using were not powerful enough, so we decided to use the  base pretrained BERT uncased model. The results from BERT were less successful than we had hoped. It even gave worse testing accuracy than our previous simpler model, which we will later discuss in our final discussion towards the end. Overall we speculate that the nature of our approach, combined with insufficient amounts of data for both Bert and our past models were the main contributors to the lack of performance.

## Related Work

In Mostafazedeh et al's original paper, *A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories*, they introduced the ROCStories dataset. The motivation for releasing the dataset was to provide a framework for evaluating understanding of common sense and of correlational and causal events. They also proposed several shallow models to complete the task. These models were not made for the purpose of solving their own task, but rather to show that more sophisticated models are needed to generalize some sort of commonsense understanding. The models are scored with a simple evaluation metric, the accuracy is the number of correct guesses a system makes divided by the number of samples it is tested on. As a baseline, they tested with a constant choose first method, so the system always guesses the first candidate ending. Accuracy is what you'd expect, 51.3%. A few examples of other simple models they showed were choosing the ending with a greater N-gram overlap with the context, choosing the ending with an average word embedding that was closest to the context, or choosing the ending that had a sentiment closer to the context. All the simple models proved to be a negligible upgrade from the constant choose first baseline. Their most complex model which gave them the best performance was the use of a Deep Structured Semantic Network (DSSM).

This model takes the four sentence context, and each candidate ending, and maps all three items to the same latent vector space, where the candidate ending with the highest cosine similarity to the original context is chosen as the models guess. This model gave 58% accuracy. The authors left it to the NLP community to develop a better system. The ROCStories team have also published a paper on their website, *LSDSem 2017 Shared Task: The Story Cloze Test* which highlights the NLP communities work on the problem. The state of the art model that they shared was developed by researchers of The University of Washington. The paper described the model as a Linear classifier based on language modeling probabilities of the entire story, and linguistic features of only the ending sentences, where these features are things such as sentence length, as well as word and character N-grams present in the endings.

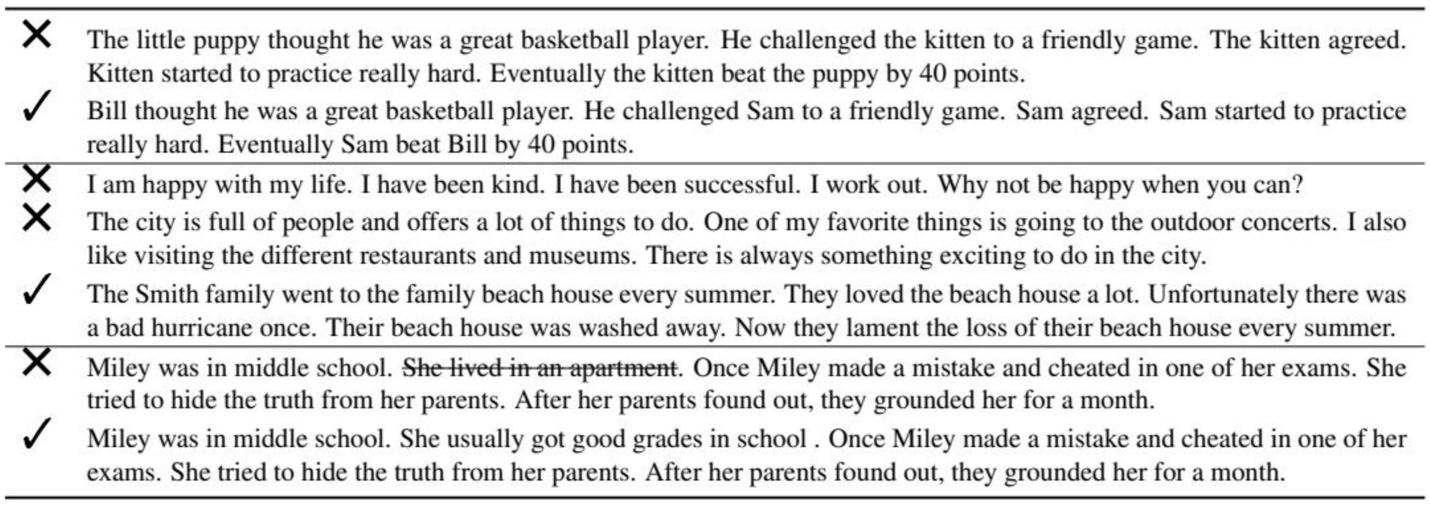
Cai et al had an interesting approach in solving the task in their paper. Researchers in the LSDSem 2017 shared task as well as Cai et al observed that better performance could be achieved with supervised learning strategies on the validation set as an alternative training corpus. The reason why this was is because the validation set has true and false endings, and the supervised learning methods involved training a system to learn the distinction between sensical and nonsensical stories.

## Experimental Setup:

## The dataset:

The ROCStories dataset has had several different releases. For our experiments, we used the Winter 2017 training corpus, with 52k complete sensical five sentence stories. We also used the 2016 release of the test and validation set. We went with the 2016 releases for the test and validation since the test set had labels identifying the correct ending, where the other releases we saw were blind tests. The sets can be obtained from an email request which can be found from their website at : https://cs.rochester.edu/nlp/rocstories/. The ROCStories team built their dataset with two goals in mind. First they wanted a corpus "containing a variety of commonsense causal and temporal relations between everyday events, to enable learning a narrative structure across a range of events, as opposed to a single domain or genre." Their second goal was to have "a high quality collection of nonfictional daily short life stories, which can be used for training rich coherent story-telling models." (Mostafazedeh et al). The dataset was crowdsourced, and to narrow down the nature of the stories in their set, they provided tight definitions of a narrative for their set. They define a narrative or story as anything which is told in the form of a causally (logically) linked set of events involving some shared characters. The resulting dataset is relatively uniform, with no edge cases.

The table below from Mostafazedeh et al's paper gives you an idea for how they controlled the quality of stories from the crowdsourced writers. Stories would be omitted if they didn't have a centralized plot or narrative. They would be rejected or modified if they contained information that was unimportant to the narrative, and they are made to reflect real life everyday scenarios, so no animal or fictional characters were allowed.

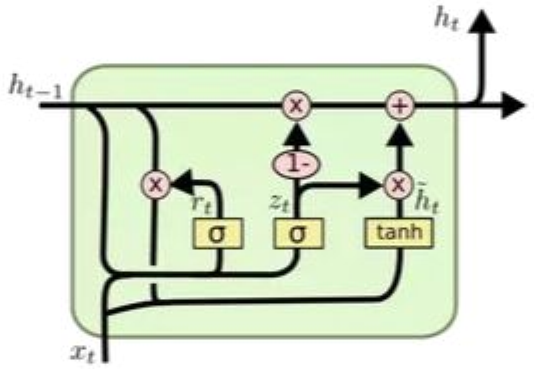


| | |
|---|---|
| ✕ | The little puppy thought he was a great basketball player. He challenged the kitten to a friendly game. The kitten agreed. Kitten started to practice really hard. Eventually the kitten beat the puppy by 40 points. |
| ✓ | Bill thought he was a great basketball player. He challenged Sam to a friendly game. Sam agreed. Sam started to practice really hard. Eventually Sam beat Bill by 40 points. |
| ✕ | I am happy with my life. I have been kind. I have been successful. I work out. Why not be happy when you can? |
| ✕ | The city is full of people and offers a lot of things to do. One of my favorite things is going to the outdoor concerts. I also like visiting the different restaurants and museums. There is always something exciting to do in the city. |
| ✓ | The Smith family went to the family beach house every summer. They loved the beach house a lot. Unfortunately there was a bad hurricane once. Their beach house was washed away. Now they lament the loss of their beach house every summer. |
| ✕ | Miley was in middle school. ~~She lived in an apartment~~. Once Miley made a mistake and cheated in one of her exams. She tried to hide the truth from her parents. After her parents found out, they grounded her for a month. |
| ✓ | Miley was in middle school. She usually got good grades in school . Once Miley made a mistake and cheated in one of her exams. She tried to hide the truth from her parents. After her parents found out, they grounded her for a month. |

*Table from Mostafazedeh et al*

## Gated Recurrent Unit (GRU):

For our first experiment we trained an RNN using the pytorch implementation of the GRU unit to make a distinction between coherent and incoherent stories. This would not be achievable with the original train corpus, as it consists of complete coherent stories, so a model couldn't receive any negative examples during training. Our first attempt at fixing this data problem was to randomly shuffle half of the story endings in the training corpus. We hypothesized that this could be sufficient in generating coherent and incoherent stories to train with. With the datasets ready, we had to prepare it for the GRU. We preprocessed the text from the training corpus with the NLTK tokenizer to construct a vocabulary for the GRU. The vocab for the model using the training corpus consisted of 28k words before pruning, and after pruning words with a frequency of less than two occurences. This reduced the training vocab to 19k words. With the vocabulary ready we used the pytorch data loader module to make the train, test and validation set using each set's respective instances and labels, and the training vocab. The data was read to be handed off to the GRU at this point. Our model was simple and consisted of three layers, the first being the embedding layer that converts the indices of tokens from the vocab into their dense vector representation. The embeddings were then handed off to the RNN layer using the GRU architecture:



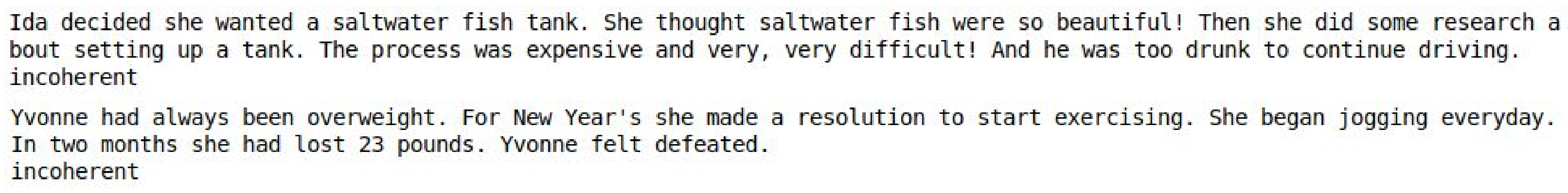
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

*GRU architecture and equations*



```
Ida decided she wanted a saltwater fish tank. She thought saltwater fish were so beautiful! Then she did some research a
bout setting up a tank. The process was expensive and very, very difficult! And he was too drunk to continue driving.
incoherent
Yvonne had always been overweight. For New Year's she made a resolution to start exercising. She began jogging everyday.
In two months she had lost 23 pounds. Yvonne felt defeated.
incoherent
```

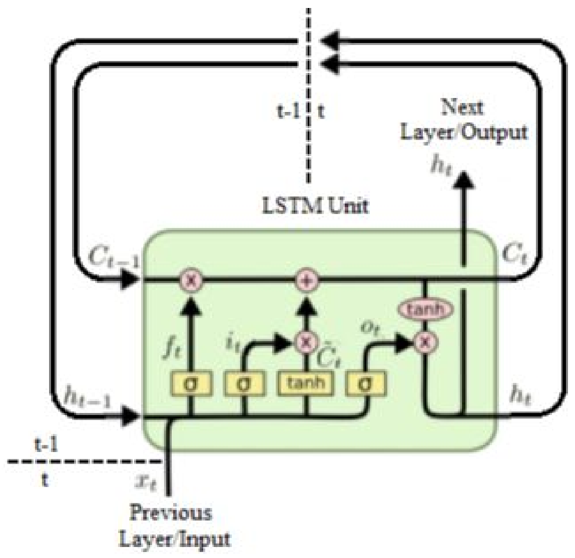
*Comparison between train stories and validation stories*

With the normal RNN architecture, input $x_t$ is multiplied with the previous output $h_{t-1}$ and pass through the *tanh* function to produce the new output $h_t$. The GRU architecture differs from the normal RNN architecture by adding an update gate which decides whether or not to pass $h_{t-1}$ to the next cell or not. This gives the GRU the ability to discard information that may not be important, and the hope was it could produce better distinctions between our stories. After going through the GRU layer, the final hidden state is max pooled within a linear layer that generates a probability distribution over the class types. As stated before, we ran the GRU with the original training set with shuffled endings. The model trained on cross entropy loss, with an embedding dimension of 64, a hidden size of 128 and trained for 10 epochs. During training, loss on the training set went down, as the validation loss started to increase.

We hypothesized that we could get better performance with our GRU model if we started using the validation set as a training set. The motivation for doing this was that the stories in the train corpus with shuffled endings produced stories that were incoherent, but even more so than those from the validation or testing set. Consider the two examples above. The first story is an incoherent sample from the training set with a random ending, and the second example came from the validation set. Notice the first story talks about a woman making a fish tank, and then the last sentence says , “And he was too drunk to drive home”. Which indeed is incoherent, but also completely nonsensical given the context. Compare this with the negative sample from the validation set. The ending of the story is still consistent in terms of its shared setting with the context. So we believed a possible explanation for the initial poor performance was that the GRU was learning the wrong kind of distinctions, as in it was separating complete nonsense from coherent stories, which isn’t the case in the validation or testing set. The negative samples in those two sets are incoherent because there is some logical contradiction or subversion of expectation, but still sensical in some sense. So we hoped to get better performance for this task by adopting the validation set as our new train set. We also began conducting all our experiments on positive and negative story endings as well, in order to see if we could confirm the findings by Cai et al that there were biases in story endings. Training with these sets was done similarly to our original experiment. With the validation set, we had a vocab of 6k before pruning, and roughly 4k after pruning. And for story endings, an initial vocab of size 4k pruned down to 2k. For full stories on either training or testing set we used a max sequence length of 50, and 13 for story endings.

## GRU with pretrained GLOVE embedding matrix:

To continue with our experiments with the GRU, we initialized the embedding layer with the 300 dimensional pretrained GLOVE embeddings. After preprocessing the corpora, each token in the training vocab had its entry in the embedding matrix of the model initialized with its corresponding GLOVE vector. Not only would the model be working with more sophisticated word representations than we could learn from training on the limited corpus, but it would also have a larger dimensionality to work with. The hope was that the richer representations for tokens would give better performance with the model, although this only gave a slight performance boost, for both full stories as well as for story ends.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \;+\; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

*LSTM architecture and equations*

## Long Short Term Memory (LSTM):

Our next attempt at improving the model was to swap out the GRU layer of the model with an LSTM layer. LSTMs are similar in nature to GRUs, with a few differences in their architecture which add more control. The LSTM has the update gate like the GRU, as well as two additional gates, forget and output gates. LSTMS also take advantage of a memory unit, which theoretically should give LSTMs a better ability to model longer sequences of tokens compared to the GRU. A possible hypothesis would be that the performance gain with full stories would be larger than the performance gain on just story endings. LSTMs can also be made bidirectional. The difference between a standard unidirectional versus a bidirectional LSTM, is that the bidirectional parses input across both time dimensions. This means input is parsed from the beginning of the sequence to the end, like in typical RNN architectures, but it is also processed simultaneously from the end to the beginning of the sequence. This added feature of the LSTM can give models a better sense of context as during any point in processing the sequence, both past and future information is retained. We were interested in seeing if this different kind of architecture would have any significant improvement in classifying stories. On one hand, the contradiction in incoherent stories only appears at the very end of the sequence, so processing from beginning to end may be sufficient. But it also seems possible that these contradictions can be detected earlier during processing of incoherent sequences. We tried training our model as before, aiming to classify full stories, or just story endings. Training was done with both unidirectional and bidirectional LSTMs to see if there were any differences in performance for this task. The model architecture was otherwise the same as in the previous case, we still used the pretrained embedding matrix with the GOVE embeddings, training to optimize cross entropy loss, vocab size of 38k, hidden size of 128 and embedding dimension of 300. With the bidirectional model, the dimensions of the linear projection layer were changed to be double the original hidden size to accommodate the doubling of hidden states that comes from the bidirectional variant. Results will be discussed more in depth, but this experiment yielded similar performance improvements from the previous experiments. The most notable finding was that unidirectional and bidirectional had similar performance.

## Swapping to pretrained model, BERT:

Dissatisfied with our results from our past experiments, we had the hope that swapping to a powerful pretrained model would help give us better performance. BERT (Bidirectional Encoder Representations from Transformers) is a neural language model made open source by researchers of the Google AI Language team in 2018. During pre training, BERT completes two different tasks, the first being Masked Language Modelling. During this task BERT is given a sentence with two words replaced with special mask tokens. The model has to predict what the original words were in the sentence. The second pretraining task is the Next Sentence Prediction Task. For this phase Bert is given 2 sentences from the original corpus, and classifies them as being sequential or not. To use Bert with our datasets, we cloned Bert into our directory on the Dandeneau servers. After that we located and installed the base Bert model and put it in the same directory. We went with the base model since the larger model would take much longer to run with the Dandeneau GPUs. Before running Bert from the Dandeneau machines we had to construct the relevant TSV files Bert expects, one for training testing and evaluation, which was done in a few lines of code in Jupyter notebook. To make these files we took all possible stories from the original training and validation set, and merged them into one larger set. This augmented set was then split 80/10/10 into a training, test, and dev set to give Bert. In total there were roughly 6k training examples, and 750 for testing and validation. We ran Bert's run_classifier.py file, and gave it the directories to the datasets. We set the flags for do_train, do_eval to true, gave it the base vocab and config files, set the max sequence length to 60, a learning rate of $2E^{-5}$ and 3.0 epochs. After running, Bert trained for approximately an hour. After the model was trained we ran the run_classifier.py again, setting the do_predict flag to true, and giving it the checkpoints saved from the trained classifier. The output was put into a seperate file, and to evaluate the results we iterated through each example and compared the class with highest probability to the true class labels from the test data.

## Experiment Results:

The table below shows the highest reported accuracy from each experiment, both on training ends as well as for full stories. Notice that the performance gain for training with solely the story ends is not insignificant compared to training on full stories . Classifying story endings is consistently ~10% more accurate than it is for full stories. We believe the main reason for this is due to the presence of an apparent bias between the pre-written true and false story endings in the validation and testing sets. Also note that the highest gain in performance for classifying entire stories came from swapping the original train corpus with shuffled endings with the validation set for training. All of our models in the table except for the first GRU used the validation set for training. We also compare our results with baselines established by Mostafazedeh, Cai, and Washington State's model.We also provide human accuracy as a reference, as reported by

| Model Description | Full Story Test Accuracy | Story End Test Accuracy |
|---|---|---|
| msap (Washington State) | **75.20%** | NA |
| Constant Choose First (Mostafazedeh et al) | 51.30% | NA |
| DSSM (Mostafazedeh et al) | 58.50% | NA |
| HIER, ENCPLOTEND, ATT (Cai et al) | 74.50% | 72.50% |
| GRU Trained on shuffled training endings (ours) | 50.45% | NA |
| GRU Trained on validation set (ours) | 52.91% | 62.75% |
| GRU GLOVE embeddings (ours) | 53.61% | 63.49% |
| LSTM GLOVE embeddings (ours) | 53.92% | 64.32% |
| BiLSTM GLOVE embeddings (ours) | 53.44% | 63.68% |
| BERT Base model (ours) | 18% | NA |
| Human Accuracy | 100% | 78% |

Mostafazedeh for full stories, and Cai for story endings. We leave the previous solutions and

baselines at the top, starting with the state of the art, and we report accuracy on distinguishing story endings as well as complete stories. As you can see from the table, we struggled to come close to the state of the art, we even performed worse than constant choose first with our first attempt with the GRU. Another thing to highlight from our results is that when using the LSTM, the use of a unidirectional vs bidirectional LSTM makes very little difference for this task, when we previously hypothesized a larger gap in performance, with bidirectional being the better model. If anything, the opposite appears to be true based on the results. We also wish to point out the poor performance of Bert. Seeing as Bert only had roughly 6k training examples, it could not be expected for performance to be over the top, but the lower than 50% accuracy implies some kind of bug in our system that had gone overlooked. Due to this

| | |
|---|---|
| correct endings: | out, !, great, new, found |
| incorrect endings: | n't, did, not, never, hated |

uncertainty we did not complete training with Bert on story endings.
*(Cai et al)*

## Discussion of results and possible directions for future work:

In hindsight, there are lots of possible explanations for the overall poor results from our experiments. The biggest one being that the approach we had to this problem was inappropriate given the task. Language modeling seems like an attractive solution to this problem, and would be a logical followup experiment to conduct. We also ended up discarding a large amount of data which also may have been unwise. We also could do more experiments with the LSTM before moving onto Bert, such as adding attention mechanisms to the model. We were most successful in confirming some of the findings by Cai et al, that there seems to be some bias present in the story endings themselves. They provided the table shown on the left, that illustrates that correct endings typically use more positive words, where false endings tend to use negation in their wordings. They showed it to be possible to increase baseline performance simply by choosing a candidate ending that uses more negations. Overall, the story cloze task has proven to be more difficult and nuanced than initially thought. We tried reducing the task at hand to a simpler one that can be solved relatively easily through several methods, although we did end up running into troubles. Our big takeaway is that the appropriate approach is crucial to tacking NLP problems, as well as having the appropriate data for it.

## References:

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, James Allen *A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories*

Zheng Cai, Lifu Tu, Kevin Gimpel, *Pay Attention to the Ending: Strong Neural Baselines for the ROC Story Cloze Task*

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, James F. Allen, *LSDSem 2017 Shared Task: The Story Cloze Test*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*