Inject Rubrics into Short Answer Grading System

Tianqi Wang[1,3] **Naoya Inoue**[1,3] **Hiroki Ouchi**[3] **Tomoya Mizumoto**[2,3] **Kentaro Inui**[1,3]

[1] Tohoku University [2] Future Corporation [3] RIKEN Center for Advanced Intelligence Project

{outenki,naoya-i,inui}@ecei.tohoku.ac.jp
hiroki.ouchi@riken.jp
t.mizumoto.yb@future.co.jp

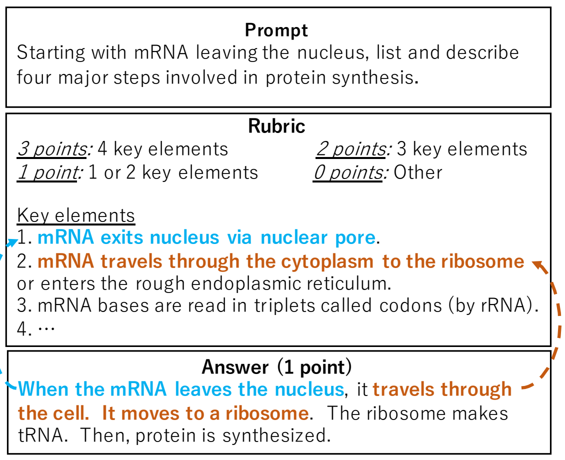
Abstract

Short Answer Grading (SAG) is a task of scoring students' answers in examinations. Most existing SAG systems predict scores based only on the answers, including the model (Riordan et al., 2017) used as baseline in this paper, which gives the-state-of-the-art performance. But they ignore important evaluation criteria such as rubrics, which play a crucial role for evaluating answers in real-world situations. In this paper, we present a method to inject information from rubrics into SAG systems. We implement our approach on top of word-level attention mechanism to introduce the rubric information, in order to locate information in each answer that are highly related to the score. Our experimental results demonstrate that injecting rubric information effectively contributes to the performance improvement and that our proposed model outperforms the state-of-the-art SAG model on the widely used ASAP-SAS dataset under low-resource settings.

1 Introduction

Short Answer Grading (SAG) is the task of automatically evaluating the correctness of students' answers to a given prompt in an examination (Mohler et al., 2011). It would be beneficial particularly in an educational context where teachers' availability is limited (Mohler and Mihalcea, 2009). Motivated by this background, SAG has been studied mainly with machine learning-based approaches, where the task is considered as inducing a regression model from a given set of manually scored sample answers (i.e., training instances). As observed in a variety of other NLP tasks, recently proposed neural models have been yielding strong results (Riordan et al., 2017).

In general, a prompt is provided along with a scoring rubric. Figure 1 shows a typical example. Students are required to answer the steps involved in protein synthesis. Each answer is scored based on a rubric, which contains several scoring criteria called *key elements*. Each of them stipulates different aspects of the conditions for an answer to gain a score. Based on the number of the key elements mentioned in an answer, its final score is determined. In Figure 1, the answer mentions two key elements, so it gains 1 point. Thus, rubrics and key elements play an essential role in SAG. Few previous studies, however, use information from rubrics for SAG.



Prompt
Starting with mRNA leaving the nucleus, list and describe four major steps involved in protein synthesis.

Rubric
3 points: 4 key elements
2 points: 3 key elements
1 point: 1 or 2 key elements
0 points: Other

Key elements
1. mRNA exits nucleus via nuclear pore.
2. mRNA travels through the cytoplasm to the ribosome or enters the rough endoplasmic reticulum.
3. mRNA bases are read in triplets called codons (by rRNA).
4. ⋯

Answer (1 point)
When the mRNA leaves the nucleus, it travels through the cell. It moves to a ribosome. The ribosome makes tRNA. Then, protein is synthesized.

Figure 1: Example prompt and rubric from the ASAP-SAS dataset.

In this paper, we present a method to incorporate rubric information into neural SAG models. Our idea is to enable neural models to capture alignments between an answer and each key element. Specifically, we use a word-level attention mechanism to compute alignments and generate an attentional feature vector for each pair of an answer and a key element.

The contributions of this study is summarized as follows:

- This is the first study that explores how to incorporate rubric information into neural SAG models.

- We propose a general framework to extend existing neural SAG models with a component for exploiting rubric information.
- Our empirical evaluation shows that our proposed model achieves a significant performance improvement particularly in low-resource settings.

2 Related Work

A lot of existing SAG studies have a main interest in exploring better representations of answers and similarity measures between student answers and reference answers. A wide variety of methods have been explored so far, ranging from Latent Semantic Analysis (LSA) (Mohler et al., 2011), edit distance-based similarity, and knowledge-based similarity using WordNet (Pedersen et al., 2004) (Magooda et al., 2016) to word embedding-based similarity (Sultan et al., 2016). Recently, Riordan et al. (2017) report that neural network-based feature representation learning (Taghipour and Ng, 2016) is effective for SAG.

In contrast to the popularity of learning answer representations, the use of rubric information for SAG has been gained little attention so far. In Sakaguchi et al. (2015), the authors compute similarities, such as BLEU (Papineni et al., 2002), between an answer and each key element in a rubric, and use them as features in a support vector regression (SVR) model. Ramachandran et al. (2015). Ramachandran et al. (2015) generates text patterns from top answers and rubrics, and reports the automatically generated pattern performances better than manually generated regex pattern. Nevertheless, it still remains an open issue (i) whether a rubric is effective or not even in the context of a neural representation learning paradigm (Riordan et al., 2017), and (ii) what kinds of neural architectures should be employed for the efficient use of rubrics.

Another issue in SAG is on low-resource settings. Heilman and Madnani (2015) investigate the importance of the training data size on non-neural SAG models with discrete features. Horbach and Palmer (2016) show that active learning is effective for increasing useful training instances. This is orthogonal to our approach: combining active learning with our rubric-aware SAG model is an interesting future direction.

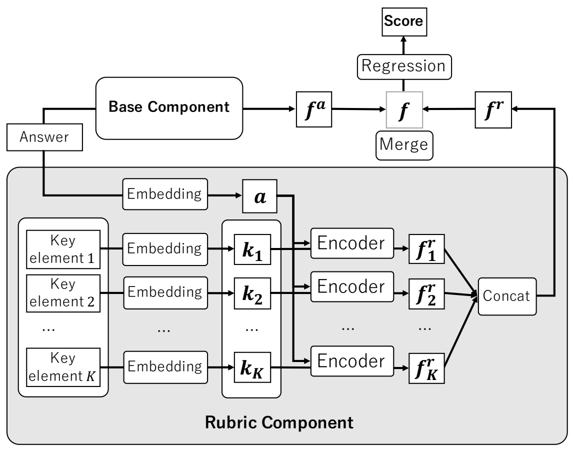


Figure 2: The proposed rubric-aware SAG architecture, consisting of base component and rubric component.

3 Proposed model

3.1 Overall architecture

Figure 2 illustrates our proposed model, which consists of (i) *base component* and (ii) *rubric component*.

We assume the base component encodes an answer into a feature vector $\boldsymbol{f}^a$. We also assume that a given rubric stipulates a set of key elements in natural language. We build a *rubric component* to encode rubric information, based on the relevance between the answer a and each key element $k \in \{k_1, k_2, \cdots, k_K\}$ provided in the rubric.

The rubric component first encodes each key element that consists of m words, $k = (w_1, w_2, \cdots, w_m)$, into its feature vector $\boldsymbol{k}$ and the answer a into $\boldsymbol{a}$. Then, it computes the relevance between the given answer a and each key element $k \in \{k_1, k_2, \cdots, k_K\}$ using a word-level attention mechanism, and generates attentional feature vectors $\boldsymbol{f}_1^r, \cdots, \boldsymbol{f}_K^r$, which represent the aggregated information of each key element. A rubric feature $\boldsymbol{f}^r$ is generated based on the obtained K attentional feature vectors. Finally, $\boldsymbol{f}^a$ and $\boldsymbol{f}^r$ are merged into one vector $\boldsymbol{f}$, which is used for scoring:

$$\mathrm{score}(a) = \beta\, \mathrm{sigmoid}(\boldsymbol{w} \cdot \boldsymbol{f} + b), \qquad (1)$$

where $\boldsymbol{w}$ is a parameter vector, β is a prompt-specific scaling constant, and b is a bias term.

Note that the model does not require explicit annotation of key elements on the training answer samples because the model implicitly estimates which key elements are included in each student answer in the course of training. It is also

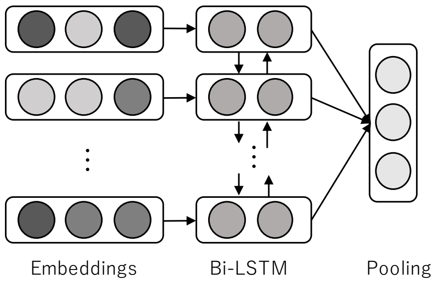


Figure 3: The base component.

important to note that our framework is encoder-agnostic; namely, any answer encoder that produces a fixed-length feature vector can be used as the base component.

3.2 Base component

As the base component, we employ the neural SAG model proposed by Riordan et al. (2017), which is the state-of-the-art SAG system among published methods. As shown in Figure 3, this model consists of three layers, namely (i) the embedding layer, (ii) the BiLSTM (bidirectional Long Short-Term Memory (Schuster and Paliwal, 1997)) layer and (iii) the pooling layer.

Given an answer $a = (w_1, w_2, ..., w_n)$, the embedding layer outputs a vector $\boldsymbol{e}^a_i \in \mathbb{R}^d$ for each word w_i. Taking a sequence of these vectors $(\boldsymbol{e}^a_1, \boldsymbol{e}^a_2, \cdots, \boldsymbol{e}^a_n)$ as input, the BiLSTM layer then produces a contextualized vector $\boldsymbol{f}^a_i = [\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}_i}]$ for each word, where $\overrightarrow{\boldsymbol{h}_i} \in \mathbb{R}^h$, $\overleftarrow{\boldsymbol{h}_i} \in \mathbb{R}^h$ are the hidden states of the forward and backward LSTM, respectively. Finally, the pooling layer averages these contextualized vectors to obtain a feature vector for the answer as follows:

$$\boldsymbol{f}^a = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{f}^a_i \tag{2}$$

3.3 Rubric component

Inspired by Chen et al. (2016), we compute word-level attention between each key element and an given answer as illustrated in Figure 4. The rubric component captures how relevant a key element is to the given answer in this way.

Given word embedding sequences of an answer $(\boldsymbol{e}^a_1, \boldsymbol{e}^a_2, \cdots, \boldsymbol{e}^a_n)$ and a key element $(\boldsymbol{e}^k_1, \boldsymbol{e}^k_2, \cdots, \boldsymbol{e}^k_m)$, the rubric component first calculates the word-level attention between $\boldsymbol{e}^k_i$ and $\boldsymbol{e}^a_j$:

- Calculate the inner-products between word embeddings from the answer and key element: $z_{i,j} = \boldsymbol{e}^k_i \cdot \boldsymbol{e}^a_j$
- Calculate softmax of $z_{i,j}$ over the rows and columns respectively:

$$\boldsymbol{\alpha}^k_i = \text{softmax}(z_{i,1}, z_{i,2}, \cdots, z_{i,n}) \tag{3}$$
$$\boldsymbol{\alpha}^a_j = \text{softmax}(z_{1,j}, z_{2,j}, \cdots, z_{m,j}) \tag{4}$$

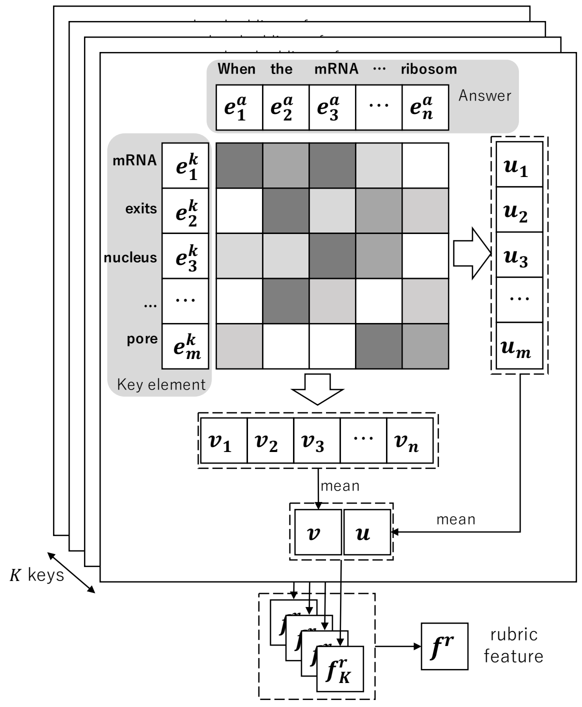


Figure 4: Calculation of rubric feature based on word-level attention. Words of answers lay on x-axes, and words of the key elements lay on y-axes.

Note that $\boldsymbol{\alpha}^k_i \in \mathbb{R}^n$ stands for the attention from the i-th word of a key element to each word in the answer a. Similarly, $\boldsymbol{\alpha}^a_j \in \mathbb{R}^m$ stands for the attention from the j-th word of answer to each word in the key element k.

Next, attentional vectors of key-to-answer ($\boldsymbol{v}$) and answer-to-key ($\boldsymbol{u}$) are calculated by the sum of word embeddings weighted by $\boldsymbol{\alpha}^a$ and $\boldsymbol{\alpha}^k$ as follows:

$$\boldsymbol{u} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} \alpha^k_{i,j}\boldsymbol{e}^a_j \tag{5}$$

$$\boldsymbol{v} = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{m} \alpha^a_{j,i}\boldsymbol{e}^k_i \tag{6}$$

Intuitively, vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are the aggregation of answer tokens that are highly relevant to a key element, and tokens in the key elements that are highly relevant to the answer. We then concatenate $\boldsymbol{u}, \boldsymbol{v}$ to obtain a feature vector for the key element.

Finally, we generate feature vectors $\boldsymbol{f}_1^r, \cdots, \boldsymbol{f}_K^r$ for all key elements in this manner, and then generate rubric feature $\boldsymbol{f}^r$ based on them.

3.4 Merge features

We introduce two methodologies to merge $\boldsymbol{f}^a$ and $\boldsymbol{f}^r$ into one single feature $\boldsymbol{f}$.

Concatenation We concatenate $\boldsymbol{f}^a$ and $\boldsymbol{f}^r$:

$$\boldsymbol{f}^r = [\boldsymbol{f}_1^r; \boldsymbol{f}_2^r; ...; \boldsymbol{f}_K^r] \quad (7)$$

$$\boldsymbol{f} = [\boldsymbol{f}^a; \boldsymbol{f}^r], \boldsymbol{f} \in \mathbb{R}^{2h+2dK} \quad (8)$$

In this case, we expect the regression layer learns weights for the two feature space at the same time.

Weighted Sum Besides, we introduce a trainable parameter λ, which represents the influence of the rubric component. We then generate a rubric-aware answer feature as follows:

$$\boldsymbol{f}^r = \frac{1}{K}\sum_{i=1}^{K} \boldsymbol{f}_i^r \quad (9)$$

$$\boldsymbol{f} = \lambda \boldsymbol{f}^a + (1-\lambda)(\boldsymbol{f}^r M), \boldsymbol{f} \in \mathbb{R}^{2h} \quad (10)$$

where $M \in \mathbb{R}^{2d \times 2h}$ is a transformation matrix to learn, projecting $\boldsymbol{f}^r$ to the space of $\boldsymbol{f}^a$. To reduce parameters to learn, we compute $\boldsymbol{f}^r$ by average instead of concatenation. λ is initialized with 0.5 in our experiments.

Finally, the answer a is scored as follows: $\text{score}(a) = \beta\text{sigmoid}(\boldsymbol{w} \cdot \boldsymbol{f} + b)$, where $\boldsymbol{w} \in \mathbb{R}^{2h+2dK}$ (or $\boldsymbol{w} \in \mathbb{R}^{2h}$ for 'weighted sum' strategy) is a model parameter, β is a prompt-specific scaling constant, and b is a bias term.

4 Experiments

4.1 Settings

We apply the proposed model on a widely-used, rubric-rich ASAP-SAS dataset[2], which includes 10 prompts, with 2,226 answers for each prompt on average, including around 1,704 training data and 522 test data. In this paper, we choose the prompts 1, 2, 5, 6 and 10, where key elements are explicitly provided in their rubric, and we randomly take 20% of answers from the training data as the development data. On average, we have 1,308 answers as training data, 327 answers as development data and 545 answers as test data.

[2]https://www.kaggle.com/c/asap-sas/data

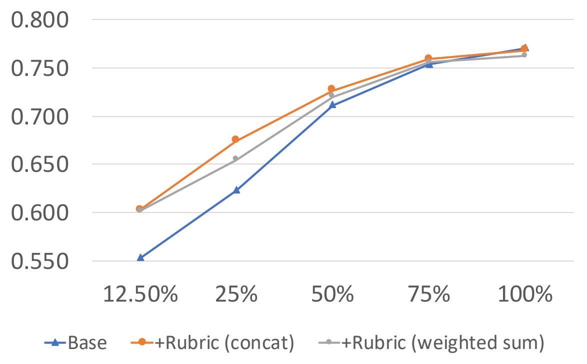


Figure 5: Mean performance across different size of training data. The performance is shown in average QWK over all prompts.

For both the base and rubric components, we use 300-dimensional GloVe embeddings pre-trained on Wikipedia and Gigaword5 (Pennington et al., 2014) to initialize the word embedding layer ($d = 300$), and update them during the training phase.

For the bi-LSTM layer of base component, we set $h = 256$, set the dropout probability for linear transformation as 0.5, and set the dropout probability for recurrent state as 0.1, following the setting of (Riordan et al., 2017).

Mean Squared Error (MSE) is used as the loss function, and optimized by RMSprop optimizer with a learning rate of 0.001. The batch size is set to 32.

The model is trained on each prompt. We first train the base component, then fix the base component and train the whole model, and run the training phase for 50 epochs to choose the best model on the development data. For each prompt, we repeat the experiments 5 times with different random seeds from 0 to 4 for initialization, and evaluate the model with Quadratic Weighted Kappa (QWK) independently, then we take average QWK over all the random seeds as the final performance of the model on the corresponding prompt.

To evaluate the robustness of our model in low-resource settings, we train our model on various sizes of the training data (12.5%, 25%, 50%, 75% and 100%).

4.2 Results

The experimental results under different sizes of training data are shown in Figure 5. The performance of the base component ('Base') with 100% training data was 0.770, which is comparable to

Table 1: Performance across different sizes of training data. * indicates a statistically significant improvement by Wilcoxon's signed-rank test ($p < 0.05$).[1] 'B' indicates baseline, and '+R' indicates our model (base component + rubric component)

(a) Merge base feature and rubric feature by concatenation.

Prompt		1	2	5	6	10	mean
12.5%	B	**.588**	.331	.617	.611	**.618**	.553
	+R	.579*	**.408***	**.723***	**.721***	.582*	**.603**
		-.009	+.077.	+.107.	+.110	-.036	+.050
25%	B	**.656**	.473	.641	.627	**.719**	.623
	+R	.652*	**.544***	**.719***	**.743***	.712*	**.674**
		-.004	+.072	+.078	+.116	-.007	+.051
50%	B	**.748**	.637	.748	.718	**.705**	.711
	+R	.745*	**.641**	**.790***	**.756***	.700*	**.726**
		-.003	+.004	+.042	+.038	-.005	+.015
75%	B	.776	**.700**	.798	**.748**	.744	.753
	+R	**.780***	.696	**.803***	.759*	**.755***	**.759**
		+.004	-.004	+.005	-.011	+.011	+.006
100%	B	**.792**	.713	**.804**	.788	**.753**	**.770**
	+R	.784	**.714***	.797*	**.793***	.751	.768
		-.008	+.001	-.008	+.005	-.002	-.002

(b) Merge base feature and rubric feature by weighted sum.

Prompt		1	2	5	6	10	mean
12.5%	B	.588	.331	.617	.611	**.618**	.553
	+R	**.599***	**.424***	**.689***	**.679***	.617	**.602**
		+.012	+.093	+.073	+.068	-.001	+.049
25%	B	.656	.473	.641	.627	**.719**	.623
	+R	**.661***	**.529***	**.687***	**.697***	.698	**.654**
		+.005	+.056	+.046	+.070	-.020	+.031
50%	B	.748	.637	.748	.718	**.705**	.711
	+R	**.747***	**.643**	**.784***	**.723***	.702*	**.720**
		+.000	+.006	+.036	+.006	-.004	+.009
75%	B	.776	.700	**.798**	.748	.744	.753
	+R	**.783***	**.704**	.787*	**.750***	**.784***	**.762**
		+.007	+.004	-.010	+.002	+.040	+.009
100%	B	**.792**	**.713**	**.804**	.788	**.753**	**.770**
	+R	.789	.695*	.786*	**.790***	.748	.762
		-.003	-.018	-.018	+.002	-.005	-.008

the best performance of QWK 0.773 on the corresponding 5 prompts reported in (Riordan et al., 2017). This indicates that we successfully replicated their best performing model.

Also, by adding the rubric component ('+Rubric'), the performance was improved especially when less training data is available. This suggests that the rubric component compensates the lack of training data. This is consistent with (Sakaguchi et al., 2015), a non-neural counter-part of our study.

Performance on each prompt is shown in Table 1. The results indicate that the benefit we obtain from rubric component varies with prompts. For instance, we achieve more improvements on prompt 2, 5 and 6 compared to the others. One of the reasons is that the rubrics vary on prompts. For instance in prompt 5 and 6, all key elements with which an answer can get points are listed, while in prompt 10 only four example answers are provided.

4.3 Analysis

Contribution of components Figure 5 demonstrates that when trained with full training data, our rubric-aware model ('+Rubric') achieved a comparable performance to the base component. To reveal reasons for this, we conduct two analyses.

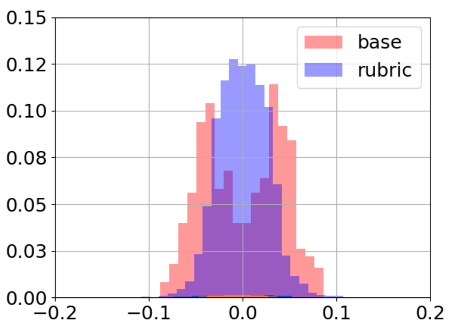
First, for '+Rubric (concat)', we investigate the distribution of the learned weights of regression layer corresponding to the base and rubric components following the idea from Meftah et al. (2019). The distribution is shown in Figure 6. When the model was trained on 100% training data, the weights for the rubric component were closer to 0, while the weights for the base component were more dispersed (Figure 6b), compared to the distribution for 12.5% training data (Figure 6a).

Second, for '+Rubric (weighted sum)', we plot the values of trained λ in Figure 7, representing the weights of base component. Generally, the values of λ grow with data size, which is consistent with Figure 6. This means that as training data increases, the rubric component makes less contribution to the performance, thus little improvement was obtained from the rubric component. Addressing this issue is an interesting direction of our future research.

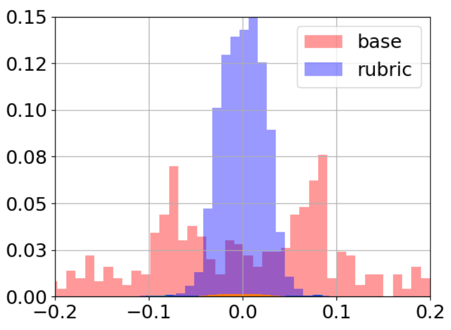
Word-level attention To get further insights on the rubric component, we analyzed 1-point answers in the test set. We show two typical examples of 1-point answers in Table 2, where each answer is graded (a) correctly and (b) incorrectly by the system trained with 12.5% training data. Both the two answers are graded incorrectly as 0-points

Table 2: Instance 1-point answers.

ID	Answer	Score	Base	+Rubric
13278	the mRNA gets transcribed, it leaves the nucleus by the ribosomes, then it travels on the Endocplasmic reticulum, and goes to the lysomes and gets translated to proteins.	1	0	1
13174	mRNA leaves the nulceus, travels to the endoplasmic reticulum, then to cell membrane and exits the cell	1	0	0



(a) Training data size: 12.5%



(b) Training data size: 100%

Figure 6: Value distribution of learned weights of regression layer corresponding to base and rubric component for prompt 5.

by the baseline.

The corresponding prompt and its rubric are shown in Figure 1. Both the answers only contain the first key element provided in rubric. The first answer is graded as 1-point correctly while the second is graded as 0-points.

The word-level attention shown in Figure 8 indicates how the proposed model identified the relevancy of the answer towards the key element. Figure 8a shows that the model successfully found words and phrases most related to the key element, helping the model improve the performance. On the other hand, Figure 8b shows that the model incorrectly aligned words in the answer and key element. Specifically, the model aligned *exists* in

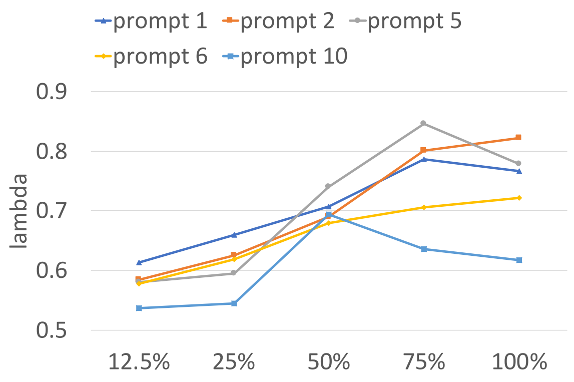


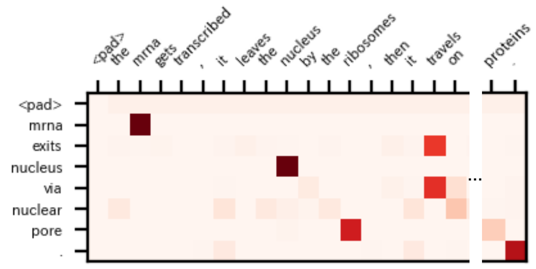
Figure 7: Values of λ trained by various of data size.

the answer with *exists* in the key element. However, these two verbs should *not* be aligned because their objects are different from each other (i.e. *the cell* in the answer, but *nucleus* in the key element). Because the attention is calculated on word-level, the model tends to simply find similar words that appear in the key element, ignoring the context around the words.

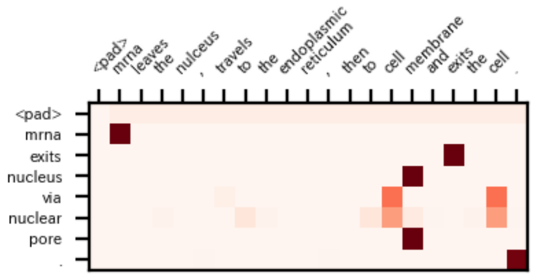
5 Conclusion

Rubrics play a crucial role for SAG but have attracted little attention in the SAG community. In this paper, we present an approach for incorporating rubrics into neural SAG models. We replicated a state-of-the-art neural SAG model as the base component, and injected rubrics (key elements) through the rubric component as an extension. In the low-resource setting where the base component had difficulty learning key elements directly from answers, our experimental results showed that the rubric component significantly improved the performance of SAG. When all training data was used, the rubric component did not have a negative effect on the overall performance.

Overall, the proposed model still has much room for improvement. For example, the approach to calculate the alignment between answers and key elements could be improved by taking context into account, instead of using word-level at-



(a) Attention for answer 13278



(b) Attention for answer 13174

Figure 8: Word-level attention. Words of answers lay on x-axes, and words of the key element lay on y-axes.

tention. Moreover, other types of rubrics could be explored in the SAG task, especially for prompts where key elements are not provided explicitly. We also expect to obtain a further improvement when full training data is available, by increasing the weights of rubric component feature, as discussed in Figure 6. Beyond SAG, we would like to explore approaches for generating feedback based on the computed attention to key elements.

References

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Michael Heilman and Nitin Madnani. 2015. The impact of training data on automated short answer scoring performance. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 81–85.

Andrea Horbach and Alexis Palmer. 2016. Investigating active learning for short-answer scoring. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 301–311.

Ahmed Ezzat Magooda, Mohamed A Zahran, Mohsen Rashwan, Hazem M Raafat, and Magda B Fayek. 2016. Vector based techniques for short answer grading. In *FLAIRS Conference*, pages 238–243.

Sara Meftah, Youssef Tamaazousti, Nasredine Semmar, Hassane Essafi, and Fatiha Sadat. 2019. Joint learning of pre-trained and random units for domain adaptation in part-of-speech tagging. *arXiv preprint arXiv:1904.03595*.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the ACL*, pages 752–762. Association for Computational Linguistics.

Michael Mohler and Rada Mihalcea. 2009. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the EACL*, pages 567–575. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

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

Lakshmi Ramachandran, Jian Cheng, and Peter Foltz. 2015. 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*, pages 97–106.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. 2017. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168.

Keisuke Sakaguchi, Michael Heilman, and Nitin Madnani. 2015. Effective Feature Integration for Automated Short Answer Scoring. In *Proceedings of NAACL*, pages 1049–1054.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. 2016. Fast and easy short answer grading with high accuracy. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891.