Predicting English Essay Results With ELMO Embeddings

Gabriele Cesa (11887524) **Davide Belli** (11887532) **Alexander Geenen** (11855924) **Linda Petrini** (11858931)
Gautier Dagan (11441348)

Abstract

In this work we address the issue of Automated Essay Scoring on the Automated Student Assessment Price (ASAP) dataset. We approached this task by building around variants of LSTM models. To represent the long sequences of text in essays, we considered different ways to generate meaningful embeddings. First, we consider a commonly used approach, consisting of fine-tuning Score-Specific Word Embeddings (*SSWE*). We then switch to *ELMo* (Embeddings from Language Models), a recently proposed contextual character-based word representation. Furthermore, we examine two different pre-processing approaches to efficiently capture and represent the information from misspelled words.

1. Introduction

Automated Essay Scoring deals with the challenge of reducing the costly and time consuming effort of manually scoring students' assignments. This is a particularly relevant social problem for several reasons. First, essays better capture students' capabilities than multiple choice tests, which are nevertheless normally preferred for convenience. Second, a unified and automatic scoring tool would guarantee fair scoring and consistency. Third, such a tool would save both time and money that could be reinvested in education.

The Hewlett Foundation addressed the problem by issuing a $\$\ 100,000$ prize Kaggle competition[1] in 2012, before the wave of interest towards Deep Learning techniques. The considered approaches for automated scoring, such as ranking and regression, have already produced human-like performances, but with the downside of time-consuming handcrafting of features. Given the complexity of the task, these features would ideally encode properties such as types of syntactic constructions, clarity of grammatical relations, and refinement of the vocabulary. Ideally, learning meaningful score-specific word embeddings might reduce the effort required.

Our goals in this project are the following:

- Replicate state of the art results on the ASAP dataset, with the approach proposed in (Alikaniotis et al., 2016) that makes use of Score-Specific Word Embeddings (SSWE).
- Compare different approaches to embed misspelled words in essays, namely replacing with the correct word and prefixing with a specific symbol and inserting a token specific for misspellings.
- Investigate the effectiveness of new word embedding techniques, namely ELMo (Peters et al., 2018) for the task of automated essay scoring.

1.1. Related Work

Given the importance of the task, many solutions have been proposed to address this problem. Some early approaches are: logistic regression over hand-crafted features, latent semantic analysis, sentence level score prediction, naive Bayes (using different distributions to model the data), voting algorithms, learning to rank. More recently, successful approaches rely more consistently on the use of Deep Learning. Long Short-Term Memory networks (LSTMs) (Hochreiter & Schmidhuber, 1997) are particularly useful for Natural Language Processing due to their ability to exploit long term dependencies between words and hence encode information about the grammatical structure of the text. Bi-directional LSTMs (Graves, 2011) extend the approach by processing the text both forwards and backwards. The current state of the art is achieved in (Alikaniotis et al., 2016), by combining LSTMs and task-specific word embeddings that encode both semantic meaning of words and extra information on which the essay score depend. A different approach to embeddings was proposed in (Cozma et al., 2018), where pre-trained word embeddings are clustered and used to generate a *super word embeddings*, that are consequently used to produce a document representation used for score prediction. Another approach that makes use of LSTMs is (Tay et al., 2017), where the model also learns the relationships between hidden states of the network and uses them as additional features for prediction.

[1] https://www.kaggle.com/c/asap-aes/leaderboard

2. Dataset and Methodology

In the following section we describe the pre-processing procedure, the models choices and the performed experiments.

2.0.1. Dataset

The dataset we used is the training data originally published by the Hewlett foundation for the 2012 Kaggle competition. The aggregated essay collection is referred to as the Automatic Student Assessment Prize (ASAP) Automated Essay Scoring (AES) dataset. It contains eight different essays categories (different prompts for each) for a total of 12,976 essays. Each essay is scored by two raters on a varying score scale (depending on the prompt). However, if the two raters disagree on the grade by a significant amount, then a third rater is brought in to evaluate the essay and the final grade assigned is that of the third rater. The final score either is computed from the first two raters (usually average or sum) or is the score given by the third grader, if there is one. Unfortunately, the ground truth for the original test dataset was not released by the organizer of the competition. Thus, we use the train, validation, and test splits as proposed in the paper *Automatic Text Scoring Using Neural Networks* (Alikaniotis et al., 2016) to separate the data-sets so as to be able to compare our results.

Finally, in order to ease the training, we normalize the scores to be in the interval $[0, 1]$ during training and de-normalize them for evaluation. Since essay sets have very different ranges of scores (for instance one has a range of $[0, 4]$ while another has scores in $[0, 60]$), normalizing the scores with respect to their range should make the task easier than directly predicting the final score. The ids of the essay sets are also provided with the essays also in the test set of the original competition, therefore we to use this information in our models in order to normalize and denormalize in the appropriate range. Note, however, that (Alikaniotis et al., 2016) did not perform this normalization.

2.0.2. Misspelled words

One of the domain specific problems of Automated Essay Scoring is misspelled words. Given the education level of who wrote the essays (middle school), mistakes are rather frequent in our dataset. This can be a problem when training a model, since it becomes harder to represent misspelled words with embeddings that contain both the semantic information about the correct word and the score-specific information about the spelling mistake. Moreover, any word can be misspelled in a number of ways. This raises other issues, for instance whether each misspelling of the same word should have a different embedding, or whether there is ambiguity between two possible correct words for a misspelled one. There might also be difficulties during the training due to the low frequency of that specific typo in the dataset.

To address these issues, in this project we propose two approaches. In both cases, during the data pre-processing, typos are identified and corrected through the SymSpell library[2]. To do this, for each misspelled word we consider the closest one among a dataset of 227,627 English words according to the editing distance. If the editing distance is larger than 2, the proposed correction is unlikely to be accurate. In this case we substitute the word with an `unrecognizable` token. We propose, however, two different ways to correct recognized misspelled words in our pre-processing:

- The word is replaced by the correctly spelled word, but an underscore symbol is prefixed. In this way, all typos of the same word are mapped to the same token and, so, the same embedding which will nevertheless differ from the one of the correct word. This allows for weighting differently the spelling mistakes in different words. This is referred as *Local* in the results section.

- The word is replaced with its correct version and the token `misspelled` is inserted after the word. Ideally, a LSTM network would be able to learn the relationship between the presence of this token and the overall score for the essay. However, this approach loses in generality as every misspelling in every word is treated in the same way. This is referred as *Global* in the results section.

2.1. Methodology

We perform a number of experiments to compare different models based on the evaluation metrics defined in 2.1.5.[3]

2.1.1. Baseline

The baseline we choose to compare against is an SVM model trained on per-document features (as in (Alikaniotis et al., 2016)). These features were extracted according to the method discussed in (Le & Mikolov, 2014). We refer to this as the `doc2vec` model. In our experiments we use the *Gensim*[4] implementation of `doc2vec`. For the baselines we did not normalize the target scores, as in (Alikaniotis et al., 2016).

2.1.2. Score-Specific Word Embeddings

The main contribution of (Alikaniotis et al., 2016) is the introduction of Score-Specific Word embeddings (SSWE). Their approach builds on the skip gram model

[2] https://github.com/mammothb/symspellpy
[3] Code is available at https://github.com/LindaPetrini/DL4NLT
[4] https://radimrehurek.com/gensim/

for `word2vec` (Mikolov et al., 2013). Each word is associated to a vector (embedding) that represents its semantic meaning and the embeddings are trained by means of a neural network whose target is to rank a word sequence (more precisely, a *9-gram* extracted from an essay) with respect to a number of noisy copies, where the central word has been changed. Moreover, this procedure was extended in SSWE by adding a new term in the loss function, that directly accounts for the score prediction.

In short, a new linear layer is introduced as a new head of the network used to train the word embeddings. This last output layer is trained to predict the score of the essay. The final loss is then taken to be a (weighted) sum of the loss coming from the skip gram procedure and the MSE between the predicted and real score. The intuition behind this choice is to make the embeddings more informative with respect to this specific task. As an example, very common words such as *and, but, is* would not benefit from this added specification, as they appear uniformly among differently scored essays. On the other side, we could expect words with similar semantic meaning and different refinement of vocabulary to be pulled further away by the added loss term. This is because, even if their semantic meaning is the same, their contribution to characterize good or bad essays is different.

2.1.3. LSTM/BLSTM

For all the other experiments, we consider LSTM models with different types of word embeddings. For each type of embedding, we experiment with both a uni-directional and a bi-directional LSTM. The latter model was introduced to address the fact that the interpretation of a word at a given timestep t might change once the model also observes a following word that might appear some timesteps later. We experiment with one and two layers networks. On the top of the recurrent network we put a linear layer performing a linear regression on the score (as done in (Alikaniotis et al., 2016)). We believe a linear regression in the output is more reasonable than a logistic regression, since a sigmoid function is likely to suppress more information on the extreme values. The hyper-parameters we choose to tune in our experiments are: *size of the embedding, size and number of hidden layers in LSTMs, dropout, batch size* and *learning rate*. In these experiments we compare embeddings trained from scratch against SSWE embeddings for the two different training procedures.

2.1.4. ELMo Embeddings + GRU

One issue with the SSWE approach is that unseen words are not mapped to any meaningful embeddings. An alternative approach to build word embeddings that can be extended to unseen words is proposed in (Peters et al., 2018). The intuitive idea behind ELMo embeddings can be understood comparing it to how stacked CNNs process data in a hierarchical manner (identifying lower level features of images in early layers, and more complex shapes in the last layers). This intuition suggests that, in the context of an LSTM with multiple layers, the activations at various layers carry meaningful information about the semantic meaning at lower levels and about the sentence structure at higher levels. The authors suggest to train a Bidirectional-LSTM with L layers using a task-specific objective and afterwards define the concatenation of the input and both the backward and forward activations for all layers in the LSTM as embeddings (for any other downstream task). In our experiments, we use a pre-trained model[5] to extract embeddings, which are subsequently fed to a bidirectional Gated Recurrent Unit networks (GRU) model.

The training procedure for ELMo results in high dimensional embeddings (1024 in the case of the pre-trained embeddings). In order to address the memory issues that arise from this fact, we decided to include a linear projection to a lower dimensional space of 256 as first layer. In addition, instead of LSTM networks, experiments with ELMo embeddings were performed with GRU (Cho et al., 2014). This type of network behaves similarly to a LSTM network, with the difference that the output gate is not present and hence GRU networks have less parameters. To ensure a fair comparison with SSWE, experiments were run with the same GRU architecture and SSWE embeddings/ELMo embeddings.

2.1.5. Evaluation Metrics

We evaluate our approaches based on the following metrics: Spearman's ρ, Pearson's r, Cohen's κ and Root Mean Square Error (*RMSE*).

The Cohen's κ coefficient with quadratic weights is the metric used in the original Kaggle competition to evaluate the winning algorithm on the unseen test data and it expresses a measure of the agreement between two raters on a classification task. The Spearman's ρ measures relationships in the rankings of two different variables. In the case of the AES dataset, this measures the agreeability between the predicted and the actual scores (i.e. the rankings) of the individual essays. Similar to Spearman's ranking coefficient, the Pearson r is a measure of linear correlation between two variables, without incorporating rankings. Measuring the performance of various techniques on the AES dataset with these metrics allows for a robust way to validate our result, given the complexity of a correct evaluation for this task.

Finally, we have used *RMSE* as our main measure. As suggested by (Alikaniotis et al., 2016), this measure, associated

[5] https://github.com/allenai/allennlp

model	Cohen's κ	Spearman's ρ	Pearson r	RMSE (Normalized)	RMSE
		Global			
doc2vec + SVM	0.808	0.820	0.887	—	4.246
LSTM	0.967	0.903	0.968	0.194	2.229
BLSTM	0.971	0.926	0.971	0.165	2.095
LSTM + SSWE	0.968	0.909	0.968	0.186	2.171
BLSTM + SSWE	0.969	0.913	0.969	0.182	2.136
		Local			
doc2vec + SVM	0.814	0.825	0.886	—	4.205
LSTM	0.968	0.920	0.968	0.174	2.183
BLSTM	0.969	0.922	0.970	0.173	2.213
LSTM + SSWE	0.968	0.922	0.968	0.174	2.186
BLSTM + SSWE	0.969	0.922	0.969	0.170	2.171
GRU + SSWE	0.962	0.904	0.965	0.192	2.301
(Alikaniotis et al., 2016)	0.96	0.91	0.96	—	2.4
GRU + ELMo	**0.974**	**0.942**	**0.981**	**0.147**	**1.855**

Table 1. Results on the test set according to different evaluation metrics.

with the correlations described above, is a better choice than Cohen's κ. This metric considers each score as an independent category and does not take neither the ordering of the scores nor the magnitude of their differences into account.

In our results we have considered both the *RMSE* computed on the de-normalized targets and on the normalized targets. The first one is provided for comparison with other results, but we believe the second one is a more reasonable choice. Since different essay-sets have different scales, errors on essays with a wider range of scores will bias the *RMSE* computed on the de-normalized targets towards them. Conversely, if the measure is computed on the normalized scores, each essay contributes equally to the result.

For this reason, we employed the *RMSE* on the normalized targets as our main measure and we used it for early-stopping during the training of our models.

3. Results and Discussion

We first performed a grid search on our LSTM and BLSTM models to determine the best hyper-parameters after 25 epochs. The hyper-parameters tested were: learning rate, number of layers, number of hidden units, dropout and types of embeddings. Adam is used as the chosen optimizer throughout our experiments because of its general success in similar tasks and affiliation to UVA. We found the lowest loss on the validation set for both the LSTM and BLSTM models using a learning rate of $1e-4$, a single layer of 128 units and a dropout of 0.4. For the rest of our experiments we fix these hyper-parameters in order to control the effects of the different approaches.

The ELMo GRU network was constrained to a single set of hyperparameters due to computational constraints. As mentioned in Section 2.1.4, the pre-trained ELMo embeddings of size 1024 were projected to a size of 256 using a linear layer, due to memory constraints with the large sequence lengths of the individual essays. The GRU network was trained using a learning rate of $3e-4$ (using Adam), with a single bidirectional GRU layer with a hidden dimension of 100 units.

In Table 1 we report the performance of the different models on the test set according to the evaluation metrics described previously. We distinguish between the normalized RMSE and original RMSE. The former is used as training loss for our model, and it is computed on the targets and predictions normalized between 0 and 1. The latter RMSE is computed on the original grades (which have different ranges in different essay sets), and it is used to compare our model with the previous state of the art results (shown at the bottom of Table 1) which used a two layer BLSTM and their custom SSWE (Alikaniotis et al., 2016).

We notice how the ELMo + GRU model was the best model according to all of the metrics. This is most likely due to the semantic and contextual information contained within the ELMo embeddings and their high dimensionality (which enable them to encompass more information). Moreover, the ELMo embedding of a word is built from a character-level language model, which means that this approach can automatically deal with misspelled words. Also, when it is compared with the performance of the GRU model using SSWE as well as the LSTM or BLSTM models using SSWE, we can attribute most of gains to the ELMo embeddings layer. Regarding the SSWE and the randomly-initialized embeddings, their lower dimensionality and the smaller hyper-parameter tuning we could perform may have also contributed to poorer performance.

Regarding the two ways of pre-processing misspelled words, we predict to see two different ways in which they affect the behavior of our models. On one hand, the general misspelled token is more likely to converge to the expected negative value when influencing the final essay score. This is because having a unique token for spelling mistakes would result in it being very frequent in the dataset and, probably, a good indicator of the total number of errors in each essay. On the other hand, a unified misspelling token would lose the semantic information about which word was misspelled. This information, captured when using the local misspelling token instead, can be used for example to discriminate between errors in common words (which could be good indicators of low grades) and the ones in infrequent words (possibly less relevant). Considering our results, we notice how the global misspelled token corresponds to mostly better scores for some of our models. This is most likely due to the fact that our models struggle to accurately learn different embeddings for the misspelled version of every word. Conversely, the unified misspelling token is both more informative and easier to learn thanks to its frequency in the training set.

3.1. Visualizing Distance between Embeddings

	good	_good	bad	_bad
good	1	-	-	-
_good	-0.029	1	-	-
bad	+0.034	-0.021	1	-
_bad	-0.145	+0.029	-0.046	1

Table 2. Cosine distance between embeddings using local misspelled token.

As discussed in Sec. 2.1.2, SSWE learned for our task are able to capture both a semantic and a score-specific information for words in the essays. In particular, we expect that part of the latent representation is responsible for describing the contribution of the word towards the grade (which depends on features like correct spelling and lexical level). The other part of the representation depends on the semantic meaning of the words itself, as in classic versions of embeddings. To better understand the difference and similarities between words, we can compute their cosine distance. For example, we expect a certain degree of similarity between two words that are semantically similar but also between ones that are both correct or both wrong. On the other way, we should find a higher distance (negative cosine similarity), between words with opposite meaning and where one is correct and one is misspelled. In table 2 we report the cosine similarity between some words in our vocabulary and we notice how the scores are in line with our observations about the embeddings. Notice how pairs of words matching in "spelling correctness" (like *good* - *bad* and *_good* - *_bad*) have a positive cosine similarity. On the other side, when words don't match neither in correctness nor in semantics (for example in *good* - *_bad*) the similarity is negative. Overall, as we motivated in the previous paragraph, representations with local misspelling tokens are not as stable and accurate as commonly adopted embeddings like word2vec or Glove. This is mostly due to the rarity of misspelled words in the training set, and the global misspelled token was designed to solve this issue.

4. Conclusion and Future Work

We have successfully replicated state of the art results from (Alikaniotis et al., 2016) using the SSWE embeddings and evaluated using different approaches to pre-processing and word embeddings. We are able to beat its best results by using the new ELMo embeddings, obtaining a final RMSE of 1.855. Even though the other metrics did not improve by significant amounts, we maintain that the RMSE is a better score by which to measure our model. It is possible, however, that the ELMo embeddings could have performed even better, had we unfrozen the embedding layer so as to fine tune it on our dataset or even had the time to explore the hyper-parameter space. We were unable to do so due to computing and time constraints and so future works could still consider this approach if additional resources are available.

Another possible line for future work would be the explainability of the model prediction. (Alikaniotis et al., 2016) propose a visualization tool that highlights which words had the most influence in the score decision, even though they also point out that their approach is based on gradients, that are calculated at the end of the sentence and are hence unable to distinguish between multiple appearances of a particular word. Such explainability would be useful in preventing and detecting adversarial inputs or essays which purposefully try to fool an automatic scorer such as the one proposed here.

Finally throughout this experiment we were able to use normalization and denormalization on the essay ranges because the essay set was given to us. If our model however came across an essay set not given in the training dataset, it would probably not be as effective. Future works could consider approaching the problem of prompt agnostic essay scoring, where regardless of the essay set or prompt the work is judged based on the level of writing. This hopefully offers a more robust approach for real world applications and remains a possible question for future works.

References

Alikaniotis, D., Yannakoudakis, H., and Rei, M. Automatic text scoring using neural networks. *CoRR*, abs/1606.04289, 2016. URL http://arxiv.org/abs/1606.04289.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Cozma, M., Butnaru, A. M., and Ionescu, R. T. Automated essay scoring with string kernels and word embeddings. *CoRR*, abs/1804.07954, 2018. URL http://arxiv.org/abs/1804.07954.

Graves, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. 2011.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Le, Q. and Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1188–II–1196. JMLR.org, 2014. URL http://dl.acm.org/citation.cfm?id=3044805.3045025.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL http://arxiv.org/abs/1802.05365.

Tay, Y., Phan, M. C., Tuan, L. A., and Hui, S. C. Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring. *CoRR*, abs/1711.04981, 2017. URL http://arxiv.org/abs/1711.04981.