1 Introduction

Formative assessments place a primary focus on providing qualitative feedback for both teachers and students. Such feedback not only allows teachers to monitor their students' progress and evaluate the effectiveness of their own instruction, but also provides insight as to how the teaching and learning process can be optimised going forward [1]. Formative assessments have been shown to produce a much greater impact on student learning as compared to summative assessments when administered frequently and in a timely manner [2]. Hence, formative assessments clearly play a beneficial role in today's modern classrooms.

However, grading certain forms of formative assessments such as short-answer questions is a complex task that requires significant human intervention and input. Furthermore, valuable feedback needs to be produced that is beneficial to both students and teachers, which is a time-consuming and tedious task for teachers. The automation of this task would allow open-ended formative assessments to be administered more frequently and easily, greatly easing the workload of teachers and allowing both students and teachers to reap their benefits through the frequent and timely report of progress that they would receive.

The advent of online learning platforms has made administering formative assessments more convenient and more importantly, has allowed for the digitalisation of student answers which opens up opportunities for automatic grading that are explored in the following sections.

2 Literature Review

Much progress has been made in developing different approaches to the semi-automatic grading of open-ended questions. Larkey [3] showed that a system utilising multiple linear regression with a combination of hand-crafted features and probabilistic models, such as Bayesian classifiers and k-nearest-neighbour algorithms, can give results comparable with human graders. Rosé, Roque, Bhembe, *et al.* [4] developed CarmelTC, a rule-based approach, combining both features obtained from deep syntactic functional analyses of texts and a "bag-of-words" classification extracted from Rainbow Naive Bayes, outperforming Latent Semantic Analysis, Rainbow Naive Bayes and a purely symbolic approach. Nguyen and Dery [5] used a deep learning approach for essay grading while Zhang, Shah, and Chi [6] used hand-picked features with linear regression on the same dataset. The former performed significantly better, demonstrating the superiority of deep learning approaches over classic Natural Language Processing (NLP) methods.

However, much of the literature focuses on grading summative assessments; the only objective is to produce an accurate quantitative prediction of the score. This is arguably easier and less complex than grading formative assessments which involves producing feedback for both teachers and students on top of merely providing a quantitative score. Thus, there is a need for new semi-automatic grading architectures that can fulfil the aims of formative assessments.

3 Methodology

3.1 Datasets

An online physics quiz was conducted on 292 Raffles Institution Secondary 2 students, comprising two short-answer physics question on the topic of thermodynamics. Thermodynamics was chosen as it is an important topic in mainstream schools which students have many misconceptions about, and this topic is typically tested using qualitative short-answer questions. The first question is a simple recall question, while the second question is a more complex application question, designed to have more variance in answers. The answers were then graded by an entire level of 3 physics teachers based on a marking scheme comprising different marking points. Student answers were then pre-processed by tokenising and converting to lowercase while punctuation, non-alphabetical characters and stop words were also removed. The answers and scores from question 1 and question 2 are denoted dataset 1 and dataset 2 respectively. The datasets are listed in Appendix A.

3.2 Components of Architecture

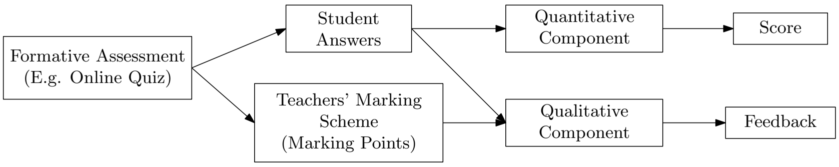


Figure 1: Outline of Architecture

We aim to design an architecture that automatically grades short-answer formative assessments by **(1)** providing an accurate quantitative score for student answers in order to expedite the marking process and **(2)** providing qualitative feedback to teachers and students in terms of the key areas of improvement. We evaluated different qualitative and quantitative models that attempt to achieve these two aims.

3.2.1 Quantitative Component

We propose a neural network model for the quantitative component. In order to design the best performing architecture, we propose different types of models below. Firstly, we compare the use of different word embeddings. Secondly, we fix our baseline model to be a simple feedforward neural network. Thirdly, we compare two types of Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks. Fourthly, we compare the use of bidirectional RNNs (BiRNNs) and their unidirectional variants. Finally, we propose an attention mechanism.

Word Embeddings. Within the neural networks, the answers are first represented in terms of word vectors that can encode meaningful semantic relationships between words. We compare 2 types of word embeddings - GloVe [7] and fastText [8], both of which are 300 dimensional.

Feedforward Neural Network (Baseline). We chose a simple feedforward neural network as our baseline. This network is trained on an "answer vector" for each answer, which is simply the sum of the word vectors for each word in the answer. While this approach does not preserve sequential information, word vectors have been shown to be able to meaningfully encode linearities such as *king - man + woman = queen* and *Paris - France + Italy = Rome* [9]. Thus, we hypothesise that the answer vectors will nevertheless be able to encode meaningful information about the answers to a certain degree. The architecture of the baseline model can be found in Appendix B.1 while that of the rest of the models can be found in Appendix B.2.

LSTM and GRU Networks. LSTM networks [10] preserve long-distance dependencies, making them ideal for processing text sequences [11]. On the other hand, GRU networks were developed as an alternative to LSTM networks and are computationally less complex [12], yet perform comparably to LSTM networks in sequence processing tasks [13]. We chose to work with RNNs as they have demonstrated excellent performance in sequence processing tasks, including sequence classification due to their ability to account for sequential information [14], making them applicable to our task of grading short-answer physics questions.

Bidirectional Recurrent Neural Networks. BiRNNs [15] such as BiLSTM networks aim to improve upon RNNs by processing the same input sequence twice, forwards and backwards. This allows the BiRNN to retain contextual information in both directions and has led to an improvement in performance in various sequence processing tasks [16], [17]. Hence, we compare the performance of BiLSTM and BiGRU models with that of regular LSTM and GRU models.

Attention Mechanism. Recently, the attention mechanism has been developed for sequence processing tasks such as machine translation and sequence classification [18], [19]. Since not all words in a sentence contribute equally to its meaning, attention is used to place more weight on more important words while placing less weight on less important words. We hypothesise that attention would be able to extract keywords from answers to aid in marking them more effectively, similar to how a teacher might mark answers. A detailed diagram showing the architecture of the attention layer on top of a BiRNN layer with an explanation can be found in Appendix B.4.

3.2.2 Qualitative Component

We propose a few different methods to give provide qualitative feedback to students and teachers using the marking schemes in our datasets by capturing common strengths and weaknesses in student answers. We mainly adapt and draw inspiration from classic probabilistic topic models such as Latent Dirichlet Allocation [20], which produces sparse and low-dimensional interpretations of topic memberships in documents. These are highly interpretable, allowing humans to gain high-level insight and intuition from them, which is especially necessary in the field of formative assessment where human understanding and feedback is key.

Vector Decomposition. Our datasets provide us with marking points and student answers. In order to manipulate these mathematically, a vector representation first needs to be defined for them. Motivated by the fact that word vectors can meaningfully encode linearities, as mentioned in Section 3.2.1, we initialise the marking point vectors to be the sum of the words in each marking point, so that they capture the meaning of the marking points: $\mathbf{t_k} = \sum_{i \in T_k} \mathbf{w_i}$, where $\mathbf{t_k}$ and T_k are the marking point vector and the set of words in the kth marking point out of n marking points respectively, and $\mathbf{w_i}$ is the word vector for the ith word. The GloVe word vectors are used for the purposes of the qualitative experiments.

A similarly suitable vector representation of student answers, is simply the sum of all the word vectors for each word in the answer: $\mathbf{a_j} = \sum_{i \in A_j} \mathbf{w_i}$, where $\mathbf{a_j}$ and A_j are the answer vector and the set of words in the jth answer respectively. However, an answer can also be represented as a complete or incomplete combination of the marking points.

Thus, we hypothesise that the answer vector $\mathbf{a_j}$ can be decomposed into a linear combination of marking point vectors:

$$\hat{\mathbf{a_j}} = \sum_{k=1}^{n} p_{jk}\mathbf{t_k} = \sum_{k=1}^{n} p_{jk} \sum_{i \in T_k} \mathbf{w_i} \tag{1}$$

where p_{jk} is the proportion of a marking point k in answer j and $p_{jk} \in [0, 1]$. $p_{jk} = 0$ would mean that a marking point is completely not within the student's answer, while $p_{jk} = 1$ would mean otherwise. Let $\mathbf{T}$ be the matrix whose columns are $\mathbf{t_1}, \dots, \mathbf{t_k}, \dots, \mathbf{t_n}$. Then $\hat{\mathbf{a_j}}$ is an element of the column space of $\mathbf{T}$, and can be equivalently expressed as $\hat{\mathbf{a_j}} = \mathbf{T}\mathbf{p_j}$. In order to then find an accurate decomposition of the answer vector, the optimal proportion vector $\hat{\mathbf{p_j}}$ can be found by optimising the mean squared error between $\hat{\mathbf{a_j}}$ and $\mathbf{a_j}$:

$$\hat{\mathbf{p_j}} = \arg\min_{\mathbf{p_j}} \|\hat{\mathbf{a_j}} - \mathbf{a_j}\|^2 \tag{2a}$$

$$= \arg\min_{\mathbf{p_j}} \|\mathbf{T}\sigma(\mathbf{p_j}) - \mathbf{a_j}\|^2 \tag{2b}$$

The logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is applied element-wise on $\mathbf{p_j}$ to enforce the constraint $0 \leq p_{jk} \leq 1$. A detailed description of the derivatives and optimisation process is given in Appendix C.2. The use of $\mathbf{p_j}$ to derive feedback is detailed in Section 4.3.2 of the paper.

Cosine Similarity. Under this method, the marking point proportions p_{jk} are defined as the cosine similarity [21] between the student answer vector and each marking point vector: $p_{jk} = \cos\theta_{jk} = \frac{\mathbf{a_j} \cdot \mathbf{t_k}}{\|\mathbf{a_j}\| \|\mathbf{t_k}\|}$. These p_{jk} values are collected into a vector, $\mathbf{p_j} = \left(p_{j0} \; p_{j1} \; \cdots \; p_{jk} \right)$, for each answer j.

Euclidean Distance. p_{jk} can also defined as the Euclidean distance [21] between the student answer vector and each marking point vector: $p_{jk} = \|\mathbf{a_j} - \mathbf{t_k}\|$ and $\mathbf{p_j} = \left(p_{j0} \; p_{j1} \; \cdots \; p_{jk} \right)$.

4 Data and Discussion

4.1 Evaluation Methodology

The datasets are split into training and validation sets, with a proportion of 0.8 and 0.2 respectively. Each model is evaluated on 3 metrics. Firstly, accuracy (Equation (D.1)) checks to see whether the model is able to give each student answer a correct score. Such a metric is useful for teachers and students as an ideal model would be able to mark all student answers exactly like a teacher would. Secondly, Categorical Cross-Entropy Loss (Equation (D.2)) is used to compare the different models as this the loss function that they were trained using. Lastly, weighted F1 score (Equation (D.5)) takes into consideration the unequal distribution of scores for the student answers as there are very few students who received full marks for each question. The definitions of these metrics can be found in Appendix D. The detailed results can be found under Appendix E and Appendix F for the quantitative and qualitative components respectively.

4.2 Evaluation of Quantitative Component

4.2.1 Comparison of Models

LSTM vs GRU. The GRU models performed better than the LSTM models on dataset 1, while the opposite is true for dataset 2. We hypothesise that this is the case because of the difference in complexity of the two questions. Since question 1 is less complex than question 2, the simpler GRU models with fewer parameters are likely to have less overfitting than the LSTM models. Likewise, the LSTM models are likely able to better capture the complexities of the second questions, giving them better performance. As shown by Weiss, Goldberg, and Yahav [22], LSTM networks are "strictly stronger" than GRU networks as they can easily perform unbounded counting, which GRU networks cannot, further supporting the hypothesis that LSTM networks are better suited for more complex questions. The difference in question complexity is likely also the reason for the disparity in performances between the two datasets for all models.

Attention. In addition, attention significantly improved the performance of the models, supporting our initial hypothesis. This is likely because physics answers are marked based on keywords, which is suited for attention as it is able to place weight on more important words and extract keywords. This is further corroborated by the attention weights extracted from the layer, visualised in Figure E.3 in Appendix E, which shows that attention places more weight on keywords such as “gases”, “solids”, and “states”.

Word Embeddings. GloVe performed better for the first dataset while fastText performed better for the second dataset. We hypothesise that this performance disparity could be to the fact that fastText was better able to capture the more complex nature of the second question, due to the "subword" information captured by fastText [8].

Baseline. Contrary to our initial hypothesis, the baseline performed surprisingly well and outperformed many of the more complex models, despite having a major disadvantage due to the fact that it does not preserve sequential information. We conjecture that this could be due to two possible reasons. The more complex models have a significantly greater number of parameters compared to the baseline. This, combined with the relatively small datasets, could have led to overfitting in these models [23], preventing them from generalising their classification task to the validation set as well as the baseline model, leading to the latter having a better performance. The baseline could have also outperformed the other models due to poor hyperparameter tuning in the more complex models, causing them to be stuck in poor local minima [24]. If this is the case, then there is a possibility that performance can be significantly improved with a more careful hyperparameter search.

4.2.2 Performance Across Different Data Environments

We explored how the number of training samples affects the models' performance. 42 student answers were put aside to be the validation set. The size of the training set was varied from 25 samples to 250 samples in increments of 25 samples and the performance of the model was monitored. The detailed results, which can be found in Appendix E.2, show that our best models perform well even in low data environments. Since one of the intended aims of our models is to reduce the marking load for teachers, this demonstrates their utility as teachers only need to mark a small proportion of the dataset for the models to perform well.

4.3 Evaluation of Qualitative Component

4.3.1 Comparison of Models

To test how accurate the set of $\mathbf{p_j}$ vectors generated by each qualitative model is, we trained a simple feedforward neural network (Appendix C.1) to predict the scores based on $\mathbf{p_j}$ as the input. This is based on the assumption that accurate proportions of each marking point should be correlated to the score of the answers, since these marking points are used by teachers to mark the answers. The results are summarised in Appendix F. It can be seen that the cosine similarity model clearly outperforms the rest and the results achieved by it are even comparable to our best quantitative models despite the relative simplicity of this approach. This could be due to the same reasons our baseline quantitative model outperformed more complex models, as highlighted in Section 4.2.1. This good performance also shows that the proportions generated are accurate.

4.3.2 Producing Feedback

The $\mathbf{p_j}$ vectors can be used to produce feedback for both teachers and students. Upon some basic visualisation, students can view their own $\mathbf{p_j}$ vector and observe which marking points they managed to incorporate into their answer and which marking points were missed out, to give them an indication of their areas for improvement. Furthermore, an average $\mathbf{p_j}$ vector across all answers can also be calculated and visualised for teachers' analysis, as seen in Figure F.1 in Appendix F. Figure F.1 was produced by averaging the $\mathbf{p_j}$ vectors from the cosine similarity model run on dataset 1. From this, teachers can immediately observe which marking points most students wrote in their answers and which they failed to incorporate. This would provide feedback to teachers about which parts of their teaching were well understood by the students and likewise, which parts might need review. We can thus see how the qualitative feedback produced by the model aids in the task of formative assessment.

4.4 Final Architecture

Based on our results, we recommend two separate architectures for different types of questions. For the quantitative component, we recommend a BiGRU network with GloVe embeddings and an attention mechanism for simple recall style questions but a BiLSTM network with fastText embeddings and an attention mechanism for more complex application-based questions. For the qualitative component, we recommend the cosine similarity model for both types of questions.

5 Conclusion and Future Work

In this work, we proposed a novel approach to grading short-answer physics questions. Both the qualitative and quantitative components are shown to perform well, especially in low data conditions, which is important in order to reduce the marking workload of teachers. Furthermore, the architectures also provide interpretable feedback for both teachers and students, aiding in the task of formative assessment.

Given such promising results, these architectures can be applied to online learning portals where teachers can deploy a system for each short-answer question they create and feed the system a small amount of marked answers as training data along with a simple marking point-based marking scheme.

In future works, we propose exploring different word embeddings such as ELMo [25], which could yield a performance upgrade. Additionally, to tackle the problem of overfitting we identified, dropout layers [26] could be employed in our models. Thirdly, more careful hyperparameter optimisation using comprehensive methods such as randomised search [27] and sequential search [28] on our current models could yield better results. Considering the good performance of the qualitative models, it may be possible to increase performance by incorporating the marking points as features to the quantitative models. Lastly, the models could also be evaluated on other datasets, such as larger datasets and datasets from different domains such as other sciences, to test its scalability and adaptability.

References

[1] P. Black and D. Wiliam, “Developing the theory of formative assessment”, *Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education)*, vol. 21, no. 1, p. 5, 2009.

[2] P. Black and D. Wiliam, “Assessment and Classroom Learning”, *Assessment in Education: Principles, Policy & Practice*, vol. 5, no. 1, pp. 7–74, 1998. doi: `10.1080/0969595980050102`.

[3] L. S. Larkey, “Automatic Essay Grading Using Text Categorization Techniques”, in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’98, Melbourne, Australia: ACM, 1998, pp. 90–95, isbn: 1-58113-015-5. doi: `10.1145/290941.290965`.

[4] C. P. Rosé, A. Roque, D. Bhembe, and K. VanLehn, “A Hybrid Approach to Content Analysis for Automatic Essay Grading”, in *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, 2003, pp. 88–90. [Online]. Available: `https://www.aclweb.org/anthology/N03-2030`.

[5] H. Nguyen and L. Dery, *Neural Networks for Automated Essay Grading*, 2018.

[6] Y. Zhang, R. Shah, and M. Chi, “Deep Learning+ Student Modeling+ Clustering: A Recipe for Effective Automatic Short Answer Grading.”, *International Educational Data Mining Society*, 2016.

[7] J. Pennington, R. Socher, and C. Manning, “GloVe: Global Vectors for Word Representation”, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information”, *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space”, 2013. arXiv: `1301.3781`.

[10] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] S. Hochreiter and J. Schmidhuber, “LSTM Can Solve Hard Long Time Lag Problems”, in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, ser. NIPS’96, Denver, Colorado: MIT Press, 1996, pp. 473–479.

[12] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation”, 2014. arXiv: `1406.1078`.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling”, 2014. arXiv: `1412.3555`.

[14] J. Y. Lee and F. Dernoncourt, “Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks”, 2016. arXiv: `1603.03827`.

[15] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks”, *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[16] A. Graves, S. Fernández, and J. Schmidhuber, “Bidirectional LSTM networks for improved phoneme classification and recognition”, in *International Conference on Artificial Neural Networks*, Springer, 2005, pp. 799–804.

[17] M. Sundermeyer, T. Alkhouli, J. Wuebker, and H. Ney, “Translation modeling with bidirectional recurrent neural networks”, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 14–25.

[18] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate”, 2014. arXiv: 1409.0473.

[19] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical Attention Networks for Document Classification”, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. [Online]. Available: https://www.aclweb.org/anthology/N16-1174.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet Allocation”, *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, Mar. 2003, ISSN: 1532-4435. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944937.

[21] A. Huang, “Similarity Measures for Text Document Clustering”, in *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, vol. 4, Jan. 2008, pp. 9–56.

[22] G. Weiss, Y. Goldberg, and E. Yahav, “On the Practical Computational Power of Finite Precision RNNs for Language Recognition”, 2018. arXiv: 1805.04908.

[23] D. M. Hawkins, “The Problem of Overfitting”, *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, Jan. 2004. DOI: 10.1021/ci0342472.

[24] N. Reimers and I. Gurevych, “Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks”, 2017. arXiv: 1707.06799.

[25] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations”, 2018. arXiv: 1802.05365.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[27] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization”, *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[28] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for Hyper-Parameter Optimization”, in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2011, pp. 2546–2554. [Online]. Available: http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

[29] F. Chollet *et al.*, *Keras*, https://github.com/fchollet/keras, 2015.

[30] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, 2014. arXiv: 1412.6980.

[31] R. A. Horn, “The Hadamard Product”, in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

[32] S. Ruder, “An overview of gradient descent optimization algorithms”, 2016. arXiv: 1609.04747.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch”, 2017.

Appendix A Datasets

A.1 Dataset 1

Question: Explain the difference between conduction and convection. (2 marks)

Conduction	Convection
1. Conduction is the transfer of thermal energy through a medium without any flow of the medium.	4. Convection is the transfer of thermal energy by means of convection currents in a fluid (gas or liquid), due to difference in density.
2. Conduction occurs in solid, liquid and gases, or all states of matter.	5. Therefore, it only occurs in liquids and gases (fluids) because their particles are able to move freely.
3. Conduction occurs via the vibration of particles (for both metals and non-metals) and free electron diffusion (for metals only). Therefore, liquids and gases are poor conductors of heat compared to solids.	6. A convection current is the movement of fluid caused by a difference in the densities of various parts of the fluid.

Table A.1: Teacher's Marking Scheme for Dataset 1

Original Student Answer	Pre-processed Student Answer	Score
Conduction is the transfer of thermal energy from one place to another without any flow of the material medium. It occurs in all states of matter. However, Convection is the transfer of thermal energy by means of convection currents in a fluid due t difference in density. It occurs in only liquids and gases.	conduction transfer thermal energy one place another without flow material medium occurs states matter however convection transfer thermal energy means convection currents fluid due difference density occurs liquids gases	2
In convection, it is the transfer of thermal energy by convection currents in a fluid such as gas and liquids. However, in conduction. heat can be transfered through solids and can transfer heat from one place to another without any flow of the material medium.	convection transfer thermal energy convection currents fluid gas liquids however conduction heat solids transfer heat one place another without flow material	1

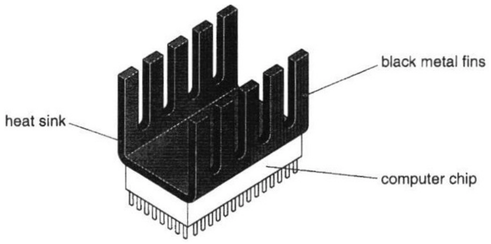
Conduction is the transfer of thermal energy from one place to another without any flow of the material medium whereas convection is the transfer of thermal energy by means of convection currents in a fluid due to difference in density.	conduction transfer thermal energy one place another without flow material medium whereas convection transfer thermal energy means convection currents fluid due difference density	1
Conduction requires physical contact for the process of thermal energy to be transferred from a body to another. Convection is the process when a fluid or gas above a hot surface gains thermal energy, expands, which causes its density to decrease and make it rise.	conduction requires physical contact process thermal energy transferred body another convection process fluid gas hot surface gains thermal energy expands causes density decrease make rise	0
Conduction is the transfer of heat energy by direct contact. However, Convection is the movement of heat by actual motion of matter.	conduction transfer heat energy direct contact however movement heat actual motion matter	0
Convection occurs through an intermediate medium while conduction occurs by a solid substance	convection occurs intermediate medium conduction occurs solid substance	0
Conduction is the transfer of heat energy by direct contact but convection is the movement of heat by actual motion of matter. Conduction is when objects have to be in direct contact in order for heat to be transferred between objects. However, convection is the transfer of heat energy through a convection current in a fluid due to difference in density. Conduction can occur is all states of matter but convection only occurs in liquids and gases.	conduction transfer heat energy direct contact movement heat actual motion conduction objects direct contact order heat transferred objects however convection transfer heat energy convection current fluid due difference density conduction occur states matter convection occurs liquids gases	1

Conduction occurs in all states of matter, as it is the transfer of thermal energy from one place to another without any flow of the material medium. Convection, however, occurs only in liquids and gases, as it is the transfer of thermal energy by means of convection currents in a fluid, due to difference in density.	conduction occurs states matter transfer thermal energy one place another without flow material medium convection however occurs liquids gases transfer thermal energy means convection currents fluid due difference density	2

Table A.2: Sample Dataset 1 Answers

A.2 Dataset 2

Question: The figure below shows a computer chip fitted with a heat sink with black metal fins. Explain the features of the heat sink that allow thermal energy to be transferred easily away from the chip. (3 marks)



Teacher's Marking Scheme for Dataset 2:

1. The fins are made of metals. Metals are good conductors of heat.
2. The heat sink has maximum contact surface with the chip, thus heat can be transferred from the chip to the heat sink effectively.
3. The fins increases the surface area exposed to the environment, hence increasing the rate of radiation.
4. The upward facing orientation of the fins increases the rate of convection.
5. The black coloured fins are good emitters of infrared radiation.

Original Student Answer	Pre-processed Student Answer	Score
The heat sink is black. Black, dull surfaces absorb and radiate heat much faster compared to bright, shiny surfaces. Thus it is a good emitter of radiation as it can absorb thermal heat from the chip and radiate it away the quickest, allowing it to be the coolest.	heat sink black black dull surfaces absorb radiate heat much faster compared bright shiny surfaces thus good emitter radiation absorb thermal heat chip radiate away quickest allowing coolest	1

Since metal is good conductor of heat, it transfer heat away from the chip faster. The metal fin is black, and black is a better emitter of radiation. Since hot air rises and cold air sinks, the chip will be cooled by the cold air.	since metal good conductor heat transfer heat away chip faster metal fin black black better emitter radiation since hot air rises cold air sinks chip cooled cold	2
The heat sink has metal fins on it to allow for a greater surface area to volume ratio that is exposed to the surrounding air, allowing the thermal energy to be conducted away from the heat sink and into the surrounding air. The heat sink is also black in colour and black is a good radiator of heat, allowing the thermal energy to be radiated away from the heat sink quickly.	heat sink metal fins allow greater surface area volume ratio exposed surrounding air allowing thermal energy conducted away heat sink surrounding air heat sink also black colour black good radiator heat allowing thermal energy radiated away heat sink quickly	2
1. There is a large surface area for heat to be transferred away from the chip to keep it cool. 2. It is black so that the chip can conduct heat away better. 3. It is able for convection to take place as when the computer heats up, the air particles near the chip gain heat, expand and rise, which forms a convection current with the cooler, denser air particles above to sink.	large surface area heat transferred away chip keep cool black chip conduct heat away better able convection take place computer heats air particles near chip gain heat expand rise forms convection current cooler denser air particles sink	0
The heat sink has a large surface area, and has holes im between allowing for heat to radiate faster and more.	heat sink large surface area holes im allowing heat radiate faster	1
The heat sink is absorber of radiation as it is black in colour. Black and dull surfaces absorb and radiate heat very fast as shiny surfaces are good at reflecting light. The black heat sink would then absorb heat from the computer chip easily.	heat sink radiation black colour black dull surfaces absorb radiate heat fast shiny surfaces good reflecting light black heat sink would absorb heat computer chip heat sink metal fins increase surface area heat sink air larger surface area higher rate heat transfer radiation heat sink would conduct	3

The heat sink has metal fins to increase the surface area of the heat sink to the air. The larger the surface area, the higher the rate of heat transfer by radiation. The heat sink would conduct heat away from the computer chip and to the surrounding cooler air faster with larger surface area and make it easier to transfer more heat away from the computer chip. The heat sink is made of metal which is a solid and solids are better conductors of heat as compared to the other states of matter. Metal is also a good conductor of heat.	heat away computer chip surrounding cooler air faster larger surface area make easier transfer heat away computer chip heat sink made metal solid solids better conductors heat compared states matter metal also good conductor heat	

Table A.3: Sample Dataset 2 Answers

A.3 Preliminary Statistics

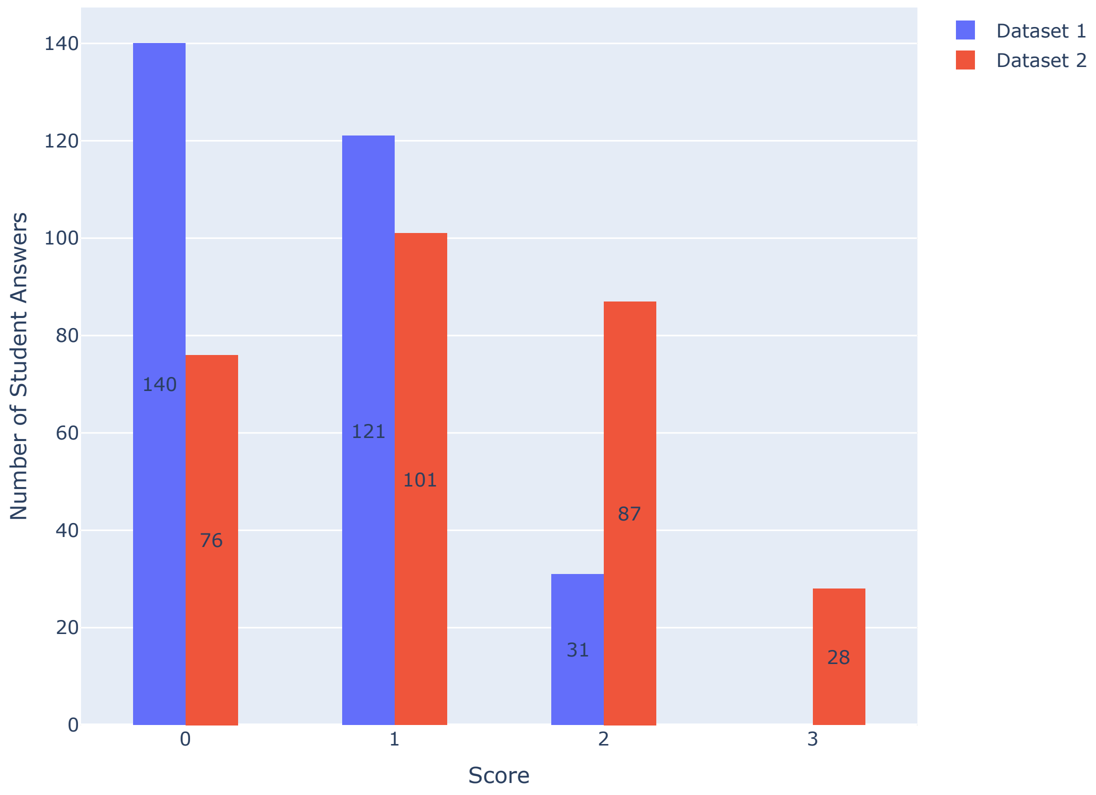


Figure A.1: Distribution of Scores

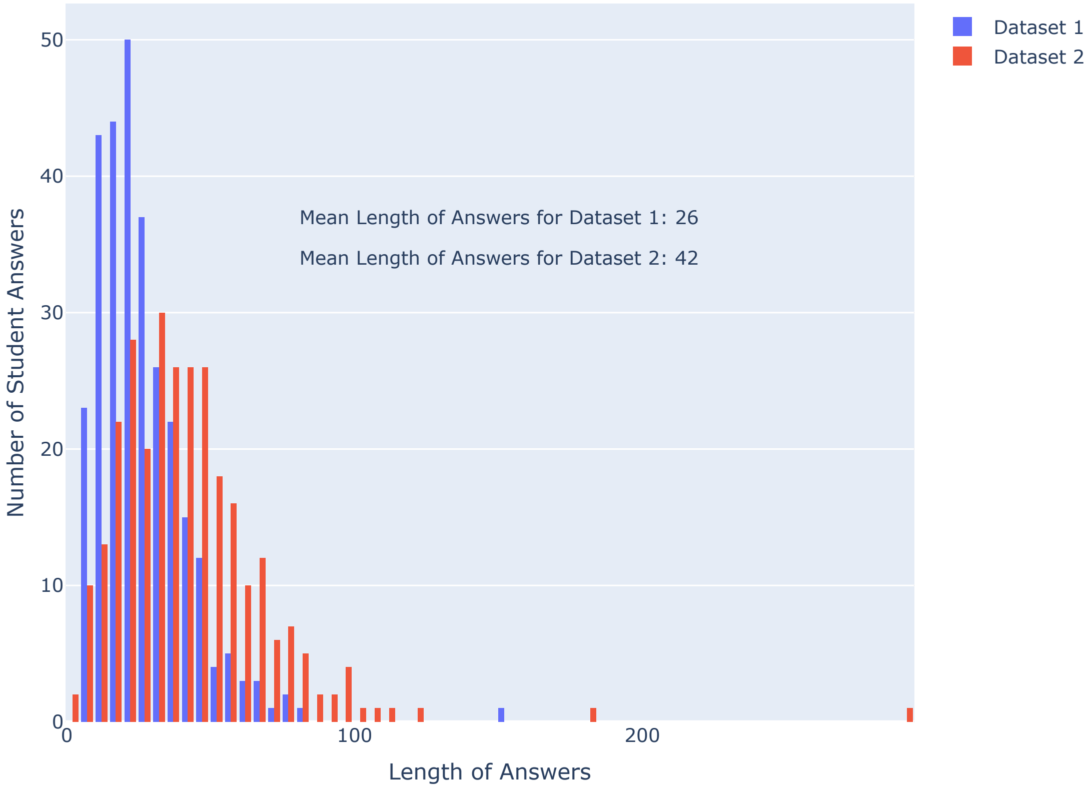
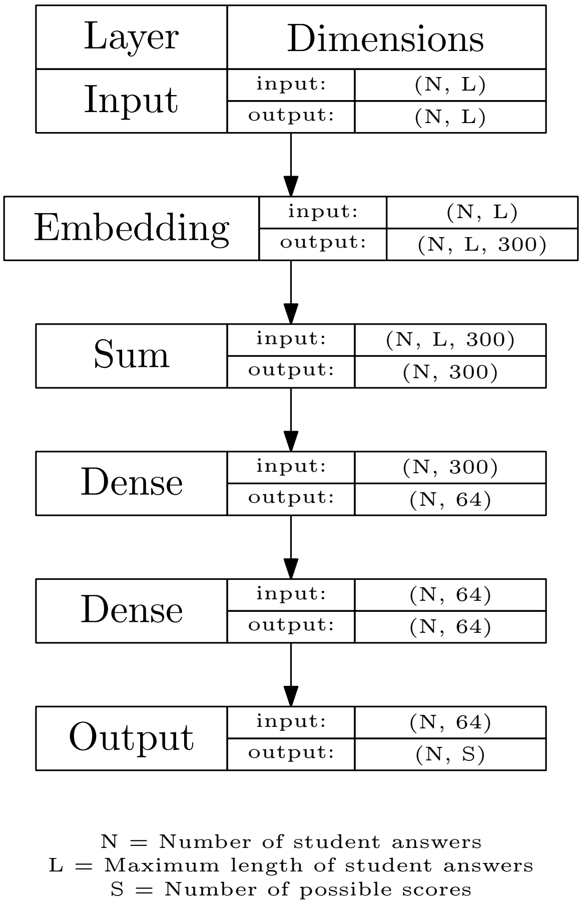


Figure A.2: Distribution of Length of Answers

Appendix B Quantitative Model Components

B.1 Baseline Neural Network Model Architecture



Layer	Dimensions	
Input	input:	(N, L)
	output:	(N, L)
Embedding	input:	(N, L)
	output:	(N, L, 300)
Sum	input:	(N, L, 300)
	output:	(N, 300)
Dense	input:	(N, 300)
	output:	(N, 64)
Dense	input:	(N, 64)
	output:	(N, 64)
Output	input:	(N, 64)
	output:	(N, S)

N = Number of student answers
L = Maximum length of student answers
S = Number of possible scores

Figure B.1: Baseline Model Architecture

The quantitative component of our short-answer grading system consists of a neural network model. The diagram above illustrates the architecture of the baseline neural network. First, the student answers are fed through the input and embedding layer to get a word vector representation of each word. GloVe is used for the embedding layer. Next, the word vector representation of each word is added together in the sum layer before this representation is fed through two dense layers with ReLU activation, followed by a softmax output layer to give a score of 0, 1 or 2 for the first question and 0-3 for the second question.

B.2 Non-Baseline Neural Network Model Architectures

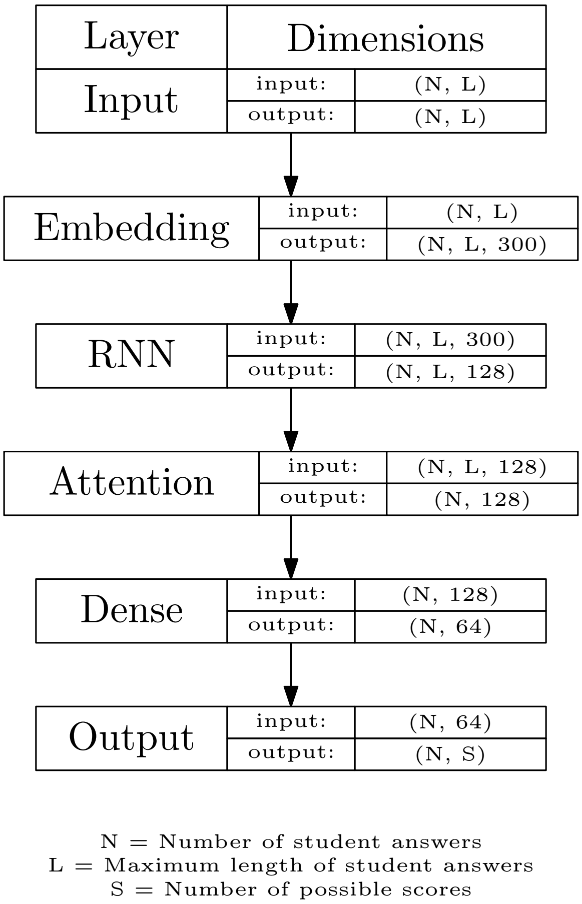


Figure B.2: Non-Baseline Model Architecture

The neural network architecture for the rest of the models is similar to the baseline. An example of an architecture is illustrated in the diagram above. The input and embedding layers are the same as the baseline model except that GloVe and fastText are used in the embedding layer for comparison. Instead of adding the word representations together, the student answers are fed through the recurrent layer of the model. 4 different types of recurrent layers are compared — LSTM, GRU, BiLSTM and BiGRU. Then, for the models with attention mechanism, the student answers are fed through the attention layer. For models without attention mechanism, this layer is not present. Finally, the student answers are fed through a densely-connected layer with ReLU activation, followed by a softmax output layer to give the final score.

B.3 Neural Network Hyperparameters

The Keras functional API [29] is used to implement the neural network models for the quantitative component of the model (both the baseline and non-baseline models). Recurrent and densely connected layers all have 64 units each. The batch size is set to 1 as we have very few training samples and thus updating the weights after every sample would allow us to train the network more effectively. The number of epochs is set to a maximum of 30. However, early stopping is implemented to prevent overfitting of the model on the training set. For early stopping, accuracy is monitored such that if the accuracy does not increase for 5 epochs, training is halted. Model checkpointing is implemented to ensure that the best model, based on accuracy, is saved during the training process so that it can be used for evaluation after training is halted. The Adam optimizer [30] is used and loss is set to categorical cross entropy loss as this is a multi-class classification problem. The loss function is defined below in Appendix D.

B.4 Attention Mechanism

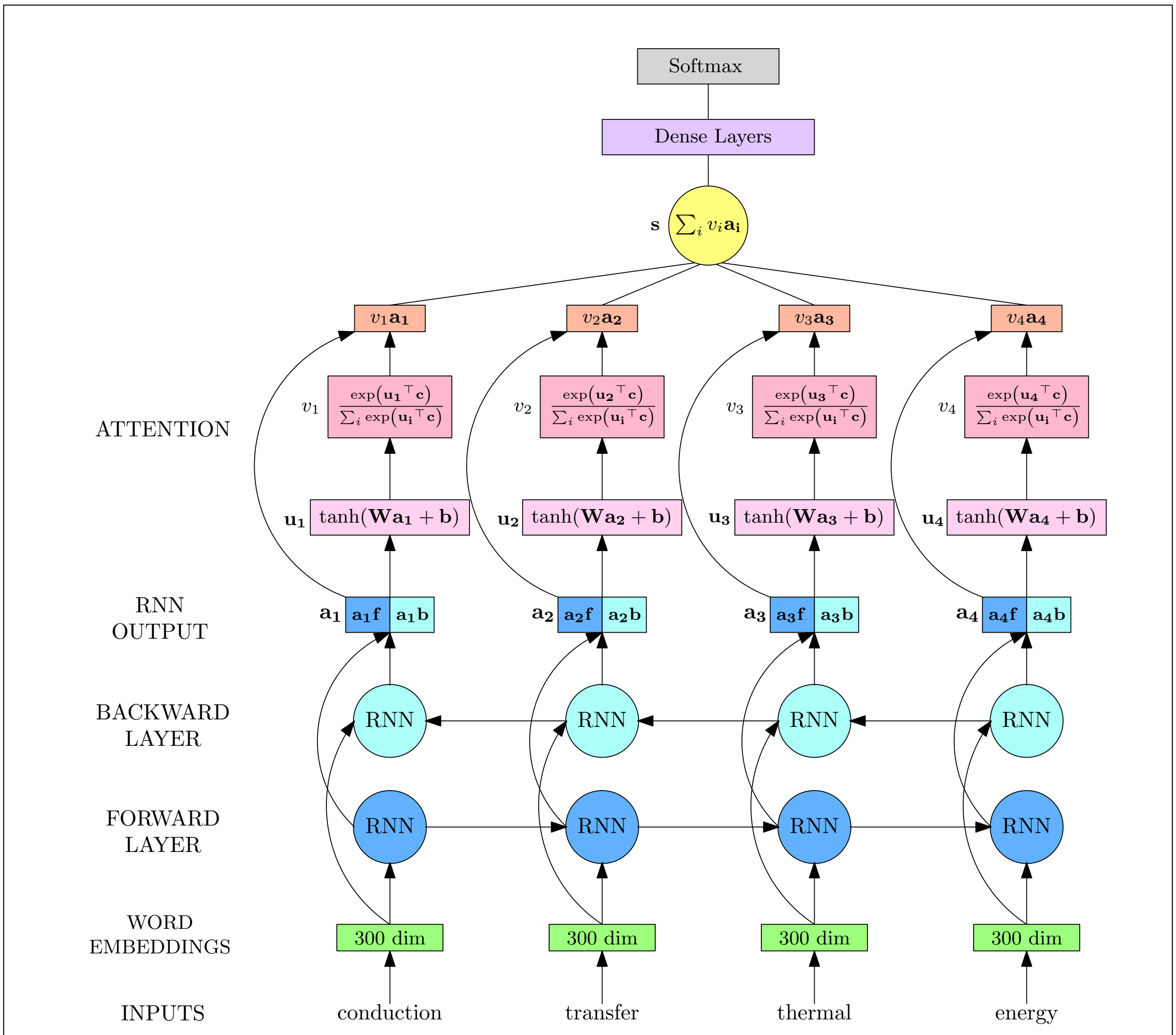

Softmax
Dense Layers
s
$\sum_i v_i \mathbf{a_i}$
$v_1\mathbf{a_1}$
$v_2\mathbf{a_2}$
$v_3\mathbf{a_3}$
$v_4\mathbf{a_4}$
ATTENTION
v_1 $\frac{\exp(\mathbf{u_1}^\top \mathbf{c})}{\sum_i \exp(\mathbf{u_i}^\top \mathbf{c})}$
v_2 $\frac{\exp(\mathbf{u_2}^\top \mathbf{c})}{\sum_i \exp(\mathbf{u_i}^\top \mathbf{c})}$
v_3 $\frac{\exp(\mathbf{u_3}^\top \mathbf{c})}{\sum_i \exp(\mathbf{u_i}^\top \mathbf{c})}$
v_4 $\frac{\exp(\mathbf{u_4}^\top \mathbf{c})}{\sum_i \exp(\mathbf{u_i}^\top \mathbf{c})}$
$\mathbf{u_1}$ $\tanh(\mathbf{W}\mathbf{a_1}+\mathbf{b})$
$\mathbf{u_2}$ $\tanh(\mathbf{W}\mathbf{a_2}+\mathbf{b})$
$\mathbf{u_3}$ $\tanh(\mathbf{W}\mathbf{a_3}+\mathbf{b})$
$\mathbf{u_4}$ $\tanh(\mathbf{W}\mathbf{a_4}+\mathbf{b})$
RNN OUTPUT
$\mathbf{a_1}$ $\mathbf{a_1 f}$ $\mathbf{a_1 b}$
$\mathbf{a_2}$ $\mathbf{a_2 f}$ $\mathbf{a_2 b}$
$\mathbf{a_3}$ $\mathbf{a_3 f}$ $\mathbf{a_3 b}$
$\mathbf{a_4}$ $\mathbf{a_4 f}$ $\mathbf{a_4 b}$
BACKWARD LAYER
RNN RNN RNN RNN
FORWARD LAYER
RNN RNN RNN RNN
WORD EMBEDDINGS
300 dim 300 dim 300 dim 300 dim
INPUTS
conduction transfer thermal energy

Figure B.3: Attention Mechanism on Top of a BiRNN

Attention is able to extract words that are important to the meaning of the sentence and aggregate the representation of those important words to form a sentence vector. Firstly, we concatenate the outputs of both the forward and backwards layer of the BiRNN to obtain the word annotations $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_i}$ for each word i in the student answer. Next, we feed these word annotations through a one-layer multi-layer perceptron with weights $\mathbf{W}$, bias $\mathbf{b}$ and a hyperbolic tangent activation to get $\mathbf{u_i}$ as a hidden representation of $\mathbf{a_i}$, as shown by the equation:

$$\mathbf{u_i} = \tanh\left(\mathbf{W}\mathbf{a_i} + \mathbf{b}\right) \tag{B.1}$$

Then, we measure the importance of the ith word as the similarity of $\mathbf{u_i}$ with a context vector $\mathbf{c}$ and obtain a normalised importance weight v_i through a softmax transform:

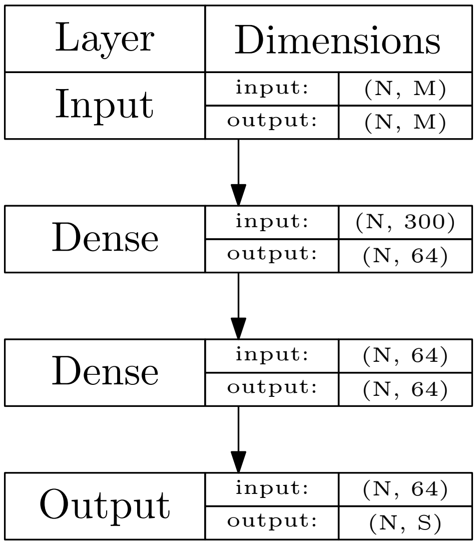
$$v_i = \frac{\exp\left(\mathbf{u_i}^\top \mathbf{c}\right)}{\sum_i \exp\left(\mathbf{u_i}^\top \mathbf{c}\right)} \tag{B.2}$$

The context vector $\mathbf{c}$, initialised randomly and learned during the training process, can be seen as a high level representation of a fixed query “which are the important words?” over all the words in the sequence. Finally, the sentence vector $\mathbf{s}$ is computed as a weighted sum of the word annotations based on the importance weights v_i. This sentence vector is then passed on to a dense layer followed by the softmax output layer to give the final score.

$$\mathbf{s} = \sum_i v_i \mathbf{a_i} \tag{B.3}$$

Appendix C Qualitative Model Components

C.1 Qualitative Testing Feedforward Model



N = Number of student answers
M = Number of marking points
S = Number of possible scores

Figure C.1: Qualitative Testing Model Architecture

To test the qualitative model, we fed the marking point proportions into a simple feed-forward neural network consisting of an input layer, two dense layers and an output layer to give the final score.

C.2 Derivative of Loss Function

We define the loss function, $J(\mathbf{p_j})$, to be minimised as:

$$\begin{aligned} J(\mathbf{p_j}) &= \|\hat{\mathbf{a_j}} - \mathbf{a_j}\|^2 && \text{(C.1a)} \\ &= \|\mathbf{T}\sigma(\mathbf{p_j}) - \mathbf{a_j}\|^2 && \text{(C.1b)} \end{aligned}$$

Differentiating with respect to $\mathbf{p_j}$,

$$\begin{aligned} \nabla_{\mathbf{p_j}} J(\mathbf{p_j}) &= \frac{\partial}{\partial \mathbf{p_j}} \|\mathbf{T}\sigma(\mathbf{p_j}) - \mathbf{a_j}\|^2 \\ &= \frac{\partial}{\partial \mathbf{p_j}} (\mathbf{T}\sigma(\mathbf{p_j}) - \mathbf{a_j})^\top (\mathbf{T}\sigma(\mathbf{p_j}) - \mathbf{a_j}) \end{aligned}$$

Let $\mathbf{v_j} = \sigma(\mathbf{p_j})$. Then,

$$\begin{aligned} \nabla_{\mathbf{p_j}} J(\mathbf{p_j}) &= \frac{\partial}{\partial \mathbf{p_j}} \big(\mathbf{v_j}^\top \mathbf{T}^\top - \mathbf{a_j}^\top\big)(\mathbf{T}\mathbf{v_j} - \mathbf{a_j}) \\ &= \frac{\partial}{\partial \mathbf{p_j}} \big(\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{T}\mathbf{v_j} - \mathbf{v_j}^\top \mathbf{T}^\top \mathbf{a_j} - \mathbf{a_j}^\top \mathbf{T}\mathbf{v_j} + \mathbf{a_j}^\top \mathbf{a_j}\big) \end{aligned}$$

However $\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{a_j} \in \mathbb{R}$, thus $\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{a_j} = \big(\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{a_j}\big)^\top = \mathbf{a_j}^\top \mathbf{T}\mathbf{v_j}$. Then,

$$\begin{aligned} \nabla_{\mathbf{p_j}} J(\mathbf{p_j}) &= \frac{\partial}{\partial \mathbf{p_j}} \big(\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{T}\mathbf{v_j} - 2\mathbf{a_j}^\top \mathbf{T}\mathbf{v_j} + \mathbf{a_j}^\top \mathbf{a_j}\big) \\ &= \left[\frac{\partial}{\partial \mathbf{v_j}} \big(\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{T}\mathbf{v_j} - 2\mathbf{a_j}^\top \mathbf{T}\mathbf{v_j}\big)\right] \frac{\partial \mathbf{v_j}}{\partial \mathbf{p_j}} \\ &= 2\big(\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{T} - \mathbf{a_j}^\top \mathbf{T}\big) \circ \sigma'(\mathbf{p_j}) \\ &= 2\big(\mathbf{v_j}^\top \mathbf{T}^\top \mathbf{T} - \mathbf{a_j}^\top \mathbf{T}\big) \circ \mathbf{v_j} \circ (1 - \mathbf{v_j}) && \text{(C.2)} \end{aligned}$$

Where $\circ$ denotes the Hadamard product [31]. The optimal proportions $\hat{\mathbf{p_j}}$ can be found by using the gradient descent algorithm [32]:

$$\mathbf{p_j} \leftarrow \mathbf{p_j} - \eta \cdot \big(\nabla_{\mathbf{p_j}} J(\mathbf{p_j})\big)^\top \tag{C.3}$$

Where $\eta \in \mathbb{R}^+$ is the learning rate.

In practice, the automatic differentiation framework PyTorch [33] was used to find the optimal proportions $\hat{\mathbf{p_j}}$, using the AdaMax algorithm [32] with an initial learning rate of $\eta = 0.002$.

Appendix D Metrics

Accuracy: Accuracy is the number of correct predictions made divided by the total number of predictions made.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{D.1}$$

Cross-Entropy Loss: For the ith sample out of n samples, given a ground truth "one-hot" vector, $\mathbf{y_i}$, and a vector of probability estimates, $\hat{\mathbf{y}}_\mathbf{i}$, the cross-entropy loss is defined by:

$$J(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i}) = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y_i} \log\left(\hat{\mathbf{y}}_\mathbf{i}\right) \tag{D.2}$$

Precision: Precision is the number of true positives divided by the number of true positives and false positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{D.3}$$

Recall: Recall is the number of true positives divided by the number of true positives and false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{D.4}$$

Weighted F1: F1 is the harmonic mean of precision and recall (defined above), used for a binary classification tasks. For multi-class classification tasks, weighted F1 is used, where the F1 score for each label is calculated and the average is weighted by the number of true instances for each label.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{D.5}$$

Appendix E Results of Quantitative Models

E.1 Performance of Models

Models		Accuracy	Loss	F1 Score
Feedforward Neural Network (Baseline)		0.733	0.849	0.719
LSTM	With GloVe	0.702	0.780	0.660
	BiLSTM with GloVe	0.705	0.837	0.699
	BiLSTM with GloVe and Attention	0.760	0.698	0.750
	BiLSTM with fastText and Attention	0.716	0.741	0.696
GRU	With GloVe	0.715	1.050	0.701
	BiGRU with GloVe	0.709	0.909	0.692
	BiGRU with GloVe and Attention	**0.781**	**0.701**	**0.775**
	BiGRU with fastText and attention	0.726	0.715	0.692

Table E.1: Performance of Quantitative Models on Dataset 1 (Embeddings Frozen)

Models		Accuracy	Loss	F1 Score
Feedforward Neural Network (Baseline)		0.716	0.875	0.695
LSTM	With GloVe	0.709	0.951	0.685
	BiLSTM with GloVe	0.709	0.995	0.688
	BiLSTM with GloVe and Attention	0.753	0.685	0.735
	BiLSTM with fastText and Attention	0.750	0.689	0.727
GRU	With GloVe	0.695	1.244	0.679
	BiGRU with GloVe	0.712	1.084	0.709
	BiGRU with GloVe and Attention	0.753	0.806	0.722
	BiGRU with fastText and attention	0.733	0.806	0.709

Table E.2: Performance of Quantitative Models on Dataset 1 (Update Embeddings)

	Models	**Accuracy**	**Loss**	**F1 Score**
Feedforward Neural Network (Baseline)		0.472	1.354	0.433
LSTM	With GloVe	0.428	1.840	0.399
	BiLSTM with GloVe	0.432	1.546	0.394
	BiLSTM with GloVe and Attention	0.520	1.530	0.500
	BiLSTM with fastText and Attention	**0.541**	**1.157**	**0.506**
GRU	With GloVe	0.414	1.658	0.373
	BiGRU with GloVe	0.421	1.459	0.392
	BiGRU with GloVe and Attention	0.462	1.435	0.447
	BiGRU with fastText and attention	0.496	1.265	0.465

Table E.3: Performance of Quantitative Models on Dataset 2 (Embeddings Frozen)

	Models	**Accuracy**	**Loss**	**F1 Score**
Feedforward Neural Network (Baseline)		0.527	1.149	0.502
LSTM	With GloVe	0.421	1.573	0.373
	BiLSTM with GloVe	0.428	1.622	0.406
	BiLSTM with GloVe and Attention	0.449	1.379	0.421
	BiLSTM with fastText and Attention	0.469	1.889	0.450
GRU	With GloVe	0.397	1.781	0.345
	BiGRU with GloVe	0.387	1.601	0.327
	BiGRU with GloVe and Attention	0.438	1.448	0.378
	BiGRU with fastText and attention	0.421	1.405	0.363

Table E.4: Performance of Quantitative Models on Dataset 2 (Update Embeddings)

E.2 Performance Against Size of Training Set

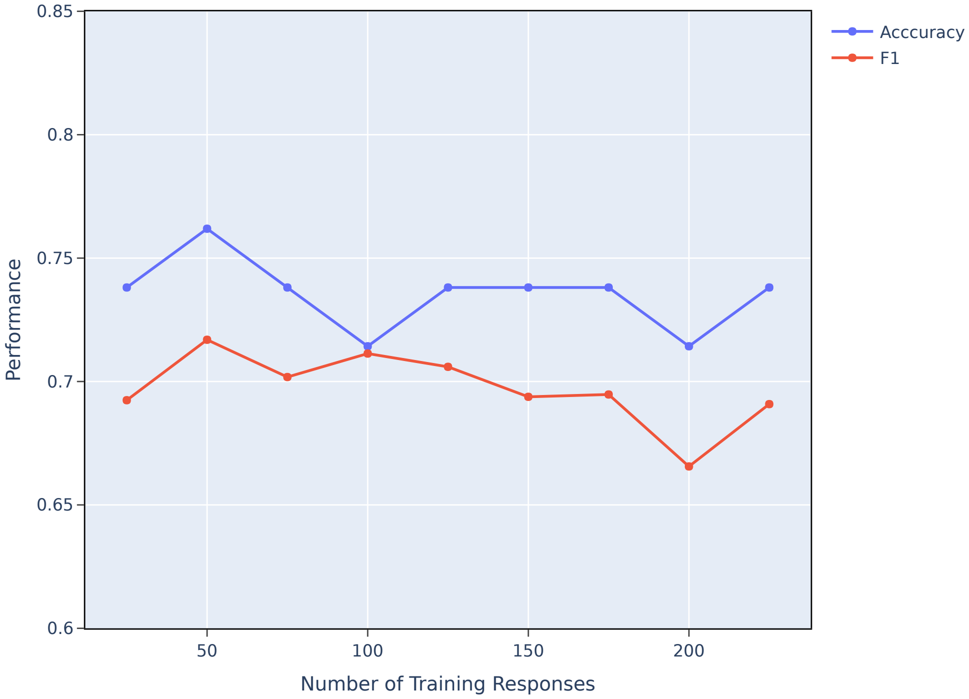


Figure E.1: Performance of Dataset 1 Best Model (BiGRU with GloVe and Attention) Against Size of Training Set

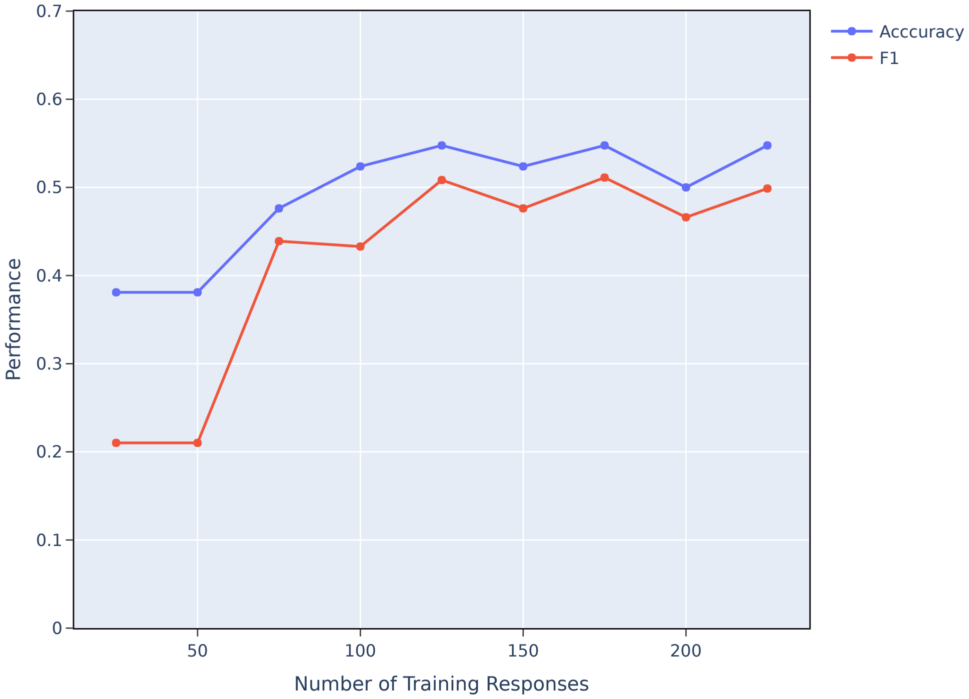


Figure E.2: Performance of Dataset 2 Best Model (BiLSTM with fastText and Attention) Against Size of Training Set

E.3 Attention Weights Visualisation

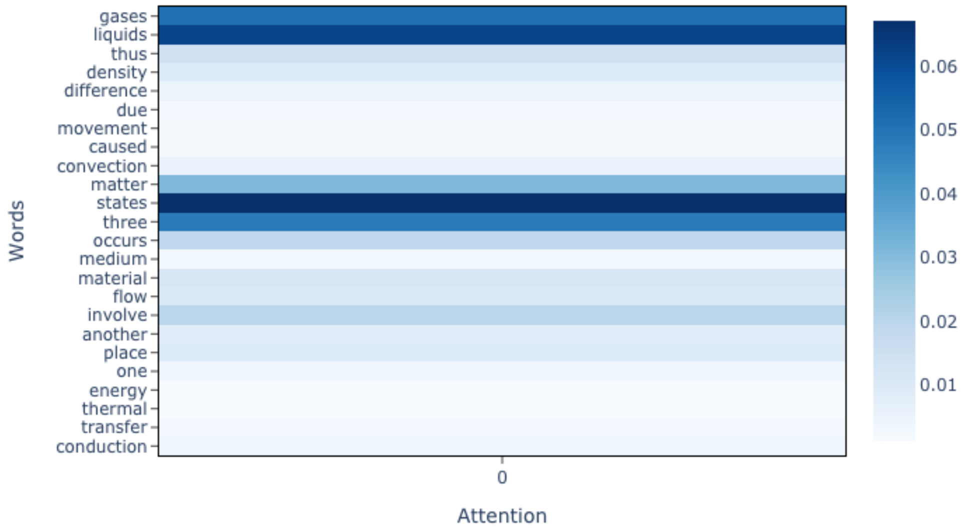


Figure E.3: Attention Weights of Answer Sample from Dataset 1

Appendix F Results of Qualitative Models

Models	Accuracy	Loss	F1
Vector Decomposition	0.591	0.931	0.552
Cosine Similarity	**0.716**	**0.652**	**0.713**
Euclidean Distance	0.648	0.769	0.608

Table F.1: Performance of Qualitative Models on Dataset 1

Models	Accuracy	Loss	F1
Vector Decomposition	0.455	1.251	0.438
Cosine Similarity	**0.541**	**1.128**	**0.527**
Euclidean Distance	0.534	1.165	0.493

Table F.2: Performance of Qualitative Models on Dataset 2

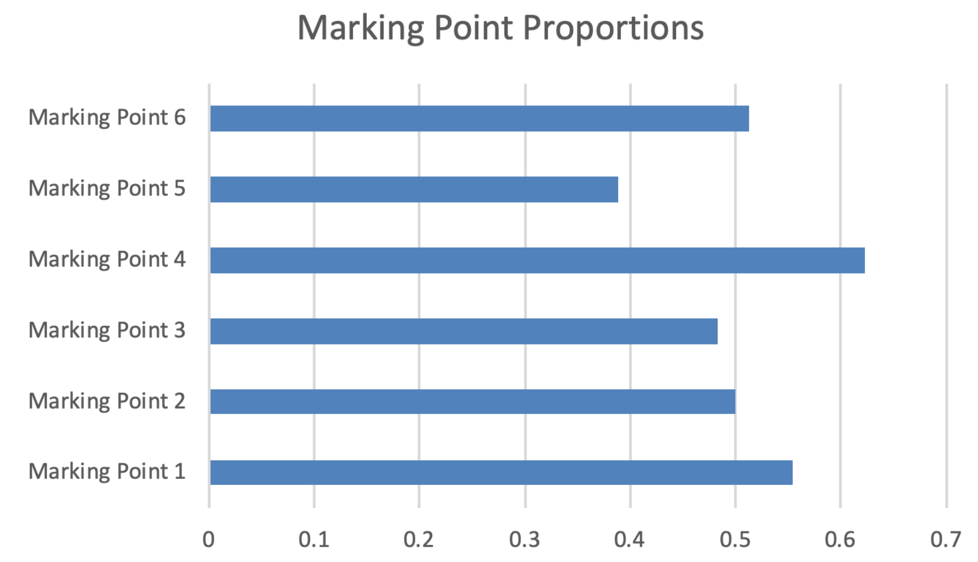


Figure F.1: Averaged Marking Point Proportions (Cosine Similarity) Across Dataset 1