NLU Project 2: *Predicting the End of a Story*

Anders Munch amunch@ethz.ch
Thomas Nilsson tnilsson@ethz.ch
Joel Oskarsson ojoel@ethz.ch
Minh Duc Tran tranmi@ethz.ch

1 Introduction

This report deals with the *StoryCloze Task*[1], in which the objective is to choose the correct ending for a short story, consisting of five sentences in total. Three datasets are given; a large training-set, a small validation-set and a small test-set. The training-set contains 88,161 stories with a single ending, namely the correct one. The validation- and the test-set both consist of 1871 stories, each of which have two candidate endings, where one is correct and the other one is wrong. Labels indicating which candidate ending is correct is also included for each story.

The goal of the task is to determine the correct story ending from two candidates, given the previous four sentences as context. The task is interesting, especially from an AI perspective, as human accuracy for the task is 100%, while many fairly advanced machine learning techniques do not perform better than approximately 70% [2] (though some dedicated state of the art methods currently achieve around 90% [3, 4]). The reason for this is that choosing the right ending depends on quite sophisticated commonsense reasoning, which necessitates a deep level of language understanding.

Apart from the difficulty of the task itself, a major challenge is imposed by the available data. The large training set only contains correct endings, meaning that it is not possible to directly approach the task with a classification model, due to missing negative samples. Previous research contributions have suggested solving the dataset issue by generating incorrect endings for the large training set by using generative models such as Generative Adversarial Networks or Language Models [2, 5, 6], or alternatively by using the ending of another random story as the incorrect ending. The far smaller validation set contains both right and wrong ending candidates, making it ideal for classification, however the danger of training on such a small dataset is that the models easily overfit the data. Results from the literature confirm that working exclusively with the single-ended training set can be challenging and most well performing approaches therefore utilize parts of the validation set as well [5, 7, 3].

With all these things in mind, our novel approach to the task is to implement a collection of models which all approach the StoryCloze Task from different angles, and then combine these models into one final model in order to draw from their different strengths.

2 Methodology

We expect that in many cases, considering a single aspect of a story might be enough to find the right ending. On the other hand, it is clear that we cannot expect that a single aspect will work for all sentences. This leads to the idea of a combiner model, which uses features from a collection of different sub-models. The combiner model should learn to weight the predictions from the sub-models based on the story being considered and ideally make each of the sub-models specialize on different topics.

This approach also allows us to combine models that need training on different types of training sets (single ending stories or stories with two possible endings). To satisfy this, the original story cloze validation set was split into 2 new sets: 80% training set with two endings, and 20% validation set.

The details of the sub-models and the combiner model is described in the next section.

3 Models

3.1 VADER Sentiment Cosine Model

We hypothesized that in some cases a simple sentiment analysis could determine the right sentence. We used a simple approach to this task by using the pre-trained sentiment tool VADER (Valence Aware Dictionary and sEntiment Reasoner) [8], which is also the approach taken in [5]. Given a sentence, VADER outputs a 3-dimensional vector of probabilities denoting the probability that the sentiment of the given sentence is either negative, neutral, or positive. We extract these probabilities from all contexts; as the probabilities must sum to 1, we only need two of these, and thus we get an 8-dimensional vector for each context. We feed these values as features to a simple feed-forward network with one hidden layer. The output is then a 2-dimensional probability-vector for the context, which is compared to sentiment probability-vector of each ending, by using the cosine similarity,

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||}. \tag{1}$$

For prediction we simply take

$$\mathbf{e}^* \in \underset{\mathbf{e} \in \{e_0, e_1\}}{\operatorname{argmax}} \cos(\mathbf{c}, \mathbf{e}),$$

where e_0 and e_1 denotes the two possible endings. For the training, however, we can use the cosine similarity above to phrase the problem as a regression problem rather than a classification task, which makes it possible to train the model on the large training set with single endings.

3.2 RNN Language Model

A simple RNN language model was developed to estimate perplexities of complete stories by learning to predict the next word. Each word is embedded in a 300-dimensional vector space by using *ConceptNet Numberbatch* word embeddings [9]. A version of the model using 100-dimensional Word2Vec embeddings was also created for comparison [10]. At each timestep, the RNN takes a word vector as input, runs this through an LSTM cell, projects the cell output onto the vocabulary and performs a softmax to get a word probability distribution. The model was trained by optimizing a cross-entropy loss and performs prediction by choosing the most probable context-ending combination. To reduce the model size and prevent overfitting the vocabulary was reduced to the 20,000 most common words. Out of vocabulary words were replaced by a special `<unk>` token.

3.3 Doc2Vec Cosine Model

The Doc2Vec Cosine Model uses pre-trained *Doc2Vec* [11] embeddings available via the *GenSim* library [1]. A *Doc2Vec* model uses a *document* as input and a *tag* as the target. The model is able to infer a meaningful embedding for a document, which allows us to measure the similarity between documents. For each story, all five sentences were used as the document and the title of the story as a tag. Using this approach means the model cannot be trained on the validation set, since the validation set does not provide story titles - it can however still predict on the validation set. Using the context as the document and the ending as the tag was also tried, as this is more analogous to how the other models were trained - however, this approach barely managed to pick up any signal in the data. A hypothesis for the title being a better tag is that a story's title often sums up what happens in the story, which can be useful for deciding the correct ending. The best accuracy for the Doc2Vec model was achieved by using a Distributed Bag-of-Words model with 50-dimensional embedding vectors and with all English stop-words removed from documents, as well as stemming each word. Similar to the Vader Sentiment Cosine model, this model also predicts by embedding the context as well as both endings and then uses the cosine similarity of equation (1) to decide which ending is most similar to the context.

3.4 BiRNN Discriminator Model

This model consists of a single layered, bi-directional RNN using GRU recurrent cells with input size 100 and hidden state size 160. The architecture is shown in figure 1. It follows the BiLSTM-V model

[1] https://pypi.org/project/gensim/

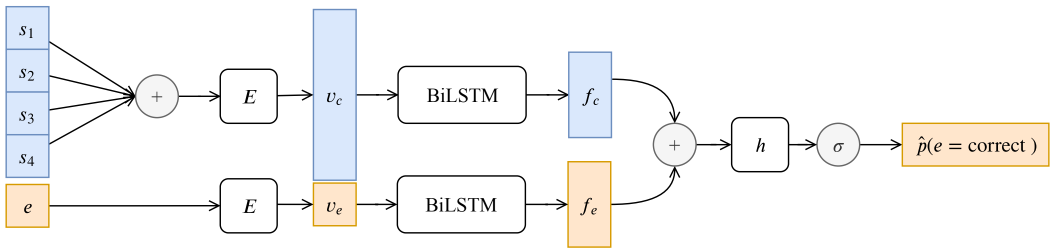

Figure 1: Architectural overview of the Bi-directional RNN Discriminator. The $+$ indicates concatenation, E word embedding, h a fully connected layer, and σ is the sigmoid activation function. The output of the RNN $\hat{p}(e = \text{correct})$ indicates the probability that the ending given in the input is the correct ending.

proposed by Bugert et al. [7]. The word-embeddings and vocabulary are the same as were used for the RNN Language Model described in section 3.2. The embedded context- and ending vectors are fed separately to the RNN and the four resulting hidden state-vectors of size 160 are concatenated to create a feature vector $\mathbf{x}$ of length $n = 4 \cdot 160 = 640$ representing the entire story. These features are then fed through a single hidden layer of dimensionality 128 and finally to an output unit with sigmoid as the activation function.

$$\mathbf{h} = \sigma(\mathbf{W_1 x} + \mathbf{b_1}) \quad \text{where} \quad \mathbf{W_1} \in \mathbb{R}^{128 \times n}, \quad \mathbf{b_1} \in \mathbb{R}^{128}, \quad \mathbf{x} \in \mathbb{R}^n \tag{2}$$

$$\hat{y} = \sigma(\mathbf{W_2 h} + b_2) \quad \text{where} \quad \mathbf{W_2} \in \mathbb{R}^{1 \times 128}, \quad b_2 \in \mathbb{R} \tag{3}$$

For each context-ending pair the model will give a probability $\hat{y} = \hat{p}(e = \text{correct})$ that the ending e is the correct ending for the context c. The Cross-Entropy between y and $\hat{y}$ was used as the loss-function for training, and since this loss function requires both positive and negative samples, the training set with two endings was used.

The architecture was also extended with an attention mechanism. Since attention typically is used for decoding over multiple time-steps [12] the concept had to be adapted to this feature extraction task. For the hidden state $\mathbf{u}_t$ at time step t a score $\mathbf{s}_t = \tanh(\mathbf{W}_a \mathbf{u}_t + \mathbf{b}_a)$ is calculated, where $\mathbf{X}_a$ and $\mathbf{b}_a$ are learnable parameters. The scores are then normalized to a probability distribution using a softmax layer $\tilde{\mathbf{s}} = \text{softmax}(\mathbf{s})$. Rather than using the final hidden state as a feature vector, the representation is instead computed as a weighted sum of the hidden state for each time-step $\hat{\mathbf{u}} = \sum_t \tilde{\mathbf{s}}_t \mathbf{h}_t$. The attention is applied independently on the hidden states of the forward- and backward passes.

3.5 Combiner Model

The final combiner model combines features from all models and tries to predict whether a context-ending pair is a valid story. The features include:

- Predicted and ground truth end sentiment probabilities from the VADER Sentiment Cosine model (section 3.1)
- Perplexity values for the story, estimated by the RNN Language model (section 3.2)
- Cosine similarity between context and ending embeddings from the Doc2Vec Cosine model (section 3.3)
- Hidden layer output $\mathbf{h}$ of the BiRNN (section 3.4)
- A small set of manually extracted features:
  - Ending length
  - Mismatch of gendered pronouns between context and ending
  - Presence of negations (not, wasn't, ...) in ending

Model	Validation	Test
Doc2Vec Cosine	65.7	58.4
VADER Sentiment Cosine	57.9	57.5
Language Model Word2Vec	54.4	56.9
Language Model NumberBatch	56.3	57.8
BiRNN Word2Vec	66.0	67.5
BiRNN NumberBatch	**70.0**	**67.6**
Combiner No Finetuning	69.2	67.1
Combiner With Finetuning	63.0	61.4

Table 1: Accuracies (%) of all models on the validation and test sets.

All these features are combined into a feature vector $x \in \mathbb{R}^n$ and fed through the same layers described in equations (2, 3), n being the number of features. The model was trained as a binary classification problem, the target being 1 if the ending is correct and 0 if wrong. The combiner assumes the other models to be re-trained, but also allows for end-to-end fine-tuning of the respective models, analogous to how pre-trained word-embeddings can be trained further.

4 Experiments

All models were trained and evaluated on the ETH Leonhard GPU cluster. In table 4 we report the accuracy on both our validation set and the given test set. For the Language Model and the BiRNN model we also report results obtained by using different embeddings (Word2Vec and NumberBatch), and for the Combiner Model with pre-trained, fixed sub-model parameters and with additional end-to-end training (no finetuning vs. finetuning).

Notable we see that the Combiner Model does not achieve better accuracy than the best of the sub-models. Also, we see that the combiner-model performs quite poorly when trained end-to-end.

5 Discussion

This section will briefly discuss various implementation variations of the models and their performance.

The attention mechanism was an interesting addition to the RNN, but in practice, we observed no substantial improvement in accuracy. It did, however, have a stabilizing effect on the validation accuracy during training and was therefore still deemed a desirable property to include.

Most importantly, we were surprised to find that the combination does not substantially improve the accuracy over the BiRNN. We tried to mend this in various ways; for example, attempts were made to more explicitly model a learning weighting over the sub-models. In these experiments, we tried to learn a probability distribution over the sub-models and used this as a weighting over their outputs. However, when using this setup the combiner converged to putting all weights on the best model (BiRNN), so it was abandoned. We still believe that there is value in harnessing the strengths of an ensemble of models, but if we cannot come up with a smart hand-crafted model for the combination step, more data is probably needed to learn this from the data alone.

6 Conclusion

We have implemented a collection of models suited for solving the *StoryCloze Task* by focusing on different aspects of natural language processing, as well as a combiner model which was intended to utilize the different strengths of these models. The *StoryCloze Task* is a challenging problem, and we were not able to come up with an approach able to achieve anywhere near human-level accuracy. We conclude that the chosen approach did not work better than the best RNN model in isolation, and so while the idea is novel, its practical effectiveness is questionable.

References

[1] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. A corpus and evaluation framework for deeper understanding of commonsense stories. *CoRR*, abs/1604.01696, 2016. URL `http://arxiv.org/abs/1604.01696`.

[2] Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain, April 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-0906. URL `https://www.aclweb.org/anthology/W17-0906`.

[3] Zhongyang Li, Xiao Ding, and Ting Liu. Story ending prediction by transferable bert. *arXiv preprint arXiv:1905.07504*, 2019.

[4] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[5] Jiaao Chen, Jianshu Chen, and Zhou Yu. Incorporating structured commonsense knowledge in story completion. *CoRR*, abs/1811.00625, 2018. URL `http://arxiv.org/abs/1811.00625`.

[6] Bingning Wang, Kang Liu, and Jun Zhao. Conditional generative adversarial networks for commonsense machine comprehension. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4123–4129, 2017. doi: 10.24963/ijcai.2017/576. URL `https://doi.org/10.24963/ijcai.2017/576`.

[7] Michael Bugert, Yevgeniy Puzikov, Andreas Rücklé, Judith Eckle-Kohler, Teresa Martin, Eugenio Martinez Camara, Daniil Sorokin, Maxime Peyrard, and Iryna Gurevych. LSDSem 2017: Exploring Data Generation Methods for the Story Cloze Test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*, pages 56–61, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://aclweb.org/anthology/W/W17/W17-0908.pdf`.

[8] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[9] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. pages 4444–4451, 2017. URL `http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972`.

[10] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL `http://arxiv.org/abs/1301.3781`.

[11] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL `http://arxiv.org/abs/1405.4053`.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.