Tackling Story Cloze Test with Wrong Ending Generation

Pascal Wiesmann Stefano Woerner Sandar Felicity Lim

1 Introduction

The training set of the story cloze test comes with only positive sample (i.e. stories with correct endings). On the other hand, the test set requires a classification as correct/wrong endings. This is one of the main difficulties with the story cloze test. We decided to generate wrong endings and to train a RNN-based classifier, using the correct endings from the training set and our own generated wrong endings. We explore several methods for the generation of wrong endings, which we will explain in section 2.

2 Methodology

2.1 Model: Classifier

Figure 1 shows the architecture of our model. This is similar to the approach by Burgert et al. except for a few differences [1]. Our model involves two Long Short-Term Memory (LSTM) units to compute the feature representation of each sentence [2], instead of a Bidirectional Recurrent Neural Network (BiLSTM). We also use the last output of the LSTM instead of the last cell state.

Firstly, we looked up the word embedding of each word in vector space for each sentence. The embeddings from the first four sentences were then passed on to the first LSTM which summarises the story in numbers. Likewise, the last sentence was passed on to the second LSTM. The results of two LSTMs are then concatenated and passed on to a dense layer. The output layer has two nodes, one for each verdict ("correct" or "wrong" ending).

For the classification on the test and validation sets, the model then picks the one with the higher score for being the "correct" ending.

2.2 Wrong ending generation

2.2.1 Shuffling

According past publications[1], permuting the correct endings of the training stories produce reasonable wrong endings. In our implementation we make sure to use a derangement, i.e. a permutation where no element stays in place. Otherwise we correct endings could appear as negative samples.

2.2.2 Simple antonyms and negation

Shuffeling obviously changes the main character and main point of the story. Therefore, we find it sensible to generate endings that are wrong but still preserve the context. Negation of the fifth sentence seemed a promising way to achieve that end. i.e. The resulting sentence would be contain at least one of the charecters of the story. The resulting sentence would still be generally realisitic, except that it would not create a meaningful story as an continuation of the previous four sentences.

As a first approach we use a few handselected rules. The rules consist of pairs like good/bad and did/didn't. For each pair we replace occurences of the first element with the second element and vice versa. One caveat of this approach is the following scenario: if one elements of the pair occurs much

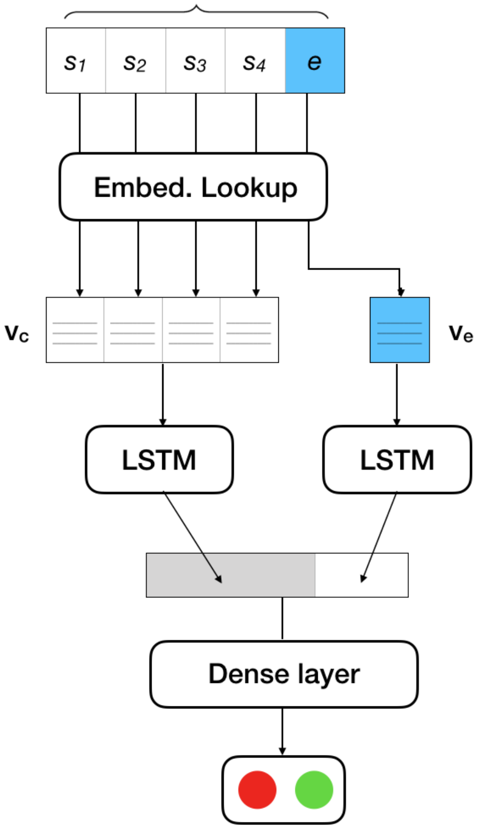

Figure 1: Our classifier

more often then the other, the rare word would appear unreasonably often the wrong endings. For this reason we make sure that the ratio between the two elements of the pairs is the same in the correct and in the wrong endings.

2.2.3 Wordnet Antonyms

Negation can be challenging for complex sentence structures because we would have to decide which part of the sentence to negate. For example, for the sentence: " Kelly quickly went home." There are at least two possibilities of negation: "Kelly slowly went home" or "Kelly did not go home quickly". If we do not keep track the number of negations in the sentence, the result would be wrong: "Kelly did not slowly go home", which is not the negation of the original sentence. Hence, our approach, as shown in Figure 2, was to first negate the adjectives and adverbs, and in their absence, negate the verbs.

A few examples of our successful sentence generations are shown in Table 2.2.3

2.2.4 Limitations

There are two main limitations of our model to generate wrong endings. First, is that our antonym replacement may not be sensible at times [3]. Table 2.2.4 shows a few examples of sentences are grammatically wrong or not sensible.

Second is that our generated wrong endings may have high occurance of the word "not" as our model was designed to insert "not" in the absence of antonyms. This would affect our classifier performance negatively.

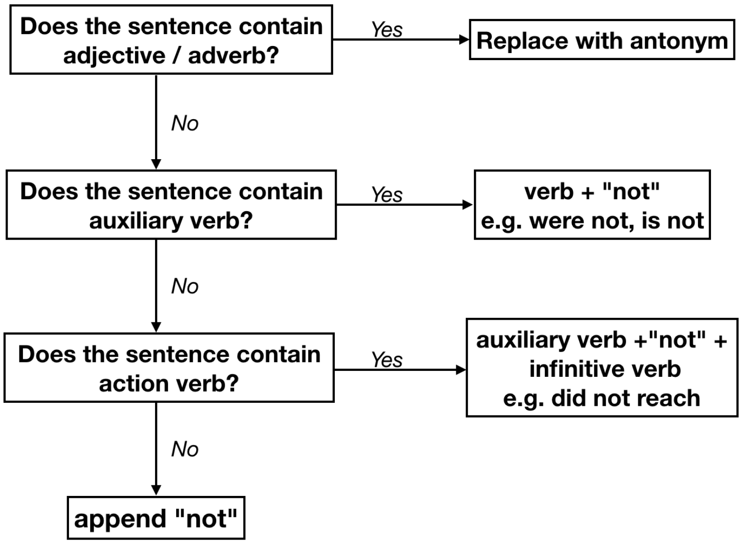
Does the sentence contain adjective / adverb?
Yes
Replace with antonym
No
Does the sentence contain auxiliary verb?
Yes
verb + "not" e.g. were not, is not
No
Does the sentence contain action verb?
Yes
auxiliary verb +"not" + infinitive verb e.g. did not reach
No
append "not"

Figure 2: Our approach to generate wrong endings

Context	Right Ending	Wrong Ending
The family was tired of not hearing when someone knocked on their door. They installed a doorbell on their front door. It would ring a pleasant tune when someone came to see them. They never missed houseguests anymore.	The family wished they'd gotten the doorbell earlier!	The family wished they'd gotten the doorbell late!
Tim was hiking up a mountain with friends. He wasn't paying attention and lost his footing. Tim tumbled down down for a few yards. He was severely hurt.	Tim's friends had to call for help and carry him.	Tim 's friends did not have to call for help and carry him .
Laura found a jar of candy in her mom's kitchen. Laura ate all of the candy. She got really sick. Laura's mom discovered why Laura was sick.	Her mom felt bad for her so she didn't punish her for eating it all.	Her mom felt good for her so she didn't punish her for eating it all.

Table 1: Generated wrong endings that works

2.3 Generating wrong endings with a seq2seq model

We constructed a simple seq2seq model following [4]. The sequence to sequence model consists of two RNNs: an encoder, which reads the input sequence to infer semantics, and a decoder, which generates an output sequence respectively.

We trained the model on 80% of the validation set. The input sequence is the concatenation of the first four sentences of each story, while the ground truth output sequence is the wrong ending to the story.

In order to be able to train on such a small dataset, pre-training on the training set was performed. This gives the seq2seq model a better understanding of the underlying language.

Because this method uses a portion of the validation set, one could think of instead directly training the classifier on the validation set. While a classifier trained directly on the validation set would require an increase in labelled data to improve its performance, our approach can simply generate more wrong endings for potentially unlimited amounts of only semi-labelled data. This is a great advantage, as stories with their right ending are typically more readily available that stories with a corresponding wrong ending.

As can be seen in the results section the generation with the seq2seq model outperforms the other generation methods.

Context	Right Ending	Wrong Ending
Gina's brother Jay had been out of the house. Now he was back. He'd fought their father. Their dad was a foot taller and 50 pounds bigger.	Gina no longer felt safe with him around.	Gina yes longer felt safe with him around.
Matthew grew up with a dad that pushed him in sports.,He thought he would grow up to be an athlete. Once he grew up he realized he wanted to do other things. His dad was very angry.	Their relationship got bad and they no longer talked.	Their relationship got good and they no longer talked.

Table 2: Generated wrong endings that are not sensible. In the first story, the phrase "yes longer" does not exists in English. In the second story, the sentence is connected by "and" but negation was only applied once.

2.3.1 Limitations

The sequence to sequence model is currently very simple, restricting its learning capabilities. This means that there is ample room for improvement in this method.

3 Training

For the training, the model just learns to predict whether a ending is consistent or not. It doesn't directly learn to decide between two possible endings. We use the validation set in order to stop training before overfitting too much.

4 Experiments

For all experiments we use the same classifier, but different wrong endings. To evaluate the quality of our classifier, we do a experiment where we use 80% of the validation set to train our classifier, and the remaining 20% as validation set.

4.1 Results

Approach	Validation	Test
Shuffle	50.3%	50.7%
Simple Antonyms	54.4%	49.5%
Wordnet Antonyms	56.4	55.6
Seq2Seq	63.1	59.7
Wordnet & Seq2Seq	59.0	58.2
All generated	56.9	56.0
Dev set & Seq2Seq	69.3	66.8
Dev set	73.3	68.0

Table 3: System accuracies on the Story Cloze datasets.

4.2 Final Model for Predictions

As a final model we use the classifier trained on the seq2seq generated wrong endings and 80% of the validation set. With this model we produced our submission for the nlu18-testset.

5 Conclusion

We showed that sophisticated generation of wrong endings can help train a binary classifier in a meaningful way. Additionaly we propose a seq2seq model which can learn to generate wrong from a very limited fully labeled dataset. The potential of the seq2seq model has been shown, but a more complex model could probably reach a much higher accuracy.

References

[1] 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. *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*, 2017.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[3] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[4] Tensorflow. Neural machine translation (seq2seq) tutorial.