Assessing Coherence via Dialogue Reconstruction Tasks

Edwin Chan
University of Michigan
edwnchan@umich.edu

Shruti Jain
University of Michigan
shrujain@umich.edu

Jai Narayanan
University of Michigan
jainara@umich.edu

Abstract

In this paper, we aim to analyze alternative coherence measurements for conversational entailment of a dialogue for a variety of models through evaluating their accuracy on two distinct tasks. We hope that this motivates other approaches to quickly adapt existing datasets for coherence measurement while minimizing the need for further human annotation.

1 Introduction

In the past few years LLMs have become increasingly better at a variety of complex language comprehension tasks. However, despite promising high performance, the question of whether the models are able to actually gain a semantic understanding of both their input and their task remains — it is unclear how much 'coherence' pre-trained language models really have. Previous work has looked at assessing model coherence through textual entailment and plausibility classification tasks, the former performed on dialogue sequences that reflect organic human conversation. In this paper we aim to extend and enhance coherence evaluation for LLMs by proposing and performing a dialogue reconstruction task in conjunction with a hypothesis entailment task across 4 different models in 3 different architectures. We hope that our results show the nuances of coherence levels via these two different types of tasks that build on each other and require the models to have a strong semantic understanding of their input to complete successfully.

2 Related Works

Coherence evaluation in language models has been a hot area of research over the last decade. Many papers and research aim to optimize a model's accuracy at entailment tasks. At the same time, other papers focus on understanding whether a model's performance on a benchmark is truly representative of a model's capability to understand a dialog. Our paper builds upon prior work in coherence evaluation by drawing insights from two prior notable studies: *Towards Conversation Entailment: An Empirical Investigation* (Zhang and Chai, 2010) and *Beyond the Tip of the Iceberg: Assessing Coherence of Text Classifiers*. (Storks and Chai, 2021) *Towards Conversation Entailment* discusses entailment in the context of conversational dialogues. Specifically, they evaluate whether a hypothesis logically follows from a transcript of a conversation. The paper's focus differs from traditional textual entailment tasks, as those often rely on isolated sentence pairs, whereas conversational entailment instead focuses on multi-turn dialogues that are more similar to organic human conversation. The paper importantly points out that conversational dialogue inherently contains more complex structure, through deeper contextual dependencies that result from the turn taking nature of conversation. Overall, *Towards Conversational Entailment* provides a valuable look into conversational entailment. *Beyond the Tip of the Iceberg* extends much of the foundation laid out in *Towards Conversational Entailment*. The paper comes at a time when large-scale pre-trained language models are able to achieve human-level and superhuman accuracy on language understanding tasks. The paper calls into question the coherence capabilities of these models. Coherence, or the ability for a model to show it's work when reaching a conclusion, is less understood, and the paper points out that this leads to model's being rewarded for shallow understandings in conversational entailment tasks. The paper asserts that for text classification tasks requiring reasoning over a discourse, models should be evaluated on their ability to show their work for how they reached that conclusion. To address this, the paper introduces a novel framework for measuring a model's prediction coherence. The framework aims to require more robust explanation from the models that go beyond understanding only surface level patterns in the semantic relationships of dialogue. We aim to build on the work presented in these papers by contributing a new design for a task that extends the ability to evaluate coherence of Large Language Models. By incorporating the insights from these two studies, our research looks to extend the overall understanding and measuring of coherence evaluation in LLMs.

3 Approach

3.1 Selecting Models

Since our purpose was to measure model coherence, we chose to conduct our research across a variety of different model architectures to see the varying capabilities of different types of LLMs. Specifically, we wanted to see the performance of models that are decoder-only, encoder-decoder, encoder-only, and RNN based. However, since our project scope was related only to evaluation rather than training a model from scratch and we were unable to find any pre-trained RNNs on platforms such as HuggingFace that fit the types of tasks we were to perform, we limited our study to the remaining 3 categories. We chose to select sufficiently state-of-the-art models to represent each category.

3.1.1 Decoder-Only

For the decoder-only category, we chose Mistral-7B-Instruct-v0.3 and Llama-3.1-8B-Instruct. We chose these models as opposed to Llama-13B, Llama-70B, and Mistral Large because we kept in mind computational constraints, especially when limited by the computational power available on GreatLakes. We also considered but eventually rejected GPT models. This was mainly due to the fact that GPT-2 was the only free and open-source option available to fine tune. From previous experiences, we were aware that GPT-2 has significantly worse performance as compared to Mistral-7B-Instruct-v0.3 and Llama-3.1-8B-Instruct across various tasks.

3.1.2 Encoder-Decoder

For the encoder-decoder model, we chose the BART-base model. We considered using Google-T5 and T5-small, however, existing research (Dharrao D, 2024) showed that BART models generally outperformed T5 across many tasks. We used the base model again due to computational constraints and because it had the capabilities to complete both of our tasks properly.

3.1.3 Encoder-Only

We chose BERT-base-cased as our encoder-only model. We chose the cased version of the tokenizer and the model because we believe that some of the semantic information in the datasets we used was better captured by the case-sensitive tokenizer. Furthermore, because BERT does not offer the same capabilities as encoder-decoder or decoder-only models, we used two different pre-trained BERT versions in our project. For the first task of dialogue sequence reconstruction, we chose to use the pre trained BertForTokenClassification with `[CLS]` tokens and for the second task of hypothesis entailment, we used the pre-trained BertForMultipleChoice. We explain in more detail the implementation and methodologies later.

3.2 Task 1

On a high-level, our first task requires the model to reconstruct the original order of dialogues in a two person conversation sequence given a permutation. Our aim with this task is to see whether a model can comprehend raw human dialogue with enough accuracy to be able to differentiate between a logical conversational sequence and an illogical, random sequence. However, rather than simply being able to pick between what is a logical conversation order and what is not (as this could rely on factors such as turn-taking analysis rather than semantic analysis), we have the model actually try and reconstruct the conversation.

3.2.1 Datasets and Preprocessing

We initially wanted to use only the SLED Research Lab's Conversational Entailment dataset from HuggingFace as it was the dataset used in previous works on model coherence assessment. However, upon further analysis we soon realized that the dataset was very small and consisted of many repeats as it had been initially formulated for subset hypothesis entailment tasks. Therefore, we decided to choose the DialogSum dataset available on GitHub (Chen et al., 2021). We came across this dataset while searching for data that included relatively long back and forth conversations, and potentially included a hypothesis to summarize the conversation. We ended up picking this dataset as we saw that it included a diverse range of conversational examples. Although this dataset was initially developed to be used for summarization tasks, it provided us with a sufficiently large database of dialogue sequences and served as a good starting dataset for our dialogue reconstruction task.

For the second task, however, we decided to use the SLED dataset for reasons we will detail later.

Because the SLED dataset generally has much shorter sequences of dialogues and due to model limitations in prompt/input length, we filtered the DialogSum dataset to include only those dialogue sequences that were 20 dialogues or less in length. We then generated a permuted file that had a randomly permuted version of each of the dialogue sequences. Our input file was a JSON file in the following format, where the correct order was the label that we wanted the labels to ultimately predict.

```
[
    {
        "items": <int>,
        "permuted_sentences":
        [<list of strings>],
       "correct_order": [<list of ints>]
    },
    . . .
]
```

The DialogSum dataset came with different splits, so we used a split of 479 inputs for our testing data, a split of 11424 inputs for our training data, and a split of 486 inputs for our validation data. We kept our training data significantly larger than testing and validation data to ensure better training and quicker evaluations.

3.2.2 Decoder-Only

The decoder-only models were trained using SFTTrainer and fine-tuned using LoRA (Low-Rank Adaptation of Large Language Models. The decision to use SFTTrainer was made as it can be easily integrated with a data collator which helps prepare batches of training data thus facilitating both training efficiency and computation time. The decision to finetune using LoRA instead of finetuning the entire model was also made to reduce the number of trainable parameters and further reduce the computational cost.

This is the prompt that was passed to the model for fine-tuning and training for Task 1.

> "Below is a list of sentences labeled from 1 to {num_sentences} that have been permuted from their original order in a dialogue. Using the numerical label for each sentence, write only the original order of the sentences as a list of numbers inside square brackets, separated by commas.
> ### Sentences:
> {permuted_dialogue}
> ### Answer:
> {label}"

3.2.3 Encoder-Decoder

When training, the BART (facebook/bart-base) model requires both input and target sequences. For tokenizing the input, the BartTokenizer was chosen. This was a straightforward decision, as it was a premade tokenizer that could be used for converting text input into tokens that the model could understand. When using the BART model, there are several options to go for. One can use the standard BartModel, or instead pick a task specific extension variant, such as BartForConditionalGeneration, or BartForSequenceClassification. These predefined options are built to perform specific tasks while leveraging the pre-trained BART encoder-decoder architecture. None of the task specific options seemed to particularly fit our use case, so we went with the base BartModel variant. Out of interest, we also tried the BartForConditionalGeneration model, to see how its performance would compare. When using the base BartModel, it involved adding a simple layer for having the model output the correct order of input dialogues.

In both cases, the model requires the dialogue inputs and labels to be strings, so preprocessing required combining the conversation into a single text, and converting the correct order of dialogues from a list of indices into a combined text.

3.2.4 Encoder-Only

The BERT model does not accept user prompts and only takes in input sequences. Furthermore, as an encoder-only architecture made for understanding text, it cannot generate natural language responses. Therefore, completing Task 1 on the BERT-base-cased model posed a unique challenge.

We considered multiple approaches on how to adapt the reconstruction task to the BERT model. The first approach was to modify the task to instead have the models simply choose which permutation of the original dialogue sequence was the closest to the correct sequence (the ground truth was to be the permutation that had the lowest mean squared distance from the original). However, this modification would significantly change the original task, which would reduce the consistency across models and would not allow us to perform the second task.

Our second approach was to add a linear layer at the end of the model to get the output in the desired shape. We initially worked to implement this idea, however, since labels were of all different lengths between a size of 3 dialogues to 20 dialogues, constructing the linear layer at the end posed many challenges.

We looked through various pre-trained BERT models and the tasks they were used for to see if any of these could be modified to suit our task. Finally, to bypass the difficulties of adding a variable length linear layer but retain the essence of the task, we finally chose to use BertForTokenClassification as our pre-trained model for the first task. Since BERT uses the `[CLS]` token for sentence level classification, we modified the input `permuted_sentences` to be as follows, using `[SEP]` separator tokens to further make the input structure clear to the model.

```
"permuted_sentences" = "[CLS] 1)
<dialogue> [SEP] [CLS] 2)
<dialogue> [SEP] . . . [SEP]"
```

Furthermore, when processing the data in a custom dataset object, we also padded the input permuted sentences to a fixed maximum length. To prepare the labels to now reflect the token classification task, we created them as follows:

1. create a label of the same length as the padded sentences, with each element being -100
2. find the positions of all the `[CLS]` tokens in the particular input
3. set those positions in the label to be the integer corresponding to the actual correct order for those dialogues

To make sure that the unnecessary -100 values were not considered when performing gradient updates, we set up our loss function (which used cross-entropy loss) to ignore the index -100 in the labels.

3.3 Task 2

On a high-level, Task 2 required that we test the reconstructions from the Task 1 models with the best hyperparameters on a conversational entailment task. We then compared the conversational entailment accuracy for 3 different test files:

1. The original (non-permuted) dialogue sequences with the hypotheses and entailment values
2. The randomly permuted dialogue sequences with the hypotheses and entailment values
3. The reconstructed dialogue sequences with the hypotheses and entailment values

The reason that we compared accuracy on the third (target) test file with the other two was to establish a best and worst baseline. The model's performance on File 1 would be the best possible conversational entailment accuracy that it could get, so we used it as our baseline for the best performance that would be possible. The model's performance on File 2 would then serve as our baseline for the worst possible conversational entailment accuracy, while File 3 would be the accuracy we were trying to test. The reason that we established these baselines is because we acknowledge that different models will have different accuracies on a conversational entailment task, and to better contextualize their performance specifically on reconstructed dialogues, we needed to understand what their performance was on other variations. Furthermore, we wanted to measure if the order of the dialogues even affected performance of the models on the conversational entailment task or if the models ignored inter-dialogue positional information and only focused on intra-dialogue position for semantic comprehension.

Because we wanted to use this task as an extension of the first task (our aim was to have a more nuanced way of measuring reconstruction value), and because we wanted to simply assess a model rather than train it for our project, we initially decided to only use a zero-shot non-fine tuned approach for Task 2. However, this approach was modified later and will be explained in detail in later sections.

3.3.1 Datasets and Preprocessing

Although we trained, validated, and tested the model in Task 1 only on the DialogSum dataset, we used the models from Task 1 on a subset of the SLED-dataset to complete our second task.

The reason for this choice even though the SLED dataset is significantly smaller than the DialogSum dataset is because the DialogSum dataset did not include any false hypotheses, and we needed a dataset that would include both false and true hypotheses to properly measure the accuracy of our models on this task.

As in Task 1, we filtered out any dialogue sequences from the SLED dataset that were less than 3 dialogues in length. The SLED dataset did not have any dialogue sequences that were larger than 20 dialogues in length. After this filtering, we combined the existing validation and test splits on HuggingFace. We chose to combine these splits and create our own splits as necessary because we wanted any splits to reflect the proportions of true/false values for the entailment in the complete filtered dataset.

Our input data was in the following format:

```
{
    "Items": <int>,
    "Hypothesis": <str>,
    "Conversation": [<list of str>],
    "Entailment": <bool>
}
```

3.3.2 SLED Dataset Peculiarities and Implications

Although that only dataset that we were able to find that sufficiently fit our criteria of having two-person dialogue sequences, hypotheses, and entailment values, we acknowledge that the SLED dataset had certain peculiarities because its initial use case differed from the use case in our project. We list out these specific properties of the dataset below:

- The SLED dataset is very small in length - and after filtering it was only 350 examples in total.
- The SLED dataset is biased towards 'True' entailment values with a 60:40 split between true and false entailment values
- The SLED datasets has occasional repeats of exact dialogue sequences with different hypotheses
- The SLED dataset has many repeats of subsets of dialogue sequences (ex. example A will have dialogues 1-5 and example B will have dialogues 1-3) because it was originally used for a subset dialogue entailment task

Due to property 1 (dataset length) it is hard to determine whether our findings will hold for larger datasets and any fine tuning is limited to very few examples. Furthermore, the conversations in the dataset were very specific types of conversations, so results may not include all possible other conversation sequences and hypotheses.

Due to property 2, we cannot establish a baseline random accuracy of 0.5, and it is possible for us to get worse accuracy than just answering 'True' (0.6 accuracy). We contemplated removing a portion of examples that had 'True' entailment, but eventually decided against it because of our already small dataset. However, to address this issue we did ensure that both our training data and testing data had a 60:40 proportional split of 'True' and 'False' entailment examples.

Property 3 does not negatively impact Task 2, however it does impact the overall accuracy when the reconstruction task (Task 1) is carried out on the SLED dataset, because the model will have less "real" data to both fine tune and test on.

Property 4 also reduces our confidence level in our results on the SLED dataset for the purposes of Task 1 (since certain examples are partial repeats), although it does not affect Task 2 much.

Due to our time constraints we were unable to construct our own datasets, however we point out these potential issues with our dataset usage as guidance for future work on this subject and encourage the creation and usage of datasets that do not come with these caveats if replicating these tasks on any model.

3.3.3 Decoder-Only

For this task, we used the base model of Mistral and Llama. For the reconstructed dialogue approach of this task, each model received the dialogue reconstruction from the output of running their respective Task 1 model. The following prompt was what was settled on.

> "Below is a dialogue paired with a hypothesis. Your task is to decide whether the hypothesis is directly supported or implied by the dialogue. If yes output only 1. If no output 0
> ```
> ### Dialogue:
> {dialogue}
> ### Hypothesis:
> {hypothesis}
> ### Answer:
> {entailment}"
> ```

3.3.4 Encoder-Decoder

For this task, we used BartForConditionalGeneration. We formatted inputs to have the dialogues, hypothesis, and the True or False answer choices concatenated. We ran it without fine tuning, and processed the output by taking the generated token IDs, and converting them back into text using the tokenizer. From there we looked for the predicted output and used that to test for accuracy. We didn't finetune the Bart model in this case.

3.3.5 Encoder-Only

For this task, we used BertForMultipleChoice, where we formatted the input such that the full dialogues and hypothesis were concatenated with each of the answer choices of 'True' and 'False'.

Although our initial plan was to not finetune the model at all, BertForMultipleChoice gave us a very low accuracy without fine tuning. To address this issue, we split the data into a finetune and a testing set (skipping validation due to the very small length of our dataset). The fine tune split had 100 examples and the testing set had 250 examples, and both of these splits reflected the 60:40 'True' to 'False' ratio. We recorded accuracies on different hyperparameter configurations for the fine tuned version as well as the initial accuracy on the non-fine tuned version.

4 Evaluation

4.1 Task

We have two different accuracy measures that we used for this task. The first was binary accuracy, which 1 if the model correctly predicted the whole sequence for an example and 0 otherwise. The second was element accuracy, which was the fraction of the sequence that the model correctly predicted (ex. $\frac{1}{3}$ if the label was [3,1,2] and the model predicted [3,2,1]).

Although our initial plan was to only calculate the binary accuracy, we also calculated the element accuracy because some models had very low accuracy and we wanted to be able to give 'partial credit' for mostly correct reconstructions.

We considered using mean squared distance and absolute distance as measures for accuracy but ultimately abandoned those because the dialogue sequence lengths differed.

4.1.1 Decoder-Only

With regards to the hyperparameters selected, we were limited due to the computational memory available to us. Both Mistral and Llama shared the following hyperparameters.

```
Epochs = 3
Sequence Length = 1024
Lora Alpha = 16
Learning Rate = 5e-5
Max Grad Norm = 0.3
Warmup Ratio = 0.03
Lora Dropout = 0.1
Optimizer = "paged_adamw_32bit"
LR Scheduler Type = “linear”
```

Batch Size	Gradient Accumu lation Steps	r	Target Modules	Binary Accuracy	Element Accuracy
4	**4**	**16**	**q, v**	**41.63%**	**62.58%**
4	8	16	q, v	41.21%	63.60%
4	8	64	q, v, k, o	32.86%	40.69%

Figure 1: Mistral performance on different hyperparameter configurations

Batch Size	Gradient Accumul ation Steps	r	Target Modules	Binary Accuracy	Element Accuracy
4	**4**	**16**	**q, v**	**42.26%**	**63.82%**
2	16	16	q, v	39.75%	61.94%
2	16	64	q, v, k, o	31.14%	40.48%

Figure 2: Llama performance on different hyperparameter configurations

4.1.2 Encoder-Decoder

The various hyperparameters used when fine tuning and testing for Task 1 can be found in the table below. Higher epochs seemed to be an interesting parameter to play with, but their long runtime only allowed for a few runs with higher epoch counts.

Batch Size	Epochs	Learning Rate	Binary Accuracy	Element Accuracy
8	2	3e-5	0.00%	20.28%
8	3	3e-5	0.00%	24.48%
16	3	5e-4	0.00%	24.04%
16	5	5e-4	14.11%	30.60%
32	**5**	**3e-5**	**14.25%**	**34.41%**

Figure 3: BART performance on different hyperparameter configurations

4.1.3 Encoder-Only

The best hyperparameters when fine tuning and testing on the DialogSum dataset are bolded below and the corresponding model was saved and used to generate the input file for Task 2.

Batch Size	Epochs	Learning Rate	Binary Accuracy	Element Accuracy
32	6	5e-6	1.46%	21.68%
16	2	3e-5	0.00%	13.04%
16	6	3e-5	7.69%	28.66%
16	10	3e-5	10.25%	40.51%
16	**15**	**3e-5**	**15.90%**	**46.05%**
32	15	3e-5	12.97%	44.11%
32	15	5e-5	13.72%	45.07%
16	20	3e-5	14.85%	45.84%
16	15	3e-4	0.00%	10.79%

Figure 4: BERT performance on different hyperparameter configurations

4.2 Task 2

Measuring accuracy for Task 2 was done by simply calculating the percentage of questions for which the model answered correctly. We have also included the reconstruction accuracy on the SLED dataset from Task 1.

4.2.1 Decoder-Only

Below we report the dialogue reconstruction task accuracy and entailment for the SLED Dataset.

Task	SLED Dataset	Accuracy
Reconstruction (Task 1)	Full dataset (with repeats)	Binary: 32.00%
		Element: 40.49%
Entailment	Original full dataset	62.00%
Entailment	Permuted full dataset	63.71%
Entailment	Reconstructed full dataset	64.00%

Figure 5: Mistral performance on SLED reconstruction and Task 2

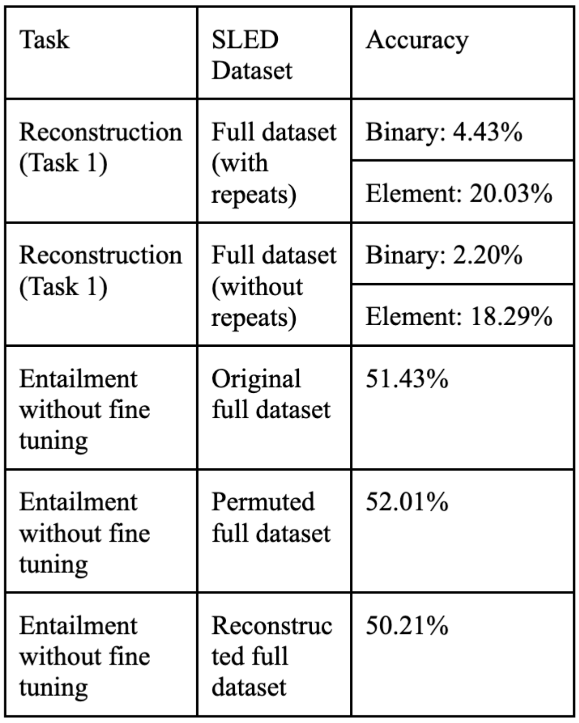
Task	SLED Dataset	Accuracy
Reconstruction (Task 1)	Full dataset (with repeats)	Binary: 32.57%
		Element: 41.37%
Entailment	Original full dataset	78.00%
Entailment	Permuted full dataset	74.29%
Entailment	Reconstructed full dataset	78.86%

Figure 6: Llama performance on SLED reconstruction and Task 2

On Task 2, Llama showed better performance than Mistral.

4.2.2 Encoder-Decoder

Below is the accuracy on the BART model.



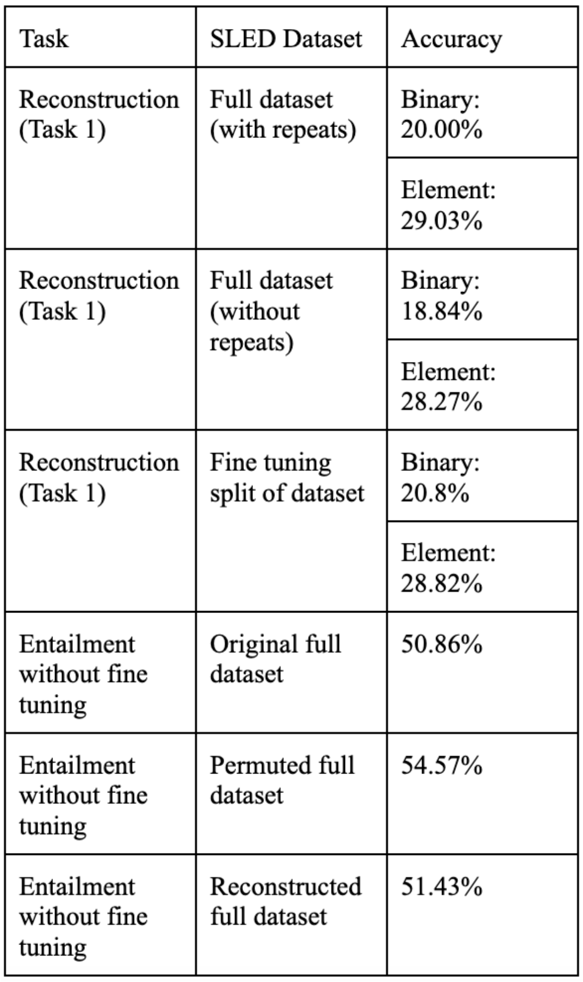
Task	SLED Dataset	Accuracy
Reconstruction (Task 1)	Full dataset (with repeats)	Binary: 4.43%
		Element: 20.03%
Reconstruction (Task 1)	Full dataset (without repeats)	Binary: 2.20%
		Element: 18.29%
Entailment without fine tuning	Original full dataset	51.43%
Entailment without fine tuning	Permuted full dataset	52.01%
Entailment without fine tuning	Reconstructed full dataset	50.21%

Figure 7: BART performance on SLED reconstruction and Task 2

BART performance was very poor on the SLED dataset reconstruction.

4.2.3 Encoder-Only

Below we report the BERT accuracy of the reconstruction and the entailment accuracy without fine tuning of the multiple choice BERT model.



Task	SLED Dataset	Accuracy
Reconstruction (Task 1)	Full dataset (with repeats)	Binary: 20.00%
		Element: 29.03%
Reconstruction (Task 1)	Full dataset (without repeats)	Binary: 18.84%
		Element: 28.27%
Reconstruction (Task 1)	Fine tuning split of dataset	Binary: 20.8%
		Element: 28.82%
Entailment without fine tuning	Original full dataset	50.86%
Entailment without fine tuning	Permuted full dataset	54.57%
Entailment without fine tuning	Reconstructed full dataset	51.43%

Figure 8: BERT performance on SLED reconstruction and Task 2 without fine tuning

Since the entailment task accuracy without fine tuning was as bad as random, we decided to fine tune BertForMultipleChoice on the fine tuning split of the original dataset (not the reconstructed or permuted versions). For a select few hyperparameter configurations that seemed promising, we noted the accuracies on the permuted and reconstructed datasets. The best hyperparameters are bolded below.

Batch Size	Epochs	Learning Rate	Accuracy
16	20	1e-4	Original: 59.2% Permuted: 59.2% Reconstructed: 59.2%
16	10	1e-4	Original: 56.8%
16	20	5e-5	Original: 58.0%
16	20	5e-4	Original: 42.0%
32	20	1e-4	Original: 52.8%
32	10	1e-4	Original: 59.6% Permuted: 59.2% Reconstructed: 59.6%
32	**15**	**2e-4**	**Original: 60.0%** **Permuted: 60.0%** **Reconstructed: 60.0%**
32	20	2e-4	Original: 55.6%
32	15	2e-4	Original: 60.0%

Figure 9: BERT performance on Task 2 with fine tuning

5 Discussion of Results

5.1 Comparing Models

Across all models, we found that the decoder models consistently performed better on both Task 1 and Task 2 even with additional finetuning for Task 2. Much of this was due to the ability of decoder-only models to better comprehend and execute given instructions in a prompt which allowed for more flexibility compared to the encoder-decoder models and the encoder models. As the encoder-decoder models are less suited to open-ended generation tasks, they often struggled to structure their outputs in a manner that can be easily parsed or would completely ignore the task goal. The encoder-only models were worse as they are not suited to language generation and could not be given system prompts.

5.2 Decoder-Only

For Task 1, we were limited on the limits of the hyperparameters that we can change as there were memory constraint issues. As a result, we decided to settle with the shared hyperparameters above. When varying the other hyperparameters, we initially believed that a higher r value along with more linear layers in the target modules would lead to increased performance. However, subsequent tests showed that it actually decreased performance in some aspects for this task. For both models, we weren't able to achieve over a 50% binary accuracy even with the best hyperparameters. We believe that this accuracy has a lot of room for improvement with more training and experimentation, such as larger epochs or different target modules.

The performance of dialogue reconstruction on the SLED dataset seemed relatively low at below 50% for both binary and element accuracy. This raised some issues about how this would affect the later entailment task as the difference may not be statistically significant between the permuted and reconstructed dataset. With Mistral, we noticed that this held true as no significant difference was observed between the accuracy on entailment for the permuted and reconstructed dialogue. Surprisingly, the model performed better on the permuted and reconstructed dialogue over the original dialogue. However with Llama, we noticed a much bigger gap between the accuracies of the permuted and reconstructed dialogue showing that the reconstructed dialogue gave the model a higher overall understanding of the dialogue. It was surprising again to see that the reconstructed dialogue performed better than the original dialogue again. The reconstructed dialogue outperforming the original dialogue for both models may be due to the reconstructed dialogue cutting out some sentences of the original dialogue and duplicating others which may place more emphasis on the sentences that contribute to the overall entailment of the hypothesis. Another issue that we faced was the tendency of the Mistral model to gravitate towards whichever integer was specified first. For example if the prompt was, "If yes, output 1. If no, output 0, then 1 would be output with overwhelming frequency. When we swapped the order, then 0 would be output with overwhelming frequency. We tried a variety of different prompts, but we were not able to change this behavior. It is difficult to see what extent Llama was affected as its overall accuracy is relatively high but this may be due to the skewed dataset. However, we believe that Llama was able to comprehend instructions in the given prompt better than Mistral resulting in a higher accuracy on Task 2.

5.3 Encoder-Decoder

The performance of BART-base for Task 1 peaked at around 14.5%. This is fairy low, showing BART was unable to reconstruct the majority of conversations in their correct order. Looking at the partial accuracy, BART was capable of achieving a slightly higher accuracy at 34.41%. While still low, this accuracy was a relatively high jump from the 20% partial accuracy that was found when running the model with only 3 epochs and a smaller batch size. Through the various runs that gradually incremented the epoch count and batch size, the model's accuracy improved as both increased. Ideally, we would have been able to test with a greater number of epochs and potentially a larger batch size. A limiting factor was the time taken and resources required for the training process to complete with more than 5 epochs. While the training of BERT in our encoder-only tests allowed for up to 20 epochs, the training for BART appeared to be much more intensive, with each epoch taking far longer to run. While speculation, it is likely that given more epochs the model could achieve a higher partial and binary accuracy as well.

For Task 2, BART achieved an accuracy of 52%, slightly worse than what it could achieve by simply outputting True for every response. While fine tuning could potentially help, the poor accuracy seems more of a result of the poor accuracy of the model from the reconstruction task.

5.4 Encoder-Only

The performance of BERT-base-cased on the first task initially seemed very low, as the overall accuracy was <16% even with the best hyperparameters. However, upon analysis of the output, we realized that the model was often reconstructing parts of the dialog correctly even if it was not correctly predicting the overall sequence perfectly. To make sure that this was captured, we added the accuracy measure that looked at each element in the predicted and label lists and calculated accuracy accordingly. To make sure that this accuracy was actually better than random (that it was not by coincidence that the permuted versions had the correct ordering), we also calculated the accuracy of the permutations before any processing - their element accuracy was 9.62% and binary accuracy was 0.00%, showing that our models were able to successfully partially reconstruct the dialogues. This is promising in showing that although it cannot compare to the performance of the decoder-only models which are built for generative tasks, the BERT model does have a fair amount of coherence in comprehending natural conversational content. We do, however, acknowledge that it may be relying on factors other than the content (ex. taking benefit of the turn-taking structure of the dialogues), but believe that such accuracy would not be possible based only on non-semantic understanding.

The reconstruction model performed reasonably well on the SLED dataset test split considering that it had been fine tuned on a different dataset entirely. However, our results on the conversational entailment task were not very promising. Without fine tuning the model performed worse than it would have if it had just answered 'True' on every question. With fine tuning we were able to get slightly better accuracy, however, this accuracy was still capped at 60%, and did not change based on the ordering of the dialogues. This accuracy may be artificially created by simply answering 'True' on most questions, but even upon analysis of hyperparameter configurations that give different accuracies, we see that the performance of the model does not change much by the input type (original, permuted, or reconstructed). This result can mean any combination of the following factors:

1. We failed in finding optimal hyperparameters for training, getting stuck at a local maximum
2. the dataset was far too small and biased to 'True' to evaluate
3. the model is bad at comprehending inter-dialogue positional information in the context of hypothesis entailment
4. the model simply was unable to properly comprehend any semantic information in a the dialogue input in a manner that would suit this task

Although we cannot conclusively say which of the factors played a role in the results we got, our takeaways on how this task can be replicated with more decisive results by addressing the above issues are as follows:

1. More hyperparameter configurations and fine tuning with experiment tracking tools
2. A larger non-repetitive dataset with equal proportions of 'True' and 'False' entailment examples
3. Different input formatting (ex. omitting `[CLS]`, `[SEP]` tokens between dialogues in conversational entailment input) to enhance model comprehension of input
4. Replicating this task on different encoder-only models (such as BERT large)

6 Conclusion

Our work establishes the performance of different model types on coherence tasks involving conversation reconstruction and hypothesis entailment. We show that the chosen decoder-only models are better at conversation reconstruction than both encoder-decoder and encoder-only models. The chosen encoder-decoder and encoder-only performed largely similar on the conversation reconstruction tasks. On the hypothesis entailment task, decoder-only models again outperformed models in both other categories, with the encoder-decoder and encoder-only models giving performance levels that were similar to random guessing. Our results conclude that the coherence of decoder-only models across a variety of tasks that involve the comprehension of human dialogue is much higher than the coherence of encoder-decoder models and encoder-only models, though we emphasize that further work is needed to see if these results hold across larger versions of models and with more diverse and balanced datasets. While not perfect, we have explored a potential approach in taking an existing conversational entailment dataset and quickly adapting it for some measure of coherence without the need for further human annotation. We hope that this inspires later work on generating alternative measures of coherence and leveraging existing datasets as we become increasingly reliant on conversing with these models.

7 Division of Work

When handling the division of work, our project was split into three parts with subtasks within each category. For clarification, the categories will be iterated again: Processing the Data, Dialogue Reconstruction, Zero-Shot Entailment prompting.

For data processing, Shruti(SLED) and Jai(DialogSum) chose one of the two main datasets and created their own data parser to handle the unique formatting of each. Edwin was responsible for generating permuted dialogues from the processed datasets.

For the subsequent model tasks, our project involved handling different model types and thus we divided the following tasks based on the model types selected as shown below.

Shruti: Encoder (BERT)
Jai: Encoder-Decoder (BART)
Edwin: Decoder (MISTRAL, LLAMA)

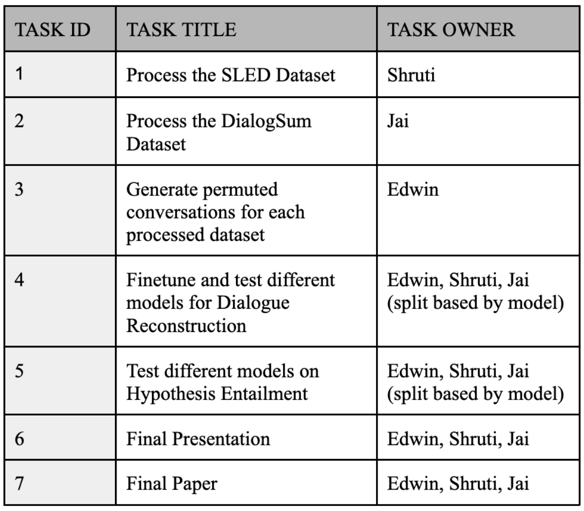
Edwin was assigned more models due to the plug and play nature of the decoder models that were observed in previous assignments in this class where much of the code needed to train the model could be reused by simply swapping the model that was loaded at the start. In contrast, the BERT and BART models required much more complex and complicated integration at times as mentioned above. Once the models were assigned, much of the work from this point forth was independent.

While the training and testing of the models were largely independent, our group agreed that we should have biweekly meetings to ensure that everything was going smoothly and address any potential issues that may arise. In addition, these meetings helped us divide tasks for deadlines that weren't inherent to just training the models, such as preparing for the final presentation and drafting our final paper submission.

For the presentation, Shruti took a large role in formatting and organizing the slides. She was especially helpful in ensuring that slides weren't too visually cluttered and keeping track of any information that may have been missed. Outside of this, the division of work on the slides were divided evenly with each member taking time to review everyone's slides and make suggestions as needed. In addition, we also set aside time to practice going through the presentation as we knew that timing may be an issue given the tight time constraints for each presentation and the amount of information that we were presenting.

For the final paper submission, each member handled the subsections related to the models they chose for approaches and evaluations. what models they handled. Shruti, again, helped a lot in formatting the final paper by splitting the paper into more manageable subsections which helped the other members know how their information should fit into the structure as a whole.

Our overall division of work was much different from what we initially set out in the project proposal due to evolving project guidelines and directions as we progressed. The final division of work can be summarized in the following figure.



TASK ID	TASK TITLE	TASK OWNER
1	Process the SLED Dataset	Shruti
2	Process the DialogSum Dataset	Jai
3	Generate permuted conversations for each processed dataset	Edwin
4	Finetune and test different models for Dialogue Reconstruction	Edwin, Shruti, Jai (split based by model)
5	Test different models on Hypothesis Entailment	Edwin, Shruti, Jai (split based by model)
6	Final Presentation	Edwin, Shruti, Jai
7	Final Paper	Edwin, Shruti, Jai

Figure 10: Task Distribution

References

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Kazi A Pangavhane M Pise P Bongale A Dharrao D, Mishra M. 2024. Evaluating bart, t5, and pegasus for effective information extraction. In *International Information and Engineering Technology Association*. International Information and Engineering Technology Association.

Shane Storks and Joyce Chai. 2021. Beyond the tip of the iceberg: Assessing coherence of text classifiers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3169–3177, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chen Zhang and Joyce Chai. 2010. Towards conversation entailment: An empirical investigation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 756–766, Cambridge, MA. Association for Computational Linguistics.