Story Cloze Test - NLU Project 2

Gabriel Hayat Hidde Lycklama à Nijeholt Yiji He Arthur Deschamps

1 Introduction

Recent advances in machine learning have vastly improved the performance of models for Natural Language Understanding [11][12][13]. Many areas such as sentiment analysis, language modeling or machine translation have already achieved impressive results. Unfortunately, the domain of machine comprehension and logical induction is a research area where significant improvements are yet to be made. The Story Cloze Test (SCT), a task to choose the correct ending of a four-sentence story, is one of the evaluation frameworks in this domain. In this project, we constructed a model to tackle this test. Unlike most common machine learning tasks, the training and validation set have a fundamental difference. The training data is composed of 80k five-sentence stories released in 2016 by [1], whereas the validation set is a one of four-sentence stories (context) given with two possible endings. [1] demonstrates that relying exclusively on shallow semantic features is not enough in order to achieve a high accuracy, with the best classification accuracy of 58.6% obtained by a Deep Structured Semantic Model (DSSM).

Narrative comprehension in the SCT problem is a challenging task due to the requirement for contextual understanding of stories. However, the biggest challenge in this task is the lack of negative endings in the training set. Training solely on the validation set has demonstrated to achieve higher scores than training on the training set, despite the important difference in the volume of data [2]. The issue has previously been addressed by generation of negative endings (i.e GAN [3]) or sampling negative endings from the training set following various heuristics [2].

We follow a discriminative approach similar, although not identical, to the one described by Roemmele et al. [2]. In contrast to [2], we use a different kind of word embeddings and sentence embeddings. Moreover, we highlight the performance differences when training on the training set versus training solely on the validation set.

2 Methodology

The input data contains around 88 thousand stories consisting of four context sentences and one ending. Furthermore, a validation dataset was provided that contains around two thousand stories with a positive and a negative ending, as well as a label value indicating which ending is correct. Because of the challenge of the training data being incomplete, the task was split up in two sub-problems, namely discrimination and generation. First, we present the story representation, after which we will continue with both our approaches to ending generation. Finally, in Chapter 3 we will describe our discriminator.

2.1 Story representation

It is beneficial to feed stories to the model in a way that may help the model understand the structure in the language better. After pre-processing the input data to sanitize punctuation, we explored two ways of representing stories in our model. Both approaches use a pre-trained representation of words, which encode stories as vectors of real numbers. This way, the meaning of the encoded words is based on an even larger corpus of training data, increasing the likelihood that the model may learn useful relations between words and sentences.

Word embeddings: Word embeddings encode each word in a sentence with a single vector. From these vectors, the mean is taken to create a sentence representation. We use two kinds of pre-trained word embeddings. In order to load the embeddings, the gensim word2vec library [16] was used.
Word2vec: Stories were encoded with word-level embeddings using two different word2vec embeddings. The first word embedding was 100-dimensional and trained on the corpus of project 1 of the Natural Language Understanding course[20] available at [14]. The second word2vec embeddings were 300-dimensional vectors and trained on the 100-billion word Google News dataset [15].
Glove: Pretrained glove embeddings of 300 dimensions trained (2.2M Vocab, 840B tokens)[4].

Sentence embeddings: Instead of encoding each word in a sentence separately, sentence embeddings return an embedding per sentence. This way, the pre-trained model can contain meanings of words relative to the sentence it is in. We use a pre-trained Skip-Thought model based on Kiros et al. [6]. Skip-Thought uses an RNN based architecture to learn a 4800-dimensional sentence representation. The pre-trained sentence encoder, provided by the authors of the original paper from the University of Toronto [18], was used.

2.2 Generative

We combined two different methods to generate incorrect endings.

Random: Sampling uniformly from the set of all fifth sentences of the training set's stories. The sample sentences will not have any semantic link to the context. Therefore, the negative endings might easily be discarded by the model.
Backward: Because the random sampling approach generates endings which are semantically far apart from the context, taking a uniform sample from the story's context sentences may be more promising. This way, the negative endings are semantically closer to the positive ending.

During training on the training set, negative endings were generated using both previously described methods. An ending sentence is sampled using one of the two methods according to a distribution specified in Chapter 4.

3 Model

RNN-based architectures to approach language models has been a popular choice in the past years.[21][13] As a starting point, an architecture similar to that described by [2] was used. A schematic overview is shown in Figure 1.
We use a binary classifier conditioned on the context sentences and two endings to classify which ending is more likely correct. After encoding the sentences of the context and both endings, each context sentence is fed as a time step into an RNN with an LSTM cell size of 1000. Then, the RNN, in its current state, is evaluated twice, once for each ending, independent of each other. Next, the LSTM's hidden 1000-dimensional state is fed into a single fully connected layer with 1 output node without any activation function. Both output nodes, one for each ending, are then activated with a softmax function.

3.1 Attention

Attention is a mechanism combined in the RNN allowing it to focus on certain parts of the input sequence when predicting a certain part of the output sequence, enabling easier learning and of higher quality. It has been proved effective in many NLU applications, such as machine perception [9] and classification [10]. We have thus decided to implement two types of attention to boost the performance of our model, namely **Bahdanau Attention**[1] [7] and **Luong attention**[2] [8].

3.2 Features

In order to further help our model develop semantic links, we computed three features that are dynamically computed between the context of the story and its two possible endings. This approach

[1]`tf.contrib.seq2seq.BahdanauAttention`
[2]`tf.contrib.seq2seq.LuongAttention`

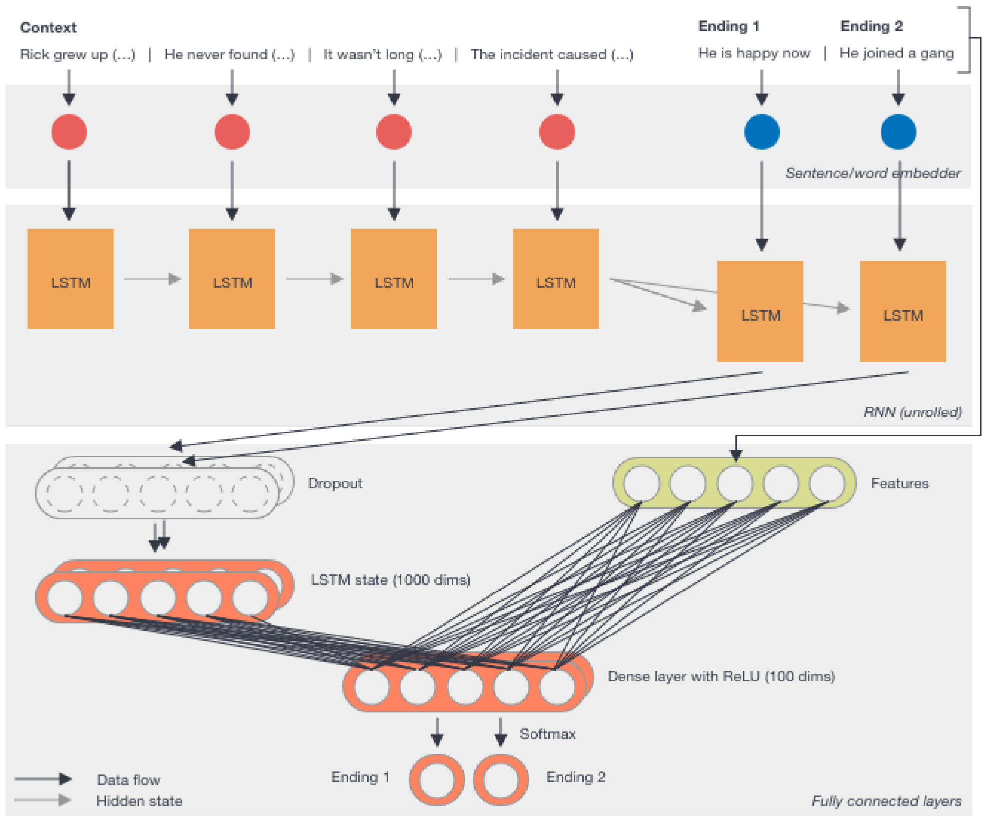

Figure 1: Model architecture.

was motivated by the results achieved in [5].

NgramOverlap: The number of overlapping ngrams between an ending and a story context for $n \in \{1, 2, 3\}$. This idea was directly inspired by [5]. To compute the ngrams themselves, we the `nltk`library [19] was used.
PronounContrast: The presence of pronoun mismatches between a story context and an ending. This helps to detect cases where the wrong ending is written in first person although the story context is written in third. This idea also came from [5] and was also integrated using `nltk` [19].
Sentiment: Extract positive, neutral, negative and compound sentiments from the last two sentences from the context and the ending, by using lexicon and rule-based sentiment analysis tool–VADER. [17]

The features are used in the model in the fully connected layer. An additional 100-dimensional layer is used to combine the RNN output with the features, as shown in Figure 1.

4 Training

We have trained the different variation of the model on the training set and validation set. The dimensions of the word embeddings depend on the type being used (100 or 300 dimensions). The attention and the RNN cell dimensions are both set to 1000. The batch size is set to 128 when training with any type of word embeddings and 16 when training with sentence embeddings. The Adam optimizer proved to achieve a better performance than the RMS optimizer. Thus, all training has been done using Adam. The learning rate is set to 0.001 and the gradients are clipped to a value of 10. We use a dropout layer with dropout rate of 0.3 after the RNN cell to minimize overfitting. After investigation, the optimal ratio of random/backward ending generation was found

Model instance	Training Validation accuracy	Training Test accuracy	Validation Test accuracy
RNN GRU [2] Rand-3 + Back-1 + Near-1 + LM-1 (orig.)	0.656	0.672	–
RNN Vanilla	0.481	0.500	–
RNN LSTM with features + w2v embeddings (100d)	0.518	0.506	0.671
RNN LSTM with features + w2v Google embeddings (300d)	0.534	0.516	0.657
RNN LSTM with features + Glove embeddings (300d)	0.523	0.507	0.663
RNN LSTM + skipthoughts embeddings	0.507	0.473	–
RNN LSTM + skipthoughts embeddings + Bahdanau [7] Attention	0.523	0.480	–
RNN LSTM + skipthoughts embeddings + Luong [8] Attention	0.510	0.485	–

Table 1: Accuracy scores achieved by different models

to be 2/1, thus this ratio has been used throughout all experiments. Finally, we use Tensorflow's `sparse_softmax_cross_entropy_with_logits` as loss function for all experiments.

To evaluate the quality of the learning of our model, an evaluation step was performed every 100 steps. This step takes a batch of stories and endings from the validation set and computes the accuracy.

5 Experiments

The model was trained on both the provided training set and validation set. When training on the training set, the training data was augmented with generated negative endings, as explained in Section 2.2. When training on the validation set, the data was split to keep 20% for validation purposes. After training, the model was evaluated on the project's Story Cloze test set. Both training mode were tested with word embeddings, sentence embeddings, attention and features. In Table 1, the **average** accuracy when training different models on the two provided sets are shown.

The RNN vanilla model does not make use of features, attention nor sophisticated embeddings. As a result, we see that the reported accuracy on both sets are even worse than random.

Next, we analyzed the effect of different word embeddings on the model. They were trained for about 2 epochs on the training set and 10 epochs on the validation set. We see that the best performance on the training set use Google's word2vec embeddings, which can be explained by the extent of the data they were trained on. As expected, the model using Glove embeddings does not perform much worse, with the 100-dimensional word2vec embeddings having the worst score. However, these embeddings surprisingly achieve the best performance when trained on the validation set. The difference in performance might be explained by the reduced complexity of the validation set compared to the training set, thus the quality of the embeddings having a smaller impact on the overall quality of the predictions.

In addition, we trained a model using Skip-Thought embeddings, with Bahdanau Attention or Luong attention. The results demonstrate that Bahdanau attention performs best, but still worse than the Google word2vec word embeddings model.

The results in Table 1 strongly emphasize the difference in performance when training the same model on the training set and validation set. Based on this finding, we can conclude that the generated negative endings were not close enough to the negative endings used in the Story Cloze test set. Therefore, the model was not able to learn from the training dataset in a way that could be extrapolated to the Story Cloze test set. An improvement to this approach would be to use near embeddings as well as KDE sampling, which tries to generate a negative ending based on the average cosine distance difference between a positive and negative ending in the test set. [5]

6 Conclusion

The Story Cloze Test is a challenging problem that requires machine comprehension and logical induction to achieve significant results. We tackled this problem by exploring several variations of the model and analyzed their performance. However, we did not manage to obtain a similar performance as Roemmele et al.[2]. This can be attributed to the poor quality of the generated negative endings.

References

[1] N. Mostafazaden, 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] Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, Andrew M. Gordon: An RNN-based Binary Classifier for the Story Cloze Test.

[3] Bingning Wang, Kang Liu, Jun Zhao: Conditional Generative Adversarial Networks for Commonsense Machine Comprehension

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

[5] Michael Bugert,, Yevgeniy Puzikov, Andreas Ruckl,Judith Eckle-Kohler, Teresa Martin, Eugenio Martinez-Camara, Daniil Sorokin, Maxime Peyrard†, Iryna Gurevych: LSDSem 2017: Exploring Data Generation Methods for the Story Cloze Test.

[6] Ryan Kiros and Yukun Zhu and Ruslan Salakhutdinov and Richard S. Zemel and Antonio Torralba and Raquel Urtasun and Sanja Fidler. 2015. Skip-Thought Vectors

[7] Bahdanau, Dzmitry and Chorowski, Jan and Serdyuk, Dmitriy and Brakel, Philémon and Bengio, Yoshua: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

[8] Luong, Minh-Thang and Pham, Hieu and Manning, Christopher D: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing

[9] Minh-Thang Luong Hieu Pham Christopher D. Manning: Effective Approaches to Attention-based Neural Machine Translation

[10] Zichao Yang, Diyi Yang1Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy: Hierarchical Attention Networks for Document Classification.

[11] Vu, Ngoc Thang and Gupta, Pankaj and Adel, Heike and Schütze, Hinrich: Bi-directional recurrent neural network with ranking loss for spoken language understanding

[12] Shi, Yangyang and Yao, Kaisheng and Chen, Hu and Pan, Yi-Cheng and Hwang, Mei-Yuh and Peng, Baolin:Contextual spoken language understanding using recurrent neural networks

[13] Liu, Bing and Lane, Ian: Recurrent neural network structured output prediction for spoken language understanding

[14] `https://polybox.ethz.ch/index.php/s/mFkjmC9EmPKDzg1`

[15] `https://code.google.com/archive/p/word2vec/`

[16] `https://radimrehurek.com/gensim/models/word2vec.html`

[17] `https://github.com/cjhutto/vaderSentiment`

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

[19] `http://www.nltk.org`

[20] Natural Language Understanding Project 1, Retrievable at: `http://www.da.inf.ethz.ch/teaching/2019/NLU/material/project1_description.pdf`

[21] Baolin Peng and Kaisheng Yao. 2015. Recurrent Neural Networks with External Memory for Language Understanding