Evaulating Student Writing

Stanford CS224N Custom Project

Anthony Riley
Department of Computer Science
Stanford University
ajriley@stanford.edu

Sohit Gatiganti
Management Science and Engineering
Stanford University
sohitg@stanford.edu

Chris Chankyo Kim
Department of Computer Science
Stanford University
chankyo@stanford.edu

Abstract

Writing is a critical skill for success and automated tools for evaluating written documents are becoming commonplace. While numerous examples of automated writing feedback tools exist, most either fail to identify writing structure or exist as proprietary algorithms that are inaccessible to educators of under-funded schools thus disproportionately affecting a number of low-income and minority students. To address this, the Bill and Melinda Gates Foundation is sponsoring a competition where NLP researchers get a chance to build a model to classify student writing into its key argumentative components, a difficult text classification problem that requires large, pretrained, state-of-the-art models. Our team presents several candidate models including a Longformer with Oversampling, and a combined Longformer and BigBird model, the best of which classified components with a 69.1% accuracy. You can find the open-source code for our experiments on GitHub: https://github.com/pythonicode/cs224n-final.

1 Key Information

- Mentor: Anna Goldie
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

In our modern world, proficiency in writing is a critical skill for success. Despite this, less than one-third of high school seniors in the United States demonstrate writing proficiency [1]. Additionally, low-income and minority groups, such as African-American and Hispanic students, show severe lack of proficiency with less than 15 percent of students demonstrating writing proficiency. Automated tools that evaluate and provide feedback on written essays presents a potential solution to help students improve their writing abilities with little to no-cost.

Identification of argumentative writing structure is a difficult natural language processing task that requires establishing abstract relationships between local and global textual contexts. Models employ these relationships to accurately predict whether a queried token contributes to a significant argumentative component in a written document. Historically, most NLP tasks have been limited to shorter lengths of text data due to the quadratic computational complexity of self-attention scoring. The state-of-the-art RoBERTa model for example, limits its query to 512 tokens as it becomes

computationally intractable for longer texts. As such, the field of automated writing feedback, and NLP analysis of larger texts, is ripe with opportunities to explore new ideas and methods.

In this project, our group seeks to explore a successful autoregressive learning model that identifies argumentative components of written documents - such as claims, evidence, rebuttal, conclusions, etc. Specifically, the scope of this project revolves around identifying components of argumentative writing by students. The methods and models presented in this project are part of our group's submission for the Kaggle competition *Feedback Prize - Evaluating Student Writing* hosted by the Bill and Melinda Gates Foundation in adjunct with Georgia State University [2]. In this paper, we present a model that achieved the 90th-percentile accuracy of 69.1% during the competition. For reference, the average inter-rater reliability of human evaluators is estimated around 73%, which is quite low for human baseline accuracy [2].

3 Related Work

Traditionally, the task of classifying argumentative components within a written corpus has been approached as a word level supervised text classification problem. As such, supervising learning models were commonly used used to distinguish basic claims and conclusions in narrative media such as news articles and personalized essays [3]. While simple in design, these models exploit the sequential nature of argumentative writing and have been demonstrated to be effective. Stab and Gurevych [4] used State Vector Machines (SVM) to annotate argumentative claims in essays and Potash et al. [5] demonstrated the use of sequence-to-sequence Recurrent Neural Networks (RNN) to infer the expected type of argument component.

In recent years, the introduction of multi-task models have been shown to be highly beneficial for the field of writing segmentation in NLP. Inspired by human ability to pursue multiple tasks in parallel while exploiting similarities between assignments to expedite completion, new machine learning models that used the multi-task framework faced significant improvements in both computational efficiency and diagnostic accuracy. Yang et al. [6] demonstrated that multi-task modeling of multi-language models can improve performance in cases where the dataset is partially labeled due to the cross-reference nature of multi-task processing. Mensonides et al. [7] further showed that the multi-task framework can be used with deep neural networks to solve complex tasks that require nuanced understanding of natural language by exploiting shared weight parameters of simple tasks, such as part-of-speech chunking.

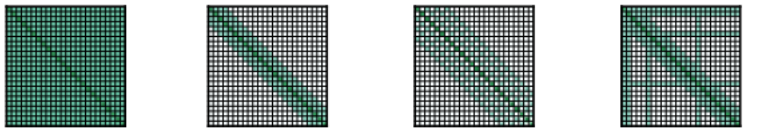
Alongside multi-task modeling, transformers such as Google's BERT [8] and its optimized pre-trained incarnation RoBERTa [9] have recently gained popularity for achieving state-of-the-art performance in both generative language modeling and discriminative language understanding. Transformers achieve peak performance by using their self-attention component to capture contextual information from the entire sequence, whereas previous models were primarily sequential. However, the computational requirements of this strategy is intractable and complexity grows quadratically with linear growth in sequence length, making it impractical to process long sequences. Fortunately, the recent LongFormer model exploits the multi-task framework to use a stripped down attention matrix that improves the transformer computational efficiency while supplying good performance [10].

4 Approach

We primarily used the Longformer architecture with some modifications during our experiments and BigBird, another long document transformer, which we've omitted from this section for brevity.

Longformer Architecture

The Long-Transformer (Longformer) is a state-of-the-art Transformer model designed for processing long sequences. Like we saw in Assignment 5, Transformer attention scales quadratically with sequence length. Since our model will be reading and classifying long pieces of text (some over 4000 words) we require a faster attention computation. Allen AI's Longformer reduces computational complexity by using a modified attention schema that scales linearly with sequence length. In particular, Longformer utilizes a sliding window attention pattern where given a window w, each token captures $\frac{1}{2}w$ tokens on either side, running at $O(n \times w)$. This scheme can be diluted by a factor l to capture a larger receptive field, and thus deeper relationships between tokens without sacrificing



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 1: (left to right) Complexity visualization of traditional attention scoring, sliding window attention, dilated sliding window, global-sliding window

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Figure 2: Equation to compute Transformer model attention score given linear projections Q, K, V. The LongFormer model uses two sets of projections Q_s, K_s, V_s and Q_g, K_g, V_g to compute attention scores of sliding window and global attention respectively

complexity, running at $O(n \times l \times w)$[10]. Since the windowed and dilated attention are not flexible to learn task-specific representations, Allen AI introduces a global attention scheme on pre-selected input locations that attends to all tokens across a sequence and all tokens in the sequence attend to it [10]. See *Figure 1* for visualized comparisons of attention models and their relative complexity.

Gradient Accumulation

Since our GPUs only have 16GB of memory and we are training very large transformer models, it became nearly impossible to use a batch size greater than 1 or 2. This meant that for each forward pass, there would be a single backpropogation. Not only is this extremely slow, but it could also lead to erroneous weight updates due to mislabeled/poor data. Gradient accumulation fixes this issue by effectively simulating a higher batch size by computing and storing the gradient for the last N forward passes, averaging them, and then doing a single backpropagation step with this mean gradient. This leads to more accurate weight updates and faster training. We used Gradient Accumulation as one of the strategies to train a model and it showed improved results which you will see in the next section.

5 Experiments

5.1 Data

Our primary dataset was provided directly by the hosts of the Kaggle competition and consisted of 15,600 labelled essays and a small sample testset with 5 labelled essays. Additionally, we used a third-party open-source dataset called Argument Annotated Essays [11], a corpus of roughly 1000 samples of student writing that we used to augment our dataset and to provided additional pretraining data to our language models using an unsupervised masked language modelling task.

5.2 Evaluation method

Our evaluation method is provided by Kaggle which runs our model on a hidden testset and evaluates the overlap between ground truth and predicted word indices [2].

Reported accuracy is calculated as follows.

1. For each sample, all ground truths and predictions for a given class are compared.
2. If the overlap between the ground truth and prediction is >= 0.5, and the overlap between the prediction and the ground truth >= 0.5, the prediction is a match and considered a true positive. If multiple matches exist, the match with the highest pair of overlaps is taken.
3. Any unmatched ground truths are false negatives and any unmatched predictions are false positives.

The final score is arrived at by calculating TP/FP/FN for each class, then taking the macro F1 score across all classes [2].

5.3 Experimental details

We ran many experiments to determine how to train the best model for the task, starting with a baseline single pretrained Longformer finetuned for text classification on the Kaggle dataset using NER token labels (e.g. "B-Claim", "I-Claim", "B-Evidence" etc).

Single Longformer NER Text Classification

Our baseline model was based on Chris Deotte's Longformer model for NER, but we modified the model to be used in PyTorch (https://www.kaggle.com/cdeotte/tensorflow-longformer-ner-cv-0-633).

```
MODEL_NAME = 'allenai/longformer-large-4096'
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForTokenClassification.from_pretrained(MODEL_NAME)
model.config.num_labels = 15 # NER labelling with 7 classes
```

For this model we still did not have access to an Azure virtual machine so we trained the model using Kaggle's free GPU hours. We used a similar training schema to the notebook linked above.

Batch Size	Epochs	Optimizer	Learning Rate	Total Iterations	Total Time
2	5	Adam	0.25e-4	38985	5.5 hours

The only difference is that Chris Deotte used a scheduler for the optimizer so that during the last epoch the learning rate was reduced by a factor of 10 which may have had some performance benefits which we explored in our further experiments. Overall, our model performed fairly similarly to the example model at **62.8%** accuracy compared to 63.1 % accuracy which was a great baseline model.

BigLongBirdformer: Combined Longformer and BigBird Model for NER

Our first idea to improve on the simple Longformer model architecture was to combine the Longformer and BigBird models into a new state-of-the-art model: the BigLongBirdformer.

The idea behind the model was that maybe using contextual embeddings from two different transformers would provide better contextual information to the feed-forward layers thus better identifying which component each token belongs to. We simply ran each sequence through a Longformer and a BigBird model, concatenated the embeddings and ran them through a feed-forward layer with dropout.

```
longformer_out = self.longformer(input_ids, attention_mask)
bigbird_out = self.bigbird(input_ids, attention_mask)
sequence_output = torch.cat((longformer_out.last_hidden_state,
                             bigbird_out.last_hidden_state), dim=2)
logits = self.output(self.sequence_drop(sequence_output))
return logits
```

What was quite surprising is that when we started training the model it was performing significantly worse, loss was plateauing at around 1.5 per step and accuracy on the training set was plateauing around 25-30% (see Figure 3). We tried to fix the model by changing the learning rate, model architecture, and scheduling function, but for an unknown reason the metrics were still plateauing.

—	Batch Size	Epochs	Optimizer	Learning Rate	Loss	Accuracy
First Attempt	2	5	Adam	0.1e-3	1.65	26%
Second Attempt	2	5	Adam	0.25e-4	1.6	30%
Third Attempt	2	5	Adam	0.3e-5	1.58	30%

Note: Loss and Accuracy columns represent the approximate value at which they plateaued.

We suspect that the issue lies within the difference between BigBird and Longformer architectures. Somehow by concatenating the BigBird embedding to the Longformer embedding, the new embedding did not allow the model to learn and despite many hours trying to debug the model, we couldn't figure out why it failed nor how to fix it.

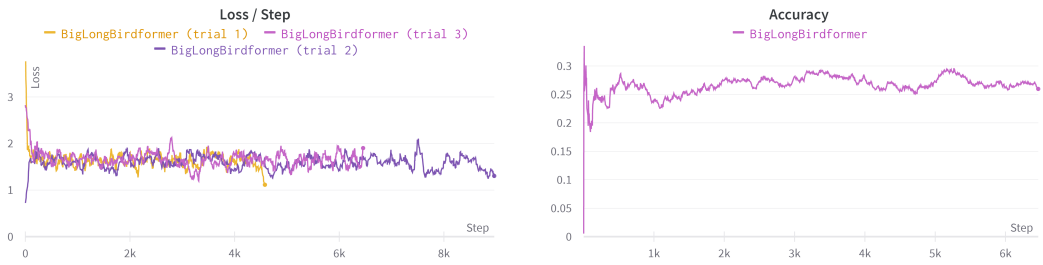


Figure 3: Loss and accuracy graphs over 1 epoch of training the BigLongBirdformer.

Longformer with Pretraining

To make use of the Argument Annotated Essays dataset, we decided to try additional pretraining of the Longformer using a Masked Language Modelling task. Our idea was that training the Longformer on student essays would make the embeddings more accustomed to student writing thus making the model make better use of the contextual word embeddings during fine-tuning.

We used the Argument Annotated Essays to create a dataset and used a data collator function to randomly masked out tokens with a 0.15 probability. We then loaded AllenAI's pretrained Longformer and trained the model embeddings to predict the masked tokens.

```
model = LongformerForMaskedLM.from_pretrained(LONGFORMER_PATH)
dataset = MaskingDataset(pretrain_df)
collate = DataCollatorForLanguageModeling(tokenizer)
tr_args = TrainingArguments(output_dir='./output')
trainer = Trainer(model, tr_args, collate, train_dataset=dataset)
trainer.train()
torch.save(model, './output/pretrained_longformer.bin')
```

There were two steps to training this model: we trained the Longformer on the Masked LM task and saved the embeddings, then we finetuned the Longformer on token classification.

Pretraining

Batch Size	Epochs	Optimizer	Learning Rate	Total Iterations	Total Time
2	10	Adam	5e-5	450	6 min

Finetuning

Batch Size	Epochs	Optimizer	Learning Rate	Total Iterations	Total Time
2	5	Adam	0.25e-4	38985	5.5 hours

After running the model on the test set we were surprised to get an accuracy of 62.7 % which was *lower* than the model without additional pretraining! This is likely because the size of the dataset was so small at only 90 samples, despite running training over 10 epochs there was not much of a difference.

Sentence-Based Classification with BERT

Since argumentative components of essays are *usually* sentences, we thought a sentence-based classifier would naturally produce respectable results. Therefore, we reformatted the dataset to have individual sentence or participles map to class labels. We then used HuggingFace to train a BERT-base models on a sequence classification task.

```
model = AutoModelForSequenceClassification.from_pretrained(MODEL_CHK,
                                    num_labels=NUM_LABELS)
...
trainer = Trainer(
    model=model,
    args=training_args,
```

```
    train_dataset=ds_train_tokenized,
    eval_dataset=ds_val_tokenized,
    tokenizer=tokenizer,
)
trainer.train()
```

We varied the model to train on both bert-base-uncased and bert-base-cased to see if lower-casing the entire essay would improve accuracy at all.

Sentence Based Classifier Training Args (for both models)

Batch Size	Epochs	Optimizer	Learning Rate	Weight Decay	Total Iterations
32	5	Adam	1.5e-5	0.01	53125

Sentence Based Classifier Model Training Results

Model Name	Training Loss	Validation Loss	Training Time
bert-base-uncased	0.721	1.071	3.2 hours
bert-base-cased	0.697	0.992	3.1 hours

Interestingly, the cased model had better performance with a 26.1 % accuracy compared to the uncased 22.4 % accuracy. We believe this is because the use of proper nouns, acronyms or any other capitalisation within essays provide valuable information for the model to distinguish between classes.

Longformer with Oversampling

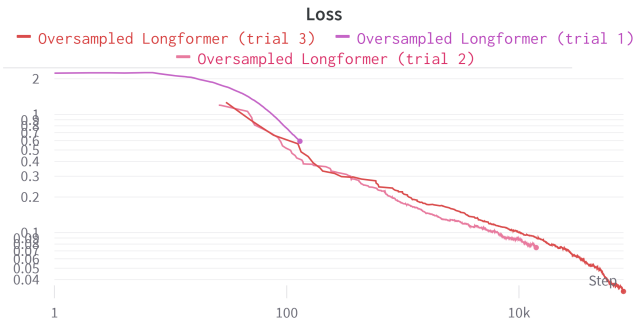
We quickly realised that using Longformers were producing the best results by quite some margin, so we settled on using an overall architecture around it. In our final experiment we again used AllenAI's pretrained Longformer with attention followed by two dense layers (see Appendix).

After finalising the model architecture, we decided to oversample the data as some classes were severely under-represented; classes such as Rebuttal and Counterclaim only represented 3% and 4% of the data respectively. Therefore, created a new dataset by randomly choosing extra examples to improve the ratio between each argumentative component. The distributions before and after can be seen below.

Oversampling the Data

Class	Before Count	After Count	Percent Increase
Claim	13691	24337	77
Concluding	12329	22274	80
Counterclaim	4243	15174	257
Evidence	14270	25175	76
Lead	8585	16008	86
Position	14090	25012	77
Rebuttal	3339	14270	327

Finally, we trained the model using this oversampled dataset and gradient accumulation (as described in the previous section). As we can see from the loss graph, we had a steady decrease in loss throughout each epoch.

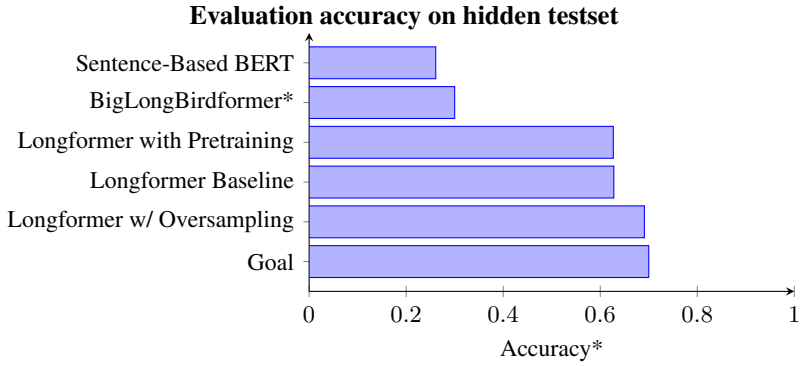


The graph is disjoint as our training crashed twice and had to be restarted.

Batch Size	Epochs	Optimizer	Learning Rate	Total Iterations	Total Time
1	8	Adam	4.5e-5	201800	16 hours

After training this new model with an additional dense layer, gradient accumulation, and oversampled data we were suprised to get a much better accuracy on the hidden testset of 69.1 %.

5.4 Results



Recall accuracy is reported as macro F1 score across classes.
** Not evaluated on hidden testset, accuracy comes from training data.*

Above are the accuracy scores for each of the models from our experiments. We did not reach our goal accuracy of 70% which was a lot more difficult to reach than anticipated.

We were quite suprised to find that the Longformer that was pretrained with additional data performed worse than when loading the original pretrained embeddings. The BigLongBirdformer also performed much worse than expected. Something about the architecture of the combined model didn't allow for it to be able to accurately classify tokens. We expected an improvement over the single Longformer because of the additional information coming from the BigBird word embeddings but the BigLongBirdformer ended up performing significantly worse than the Longformer at approx. 30 % accuracy vs 62.8 % accuracy.

The Longformer with Oversampling model surprisingly boosted our accuracy by over 6% a hefty improvement to the baseline Longformer. We attribute this to the additional feed-forward layer and extra training time on data that was better distributed because of the oversampling.

6 Analysis

It was immediately clear from discussion with other competitors that Long-document Transformers like BigBird and Longformer were performing significantly better than other transformer based models like BERT and RoBERTa.

These reasons were discussed in a previous section, but the benefits of using both local and global attention across a sequence provides valuable information about the broader context of a token's role in an essay, and when it comes down to making predictions as to whether a token belongs to a larger component, this long distance context becomes increasingly important.

If we take a look at the actual outputs of our model we see that it does a relatively good job at identifying components (see Appendix below), with a few critical mistakes. The model is not good at identifying the start and end of argumentative components, often being off by a few tokens. This could've potentially been solved by the method described above where the model learns to classify the start and end of components rather than all the tokens that are part of a component. It also sometimes completely fails at identifying a component, instead leaving it unlabelled. This is especially comfusing in the first example where the concluding statement isn't labelled when it is quite obvious to a human that it's a concluding statement.

7 Conclusion

7.1 Achievements

With 3 days remaining in the competition, we are **placed 188 out of 1997 teams**. We got close, but did not reach the top 5% of teams in the competition like we'd hoped, although our goal seemed a bit ambitious considering the size and complexity of the models the top competitors were experimenting with.

Despite not reaching our goal, the model performed with very high accuracy: the best models in the competition performed less than 5% better than our model and we performed less than 4 % worse than human annotators. We are particularly proud of how well the model seems to perform on example essays. Looking at a sample from the Appendix, our model generally does a great job capturing components and it's very rewarding to see our model generating such plausible output.

7.2 Limitations

There were many difficulties, challenges and limitations that came with performing our experiments.

Despite having 16 GiB on our GPU we frequently ran out of memory when trying to train our models. This was a huge issue and led us to adapt our models with gradient accumulation and using small batch sizes. This did not fix all our problems however, many of the other competitors used AllenAI's large pretrained Longformer while we could only use the base model because of memory limitations. Competitors using the large model achieved significantly better results which is not unexpected because the model has 50 % larger embeddings and 4x the number of parameters. We still ran out of credits several times so it was definitely a surprise to see how much compute played a role in our progress.

7.3 Future Work

An interesting idea is whether or not using NER token classification was the best choice for the task. Our group took the NER method at face value and decided to use it for all of our models, but it's not clear whether that's the best way to teach a model how to label argumentative components. NER definetly has it's strengths: it allows for more variations in results based on post processing the data to create labels, but it also doesn't really line up with the way that a human being would identify an argumentative component. There was some discussion and interest in trying a model that tried to predict the start and ending token of each argumentative components which better matches how a human would classify them, but we didn't get a chance to explore this model.

Another things that we would've tried if we had more computing resources is training a model using k-fold Cross Validation. Because the training dataset was relatively small (15600 essays at around 10 million tokens total) we would've liked to use all of the training data in our model. In order to do so while maintaining a validation set, we could've created 5 folds of the data. If you are unfamiliar with cross validation, the basic idea is that we would take our dataset and split it 5 different ways (thus the 5 folds idea) where 20 % of the data is for validation and 80 % of the data is for training. The validation set would be rotated so that each piece of data is in 1 of the 5 validation sets. Thus the model can train and validate on all of the data making better use of a small dataset.

References

[1] National Assessment of Educational Progress. 2011 Writing Assessment. U.S. Department of Education, 2011.

[2] Georgia State University. Feedback Prize - Evaluating Student Writing. Kaggle, 2022.

[3] Roland Kluge Judith Eckle-Kohler and Iryna Gurevych. On the role of discourse markers for discriminating claims and premises in argumentative discourse. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

[4] Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. Computational Linguistics, 2017.

[5] Alexey Romanov Peter Potash and Anna Rumshisky. Here's my point: Joint pointer architecture for argument mining. arXiv preprint arXiv: 1612.08994, 2016.

[6] Ruslan Salakhutdinov Zhilin Yang and William Cohen. Multi-task cross-lingual sequence tagging from scratch. arXiv preprint arXiv: 1603.06270, 2016.

[7] Jacky Mountmain Jean-Christophe Mensonides, Sebastien Harispe. Automatic detection and classification of argument components using multi-task deep neural network. Proceedings of the 3rd International Conference on Natural Language and Speech Processing, 2019.

[8] Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. volume abs/1810.04805, 2018.

[9] Naman Goyal Jingfei Du Mandar Joshi Danqi Chen Omer Levy Mike Lewis Luke Zettlemoyer Yinhan Liu, Myle Ott and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. 2019.

[10] Arman Cohan Iz Beltagy, Matthew E. Peters. Longformer: The Long-Document Transformer. In *Computation and Language*, 2020.

[11] Christian Stab and Iryna Gurevych. Argument Annotated Essays. Technische Universität Darmstadt, 2014.

A Appendix

Longformer with Oversampling Architecture

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_ids (InputLayer)          [(None, 4096)]       0
__________________________________________________________________________________________________
attention_mask (InputLayer)     [(None, 4096)]       0
__________________________________________________________________________________________________
tf_longformer_model (TFLongform TFLongformerBaseMode 148659456   input_ids[0][0]
                                                                 attention_mask[0][0]
__________________________________________________________________________________________________
head/fc1 (Dense)                (None, 4096, 768)    590592      tf_longformer_model[0][13]
__________________________________________________________________________________________________
head/fc2 (Dense)                (None, 4096, 192)    147648      head/fc1[0][0]
__________________________________________________________________________________________________
head/classifier (Dense)         (None, 4096, 9)      1737        head/fc2[0][0]
==================================================================================================
Total params: 149,399,433
Trainable params: 149,399,433
Non-trainable params: 0
__________________________________________________________________________________________________
```

True Annotations

6FD0E82AB64D

While the idea of a computer being able to reconize the emotions of sudents in the classroom is an intresting one, it simply wouldn't work well enough when put into action. **Position**

When it tells of how it could modify the lesson when it recognizes a student becoming bored sounds reasonable, when you think about it, having a computer changing a lesson simply due to possibly a single student becoming bored with it, would not only make the lesson longer, but maybe even worse off than it might've been. **Claim**

There is also te question of what if the computer gets something wrong or has a glitch? what if it accedentily misinterpruts sadness as anger for example. Not only will this cause a plethera of complications, it may also lead to the computer making other mistakes in the process. **Evidence**

While it is an intresting idea, it really needs to be utterly perfect in every way or else it could cause many problems. **Concluding Statement**

FAAC3CC5476F

Hello today I'm going to be talking about the majority of humans that own and operate cell phones on a daily basis. **Lead** First I would like to say that driving and operating a cell phone at the same time wouldn't be a great idea. **Position** In this day of age 2020 everyone rely on there cell phones to do everything some people rely on it to get their destination using a gps located on your cell phone. Or either texting someone on there phone instead of going to them face to face and talking. **Evidence**

Now that I've told yawl about the use of cell phones in are day and age lets talk about should or shouldn't people not be able to use cell phones in a vehicle. Personally I think people shouldn't be able to be on there phones while driving. It's a safety hazard **Claim** and if you would look at the news the you would see how many car accidents there are. People lost there lives because other people want to be on their cell phones that's not fair at all. That's why I feel the way I feel about cell phones and driving it doesn't have to be like that if people would put there phones down and drive. **Evidence**

And for all the people that can't stop texting and driving should be punished. **Claim** Cause they have other people family die it isn't right. If there was no cell phones then all the lives and car accidents could have been prevented. **Evidence**

Model Annotations

6FD0E82AB64D

While the idea of a computer being able to reconize the emotions of sudents in the classroom is an intresting one, it simply wouldn't work well enough when put into action. When it **Position** tells of how it could modify the lesson when it recognizes a student becoming bored sounds reasonable, when you think about it, having a computer changing a lesson simply due to possibly a single student becoming bored with it, would not only make the lesson longer, but maybe even worse off than it might've been. There is also **Claim** te **Evidence** question **Claim** of what if the computer gets something wrong or has a glitch? what if it accedentily misinterpruts sadness as anger for example. Not only will this cause a plethera of complications, it may also lead to the computer making other mistakes in the process. While it is an **Evidence** intresting idea, it really needs to be utterly perfect in every way or else it could cause many problems.

FAAC3CC5476F

Hello today I'm going to be talking about the majority of humans that own and operate cell phones on a daily basis. First **Lead** I would like to say that driving and operating a cell phone at the same time wouldn't be a great idea. **Position** In this day of age 2020 everyone rely on there cell phones to do everything some people rely on it to get their destination using a gps located on your cell phone. Or either texting someone on there phone instead of going to them face to face and **Evidence** talking. Now that I've told yawl about the use of cell phones in are day and age lets talk about should or shouldn't people not be able to use cell phones in a vehicle. Personally I think people shouldn't be able to be on there phones while driving. It's a safety hazard and if you would **Claim** look at the news the you would see how many car accidents there are. People lost there lives because other people want to be on their cell phones that's not fair at all. That's why I feel the way I feel about cell phones and driving it doesn't have to be like that if people would put there phones down and drive. And for all the people **Evidence** that can't stop texting and driving should be punished. Cause they have other people family die **Claim** it isn't right. If there was no cell phones then all the lives and car accidents could have been prevented. **Evidence**