A discriminative and generative approach to the Story Cloze Task

Jérémy Scheurer **David Yenicelik** **Laurin Paech** **Jonathan Unger**

Abstract

In this report we present two approaches to the Story Cloze Task[1]. The first is a discriminative and the second a generative method. The argument for the first approach is, that it is arguably easier to model a separating hyperplane between two distributions, rather than modeling the distribution of a certain model itself. However, if one is able to model the distribution of stories, it might be easier to detect outlier sentences. Also we followed this second approach both for the fun and creativity part of the project, as well as because we wanted to learn more about outlier detection models for text. The best accuracy rate we got it 68.8%.

1 Definitions and Keywords

This section is intended as clarification on details for the reader.

- Training data: With the training data, we refer to the stories given in the train-stories.csv file.
- Training on the Story Cloze dataset: By this, we mean that we do a train-test split on the Story Cloze validation dataset. Usually, the proportions are at 75% to 25% percent.

2 Discriminative Approach

2.1 Introduction

We want to emphasize that we only use the validation data for training with the discriminative approach.
After conducting some research on existing solutions, we decided to go with a solution that seemed to be easy to implement, and allowed to easily improve on the initial model. In the methodology, we will shortly draw conclusion on the contents of the paper by [2]. We also offer a novel extension to that model which allows for similar accuracy models as the model proposed in the above paper, except that our model needs about 300x less parameters.

2.2 Methodology

Initially we lost a lot of time by trying to use the training data, and using the Story Cloze dataset as validation data only. We tried generating artificial fake endings, by taking the training set, and swapping the last sentence with any other sentence. There are multiple reasons why this could not work on a theoretic level, but we decided to go test this approach anyways. We then trained a model based on [2] and described in the Model section on this data.

Later on, we trained the same model on the Story Cloze validation Set only and were able to consistently achieve better-than-random accuracy (see Appendix A). From that point on, we decided to only use the validation set for the discriminative approach, as we had lost a few days trying to incorporate the training data from train-stories.csv .

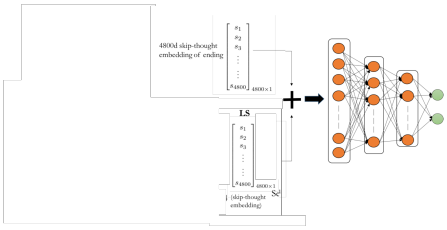

Figure 1: **Base model architecture** The base model of our architecture. Instead of using the skip-thought embeddings, we learn our own embeddings by taking the output of a bidirectional lstm. The difference between our model, and the model proposed in their paper is that we have smaller hidden weight sizes, but two layers of lstm's before we retrieve the output vectors.

2.3 Model

The discriminative architecture with which we started off is based on [2], especially we used only GLOVE embeddings and not Skip-Thought embeddings with which they were able to get nearly a 10 percent increase in accuracy. Here is how it works:

1. We feed the last context sentence through a bidirectional LSTM, whose last output(concatenate backward and forward output vectors) we refer to as *context vector*.
2. We also feed one of the two optional endings through the same bidirectional LSTM, whose last output(again concatenate backward and forward output vectors) we refer to as the *ending vector*.
3. We now have the context vector, and the ending vector. We now concatenate these two vectors, and pass this vector through three fully connected layers.
4. The output of the Neural Network is a **0** or **1**, where 1 denotes the real ending, and 0 denotes the fake ending.

Using our methodology, we found out that the model proposed by [2] is likely not able to capture the entire information of the data. Thus, we modified the network. We decreased the hidden sizes of the LSTM's, because we have a relatively low number of samples. Secondly, to make the model more expressive, we add a second biLSTM layer.

2.4 Training

When using the artificial fake endings, we did not consistently get over a random baseline. In theory, the artificial and real endings are so different (if viewed as distributions), that training on the training data, and validating on the validation data leads to the model being adapted to an entirely different distributions.

We use tf.train.GradientDescentOptimizer with a learning rate of 0.01 [2], and train on 17 epochs on a cross-entropy loss.

2.5 Experiments

Our best accuracy is 68.8% on one quarter of the validation set (we used the rest of the validation set for training).

The idea was that deeper models can capture expressiveness much better. The following shows that parameters for our novel model, which is able to get similar accuracy as their [2] which get 69.7 percent, but with 300x less parameters and after only 20 minutes of training.

2.6 Conclusion

The accuracy rate that we have is 68.8%. We achieved a similar accuracy rate to [2] results with GLOVE embeddings, with the difference that we need 300x less parameters and only 20 minutes of

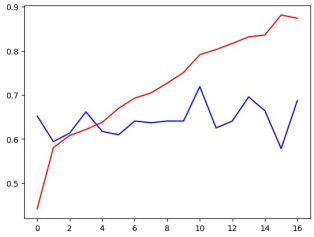

Figure 2: **Accuracy of our model** The red line shows the training accuracy, and the blue line shows the validation accuracy. We had to carefully choose the number of epochs (x-axis) to account for easy overfitting due to the small training data.

training. This, however, is a strong indication that the network found a "hacky" way to assess if an ending is real or not, as it is arguable difficult to learn meaning within 20 minutes in such a restricted dataset (remember we only look at the last 2 sentences).

Given our graphs, we also argue that the main bottleneck of this problem is the restricted amount of data. Our models improved until they started over-fitting (after which the validation accuracy decreased again). Because data was the main bottleneck, and we failed to appropriately account for the training data in the discriminative approach, we also worked on a generative approach in parallel.

3 Generative Approach

3.1 Introduction

Our goal with this generative approach is to attack the Story Cloze Task with an outlier detection model. We introduce a LSTM-Autoencoder which is applied on the train data to learn the structure of 5-sentence stories. We then proceed with two different procedures:

1. We apply the trained Autoencoder to the evaluation stories, once with the correct ending and once with the incorrect ending. We consider the sample with the smaller reconstruction error to be the story with the correct ending.
2. We use the Autoencoder to extract a latent representation of stories. We then use these representations as features to a One-Class SVM Model.

Although we did get acceptable results with a discriminative approach, we think that only training on the validation data, does not make good use of all the data available. Also by only looking at the last two sentences, we ignore possibly crucial information about a story.

3.2 Methodology

The ideas for this approach were taken from [2], but were nonetheless never applied to the Story Cloze Task. This makes our approach much more experimental.

3.3 Model

Our model consists of an LSTM-Autoencoder, where in one approach we look only at the reconstruction error, and in another we use the latent space as features to a One-Class SVM. The idea is that the Autoencoder learns the distribution of a "correct" story and when given a "false" ending, not fitting the story plot, it will have more trouble reconstructing it.

The raw input is given to an embedding layer, which tries to learn an embedding vector of our words. The Encoder takes the calculated embedding vector and a zero state. Then it passes its final state to the Decoder. The Decoder takes the Encoders final state and a "beginning of sentence" tag as input. Each output was feed into another Neural Network layer which projected the output, from the hidden size back to the vocabulary size. We then applied a softmax function to this projection and took the largest value of that. This was then feed into the next cell with the previous state as input.

Accuracy on Validation Set	5 Sentences	Last 2 Sentences
Reconstruction loss	0.5	0.5
OC-SVM	0.47	0.47

Figure 3: The accuracy of our two generative approaches

We then took two approaches to use this basic architecture to predict story endings.

1. We simply took the 4-sentence stories of the evaluation set and appended the optional endings 1 and 2. We then fed both options into our Autoencoder and looked at the reconstruction error. We consider the option with the smaller reconstruction error to be the "correct" story.
2. With the second approach, we want to make use of the latent representation that is created by an Autoencoder to decide if a story is an outlier(i.e. has a "false" ending) or not Specifically we again took the 4-sentence stories of the evaluation set and appended it with the optional endings. We then fed this into our architecture and then take the final output of our Encoder. We then feed this into a One-Class SVM model. This model works in an unsupervised manner, i.e. during training it assumes you give it "correct" samples and then tries to learn its distribution. When it is given test data, it then calculates the distance of the samples to the hyperplane and decides if it is an outlier.

3.4 Training

For training we used a cross entropy loss function. Also, we trained for 20 Epochs, we used a hidden size of 512, an embedding size of 100 and a vocabulary size of 20. When preprocessing we allowed a maximal sentence length of 18, which was enough to keep all sentences without trimming them. This means that our training input was a story of length 90. We realize that this is a very long input for an LSTM, which might have trouble to unroll for 90 steps. Thus we also tried another approach, where we only used the last two sentences of a story. This then yields a story length of 36 words, which should be much more doable by an LSTM. Also note that for our One-Class SVM model we used an RBF Kernel.

3.5 Experiments

Unfortunately our results turned out to be very bad. As one can see in the Figure 3, all of our models learned nothing at all and thus the predictions are just random guesses. There might be several reasons for this poor performance. It might be quite possible that an Autoencoder which tries to reconstruct 2 or even 5 sentences, just has too much variance in it. That is, the influence of the words in a sentence on the reconstruction loss is much larger than the fact that a sentence ending might semantically be correct or incorrect. This then means that the reconstruction loss, or the distribution learned has nothing to do with the correctness of the ending. Also it could be quite possible that even 2 sentences are too long to learn any distribution about it.

3.6 Conclusion and Further Work

Even though our results show very poor performance, we still believe that generative models are the right way to go. At least one should consider all the data available and not just ignore the training dataset. Although the poor performance of our approach does not set us optimistic about further work with this particular methods, one might implement the One-Class Neural Network, proposed by [1] and maybe get better results than with the more simplistic methods we chose.

References

[1] R. CHALAPATHY, A. MENON, S. C. Anomaly detection using one-class neural networks. *arxiv* (2018).

[2] SIDDARTH SRINIVASAN, RICHA ARORA, M. R. A simple and effective approach to the story cloze test. *arxiv* (2018).