Story Cloze Test Project Report

Thomas Diggelmann
ETH Zurich
thomasdi@student.ethz.ch

Doruk Cetin
ETH Zurich
dcetin@student.ethz.ch

George Mtui
ETH Zurich
gmtui@student.ethz.ch

1 Introduction

One end goal of Natural Language Understanding (NLU) as a branch is to develop systems that can understand structure and meaning of the human languages. The NLU community has achieved milestones in many application areas such as sentiment analysis, machine translation and question answering. As the evaluation of NLU systems are hard, researches have been designing proxy tasks for many subfields to train and evaluate such systems.

We tackle one such task, named Story Cloze Test (SCT) [Mostafazadeh et al., 2016], which considers story understanding to assess systems' capabilities of reading comprehension. The authors introduce the ROCStories corpus, where each entry is a five-sentence commonsense story. The SCT framework is defined on this corpus, where they randomly sampled a subset of the stories and created couples of *right* and *wrong* endings (with respect to their plausibility, entailment and relevance) on top of their four-sentence contexts. Here, a system is expected to choose between a right and a wrong ending, given four preceding sentences as the context.

Authors [Mostafazadeh et al., 2017] report results from eight systems on SCT and discuss the interesting outcomes of different approaches. Previous baseline on the task was improved from 58.5% to 75.2%. One major complication of the SCT is the lack of negative examples in the training data, as the corpus only contains stories with correct endings. Roemmele et al. [2017] and Bugert et al. [2017] choose to generate fake incorrect examples by sampling through endings from the training corpus itself and Wang et al. [2017] use a GAN to generate fake endings. However, models that are trained only using the training dataset achieved significantly less accuracies than the ones that utilized validation data (which has both positive and negative examples) in training. One such extreme case is explored in Schwartz et al. [2017], where they completely ignore the context sentences and train a model on the features extracted only from the validation dataset, which achieved a surprising accuracy score of 72.4%. Similarly, Mostafazadeh et al. [2017] claim sentiment is one factor in correct detection along with the stylistic features isolated in ending sentences. Since the task was not conducted as a blind challenge, it is not clear if the aforementioned methods capture artifacts of the test set and whether they can generalize well to unseen data.

2 Methodology & Model

We wanted to implement a model where we do not leverage the implicit biases (c.f. effects of directly using sentiment and stylistic features) but keep as close as possible to the original intent of the SCT. Our approach is to train a robust language model (LM) on the training dataset, with the help of multi-task learning, to indirectly integrate knowledge about sentiment. Our aim is to obtain a model that "understands" the text rather than relying on the stylistic differences in the validation corpus.

Our system is generative in nature: we train a language model on the training dataset and do predictions on the probabilities generated by the LM. Figure 1 illustrates the architecture of our model. The LM is modeled by a recurrent neural network (RNN) that utilizes word embeddings. Sentiment information is only used in the multi-task training of the language model. Lambda layer is used to retrieve the final output of the RNN to base the sentiment prediction.

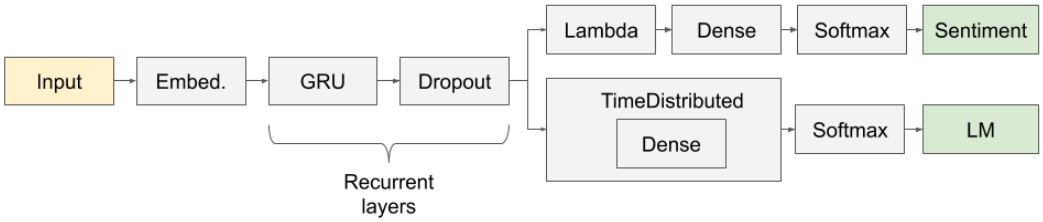

Figure 1: Architecture of the multi-task learning model, which additionally uses sentiment prediction as a second task besides the classic RNN-predict-next-word language model task.

There have previously been a few approaches to incorporate language models to SCT. Roemmele et al. [2017] used language models to generate fake endings on the training dataset. Chaturvedi et al. [2017] and Peng et al. [2017] trained semantic language models to generate conditional probabilities of the sentence sequences, which are incorporated into the final predictor as features. Lastly, Schwartz et al. [2017] trained a recurrent neural network language model, which they combine with the extracted stylistic features to reach the state of the art in Mostafazadeh et al. [2017]. Our work follows most closely the work done on language models by Schwartz et al. [2017].

We test with three different ways to predict the right endings using the outputs of our model. First method is quite straightforward: we compare the output perplexities and classify the ending that generates smaller perplexity as the right one. Second method uses the comparison of the probability ratios, as it is motivated by Schwartz et al. [2017]. Probabilities are calculated as in the Equation 1, where we classify ending with high probability as the right ending. The intuition is again the right ending should be less surprising than the wrong one: numerator denotes how unsurprising is the ending given the first four sentences, whereas the denominator accounts for the rare endings sentences that are inherently surprising. Lastly, we train classifiers on the validation dataset, using the features extracted by our trained language models. In other words, we evaluate our model on the validation dataset —producing sentiment predictions, probabilities and hidden states —and use the resulting “features" as input to a classifier to be trained on the validation dataset.

$$\frac{p_\theta(\text{ending} \mid \text{story})}{p_\theta(\text{ending})} \tag{1}$$

3 Training

In the preprocessing step we convert all the words into lowercase and tokenize them using nltk package. During tokenization, we use a maximum sequence length of 90 words and (pre-)pad all shorter sequences using a prespecified <pad> token. Similarly to Schwartz et al. [2017], we use a maximum vocabulary size of 20000. Both of these preprocessing steps are done due to computational reasons. Our models utilize the GloVe embeddings [Pennington et al., 2014] as the embedding layer of the network. Specifically, our experiments use 100 dimensional “glove.6B" embeddings.

We use the VADER [Hutto and Gilbert, 2014] sentiment analysis tool to obtain polarity scores for the sentences. We discretize the polarity scores by the typical thresholds suggested by the authors: we classify it as neutral statement if the compound score is in between $(-0.05, 0.05)$, if it is either above or below that interval we classify it as positive or negative sentiment, respectively.

We use a compound loss as our objective function as mentioned above. It is simply the weighted average of the two multi-task losses: the language model loss and the sentiment classification loss. Adam optimizer is used to optimize the parameters of our network.

4 Experiments

We ran all models for 5 epochs and created model checkpoints after each epoch. We then evaluate the performances of the models on the test set by using their respective best checkpoint on the validation dataset. We use a learning rate of 0.001 over mini-batches of size 50 and the dropout rate is consistently 0.5 for all the models. All experiments are ran on GeForce GTX 1080 Ti on ETH

Leonhard cluster. Table 4 summarizes the results of different models on validation and test datasets using different evaluation strategies explained in the previous section. Names denote the number of GRU layers and their respective number of units, e.g. $3x512$ stands for three layers of GRUs with 512 units each.

	Validation Accuracy		Test Accuracy		
Model	Perplexity	Prob. Ratio	Perplexity	Prob. Ratio	Features
1x512	0.548370	0.648316	0.547301	0.639230	0.672367
1x512 (w/o sentiment)	0.539284	0.631213	0.538215	0.629075	0.681988
1x512 (7/3)	0.541422	**0.649920**	0.548370	**0.645644**	**0.691608**
1x512 (attention)	0.538215	0.637092	**0.548904**	0.625869	0.669695
1x512 (7/3, attention)	**0.548904**	0.627472	0.546232	0.632817	0.687332
2x1000	0.535008	0.637627	0.548370	0.626937	0.687332
3x512	0.540887	0.636024	0.543025	0.630144	0.669695

Table 1: Validation and test accuracy scores of different configurations of our architecture

We keep the general architecture introduced in Figure 1 intact and make small modifications. Unless specified, the models use an equal weighting of the loss functions while training. Ratios in parentheses denote the weight factors of language model and sentiment losses, respectively. For some models we replace the lambda layer with a weighted attention layer (source) over the word sequences. We also experiment with deepening the network by replicating the RNN-Dropout layers. We always use GRU as the recurrent cell implementation as our experiments with LSTM (not shown here) did not bring any improvement.

Multiple RNN layers slightly under-performed the one layer models, so we choose to stick to a single layer of recurrent units. We have observed that weighting the language model loss more than the sentiment loss helped the model to learn more useful representations of the data. This is in line with our expectations as the LM loss represents our main objective and the sentiment prediction proved to be comparably an easier of a task; all models achieved over 0.98 accuracy when predicting sentiments over the validation dataset. Still, we can confirm the addition of the sentiment is beneficial and the improvement due to it cannot be ignored. Attention could not improve upon the results of the rather simplistic Lambda layers. We believe this trend —together with the fact that most of the changes only result in marginal differences —shows that we already utilize what we can mostly infer from our training setup (i.e. training data, embeddings and sentiment classes).

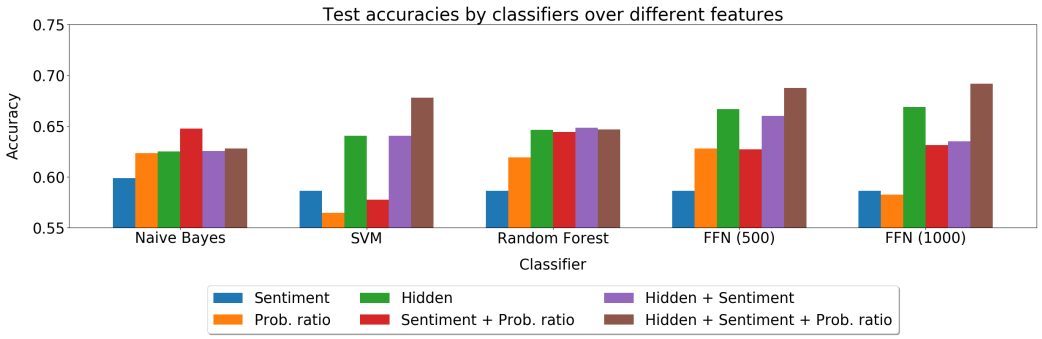

Figure 2: Comparison of the different classifiers trained on different subsets of features. FFN stands for “feedforward network" and the number inside the parentheses denote the width of their only hidden layer. Results are from the $7/3$ weighted run without attention, arguably our best model.

Lastly, Figure 2 illustrates the third evaluation method. While interestingly probability based classifiers cannot utilize the given features well, SVMs and neural networks can obtain competitive scores on the test set. We can also observe the hidden representations being the most informative set of features, proving it to be very useful for even relatively simple classifiers.

5 Future Work

As an extension of the current model, one could also think of integrating further tasks to the multi-tasking approach, such as named entity recognition (NER) or part-of-speech (PoS) tagging. We also believe replacing our "vanilla LM" with a semantic one that encodes more prior knowledge (e.g. through frame semantics, hierarchical relationships) can lead to models that are more capable at reading comprehension. Similarly, replacing the embedding layer with more powerful word embeddings that are trained on larger corpora could be beneficial to the model.

6 Conclusion

We have trained a language model for the Story Cloze Task through multi-task learning, using sentiment polarity scores. Together with the previous studies on the task we have shown that incorporating semantic elements helps the model in reading comprehension, while staying true to the spirit of the task. Although we have trained a generative model to solve this task by proxy, we also confirm that the learned hidden representations can be further exploited by discriminative models.

We are providing the predictions on the LM output using probability ratio evaluation method, rather than using the classifier output on extracted features which achieves higher accuracy on test data. We are not after a SOTA result in terms of accuracy but rather wanted to see if we could use an agnostic model that learns from the data and background knowledge.

References

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. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 56–61, 2017.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614, 2017.

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.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*, 2016.

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, 2017.

Haoruo Peng, Snigdha Chaturvedi, and Dan Roth. A joint model for semantic sequences: Frames, entities, sentiments. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 173–183, 2017.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew Gordon. An rnn-based binary classifier for the story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 74–80, 2017.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A Smith. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. *arXiv preprint arXiv:1702.01841*, 2017.

Bingning Wang, Kang Liu, and Jun Zhao. Conditional generative adversarial networks for commonsense machine comprehension. In *IJCAI*, pages 4123–4129, 2017.