Story Cloze Test

Ali Hosseini **Mounir Amrani** **Srikanth Gurram**

Abstract

The Story Cloze Test (Mostafazadeh et al. [1]) is a framework that was developed for evaluating story understanding and script learning. In this task, given the first 4 sentences of a story, we are tasked to choose the 5th ending sentence between two possible endings to the story. We describe in this paper different Neural Models that we used to solve this task: a BERT Model, a Logistic Regression Model and a Feed-Forward Neural Network (FFNN) Model. Our best performing model was the BERT model with an accuracy of 86.8% on the test set

1 Introduction

Story narration is an important part of the human language and culture. It allows sharing informations, experiences and sentiments. Designing a system for story comprehension and commonsense understanding is an interesting and challenging task. As an evaluation framework for this task, [1] introduces the Story Cloze Task in which given a four-sentence story context, the task is to pick the 'right' commonsense ending from two options. While humans achieve an accuracy of 100% on this task, [1] shows that many systems still struggle with this task and perform close to a random guess.

The data provided for this task is divided to training, validation and test data. While the validation and test sets are labeled data sets containing stories with both the correct and wrong endings as well as a label specifying the correct ending, the training set only contains stories with the correct endings. This poses a challenge as to how to augment the training set in order to generate wrong endings that are semantically close to the story context. One approach that showed good results to generate wrong endings is to randomly sample an ending from other stories in the training set. Some approaches [2],[3] showed that good results can be achieved by completely ignoring the training set and only using the validation set for training.

Another challenging part about story cloze task is that stories have a narrative sequence that maintains coherence of events, consistency of topics and commonsense. For a model to perform well on story cloze task, it must learn these aspects of stories (explicitly or implicitly). In this paper, we explore 3 different models that try to tackle the story cloze task using different approaches.

The first approach goes along the lines of the new pre-trained language models (for example [6]) which train a general-purpose "language understanding" model on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks.

The second approach involves learning the different aspects of stories explicitly. This is fairly inspired by [5] who extract features that pertain to three key aspects of stories: event sequence, sentiment trajectory and topical consistency. The best accuracy reported by [5] was 77.6%. In our second approach, we use best two of the three aspects used by [5] namely: event sequence and sentiment trajectory. We explore the use of a language model to learn sequence of nouns and verbs as a proxy to learn event sequence. We observe that the this approach is only detrimental to the performance.

The third approach closely follows the best performing model in [3]. This approach relies on the publicly available skip-thoughts model [1] by [4] for sentence embedding, followed by a FFNN to

[1]https://github.com/ryankiros/skip-thoughts

predict for each 5-sentence story 2 probabilities: the probability of having a wrong ending and the probability of having a right ending.

2 Model

2.1 BERT Model

BERT [6], or Bidirectional Encoder Representations from Transformers, is a method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. In short, BERT provides contextual word embeddings (not really, see below) for sentences or pairs of sentences.

BERT operates on wordpieces and was pretrained on a huge corpus with two target tasks: masked language model on word pieces and next sentence classification.

The next sentence classification is closely related to our task and allows to learn relationships between sentences. It is trained on a simple task which can be generated from any monolingual corpus: Given two sentences A and B, is B the actual next sentence that comes after A, or just a random sentence from the corpus?

- Sentence A: the man went to the store .
- Sentence B: penguins are flightless .
- Label: NotNextSentence

- Sentence A: the man went to the store .
- Sentence B: he bought a gallon of milk .
- Label: IsNextSentence

This latter task is very precious in our context because as we can see, true continuation classification is inherently part of the BERT model.

Finally, BERT gives a (contextual) representation for each word piece of the input, which can be a single sentence or a pair (A and B). The representation of the first word piece, the [CLS] token (a special word piece representing the start of the sentence) can be used in downstream tasks to perform classification.

2.2 Logistic Regression Model

The second approach involves extracting event sequence and sentiment trajectory features.

2.2.1 Sentiment Trajectory

In order to extract sentiment features, we divide the context of each story into 3 parts: beginning (1st sentence), body (2nd and 3rd sentences) and climax (4th sentence). The following are the features we extract:

$$\{P(S(option)|S(climax), S(body), S(beginning)),\ P(S(option)|S(climax), S(body)),$$
$$P(S(option)|S(climax))\ P(S(option)|S(context))\}$$

Where, $S(sentence) = sign(N(+ve\ words) - N(-ve\ words))$.

2.2.2 Event Sequence

Event sequence features try to track the series of events that take place in a story. [5] use framenet frames to track the evolution of events. Due to the lack of ways to parse sentences to extract evoked frames, we use a proxy method that uses nouns and verbs (event words) as a way to track the sequence of events. We train a language model to predict the sequence of event words. Let $< e_1, e_2, ...e_D >$ represent the event words in the context (1st 4 sentences of the story). Using this language model, we obtain the following features

$$\{P(event\_option_i)|e_D),\ P(event\_option_i|e_D e_{D-1}),\ \ldots\ P(event\_option_i|e_D e_{D-1}\ldots e_1)\}$$

Both these set of features for both the options are concatenated along with one comparative binary feature for every of the above features. Finally, using these features, a logistic regression model is trained to predict either option1 or option2.

2.3 Feed-Forward Neural Network with skip-thoughts embeddings

This model is trained on the labeled validation data only, referred to as development data set. The development data set is split into 90% training and 10% validation for computing the accuracy. As a preprocessing step, we embed all sentences in the development and test sets using the skip-thoughts model (pre-trained on the BookCorpus dataset) [4]: this embeds each sentence into a 4800 dimensional vector.

This model relies only the 4th sentence of the story context together with the story ending as input to the network. We experiments with 2 versions of our network: version v1 where the input is the 9600 dimensional concatenation of these 2 sentences embeddings, and version v2 where the input is the 4800 dimensional sum of these 2 sentences embeddings. Version v1 has 4 hidden dense layers with the following dimensions: 4800, 2400, 1200, 600. Version v2 has 3 hidden dense layers with the following dimensions: 2400, 1200, 600. Each hidden layer is ReLU activated and followed by a dropout layer with dropout rate 0.3. In both version, the output layer of the network has 2 units with softmax activation and are interpreted as: p1="the probability of the ending being wrong" and p2="the probability of the ending being right".

During training, we train the model to output the pair (p1, p2) = (1.0, 0.0) for inputs constructed with a wrong ending and the pair (p1, p2) = (0.0, 1.0) for inputs constructed with a right ending.

During evaluation and testing, we compute the choice between the 2 endings of each story by choosing the ending whose p2="probability of being a right ending" is the largest.

3 Training

3.1 BERT

We continue the pretraining of the language model for 50’000 steps using a batch size of 16 and a learning rate of 2e-5.

The finetuning procedure is trained using cross-entropy loss. We again use a batch size of 16 and a learning rate of 2e-5. We train for 20 epochs.

3.2 Logistic Regression

For training this model, we use cross-entropy loss that is minimized using standard stochastic gradient descent with learning rate 0.01.

3.3 Feed-Forward Neural Network with skip-thoughts embeddings

For training our model, we use the cross-entropy loss that we try to minimize using Adam optimizer with learning rate 0.001.

4 Experiments

4.1 BERT

We leverage the pre-trained BERT model in the following way: first, starting from a pre-trained BERT model[2], we continue pre-training the language model on the training set. Then, we finetune

[2]We use the ’BERT-Base, Uncased’ model which you can find here: https://github.com/google-research/bert#pre-trained-models

this pre-trained model on the validation set alone, which contains both correct and incorrect endings. For this latter task, we create two samples per story by creating a BERT input where the sentence A is the fourth sentence and sentence B is either the correct (labeled as 0) or incorrect (labeled as 1) ending. We finally do a binary classification task by plugging a simple feed-forward layer on top of the BERT representations. Note that the context we use is only the forth sentence, the beginning of the story is completely ignored. This is however enough to obtain good results on the test set. In the end, we obtain an excellent accuracy of 86.8% on the test set.

4.2 Logistic Regression

First, we consider the sole effect of using sentiment trajectory features. To exract the sentiment trajectory features, a conditional probability model is trained on the train data. A logistic regression model is trained on the features extracted from valid data using SGD with a learning rate of 0.01 for 1000 epochs. It was observed that sentiment features alone achieve a validation accuracy of 58.4% and test accuracy of 57.13%.

Including the event sequence features dropped the testing accuracy to 48% which is worse than random assignment. We suspect that the model performs bad because the features aren't clean due to the way the event sequence model is trained. Instead of using frame representations as done by [5], we train a language model directly on the sequence of nouns and verbs. Since most nouns are non-repeating, they're encountered very rarely by the language model during training and hence, the representations learned are not accurate. This we suspect is a reason for the ill performance of the final classification model. It is observed that the prediction is always option 2 when both the event and sentiment features are used hence, the bad performance.

4.3 Feed-Forward Neural Network with skip-thoughts embeddings

As described above, we experimented with 2 versions of our network: version v1 with concatenated embeddings and version v2 with summed embeddings. We perform the following for each version:

Due to the small size of the training set, we were able to train the model for several epochs. We trained the model for 50 epochs (although consistent improvement in the validation and test accuracies was only observed during the first 20 epochs approximately). After each epoch, we computed the train, validation and test accuracy and if the computed validation accuracy is higher than that of the previous epochs, we saved the new model weights. We refer to the last saved model weights as the best model.

While version v2 was faster to train and eventually achieved better validation accuracies, version v1 had better and more stable test accuracies after 20 epochs whereas the test accuracies for version v2 started to drop.

In the following, we report the train, validation and test accuracies for both versions on the best model:

Version	Train accuracy (%)	Validation accuracy (%)	Test accuracy (%)
v1	99.88	79.25	76.91
v2	97.44	80.85	75.78

5 Conclusion

- The BERT Model by a huge margin performs better than all other models explored in this paper. The best test accuracy achieved was 86.8%.
- The logistic regression model suffers from noisy features and would benefit from better event sequence features such as framenet based approach discussed in [5].
- The feedforward neural network, although simple, was able to achieve good validation and test accuracies. This is mainly due to the skipthoughts embeddings that encode well the semantics of the sentences.

References

[1] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, J. Allen (2016): A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories.

[2] R. Schwartz, M. Sap, I. Konstas, L. Zilles, Y. Choi, N. A. Smith (2017): Story Cloze Task: UW NLP System.

[3] S. Srinivasan, R. Arora, M. Riedl (2017): A Simple and Effective Approach to the Story Cloze Test.

[4] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, S. Fidler (2015): Skip-Thought Vectors

[5] S. Chaturvedi, H. Peng, D. Roth (2017): Story Comprehension for Predicting What Happens Next

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2019): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding