Automated text scoring, keeping it simple

Chané S. Moodley[1,2] and 'Maletšabisa Molapo[1]

[1]IBM Research | Africa, Johannesburg, South Africa, 2000
[2] University of the Witwatersrand, Johannesburg, South Africa, 2000

chane.simone.moodley1@IBM.com

In traditional automated text scoring approaches stop-words are either immediately removed or authors do not give importance to the handling of stop-words [1, 2, 3]. Recent studies have, however, found that removing stop-words may adversely affect certain models and should not be considered a standard component of the text pre-processing pipeline [4]. Given an essay, the task is to predict a numerical score or grade. To improve the accuracy of existing neural network approaches for essay scoring, recent attempts have focused on developing increasingly complex neural networks with little to no consideration of the text pre-processing pipeline [5, 6, 7].

In this work we investigated the text pre-processing pipeline for automated text scoring (ATS). We investigated how stacked LSTMs coupled with an adjustment to the text pre-processing pipeline and basic word embedding models can achieve results on par with the state-of-the-art. We used the ASAP dataset to train a basic LSTM deep learning model. For automated text scoring, which is concerned with the quality of writing, stop words contain crucial information for the system to predict accurate scores and should therefore, remain in text. We compared cases with and without stop-words removed to determine if there are any significant changes in the score prediction accuracy. We also compared two- and three- layer deep LSTMs to identify any significant differences.

We found that keeping stop-words present significantly improves the prediction accuracy of the model while increasing the depth of the neural network shows no statistical significance. We showed that simple deep learning models coupled with tailored text pre-processing achieve results on-par with state-of-the-art models reducing the need for complex models and feature engineering for automated text scoring.

The ASAP dataset [8], sponsored by the Hewlett Foundation for a Kaggle competition, has been used extensively for neural text scoring and contains 12,976 essays, marked by two raters. The essays range in length between 150 – 650 words and were written by students in Grades 7 – 10. For this experiment, we used this dataset with the resolved combined domain score between the two raters.

We constructed two- and three-layer LSTMs. We did not lemmatize or stem the essays. We cleaned the text by removing punctuation and special characters. All the text was converted to lower-case. Importantly, in the case where we kept stop-words present, we did not identify dataset-specific stop-words and opted not to remove stop-words via the use of stop-word lists. For the removal of stop-words we used a popular word list from a python NLP library (NLTK[1]). We tokenized the essays into word and

[1] www.nltk.org

sentence lists for each essay respectively. The sentences were passed into the Word2Vec [9] embeddings model where feature vectors were created. Each essay was then treated as a set of tokens at sentence level. The tokens, token length, vocabulary size and embeddings were then used as input to the deep LSTM networks. The networks were regularized using both regular and recurrent dropout set to a probability of 40% each. The mini-batch was set to 64 and the networks were trained for 10, 20 and 30 epochs, respectively. We optimized using both RMSProp and Adam. We tested the difference in output of 2- and 3-layer LSTMs and evaluated using the official evaluation metric of the Kaggle competition, the Quadratic Weighted Kappa (QWK), which we computed over the whole test set. We conducted an independent two-tailed t-test between removing stop-words and keeping them present for each model.

Special characters and punctuation are thought to be non-informative features contained within the text corpus, but they also concatenate with the words they are close to. This renders the words unavailable in the dictionary and contributes adversely to the vector space generated by the word embeddings model. For stop-word removal we applied the NLTK stop-word list to remove all stop-words within the corpus. Stop-words are words that are thought to be common and non-informative, examples include: “and”, “then”, “is”, “a”. Contrastingly we opted to train and test the models without removing stop-words at all. In both LSTM networks we see statistical significance in the comparison between removing and not removing stop-words for each respective model ($p < 0.0001$). We used the two most common optimizers for ATS, RMSProp and Adam. Results fluctuate ever so slightly between optimizers, however, Adam shows better results overall. Adam also proves to be the most efficient, reducing training time, on average, by 10s per epoch.

Unlike the approach of Alikaniotis et al. [10], we did not create Score Specific Word Embeddings (SSWEs) but employed the use of basic Word2Vec to create word embeddings for us. Notably Alikaniotis et al. reported their state-of-the-art model scored a QWK of 0.96 for a 2-layer BiLSTM using SSWEs. Comparatively our model scores a QWK of 0.70 with stop-words removed and 0.956 without removing stop-words using Word2Vec embeddings. Our 2-layer LSTM model provides results already on par with current state-of-the-art, our deeper 3-layer LSTM gives us a QWK of 0.959. In our experiments we did not find any statistically significant difference in increasing the depth of LSTM models, both with and without stop-words removed ($p > 0.05$).

In conclusion, we explored the text pre-processing pipeline for ATS. Our findings show that for ATS on the ASAP dataset not removing stop-words not only significantly increases model performance but allows LSTMs to achieve very promising results. We found that for ATS as much content as possible needs to be preserved. ATS is meant to support teaching and learning by providing quick and accurate feedback. Accuracy can only be achieved if the model sees as much content as a teacher would see. Removal of stop-words, therefore, diminishes the quality and content of an essay.

References

1. Taghipour, K., Ng, H. T.: “A Neural Approach to Automated Essay Scoring,”. In: *Empir. Methods Nat. Lang. Process.*, pp. 1882–1891 (2016).

2. Nguyen, H., Dery, L.: “Neural Networks for Automated Essay Grading,”. In: *CS224d Stanford Reports*, pp. 1–11, (2016).
3. Cummins, R., Rei, M.: “Neural Multi-task Learning in Automated Assessment,”. In: *arXiv Prepr. arXiv 1801.06830*, pp. 1–9 (2018).
4. Medium, https://medium.com/@limavallantin/why-is-removing-stop-words-not-always-a-good-idea-c8d35bd77214, last accessed 2019/11/03.
5. Farag, Y., Yannakoudakis, H., Briscoe, T.: “Neural Automated Essay Scoring and Coherence Modeling for Adversarially Crafted Input,”. In: *North Am. Chapter of Assoc. Comput. Linguist. Hum. Lang, vol. 1*, pp. 263–271 (2018).
6. Dong, F., Zhang, Y., Yang, J.: “Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring,”. In: *Comput. Nat. Lang. Learn.*, pp. 153–162 (2017).
7. Zhang, H., Litman, D.: “Co-Attention Based Neural Network for Source-Dependent Essay Scoring,”. In *Proc. of the Thirteen. Work. Innov. Use o fNLP Build. Educ. Appl.*, pp. 399–409 (2018).
8. Automated Student Assessment Prize, https://www.kaggle.com/c/asap-aes, last accessed 2019/05/01.
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: “Efficient Estimation of Word Representations in Vector Space,”. In: *arXiv Prepr. arXiv 1301.3781*, pp. 1–12 (2013).
10. Alikaniotis, D., Yannakoudakis, H., Rei, M.: “Automatic Text Scoring Using Neural Networks,”. In: *arXiv Prepr. arXiv 1606.04289* (2016).

Appendix: Tables of results

Tables 1 and 2 describe the results obtained from the two- and three-layer LSTM models respectively.

Table 1. 2-layer LSTM stop-word comparison.

Training	Stop-words removed	Stop-words present	Optimizer
10 epochs	0.6432	0.8896	RMSProp
	0.6657	0.9126	Adam
20 epochs	0.6947	0.9433	RMSProp
	0.6999	0.9481	Adam
30 epochs	0.7057	0.9540	RMSProp
	0.7075	**0.9561**	**Adam**

Table 2. 3-layer LSTM stop-word comparison.

Training	Stop-words removed	Stop-words present	Optimizer
10 Epochs	0.6569	0.8930	RMSProp
20 Epochs	0.7038	0.9508	RMSProp
30 Epochs	0.7082	0.9563	RMSProp
	0.7098	**0.9585**	**Adam**