Mistake Captioning: A Machine Learning Approach For Detecting Mistakes and Generating Instructive Feedback

Anton Vinogradov
University of Kentucky
Lexington, KY 40506
anton.vinogradov@uky.edu

Andrew Miles Byrd
University of Kentucky
Lexington, KY 40506
andrewbyrd@uky.edu

Brent Harrison
University of Kentucky
Lexington, KY 40506
harrison@cs.uky.edu

Abstract

Giving feedback to students is not just about marking their answers as correct or incorrect, but also finding mistakes in their thought process that led them to that incorrect answer. In this paper, we introduce a machine learning technique for *mistake captioning*, a task that attempts to identify mistakes and provide feedback meant to help learners correct these mistakes. We do this by training a sequence-to-sequence network to generate this feedback based on domain experts. To evaluate this system, we explore how it can be used on a Linguistics assignment studying Grimm's Law. We show that our approach generates feedback that outperforms a baseline on a set of automated NLP metrics. In addition, we perform a series of case studies in which we examine successful and unsuccessful system outputs.

1 Introduction

Giving feedback is one of the most critical parts of education and training. It allows the instructor to correct errors in students' understanding and guide them to the correct solution. In automated learning systems, such as intelligent tutoring systems (ITSs) the feedback and hints are focused on getting the student to complete the assignment, but this may not take into account their mastery of that skill. While better feedback can be made by considering student intent, we find that current models are lacking in this regard. They are typically focused on learning objective mastery and may not necessarily give the whole picture when determining *why* students answered something the way they did.

But there is a natural solution to this, in that skilled educators are capable of intuitively modeling a student's thought process and determining mistakes in it. This skill is often developed over time. The difficulty of this task for educators depends on the field of study, with some more free form fields like writing and programming being particularly difficult. By indirectly modeling a student's intent by instead modeling how the instructor gives feedback, we can use this model to generate novel feedback sentences.

We propose a system that when given a mistake that has been identified in the task of deriving sound changes from Proto-Indo-European to Proto-Germanic, will automatically generate an explanation that identifies the mistake and the reasoning to why this sound change should not apply in this case. We call this method *Mistake Captioning* as it takes inspiration from image captioning methods and post-hoc AI rationale generation methods such as Ehsan et al. (2019) Automated Rationale Generation.

We apply a suite of automated machine translation and image captioning metrics to test our system. We also take a closer look at 3 cases from a more complete retraining of the network to better understand how the network is performing and how well the metrics perform regarding our task. We find that this method is a good start, but more data and work need to be done to be able to integrate this into a serious learning environment.

2 Background and Related Work

In the space of ITSs a problem is how to get a student to the correct solution. One of the ways to achieve this is by having an expert to author the correct solution and the path to take to get there (Marwan et al., 2020; Unnam et al., 2019; Ariely et al., 2020). These all require the use of an expert to either structure the assignment such that feedback can be extracted or to label assignments to create a system that can learn the expert knowledge. In some domains this is not necessary as the students can provide the data for the path to the solution themselves. This data-driven approach

Proceedings of Recent Advances in Natural Language Processing, pages 1455–1462
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_163

uses data from multiple students to create all possible paths that lead to a solution, as is shown by Stamper et al. (2008) in their work on *Hint Factory*. This was later extended to create paths that optimize for productivity metrics by Maniktala and Barnes (2020). In vast solution spaces generating all possible paths to the solution is not feasible, and in Rivers and Koedinger (2017) work this is done by abstracting states so that a single state can represent a large space of states.

Our method does not attempt to establish a correct answer like these previous methods as they do not tackle an underlying problem as to why students make mistakes. We structure our problem not as a method of guiding the student to the correct path, but by trying to correct the thought process that led them to their current answer. This is similar to the methods used in image captioning (You et al., 2016) where they train a model to recognize what is in an image by relating human authored captions to images. This model is then used to generate captions on similar features in new pictures. This is also like the work done in post-hoc AI rationale generation like that of Ehsan et al. (2019). In this work they relate a set of human actions to human rationales of those actions for the use of explaining AI behavior. In our system, we make use of human actions in the form of mistakes and relate them to human explanations of the mistake.

2.1 The Proto-Indo-European Language and Grimm's Law

Our application area for this paper is on an assignment concerning Grimm's Law, which is a series of sound changes that occurred in the evolution of Proto-Indo-European (PIE) into Proto-Germanic (PGmc). PIE is a reconstructed language that attempts to recreate the common ancestor of Indo-European language family. While no direct evidence remains of PIE, the similarities between the languages in this family indicate that they come from a common ancestor. Despite these similarities, there are still differences in the languages that can be attributed to shifts in pronunciation as time went on and communities speaking this language became isolated from others. PGmc. is also a reconstructed language, serving as the source of all Germanic languages. Sound changes can be traced from PIE to PGmc.

We focus our task on a set of sound laws that describe how stop consonants change from PIE to PGmc collectively known as Grimm's Law (Campbell, 2013) and also include instances where Grimm's Law does not occur. Grimm's Law consists of three different changes that all contribute towards a single shift known as a chain. These changes are as follows:

1. PIE voiceless stops change to voiceless fricatives.
2. PIE voiced stops change to voiceless stops.
3. PIE voiced aspirated stops change to voiced stops or fricatives.

Together they are chained such that affected sounds in PIE are shifted one step to their form in PGmc. This limited set of shifts is what is represented in our data and is the reason for some of the limitations.

3 Methods

3.1 Data

The data used in this paper is centered around the linguistics PIE rule, Grimm's Law (Campbell, 2013). Since the task is to generate a reasoning for a mistake made by students, we have opted to organize the data as a fill-in-the-blank task, as one would show up as an assignment. Since we are focusing on captioning the mistake, the changes from PIE to PGmc must contain a mistake and an explanation for that mistake. To create these erroneous responses, we had a linguistics expert generate a number of entries fulfilling these categories: a PIE word; a PGmc form of that word with blanked areas demarked by underscores; a faulty PGmc word that contains a mistake; and an explanation for what that mistake was and how to fix it. The PIE word is the original form of the word that the student is tasked to change. The blanked PGmc form is blanked only in the areas that the student would be tasked to change. There may be multiple blanks if there are multiple spots where the change occurs. The faulty PGmc form contains only a single mistake each, so that the explanations can be more focused on that specific mistake. When multiple mistakes are identified for each PIE word, they are separate entries with separate explanations. Likewise, if there are multiple blanks to fill in for a word, then each blank will have a separate set of entries for its possible mistakes. The explanations were written with a simplified style as to not introduce too much detail into the explanations that

PIE	Blanked PGmc	Faulty PGmc	Explanation
pisḱós	_is_az	fishaz	ḱ ultimately does become h, but not when immediately following stops & fricatives
pisḱós	_is_az	biskaz	p can shift to b, but only when the middle of the word preceding an accented vowel
pisḱós	_is_az	fisgaz	ḱ can shift to g, but only when preceding an accented vowel in the middle of a word
pisḱós	_is_az	fisxaz	ḱ normally changes to x, but not when preceded by a stop or fricative

Table 1: The data entries for the mistaken sound changes in the PIE word for "fish"

resulted in each explanation being a single sentence following a general format. This sentence is in the rough form of "the mistake that was made, reason why it is not applicable in this case". For example, in Table 1 the explanation for why *fishaz* is wrong has *ḱ ultimately does become h* as the mistake that was made and *but not when immediately following stops & fricatives* as the reason why it is not applicable in this case.

This data was gathered by a single Linguistics Expert, with minor input from the authors as to style and structure to better suit this data for machine learning. Our linguistics expert is an associate professor of linguistics that frequently teaches classes on PIE. There were no attempts to control for consistency in tone, tense, or voice style, so such variations do occur in the data. Because the sentences follow a general format, there are some exact explanation matches in the data. Since this was the application of a small set of rules over a larger set of words, finding common letter changes and explanations could not be avoided. We collected 163 entries of PIE words, fill-in-the-blanks, faulty PGmc words, and their corresponding explanations. These came from 55 unique PIE words and contained 53 unique explanations to cover all cases found in the data. The most common explanation occurred 15 times, while others only occurred once in the entire dataset. Even those that occurred once often shared similar phrasing with other explanations since they were drawn from a common set of rules and a common set of consonants.

The data was further structured for training as seen in Table 2. To preserve the question that was being hypothetically asked to solve and the answer, we combined the PIE form and the faulty PGmc form, separated by a space, as our model input. For our model output we used the explanations without adding anything, but we did remove and change some characters. We removed all instances of ending punctuation like periods and exclamation points and spaced out commas to count them as separate tokens. The white space was also normalized and any leading or trailing white space was removed. While not shown in the table, the input was tokenized character by character to help preserve the differences between the faulty PGmc, while the output was done word by word.

3.1.1 Data Noise

Our data is very limited in scope due to the effort it requires to generate it. If we were to train our model only using the data as it is, we would likely run into issues with overfitting and overtraining. To alleviate this, we have injected noise into the data to be able to train longer. For our training output, creating noise is a straightforward process. In every training iteration, the output has 30% of its words masked out. This value was chosen to keep most of the explanation intact while still having a significant amount noised as overfitting is a serious concern. This means that at no training iteration has the model ever seen a complete output sentence, but with enough training iterations the unmasked portions should have overlapped enough to reveal the complete sentence.

The input was a more complicated process. Since the input is smaller than the output, we made the attempt to preserve some of the important parts, namely the actual changes from PIE to PGmc. Since these are fill-in-the-blank tasks, we have access to the parts that contain no mistakes and using this part of the dataset we identified the portions

Input	Output	Maskable Indices
pisḱós fishaz	ḱ ultimately does become h , but not when immediately following stops & fricatives	[10, 11, 13, 14]
pisḱós biskaz	p can shift to b , but only when the middle of the word preceding an accented vowel	[10, 11, 13, 14]
pisḱós fisgaz	ḱ can shift to g , but only when preceding an accented vowel in the middle of a word	[10, 11, 13, 14]
pisḱós fisxaz	k normally changes to x , but not when preceded by a stop or fricative	[10, 11, 13, 14]

Table 2: The processed data entries for the word "fish".

of the Germanic words that are considered safe to mask out in the form of maskable indices. These are shown in the 3rd column of Table 2, though these may not visually line up to the index of the words since letters that are accented are encoded as multiple characters. Since this only affected the PGmc part of the already smaller input, instead of masking 30% of the characters we only mask a single character at a time and limit it to only 30% of the time. This hopefully allows us to make a more robust set of training examples without disrupting the underlying meaning too much.

3.2 Model

For our model we use a sequence-to-sequence network. Sequence to sequence networks utilize two recurrent neural networks: an encoder and a decoder. The encoder encodes the sequential data into a fixed length context vector that is meant to represent the important elements of the input. This context vector is then used by the decoder to generate a natural language output. In our case, this output is an explanation, but this can be used for other tasks. Sequence to sequence networks have been used for translation (Sutskever et al., 2014) and post-hoc rationale generation (Ehsan et al., 2019) tasks, and while our task is not a traditional translation or rationale generation task it can be modeled as one. The network learns to take the input sequence of our PIE and faulty PGmc words as a character sequence and translate them into a word sequence for our explanations. By connecting the input to the explanation, we are learning human-like reasoning for why the mistakes happened. We use Gated Recurrent Units (GRUs) to take advantage of their faster training time and better use with smaller data (Chung et al., 2014). We also make use of an attention mechanism (Graves, 2013) on the decoder to learn to focus on the important parts of the output explanation.

The model was trained on 75,000 iterations using the same 163 question and explanation pairs, but as mentioned previously in each iteration some characters and words were masked to prevent the model from seeing them. During each training iteration a random sample from the data is run through the network after having noise applied.

4 Experiments

We conducted two experiments to evaluate our method: A quantitative approach using a set of automated metrics to judge the candidate explanations, and a qualitative case study to demonstrate what types of outputs were generated. For both the qualitative and quantitative evaluation we used 5 different automated metrics:

BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015), and BERTScore (Zhang et al., 2019).

For our quantitative evaluation, we used k-fold cross-validation to control for the variance in our data. We compare our approach against a majority baseline. Since several of the explanations in the data were repeated, we decided that a better baseline would be the majority baseline and used the most repeated explanation in our data. This explanation was used 15 times in the data, and for the majority baseline we set the generated output to all be this exact sentence. This explanation can be seen in Table 4 in the baseline incorrect, candidate sentence. While the majority baseline has a low

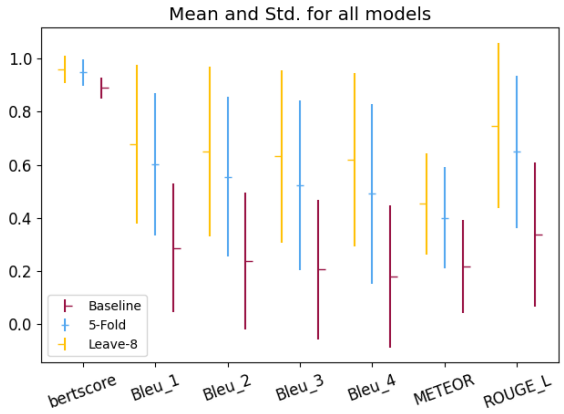

Figure 1: The means and std. displayed visually (CIDEr was not included since it is unbounded).

accuracy, about 9.2%, it was chosen to show that the method is viable before continuing to a human evaluation.

For our qualitative analysis, we opted to look at outputs for a retrained model that was trained on all but 8 random examples of the data. Since our dataset is small, we decided that training on all but a small number of examples would give better insights on what the network was learning. We examined how the model performed in generating explanations for each of the 8 examples that were held out during training. From these 8, we used their automated metric performance in conjunction with their correctness to divide them into three categories: correct, partially incorrect, and incorrect. We then explore why this occurred and reason about the implications of these results in practice.

During training, we use the following hyperparameter values for our model. We used a teacher forcing ratio of 0.5, and a learning rate of 0.01. For the hidden layers (consisting of the attention and GRU layers), a fixed size of 256 was used. All outputs were limited to a maximum length of 30 but this limit was not reached. To test the model, we implemented beam search to select the best candidate explanation instead of the default greedy search. In addition to improving the accuracy of the model this allowed us to look at multiple final outputs, each scored by the model. This is useful in manually judging if the alternatives to the top output were more correct. We limited the beam width to 5.

5 Results

We found that our method consistently outperformed the baseline in all metrics as can be seen in Table 3. This provides evidence that our model is working to learn when to apply and how to recreate humanlike feedback in response to mistakes. We also found that the leave-8 retraining performed better than the 5-fold models. This trend can be visually seen in Figure 1.

Applying a 2-sided t-test we find that the difference between the baseline and the 5-fold is statistically significant ($p < 0.0001$) in all cases. When comparing the baseline to 8 case studies this also holds true ($p < 0.001$). There is also no statistically significant difference between the case studies and the 5-fold.

6 Discussion

In this section, we discuss the results in more detail, going over the results of the automated metrics and case studies.

6.1 Automated Metrics

While the automated metrics do show that our method outperforms our baseline, they are not perfect. One finding that came out of this evaluation is that automated metrics can be misleading when evaluating performance. This is because automated metrics for evaluating language generation systems often measure how well generated sentences overlap with reference sentences. The change of a single word alters the semantic meaning of an explanation, however, which can fool the metrics into scoring it higher than it is. Likewise, a generated explanation can be nearly correct but contain little similarity in structure to the original. This is ultimately a limitation of the data since we only have a single reference explanation for each faulty PGmc example. In the data there exist explanations that explain a concept similar to other explanations but are differently structured. If the model lifts the structure of one explanation to the correct change and reasoning of a change, there is no guarantee that this will match the reference explanation and will be scored more poorly. This does not mean that these metrics are not useful, though, just that the limited scope of our data makes it less suited to these metrics. There is a general trend between how well the explanation performs compared to how correct it is, as we will see in the case studies. In general, this seems to suggest that how

	Baseline		5-Fold		Case Studies	
	Mean	Std.	Mean	Std.	Mean	Std.
bertscore	0.8896	0.0397	0.9479	0.0502	**0.9593**	**0.0521**
Bleu_1	0.2872	0.2424	0.6019	0.2690	**0.6776**	**0.2999**
Bleu_2	0.2377	0.2570	0.5558	0.3009	**0.6503**	**0.3202**
Bleu_3	0.2059	0.2633	0.5228	0.3198	**0.6322**	**0.3239**
Bleu_4	0.1802	0.2676	0.4913	0.3378	**0.6180**	**0.3266**
METEOR	0.2175	0.1749	0.4003	0.1908	**0.4532**	**0.1917**
ROUGE_L	0.3377	0.2709	0.6495	0.2864	**0.7481**	**0.3112**

Table 3: Mean and Std. for all models generated.

well these metrics will perform will change if we get more data, both on the number of PIE words and the number of possible explanations for each mistake.

6.2 Case Studies

To take a better look at the generated explanations we have a separate leave-8-out training. In these 8 that we left out and tested on we found three categories: correct, partially incorrect, and completely incorrect. Correct explanations include the correct mistaken change and the correct reason it was not applicable, while partially incorrect only include one of the two. Incorrect explanations contain neither. Of the 8, 4 of these where correct, 3 were partially incorrect, and only one was completely incorrect. We have selected 3 examples to match these 3 categories to look more closely at.

6.2.1 Completely Incorrect

There was only a single example of the completely incorrect case, which is shown in Incorrect case in Table 4. This explanation does not identify the sound change giving *ḱ* instead of *t* for the sound change. For the reasoning it seems like it is on the correct track by mentioning *preceded by stop or fricative* but reverses whether the change occurs because of this. In fact, this is a case of the candidate explanation having no relation to the reference explanation, nor is close to any reference explanation for that PIE word, nor the PIE and PGmc input. The candidate explanation refers to a *ḱ* which does not appear in input but, as will be expanded on later, this may be a case of encoding. Due to the way that the input is interpreted on a character-by-character basis the model may have recognized the presence of the *k* and conflated it with a *ḱ*. It also used the explanation that is most repeated in the dataset, which suggests that it picked the most popular explanation as opposed to correctly learning this change.

6.2.2 Partially Incorrect

For the partially incorrect case we find that the model could identify something correct but not everything. Sometimes this was only identifying the correct sound change or the correct reasoning, but in the case that we are focusing it generated a completely new explanation that was one word from being correct. This candidate explanation changed the word *must* to *only* which completely changes the meaning of the explanation. In our explanation generation we saved the top 5 sentences, and the correct explanation does appear as the second. This may be another encoding issue since the phrase *k only* appears several times in our dataset. We used beam search to attempt to avoid this problem, but in this case, it does not seem to have been enough. The complete phrase that was generated does not appear as a reference explanation in any of the examples, so it is unknown as to why it chose this one. It is likely that having more reference explanations would help as it would force the model to learn more important features.

In another case of partially incorrect, we found that the exact reference explanation that never appeared in that training data, which means that the model could not have possibly generated a completely matching explanation.

6.2.3 Correct

The correct category is marked by when the model was correctly able to generate an explanation for the mistake that was made, matching both the change and reasoning. This is also the largest category, comprising 4 of the 8 examples. These score highly on all the metrics. This category has sound changes that are well represented in the training data, with no reference explanation appearing less than 5 times. In the case shown in Table 4, the

Case	PIE PGmc	Reference Sentence	Candidate Sentence
Completely Incorrect	skot**t**ó ska**t**a	t only remains t if it's preceded by a stop or fricative	while ḱ does become a velar sound , it also shifts if it isn't preceded by a stop or fricative
Partially Incorrect	**ḱ**lew- **k**lew	ḱ must shift if it isn't preceded by a stop or fricative	ḱ only shift if it isn't preceded by a stop or fricative
Correct	pén**k^{w}**e fin**k^{w}**e	k^{w} only remains k^{w} if it's preceded by a stop or fricative	k^{w} only remains k^{w} if it's preceded by a stop or fricative
Baseline Incorrect	h$_2$yuh$_2$n**ḱ**ó- yun**h**a-	ultimately k becomes h , but it first changes into a velar sound	while ḱ does become a velar sound , it also shifts if it isn't preceded by a stop or fricative

Table 4: Specific examples of cases that we found in our data. Included is an example from the baseline.

reference explanation appeared 8 times which is relatively high, but not unique to this category. The completely incorrect case had its reference sentence appear 12 times in the data, meaning that this is likely not the most prominent factor in whether a generated explanation will be correct or incorrect.

In all these cases we find that the automated metrics to judge these explanations are generally higher when the answer is more correct, and generally lower when it is more incorrect. This holds in cases when there is only partial correctness. As mentioned before this is likely due to our limited data and may break if we have more reference sentences. We hope to expand the data to cover other changes in linguistics, though the method should be able to be transferring to other domains too.

7 Future Work

Due to the way that our method works in identifying a single mistake at a time, we can use the fill-in-the-blank task to generate explanations on multiple mistakes. This is done by first comparing the answer given to the correct answer and finding the difference between the two and isolating each individual mistake. Each of these individual mistakes can be then applied to the original correct answer to create separate faulty answers which can subsequently be run through our system. Using this method, we can also change the fill-in-the-blank task to a simple response, where the student is tasked to correctly make all the changes to the PIE word to produce a PGmc word.

Either through separate models for separate rules (and enough annotations on the input) or by training a single more complex model, it may be possible to create a system that can caption all mistakes for a given task. This could work in conjunction with automatically grading the assignments to quickly provide feedback on the mistakes.

8 Conclusion

In this paper we show a method of the novel task of creating automated captions of mistakes. These captions serve as explanations for what the thought process behind the mistake was, and why it is not applicable in this case. We apply two methods to test our model, comparing against a baseline with a suite of automated metrics, and manually identifying and analyzing the sentences in a series of case studies. Our experiments show that our method has promise in creating these automated captions but that there are significant challenges that need to be overcome.

We set out tasks in the domain of Linguistics pedagogy, and in this we can make improvements with representation of the input data and with gathering more and more varied data. We also hope to expand this to include a wider set of rules covering a larger set of sound changes, and eventually evaluate the method using a human study. We hope that this method of mistake captioning can be applied to other fields of study.

References

Moriah Ariely, Tanya Nazaretsky, and Giora Alexandron. 2020. First steps towards nlp-based formative feedback to improve scientific writing in hebrew. *In proceedings of the 13th International Conference on Educational Data Mining*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Lyle Campbell. 2013. *Historical Linguistics: An Introduction*, ned - new edition, 3 edition. Edinburgh University Press.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O Riedl. 2019. Automated rationale generation: a technique for explainable ai and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 263–274.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Mehak Maniktala and Tiffany Barnes. 2020. Extending the hint factory: Towards modelling productivity for open-ended problem-solving. In *In proceedings of the 13th International Conference on Educational Data Mining (DC paper)*.

Samiha Marwan, Ge Gao, Susan Fisk, Thomas W Price, and Tiffany Barnes. 2020. Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 194–203.

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 of the Association for Computational Linguistics*, pages 311–318.

Kelly Rivers and Kenneth R Koedinger. 2017. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *International Journal of Artificial Intelligence in Education*, 27(1):37–64.

John Stamper, Tiffany Barnes, Lorrie Lehmann, and Marvin Croy. 2008. The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track*, pages 71–78.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Abhishek Unnam, Rohit Takhar, and Varun Aggarwal. 2019. Grading emails and generating feedback. *International Educational Data Mining Society*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.