Giving *Attention* to the Colors-in-Context Encoder-Decoder Model

Sylvia Yuan
Stanford CS Department
luyuan@stanford.edu

Bryan Chia
Stanford Statistics Department
bryancws@stanford.edu

Nicholas Paul Brazeau Sanchez
Stanford CS Department
npbsanc@stanford.edu

Abstract

We approach the colors in context task first proposed by Monroe et. al with transformer based encoding and decoding scheme. We leverage generic transformers to encode and decode our model, but we also creatively adapt the T5 encoder-decoder foundation model to accept color encodings as input. We find that the additional power of these models, and in particular, the ability of the decoder to attend perfectly to all encoded inputs without a vanished gradient, allows for these models to improve modestly over the accuracy metrics of the base listener in Monroe et. al's paper. Still, we challenge the notion that these models produce color descriptions that are strictly better at helping humans discern colors in context by comparing and contrasting weaknesses of these models.

1 Introduction

For language models to truly understand natural language as humans do, it needs to be grounded in real world representation. The current paradigm for word embeddings, the quanta by which natural language models represent elements of a piece of natural language, is that these embeddings represent the co-occurrence of these units in some way, which can often be an effective ersatz metric for the similarity or difference of given words. Although understanding the similarity and differences of these words is a necessary condition for understanding language, it is not sufficient. Proponents of grounding in natural language posit that by grounding our language in real world representations of meaning, we can develop models with a deeper and more effective understanding of their wielded word embeddings.

In Colors in Context, Monroe et. al developed a natural language model with a rich, grounded understanding of colors by giving it color based, non-text inputs. The task they developed for their paper, color reference, requires a speaker to, given an arbitrary set of colors and knowledge of a "target" color in the aforementioned set, to effectively describe the target color with natural language such that another agent could identify it. This task is far more complex than simply describing a color, which only requires that a model learn to map representations of color to word embeddings. To accurately describe a color in context, a model must develop an understanding of how to pragmatically describe the target color such that a user could select it, which requires nuanced, natural language based understanding of differences between colors.

Monroe et al. developed an LSTM based encoder-decoder model that, while performing to subhuman standards on their metric of speaker accuracy, still performed well. Yet one of the weaknesses of RNN based encoder-decoder models is their vulnerability to vanishing gradients. As encoder-decoder models are trained on longer sequences, the error for an RNN cell earlier in the sequence becomes more muted by the error later in a sequence, resulting in the resulting gradients to correct these errors vanishing. This may not seem like a pernicious problem in the case of the color reference task, since inputs will typically be only three colors, and outputs will typically be a curt word or phrase that describes a color ("bright blue", for example). However, in the "close" version of the dataset, color inputs tend to be very similar to each other. In these instances, training descriptions for the colors are not only the most verbose, but these also tend to be the examples in which the model is least accurate. Thus, in the most challenging cases a model could face, the vanishing gradient could be a significant problem.

A common antidote to models vulnerable to vanishing gradients is leveraging transformer models, which allow future model outputs to perfectly attend to all prior input and output states. These models have defined the NLP state of the art for some time, and while support exists for transformers to be trained from scratch on new tasks, pretrained foundation transformers have also been leveraged to great success on numerous NLP tasks. In this paper, we apply various transformers, pretrained and otherwise, to the color reference task. We hypothesize that these models' ability to sidestep the vanishing gradient problem, and further, the extra language information embedded within pretrained transformer models, will afford greater understanding and model accuracy over the base encoder-decoder color reference models.

2 Prior Literature

Monroe et al. (2017) present an LSTM-based encoder-decoder model for interpreting referring expressions in grounded communication, specifically identification of colors from textual description. The original paper introduces a colors reference dataset, which we used for this paper. This dataset is of the following format: three colors are presented to the agent, and number of trials are dispersed evenly according to three different conditions (far, split, and close). The far condition means that the colors are far apart (very different), the split condition means that two colors are similar while the remaining one is far from them, and the close condition means that all three colors are similar. This dataset enables testing of pragmatic language interpretation. We also gained detailed information about the dataset like the expected accuracy of the tasks in the dataset. As introduced in the paper, human accuracy is different on the three conditions and can be as high as 83 percent on the close condition.

We contrast this paper with Du et al. (2021), which introduces a transformer based approach for visual grounding and introduces a method that does not rely as heavily on pretrained detector or proposal-free frameworks. The method (Visual Grounding with Transformers - VGTR) is independent of pretrained detectors and pretrained word embeddings. Given textual description, VGTR can capture context semantics of the visual and linguistic aspects and locate objects of interest. It encodes visual and textual input simultaneously, allowing for input text embeddings to attend over themselves, and then for portions of the image to attend to those post-encoded attention embeddings. The model then decodes over the final bounding box output by attending over the previously transformer encoded image embeddings. The model encodes the input text and image via a convolutional neural network and an LSTM, respectively. It suggests that transformers can be leveraged effectively in tasks that harness visual and textual data.

This result, that transformers can be effectively leveraged to encode and decode visual and textual data simultaneously, is corroborated with the findings of Deng et al. (2021), who propose TransVG, a transformer-based framework for visual grounding to address the task of grounding a language query to the corresponding region onto an image. For instance, when asked "where is the apple" in a basket of various fruits, the model tries to draw a bounding box around the correct fruit. Rather than splitting up the creation of visual and linguistic features to two separate models and limiting the interaction between both modalities, TransVG leverages transformers to allow for interactions between the two. It also shows that using a simple stack of transformer encoder layers has a higher performance than more complex fusion modules. These two papers give us confidence to approach the colors in context task with transformers.

3 Data

We used the colors in context dataset created by Monroe et al. (2017). The dataset was sourced from volunteers playing a pragmatic color reference game in English, in which one user was given a set of three colors, along with knowledge of a target color in the set, and had to describe this target color such that their partner could accurately guess it. The dataset spanned 938 different games, and is comprised of 53,365 utterances. Moreover, the dataset was split into three different conditions, a far, split, and close far subset.

4 Model

To represent our colors, we followed Monroe et al. (2017) approach of applying the Fourier transform over the HSV representation of different colors in the dataset. We also preprocessed text labels using the Monroe et al. (2017) approach by removing punctuation and cutting off the -er, -est, and -ish suffixes from all words.

4.1 Training a Transformer Encoder-Decoder from Scratch

We start off with a basic transformer encoder and transformer decoder architecture as seen in

`PyTorch's Transformer Module`. This replaces the LSTM-based encoder and decoder architectures that was used in Monroe et al. (2017). We experimented with a couple of different preprocessing methods (for colors and text), as well as hyperparameters.

1. **Preprocessing the Color Sequence**: Passing the color sequence through an LSTM, *or* adding position embeddings and passing the color encodings through a linear projection as the entry point for the transformer encoder.
2. **Preprocessing the Text Sequence**: GloVe and BERT embeddings (fine-tuned or fixed during training)
3. **Model Architecture**: Hyperparameter tuning(we vary encoder layers, decoder layers, feed-forward dimensions, dropout probability, and the number of multi-head attention models).

4.2 The T5 Transformer

The T5 model is an encoder-decoder foundation transformer designed specifically to be generalized and fine-tuned for various downstream contexts involving texts as inputs, outputs, or both Raffel et al. (2019). To this end, the T5 model treats every task as a text-to-text task. If the inputs of a task aren't already neatly expressed as text, then they are to be converted to or deconverted from text, or at least a text embedding. In our case, we needed to convert representations of colors into tokens that our model could interpret. We tested several approaches to encoding colors into our model, and found that the following works the best: each color is put through a three layer perceptron which outputs a single T5 embedding representing each color. We then append to this sequence another T5 embedding generated from the perceptron output of all three color embeddings stacked atop each other, an embedding that references all three colors, in other words. This extra word allows the model auxiliary information about the colors in context. We used T5-smallv1.1 to process these four tokens. This modeling approach for T5 inputs exceeded other approaches, such as the base T5 model, or using a one dimensional projection layer for inputs, in terms of numerical accuracy. T5 comes in-built with its own tokenizer and end tokens, which we also used.

4.3 Fine-tuned GPT-2 Transformer

Generative Pre-trained Transformer 2(GPT-2) is a decoder-only transformer model trained for text generation on internet texts (Radford et al., 2019). Trained on a large dataset, GPT-2 can predict the next token based on an input sequence. GPT-2 can also be used for classification: since every token carries information from previous tokens in the text sequence, the last hidden state of the model contains information about the entire sequence and can be used for classification. With this idea in mind, we experimented with the GPT-2 model for this task. We finetuned a pretrained GPT-2 model from the Huggingface transformers library on texts from the Colors-in-Context dataset. The GPT-2 model is designed for handling the text sequences, but not the color sequences. We attempted two different methods for handling the color sequences inputs: (1) putting the color sequences through an LSTM layer and pass the resulting encoder hidden states through the GPT-2 model, and (2) using a similar method based on passing color through a three layer perceptron as explained in the T5 transformer section to obtain a color embedding as part of the input through the GPT-2 model. The second approach with the three-layer perceptron structure achieved better results. Unfortunately, a pre-trained decoder-only model did not turn out to be the best architecture for this task and did not achieve comparable accuracy on the task as the T5-based transformer model.

5 Methods

For all models, we initially optimized over 1000 epochs, activating early stopping when a model failed to improve accuracy on a validation set for 10 epochs.

5.1 Transformer Encoder-Decoder

As mentioned earlier, we wanted to investigate the best way to preprocess data, as well as the best hyperparameters suited for our task.

1. **Preprocessing the Color Sequence**: One early finding is that adding position embeddings to the color sequence and passing them through a linear projection to the transformer encoder prevented our model from converging. We found that the model was only learning about word sequences (for example what words go together and what words are com-

mon), and not about the colors. Usually, position embeddings are used because we want to consider relative and not absolute positions. However, that is overkill in this case where the color sequences are always standardized. Furthermore, the positional embeddings could have created additional noise for the sensitive color representations. Using positional embeddings also went against a suggestion from Monroe et al. (2017) to allow the target color to have the most influence on the word sequence during training. After switching the positional encoding and linear projection combination to an LSTM that creates a hidden state for the three colors passed in sequence that would be passed to the transformer encoder, our model started converging.

2. **Preprocessing the Text Sequence**: We experimented with using GloVe and BERT embeddings, and further explored allowing the embeddings to be fine-tuned or fixed during training. We found that allowing fine-tuning has a modest (1%) improvement during validation. BERT embeddings required too many parameters for a task that required relatively little encoding dimensions, and thus also converged to values far smaller than using just a 50-dimensional GloVe embeddings.

3. **Model Architecture**: In short, only dropout mattered. Fig. 1 shows the different training accuracy scores for different levels of dropout, which we address further in Sec. 7.1. Feed-forward dimensions played the smallest role in helping convergence. We chose 128 dimensions to be in line with Monroe et al. (2017), which used roughly 100 dimensions to represent the hidden state of the color dimension. Generally, lesser encoder layers (3) than decoder layers (6) were necessary since the color sequence required less encoding than the word sequences. We also selected a higher multi-attention head number (10) as it was seen to be marginally helpful.

5.2 The T5 Transformer

For the T5 model, our best model applied 20 percent dropout at the perceptrons' 2nd layer, using ReLU as our nonlinearity and hidden size 300. We also found that the small, unfrozen version of T5 works best for the model. Using t5-basev1.1 saw a slight decrease in the accuracy metric, and took much longer to train; we believe that the small version of the model is powerful enough to handle the task.

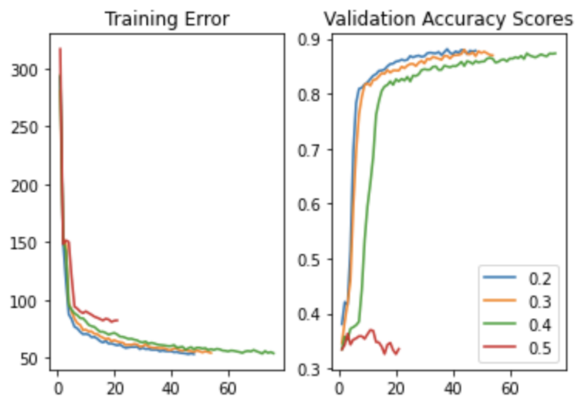


Figure 1: **Effect of Dropout on the Transformer Encoder-Decoder**: the Transformer Encoder-Decoder is unable to converge at dropout levels higher than 0.5, and performance is considerably similar for dropout levels from 0.2-0.4.

5.3 The Fine-tuned GPT-2 Transformer

We fine-tuned the pre-trained GPT-2 model for 5 hours on a single-gpu on the Colors-in-Context corpus. With a similar 3-layer perceptron structure, we converted the input color sequences to embeddings for the GPT-2 model and propagated the text sequences through GPT-2 to get output logits. We found that this fine-tuned GPT-2 model performs much better than a smaller GPT-2 model trained from scratch from the Colors-in-Context dataset, which achieved only slightly better results from random guesses. In this experiment, the hidden size of perceptrons was also 300. For fine-tuning the GPT-2 model, we used an AdamW optimizer with learning rate 0.00001 and applied gradient clip norm of 1.0 and a batch size of 32. For training the transformer, we used a batch size of 16 and an AdamW optimizer with the same parameter values.

6 Results

Tab. 1 and Tab. 2 show the test and validation results respectively. On the test set, the model with the highest BLEU and Listener Accuracy was the baseline GRU model. Its accuracy was trailed, slightly, by the T5 base transformer's, but the T5 model's BLEU was the worst of any model observed. On the validation set, the T5 and transformer model exceeded the baseline model's speaker accuracy by at least 3 percentage points,

Model	Listener Accuracy	BLEU Score
Human Performance From Monroe et al. 2017	91.1%	
Baseline Model GRU with GLoVE Embeddings	92.5%	0.72
Transformer Encoder-Decoder	91.6%	0.71
T5 Transformer	92.3%	0.63
GPT-2 Transformer	73.5%	

Table 1: Results on Colors Data Test Set

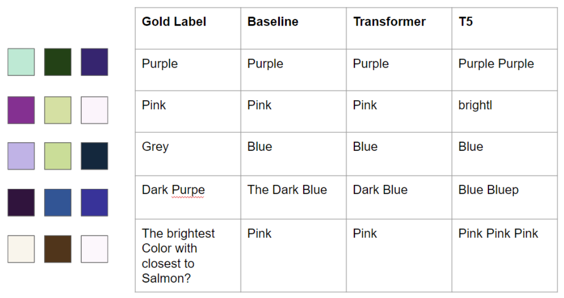
Model	Listener Accuracy	BLEU Score
Baseline Model GRU with GLoVE Embeddings	83.9 %	0.500
Transformer Encoder-Decoder	87.7 %	0.501
T5 Transformer	87.1%	0.506
GPT-2 Transformer	73.7%	

Table 2: Results on Colors Data Validation Set

and the BLEU scores were all equal. Fig. 3 also shows that the T5 Transformer and Transformer Encoder-Decoder converge quicker than the GRU baseline even though each epoch takes marginally longer; this could be due to overfitting, though.

We postulate that another reason that the transformer models did relatively better than baseline on the validation, but were only on par with the baseline on the test set was due to the underlying distribution of split-close-far examples in each of the validation and test sets. The validation set has 2% more split and close examples than the test set. There are many more difficult examples by raw counts as well as the validation set includes roughly 12,000 examples as supposed to 2000 examples in the test set.

The GPT-2 model performed worse on both sets, with around 20 percent lower on the test set and around 15 percent lower on the validation set. We suspect the reason for the difference to be that GPT-2 is a decoder-only model, unlike the other models we experimented with, which may not perform well given a task with color sequences as inputs as well as the text sequences. Another reason might be the size of the pre-trained GPT-2 model. The pre-trained version from the Huggingface transformers library has a vocabulary size of 50257, with many tokens not appearing in the Colors-in-Context corpus. Relating to the fact that the small T5 transformer achieved the best performance, we suspect that the size of the GPT-2 model and the excess vocabulary in the original training corpus might be one reason for the difference in performance.



			Gold Label	Baseline	Transformer	T5
🟩	🟩	🟪	Purple	Purple	Purple	Purple Purple
🟪	🟨	⬜	Pink	Pink	Pink	brightl
🟪	🟩	⬛	Grey	Blue	Blue	Blue
⬛	🟦	🟦	Dark Purpe	The Dark Blue	Dark Blue	Blue Bluep
⬜	🟫	⬜	The brightest Color with closest to Salmon?	Pink	Pink	Pink Pink Pink

Figure 2: **Dev Set Model Outputs of Transformer Encoder-Decoder and T5 Transformer**. Target color is on the right.

7 Analysis

7.1 Transformer Encoder-Decoder

In order to improve the test set performance, one avenue that could have been explored was increasing dropout. Instead of using our selected dropout of 0.2, our model should have used the highest possible dropout that does not affect model performance. This method is formalized in the paper by Chen and Yang (2021), which suggests that using a more regularized or parsimonious model one standard error away from the maximum score or minimum loss you are optimizing over works well for better performances on unseen test sets. As seen in Fig. 1, using a dropout of 0.4 might slow down training convergence but leads to an accuracy that is almost on par with using a dropout of 0.2, and possibly within 1 standard deviation of it.

We take a closer look at instances where the GRU Baseline model did better than the Transformer Encoder-Decoder. There are 73/2023 instances in the test set where the GRU Baseline made the correct prediction, but the Transformer Encoder-Decoder did not. In 53/73 instances, the predicted utterance of both models are the exact same. In these examples, only 8/53 had a "far" condition. We note that for most of the "close" or "split" examples, neither model seemed descriptive enough as compared to the labelled utterance. We analyze some of these examples in Fig. 4. Cases 1-

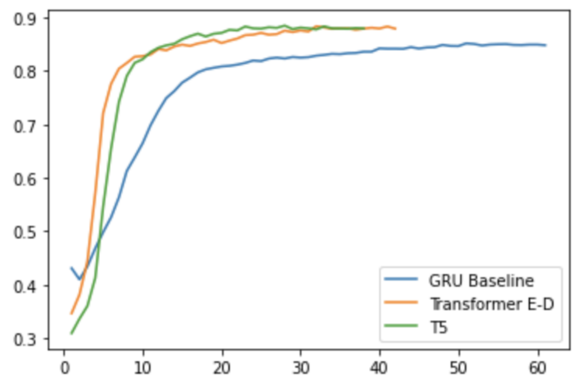
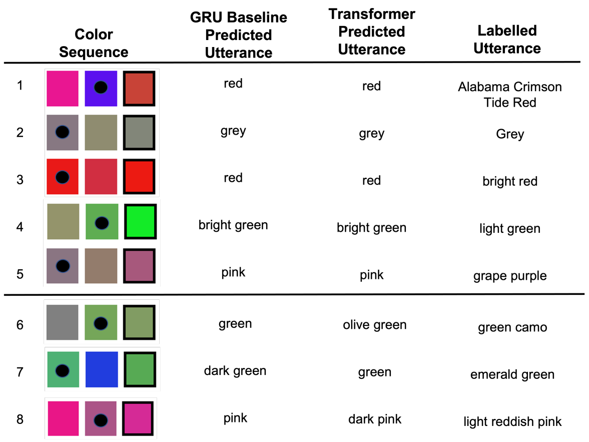


Figure 3: **Validation Listener Accuracy over Epochs**



	Color Sequence	GRU Baseline Predicted Utterance	Transformer Predicted Utterance	Labelled Utterance
1	🟪 🟪● 🟥	red	red	Alabama Crimson Tide Red
2	⬜● 🟫 ⬜	grey	grey	Grey
3	🟥● 🟥 🟥	red	red	bright red
4	🟫 🟩● 🟩	bright green	bright green	light green
5	⬜● 🟫 🟪	pink	pink	grape purple
6	⬜ 🟩● 🟩	green	olive green	green camo
7	🟩● 🟦 🟩	dark green	green	emerald green
8	🟪 🟪● 🟪	pink	dark pink	light reddish pink

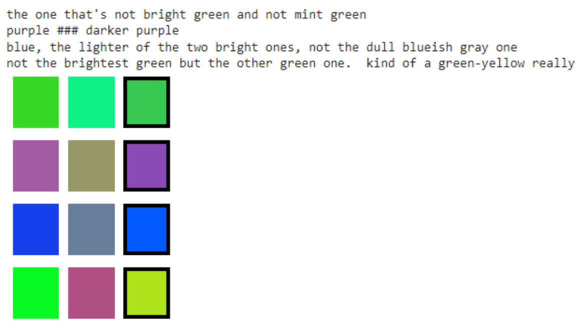
Figure 4: **Examples where the GRU was correct but the Transformer was incorrect**: Examples 1-5 are cases where the GRU and Transformer had the same utterance, while 6-8 are cases where the two models had different utterances. Transformer predictions are indicated with the black dot while the target colors are indicated with a black border.

5 show instances where the GRU and Transformer both had the same predicted utterance, but the GRU was accurate and the Transformer was not. Example 1, which shows a clear mistake by the Transformer is rare. Still, this is worrying as it is possible that the model has not quite learnt to represent certain hues correctly unlike the GRU. This suggests that the higher number of parameters in the Transformer model might overfit on artifacts or mistakes in the data. For examples 2-5, one might suggest that the discrepancies between the two model predictions are due to random chance. Examples 6-8 show instances where the two models had different utterances. Generally, the Transformer was more descriptive, as can be seen examples 6 and 8, with example 7 being a rarer occurrence. Still, the fact that the descriptiveness of the transformer model does not lead to higher accuracy suggests that the model is still weak on distinguishing colors in context (or mapping text to colors), and more training examples might be necessary.

Out of 53 of the instances where the Transformer was correct but the GRU was not, there was a more even split in terms of them having the same or different predicted utterances. Again, there is the general trend of the Transformer being more descriptive and hence more helpful to the listener than the GRU.

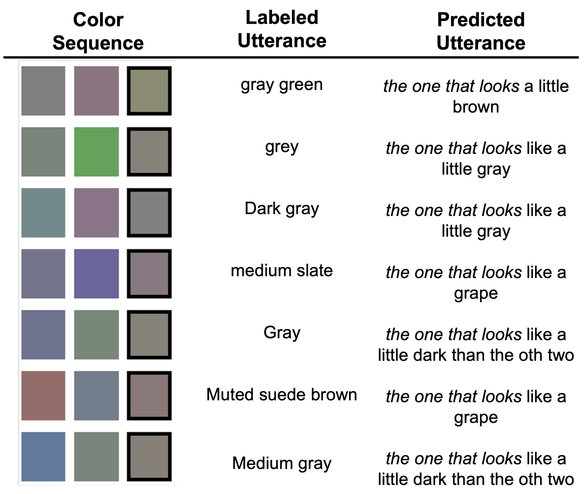
One general trend is that the Transformer Encoder-Decoder simplifies utterances as supposed to the labelled utterance, although this is not always helpful when looking at colors in context. The Transformer had an average of 1.26 words per predicted utterance (vs. 1.22 words for the GRU), while the labelled utterances had an average of 1.68 words. Since the Transformer as parsimonious as the more streamlined GRU, it attains a similar BLEU score. However, this could also make it less descriptive than the T5 which tends to have longer predicted utterances (we discuss this further in Sec. 7.2).

Despite the aforementioned advantages, the Transformer Encoder-Decoder's high accuracy does not translate to a higher BLEU score for many reasons, one being data artifacts. Fig. 6 shows an example, where predicted speaker utterance with the words *"the one that looks"* only includes purple-grey colors. While the model picked the correct target color even among difficult sequences, this is an instance where the speaker has learnt that the semantic meaning of "the one that looks" as some variant of a purplish-grey. Peeking into the training data: out of 35 training examples with the utterance "the one that looks," roughly 25 of the utterances have some association with grey or dull purple. These might also constitute a high percentage of cases where the target color is a purplish grey in the training dataset. If model associates data artifacts with colors during training, it becomes less generalizable for other uses. For instance, if we fed the model the words "the one that looks a like the lightest purple" amongst different shades of purple, the model might very well pick the dullest purple as it has learnt incorrect semantics. In future, we could analyze this further using adversarial examples and a possible solution is using a corpus of stopwords.



```
the one that's not bright green and not mint green
purple ### darker purple
blue, the lighter of the two bright ones, not the dull blueish gray one
not the brightest green but the other green one.  kind of a green-yellow really
```

Figure 5: Descriptions with longer sentences tended to repeat the target color multiple times



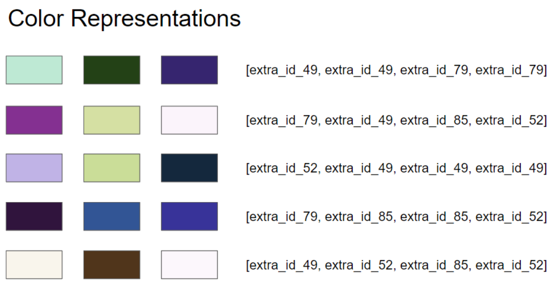
Color Sequence	Labeled Utterance	Predicted Utterance
■ ■ ■	gray green	*the one that looks* a little brown
■ ■ ■	grey	*the one that looks* like a little gray
■ ■ ■	Dark gray	*the one that looks* like a little gray
■ ■ ■	medium slate	*the one that looks* like a grape
■ ■ ■	Gray	*the one that looks* like a little dark than the oth two
■ ■ ■	Muted suede brown	*the one that looks* like a grape
■ ■ ■	Medium gray	*the one that looks* like a little dark than the oth two

Figure 6: **Example of artifacts in the data**: the Transformer Encoder-Decoder's predicted utterances with the words "the one that looks" only includes purple-grey colors, indicating that the model might have learnt incorrect associations and semantics.

7.2 The T5 Transformer

The model learned to output input colors to output tokens, and generally didn't deviate from giving short, one-word descriptions for colors in context. When it did output multi-word color descriptions, they were usually combinations of color, and less frequently adjectives appended to colors, such as dark blue or light pink. One unpleasant aspect of our model is that it tended to redundantly output color dozens of times, especially for those colors which required more verbose descriptions for the target color. We fear that the model has learned repetition of a color as a stand in for complexly describing the target. This could be due to artifacts from the dev set examples for close or split examples in which the speaker repeats colors to describe their target (See Fig. 5).

We also noticed that the T5 model output random



Color Representations

■	■	■	[extra_id_49, extra_id_49, extra_id_79, extra_id_79]
■	■	■	[extra_id_79, extra_id_49, extra_id_85, extra_id_52]
■	■	■	[extra_id_52, extra_id_49, extra_id_49, extra_id_49]
■	■	■	[extra_id_79, extra_id_85, extra_id_85, extra_id_52]
■	■	■	[extra_id_49, extra_id_52, extra_id_85, extra_id_52]

Figure 7: Color Representations

characters to words. Take, for example "Brightl" or "Blue Bluep" as listed in Fig. 2. This is a matter of how T5's tokenizer works. While many words, such as purple, blue, or green are broken down into single tokens in the T5 tokenizer, certain colors are broken into multiple tokens. For example, T5 tokenizes "teal" into two: "tea" and "l". Therefore, it's no surprise that "l" is outputted randomly within the T5 model; our model has learned to associate it with certain colors.

One interesting facet of the T5 architecture is its color-to-text conversion. T5 is fine tuned on text input and text output; we have to project our color embeddings into this word-embeddings space of the model to meaningfully interpret our colors. However, the fact that our model not only outputs colors correctly, but gets a speaker accuracy score that exceeds the encoder-decoder model put forth by Monroe et. al implies that T5 is capable of this conversion.

We were interested, then, in understanding where and how this conversion happens. We believed it to happen in the perceptron-projection step. Because T5 was trained on text inputs, the easiest way for our model to learn to process these texts was to project them to be similar to preexisting embeddings that resembled words relevant to colors, and then leverage the model's pretrained understanding of these words to produce outputs. To test this theory, we took each of the four embeddings our model input to T5 for some examples, found the word embedding with the minimum euclidean distance in its embedding for each of our four model's tokens, with the idea being that this token best approximates our input embeddings' meaning.

We find that the input embeddings T5 accepts are closest to utility tokens used by T5 (See Fig. 7). Specifically, they are closest to mask tokens, which are used in T5's training to mask and predict spoken language tokens during the unsupervised pretraining stage. These embedding distances are nontrivial; while the embeddings projected into the model tend to have a shortest euclidean distance to preexisting embeddings of around 450, these vectors' average distance across all embeddings vectors is 65,000. Yet these embeddings were even closer to each other than they were to any preexisting words. Consider in the second sets of examples, for example, that the 1st and target color embeddings had a euclidean distance of 12.68! This is intriguing, as it suggests that the model's low-level understanding of "purple" generally maps to the same space, which may be far away from the model's embeddings for the words "light" or "purple", but is nevertheless processed by T5's encoder layers such that the model might eventually output purple. Our initial hypothesis seems to be refuted, then; T5 eventually converts color embeddings that don't resemble words into words within the machinations of the model.

7.3 The Fine-tuned GPT-2 Transformer

The predicted utterances of the GPT-2 based transformers revealed in part why it did not achieve the same level of performance as transformer encoder-decoder and T5. There is an alarmingly high rate of utterances composed of just articles and made no sense with respect to the color sequences (for example, one of such generated utterances is "the"), which made analyzing errors as in the previous sections not as meaningful. The GPT-2 model did not seem to learn to generate the correct utterances in a lot of cases, and hence it made sense that it achieved lower accuracy than the rest on both sets of data.

To improve the listener accuracy, we attempted to train a smaller GPT-2 model on the Colors-in-Context corpus directly. Unfortunately, after hours of training, the model did not generate sensible outputs. When incorporated into the transformer, the GPT-2 model trained from scratch did not produce better results than the pretrained and fine-tuned version, but produced a listener accuracy just a little better than random guesses (around 40 percent). Training the GPT-2 model on a new corpus may require more careful parameter tuning and much longer training time. However, we believe that if we would be able to train such a model successfully, the accuracy of generated utterance will improve greatly and therefore improve the listener accuracy. It would be interesting to experiment with training GPT-2 from scratch and observing the change in generated tokens.

8 Conclusion

The Transformer Encoder-Decoder did well on the validation set, but some improvements could be made. Besides a more careful use of dropout to improve the Transformer Encoder-Decoder performance on unseen test sets, another extension that would be interesting to explore is increasing the interaction between the encoder and decoder using a mesh-like structure as suggested in the paper by Cornia et al. (2019). The paper suggests that allowing each layer of the decoder to attend to low and high level features of the multiple layers of the encoder could improve performance.

Future attempts to leverage T5 in color identification might more directly try to interface with the language-to-language encoding and decoding that the model was designed around. One approach could be as follows: one could create a decoder for individual colors that, given a color representation, directly outputs a rich textual description for that color. One would then have 3 color descriptions, A, B, and C corresponding to each color in context. T5 would then receive input as "A x B y C", where x and y are special tokens inputted to T5 to demarcate unique color descriptions from each other. Such a model that outputs text descriptions for colors, given colors, already exist; by training on a Sherwin-Williams dataset of paint color names and pretrained word embeddings, author Michael Alcorn was able to create an RNN decoder that outputs color descriptions (Alcorn, 2017). Then again, T5's pretraining might not help considering that the colors in context dev set is made up of short, sentence fragment phrases that don't at all resemble the majority of the C4 internet text that T5 was pretrained on Alcorn (2017).

The less than satisfactory performance of the GPT-2 model may be due to its decode-only nature, but also may have resulted from the size of the model. With more resources and time allocated to training and fine-tuning, the performance of the model may yet improve. Future progress on the GPT-2 based transformer may lie in training the GPT-2 model on the Colors-in-Context corpus directly and hence obtaining a smaller model without the excess vocabulary. Reduction in size of the pretrained GPT-2 model may further improve accuracy of this transformer to on par with the T5.

Known Project Limitations

As noted before, the Transformer Encoder-Decoder is highly susceptible to artifacts in the data. Therefore, detecting artifacts in the data after training is important, and designing adversarial examples to see if these artifacts indeed harms model performance could be an important exercise as well. Furthermore, as we have shown, there are still certain shortcomings in terms of understanding what the color means in context despite having a good description of it, meaning that perhaps more and better labelled training examples and training time is necessary for the Transformer to perform better.

Next, although the T5 is a powerful encoder-decoder transformer, it should be cautioned that this model has demonstrated social biases. In (Katsarou et al., 2022), authors Katsarou et al. found that within T5's embedding space, higher paying jobs are more associated with men than women. While this bias against women may seem trivial to invoke with respect to the color identification task, which deals with the English language in the very narrow context of colors, consider that the word "girl" features within our dataset, often very frequently when describing lighter and pinker colors. Boy is also used to describe blue or light blue within the dataset. The gender bias of our dataset may therefore interact negatively with the preexisting gender bias extant within T5.

Aside from similar known biases as in the T5 model and the limitations in the fixed size of pre-trained GPT-2 mentioned in previous sections, the large size of GPT-2 also posed a challenge due to limited resources. We were not able to conduct hyperparameter searches due to limitations of GPU resources and time. Hyperparameter values and longer training time may help improve the GPT-2 based transformer in color identification.

Authorship Statement

Bryan developed the Transformer Encoder-Decoder model, and wrote the analysis on said model. Nicholas developed the T5 based colors in context model, and wrote analysis on said model. Sylvia fine-tuned the pretrained GPT-2 model, trained the GPT-2 based transformer, and wrote analysis on said model. Professor Michael Potts, one of the co-authors in the original Colors in Context paper, graciously provided us with the python code that the team used to run the model, which we used to ensure our tokenization and color representations were correctly implemented as per their paper (Monroe et al.). We were also able to reference code from German Enik, Hannah Zhang, (Enik and Zhang) Paridhi Maheshwari, Pranav Jain, and Swastika Dutta (Maheshwari and Dutta) to ensure our colors pre-processing was done correctly.

References

Michael A. Alcorn. 2017. Using machine learning to name colors.

Yuchen Chen and Yuhong Yang. 2021. The one standard error rule for model selection: Does it work? *Stats*, 4(4):868–892.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2019. M^2: Meshed-memory transformer for image captioning. *CoRR*, abs/1912.08226.

Jiajun Deng, Zhengyuan Yang, Tianlang Chen, Wengang Zhou, and Houqiang Li. 2021. Transvg: End-to-end visual grounding with transformers. *CoRR*, abs/2104.08541.

Ye Du, Zehua Fu, Qingjie Liu, and Yunhong Wang. 2021. Visual grounding with transformers. *CoRR*, abs/2105.04281.

German Enik and Hannah Zhang. Colors-in-context-submission-1.

Styliani Katsarou, Borja Rodríguez-Gálvez, and Jesse Shanahan. 2022. Measuring gender bias in contextualized embeddings. *Computer Sciences amp; Mathematics Forum*, 3(1).

Jain Pranav Maheshwari, Paridhi and Swastika Dutta. Colors-in-context-submission-2.

Will Monroe, Robert Hawkins, Noah Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5.

Will Monroe, Robert Hawkins, Noah Goodman, Christopher Potts, Jennifer Hu, and Andrew Jong. Colors-in-context.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.