Deep learning for Short Answer Scoring

Surya K, Ekansh Gayakwad, Nallakaruppan, M.K

Abstract: ***Automated scoring of descriptive answers can find applications in educational assessment and is one of the applications of Natural Language Processing. Deep learning has contributed significantly to the growth of NLP in recent years. Deep NLP techniques are ideal for automated scoring especially short answer scoring tasks. We compare some common deep learning models for the SAS task.***

KeyWords: NLP,DLP, SAS task.

I. INTRODUCTION

Deep learning has shown impressive results on various NLP tasks such as Machine translation, question answering, text summarization etc. [1]. Traditional machine learning requires heavy feature engineering which mostly involves extraction of handcrafted features using techniques like regular expression matching, lemmatization/stemming, tokenization etc, which can be time-consuming and cumbersome, to achieve significant results. They are generally trained on high dimensional sparse features and can be computationally expensive. With successes in feature embeddings, which are low dimensional dense representations of textual data, deep learning models are robust and easier to train. They outperform most approaches in NLP tasks with minimal feature engineering. The computational requirements that most deep learning architectures bring with them are well accommodated with the use of GPUs. Further, transfer learning has greatly helped overcome several common challenges in deep learning including lack of datasets of sufficient size and lack of computational resources to train very deep networks. Recently, some works[29][30][31] have shown the success of transfer learning in NLP. In short answer scoring, we assign scores for brief answers to the corresponding question prompts. We train our model on multiple individual responses for each question prompts using the scores assigned by annotators as a target. The responses typically consist of one or few sentences. The question can be from multiple or single domains based on the assessment. The prompts might also include open-ended questions. In short answer scoring, we mainly consider the content of the responses over grammar, spelling, and vocabulary.For our task, spelling and grammar are not considered for scoring but might affect the clarity of the responses.

Revised Manuscript Received on March 25, 2019.
Surya K, Vellore Institute of Technology, *Vellore, India.*
Ekansh Gayakwad, Vellore Institute of Technology, *Vellore, India.*
Nallakaruppan M.K, Vellore Institute of Technology, *Vellore, India.*

Deep learning approaches suit SAS tasks better because of their ability to learn a complex hierarchical representation of data automatically. In this work, we investigate some common deep learning approaches to short answer scoring. The dataset and evaluation metric used are discussed in the following sections followed by the methods and results.

II. DATASET

We used the automated student assessment prize - short answer scoring dataset by the Hewlett Foundation [2]. The dataset consists of 10 question prompts. The training set contains 17000 examples with around 1700 examples per question prompt and test set contains 5100 examples with 300 to 600 examples per question prompt. The responses

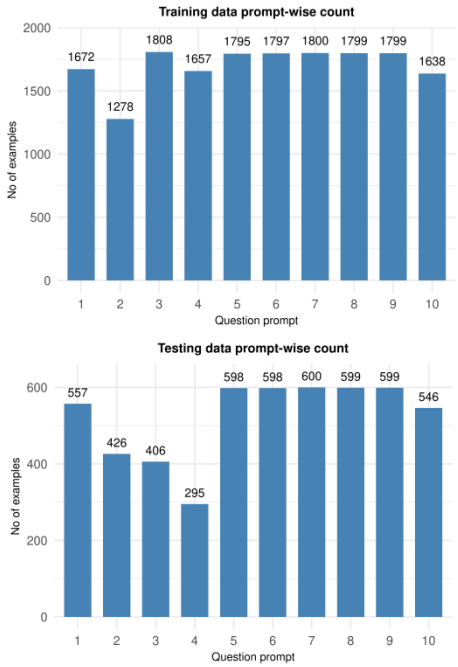

Fig :Frequency of responses across question prompts

have an average length of about 50 words. High variability among question sets is present as they cover a broad range of disciplines. For all the models we use 20% of the training data as the validation set. Refer to figure 1 and 2

III.EVALUATION

We use the quadratic weighted kappa error as the Fig. evaluation metric as per the competition guidelines. It measures the agreement between 2 raters. We average the quadratic weighted kappa across the question sets using fisher transformation as instructed by the Kaggle competition guidelines.

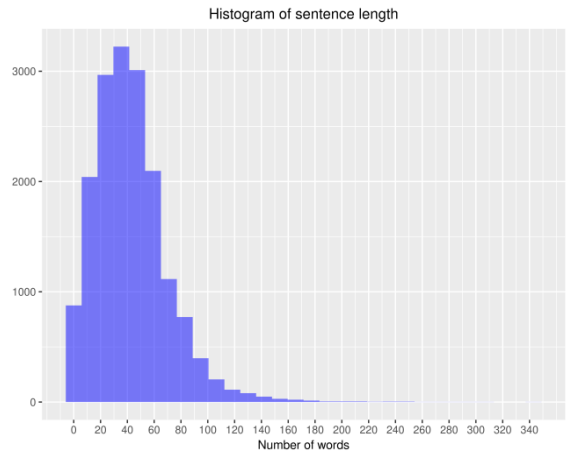

Fig. 2 Histogram of sentence length

IV. LITERATURE REVIEW

Some attempts to develop scoring engines for short answers include c-rater[3] by ETS technologies.

Microsoft's Powergrader[5] uses a learned similarity metric to cluster answers for a question. The grader can score the responses in each cluster collectively greatly reducing the effort required for grading. It normalizes responses into a canonical form based on variations among sentences for concept matching. The model is constructed by hand by content experts.

[7] also use an unsupervised approach and explore knowledge-based (WordNet) and corpus-based (LSA and ESA) measures. They also propose a feedback technique to improve the performance of the system.

[4] evaluates n-gram and word level matching and Doc2Vec based similarity methods. [6] use Maximal Marginal Reference to obtain reference answers and find similarity between the reference and student answer using GAN-LCS.

[8] extract features from the question, answer and student models and evaluate on 6 different algorithms. They show better performance using Deep Belief Networks[9] and significant improvement by using composite features from the question and student model.

The winners of the asap-sas competition[10] use heavy feature engineering and use ensemble methods on different types of models.

(Tandalla, 2012) used the Boruta algorithm to determine the relevant words, bigrams,and trigrams that helped to predict the score. This was a key step in the model's performance. Further, he used regular expressions to look for acceptable responses. He trained multiple models using random forest and gradient boosting machine and averaged their predictions.

(Zbontar, 2012) used stacking with ridge regression, SVM, K-Nearest neighbor, Random Forests and Gradient Boosting Machines as base learners to train. Features were created using character level four-grams and 6-grams and latent semantic indexing.

(Conart, 2012) used 6 ensembles of about 81 individual models, trained per answer set. The models included GLM, SVM, RF used with different tokenization, scaling and reduction techniques. They were stacked at multi-levels with GLM and Generalized additive models at second-level and ordinary least squares at the third-level. 3 other ensembles based on Nelder-Mead optimization to obtain coefficients. The best of the 6 ensembles were selected as a final model.

(Jesenky, 2012) extracted features from several bag-of-word and string matching and different models were run of subsets of the features. They use genetic algorithm on the weak learners to find the best performing subset. The final predictions were selected through voting.

(Peters & Jankewicz, 2012) extracted features using unsupervised algorithms along with some text statistics, compression-based text similarity. They stacked models like SVM, GBM, cubist and Sofia using a GBM model.

[11] proposed an improvement to Tandalla's approach by automating the generation of patterns. They extracted the content tokens and structure information using word-order graphs. They group the semantically similar related words to facilitate alternative responses. Their performance of their approach is on par with that of Tandalla's.

V. ARCHITECTURE

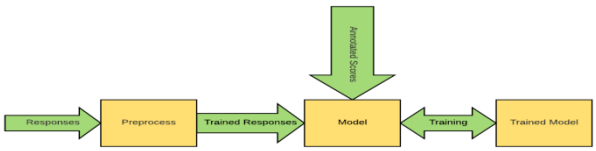

Figure 1 Training architecture

We adopt a typical supervised learning architecture. Models were trained on preprocessed response text with the annotated scores as the target. Some models also use question set as input.

Figure 2 Testing architecture

Models trained were saved and then used for prediction on the test data. Inputs were preprocessed similar to that in the training process. The model then outputs the predicted score.

VI. METHODOLOGY

Minimal preprocessing was applied to the dataset including converting all characters to lower cases. Other method-specific preprocessing methods are mentioned in their corresponding sections.

All methods use a batch size of 32 for training. Learned word embeddings were used over pre-trained word embeddings as they showed slightly better performance. The outputs of regression models are rounded to an integer value.

A. Character level CNN

We use the character level CNN from (Zhang, X. and LeCun, Y. 2019). CharCNN [18] learns to predict a target without the use of any knowledge embeddings or syntactic and semantic structure of the language.

We use a maximum input length of 1800 characters with zeros padded at the end. We use the model design described in the paper with the eighth layer replaced by a fully-connected layer that predicts the score. Particularly, we use the small version of the ConvNet. We merge the

question set in the final layer.
We use Adam [19] optimizer with an initial learning rate of 0.001. We use a multi-step linear learning rate decay. The learning rate is reduced by 0.1 times once every 3 epochs. The model converged in about 15 epochs.

B. Word level CNN

For the method punctuations and stop words were removed and the text was padded and truncated at the end before passing as input. We use a maximum input sequence length of 90.

We use a 1D Convolution layer with 64 filtersof window size 5 and a max pooling layer with pool size 4 followed by a dropout [20] layer with dropout probability 0.5 and batch normalization [21]. The question set value is merged with the feed-forward layer which is l2 regularized and outputs a single value.

We use RMSProp [22] optimizer with an initial learning rate of 0.01. We use an exponential learning rate decay at a rate 0.1 applied once every 3 epochs. The model converged inabout 15 epochs.

C. Word level bi-LSTM

The text is pre-processed similar to that in the CNN model.We use a maximum input sequence length of 90.

We use a 250-dimensional bi-LSTM [23][24] model with learned50-dimensional embeddings. The bi-LSTM is followed by a global average pooling and dropout layer with dropout probability 0.5 and batch normalization. The question set value is merged with the feedforward layer which is l2 regularized and outputs a single value.

We use the RMSProp optimizer with learning rate and learning rate decay similar to the CNN model. We clip the gradients at 10 to prevent exploding gradients. The model converged in about 15 epochs. We use the cuDNN [25] implementation of the LSTM to speed up the training on GPU.

D. BERT

BERT [26] is a bidirectional language representation model pre-trained with deep Bidirectional Transformers. They can be easily fine-tuned for many downstream tasks.

We use the BERT base model. We add a fully-connected layer for classification with softmax activation. We use a maximum input sequence length of 90.

We use Adam optimizer with a learning rate of 5e-6 and an epsilon value of 1e-9. We clip the gradients by their global norm using a threshold of 1. The model converged in about10 epochs.

Training was faster compared to other methods and it achieved better performance in lesser epochs.

VII. RESULTS AND ANALYSIS

The results of evaluation on the test data are shown in table 1.

All experiments were performed on an NVIDIA 940mx

GPU. The models took about 15 to 20 minutes to train on the same.

TABLE I
METRICS ON TEST DATA

Prompt	CharCNN	CNN	LSTM	BERT
1	0.68	0.68	0.70	0.79
2	0.58	0.67	0.66	0.70
3	0.29	0.27	0.28	0.37
4	0.56	0.55	0.59	0.69
5	0.72	0.75	0.78	0.75
6	0.80	0.74	0.74	0.84
7	0.55	0.58	0.60	0.66
8	0.43	0.50	0.54	0.60
9	0.67	0.64	0.70	0.80
10	0.61	0.67	0.71	0.74
QWK	0.70	0.73	0.75	0.79
Mean QWK	0.60	0.62	0.65	**0.71**

BERT performs better than the other models which can be attributed to its ability to learn deep bidirectional representations and the rich knowledge representations in its pre-trained language model.

The LSTM model performs better over CNN models as RNN models encode long-range context dependency [27] which help them handle information across multiple sentences better. (Yin et al., 2019) show that RNNs perform better than CNNs when the input sequence is long.

We observed that using pre-trained sentence embeddings like USE [28] did not produce satisfactory results.

We observe that all models perform poorly on question prompt 3. This is because of the fact that it is an open-ended question and assesses the student's conceptual understanding and interpretation skills. It's also common to find less agreement between scorers for such type of questions which affects the data annotations itself.

We further performed simple experiments to test the reliability of the models by modifying a few examples from the data that has a full score and was scored correctly by all the models. We performed the following changes in the responses

1. Introducing spelling mistakes
2. Paraphrasing the documents
3. Replacing few words with a synonym

We found that the RNN model performed better than the other models in the case where minor spelling mistakes were introduced and, in the case, where words were replaced with their synonyms.

BERT performed significantly better when the responses where paraphrased. This is because of BERT's deep contextual representations conditioned in both directions which allow the context to be preserved despite the change in structure. The performance of BERT was still poor compared to its performance on the test data.

I. CONCLUSIONS

We use different types of deep learning models to compare their performance among themselves and demonstrate the simplicity and efficiency of the architectures over non-neural approaches which require extracting several features from the text data manually as can be observed in the methods used by the winners of the Kaggle competition. They also train faster and are resilient to minor variations within the data.

Several challenges still exist in adopting automated scoring systems including lack of robust and generic approaches and digitized data for training, lack of simple tools and resources that can help adopt the system without requiring expertise in NLP.

REFERENCES

1. Young, T., Hazarika, D., Poria, S. and Cambria, E. (2019). Recent Trends in Deep Learning Based Natural Language Processing.
2. https://www.kaggle.com/c/asap-sas/
3. Leacock, Claudia, and Martin Chodorow. "C-rater: Automated scoring of short-answer questions." Computers and the Humanities 37, no. 4 (2003): 389-405.
4. Nogaito, Izuru, Keiji Yasuda, and Hiroaki Kimura. "Study on Automatic Scoring of Descriptive Type Tests using Text Similarity Calculations." In EDM, pp. 616-617. 2016.
5. Basu, Sumit, Chuck Jacobs, and Lucy Vanderwende. "Powergrading: a clustering approach to amplify human effort for short answer grading." Transactions of the Association for Computational Linguistics 1 (2013): 391-402.
6. Pribadi, Feddy Setio, Adhistya Erna Permanasari, and Teguh Bharata Adji. "Short answer scoring system using automatic reference answer generation and geometric average normalized-longest common subsequence (GAN-LCS)." Education and Information Technologies (2018): 1-12.
7. Mohler, Michael, and Rada Mihalcea. "Text-to-text semantic similarity for automatic short answer grading." In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, pp. 567-575. Association for Computational Linguistics, 2009.
8. Zhang, Yuan, Rajat Shah, and Min Chi. "Deep Learning+ Student Modeling+ Clustering: a Recipe for Effective Automatic Short Answer Grading." In EDM, pp. 562-567. 2016.
9. Hinton, Geoffrey E. "Deep belief networks." Scholarpedia 4, no. 5 (2009): 5947.
10. https://www.kaggle.com/c/asap-sas#winners
11. Ramachandran, Lakshmi, Jian Cheng, and Peter Foltz. "Identifying patterns for short answer scoring using graph-based lexico-semantic text matching." In Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 97-106. 2015.
12. Selvi, P., and A. K. Bnerjee. "Automatic short-answer grading system (ASAGS)." arXiv preprint arXiv:1011.1742 (2010).
13. Gomaa, Wael H., and Aly A. Fahmy. "Tapping into the power of automatic scoring." In The Eleventh International Conference on Language Engineering, Egyptian Society of Language Engineering (ESOLEC). 2011.
14. Brew, Chris, and Claudia Leacock. "Automated short answer scoring." Handbook of automated essay evaluation: Current applications and new directions 136 (2013).
15. Pribadi, Feddy Setio, Teguh Bharata Adji, Adhistya Erna Permanasari, Anggraini Mulwinda, and Aryo Baskoro Utomo. "Automatic short answer scoring using words overlapping methods." In AIP Conference Proceedings, vol. 1818, no. 1, p. 020042. AIP Publishing, 2017.
16. Siddiqi, Raheel, Christopher J. Harrison, and Rosheena Siddiqi. "Improving teaching and learning through automated short-answer marking." IEEE Transactions on Learning Technologies 3, no. 3 (2010): 237-249.
17. Kudi, Pooja, Amitkumar Manekar, Kavita Daware, and Tejaswini Dhatrak. "Online Examination with short text matching." In Wireless Computing and Networking (GCWCN), 2014 IEEE Global Conference on, pp. 56-60. IEEE, 2014.
18. Zhang, X. and LeCun, Y. (2019). Text Understanding from Scratch. Available at: https://arxiv.org/abs/1502.01710 [Accessed 9 Jan. 2019].
19. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
20. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), pp.1929-1958.
21. Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167
22. Tieleman, Tijmen, and Geoffery Hinton. "RMSprop gradient optimization." URL http://www. cs. toronto. edu/tijmen/csc321/slides/lecture_slides_lec6. pdf (2014).
23. Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735-1780.
24. Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), pp.2673-2681.
25. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B. and Shelhamer, E., 2014. cudnn: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759
26. Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [online] arXiv.org. Available at: https://arxiv.org/abs/1810.04805 [Accessed 9 Jan. 2019]
27. Yin, Wenpeng, Katharina Kann, Mo Yu, and Hinrich Schütze. "Comparative study of cnn and rnn for natural language processing." arXiv preprint arXiv:1702.01923 (2017).
28. Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. "Supervised learning of universal sentence representations from natural language inference data." arXiv preprint arXiv:1705.02364 (2017)
29. Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 328-339. 2018.
30. Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).
31. Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf (2018).

shivdas18@gmail.com