Project Overview

The initial phase of the research for the question-rewrite model began by examining the research article DISFL-QA: A Benchmark Dataset for Understanding Disfluencies in Question Answering. This paper was instrumental in clarifying the purpose behind the dataset's creation and its utility in addressing disfluencies in spoken language.

The study highlighted that current state-of-the-art models, such as BERT and T5, struggle with disfluent inputs. It advocated for the use of this dataset to enhance model training. The paper also explored strategies like data augmentation and model fine-tuning to improve the capacity of NLP models to process and comprehend disfluencies, which could greatly benefit real-world applications where disfluent speech is prevalent.

After familiarizing myself with the dataset, I explored various models for fine-tuning. Although initially inclined towards fine-tuning LLMs like Llama 3.1 8B and Mistral 7B, I opted to start with simpler models due to the dataset's limited size of approximately 7,182 entries. Given that BERT and T5 were extensively discussed in the article, I chose to work with BART, which is not only straightforward to fine-tune but also excels in text correction and generation tasks.

My goal was to log all results to Weights & Biases and upload the fine-tuned model to Hugging Face (HF) to facilitate faster result reproduction and track model development progression.

The process began by downloading and segmenting the data, then transforming these into HF Datasets through the `load_and_process_json_datasets` function, which converts JSON files into HF Datasets format. I then downloaded the pretrained `bart-base` model and its tokenizer from HF, and created the `preprocess_function` to tokenize both fluent and disfluent sentences.

I chose three text generation metrics to optimize model performance and evaluation: SacreBLEU for lexical similarity consistency, BERTScore for contextual adequacy in question rewriting, and ROUGE to ensure critical information from the original query was neither lost nor misrepresented. For initial trials, I selected commonly used hyperparameters: a learning rate of 2e-5, a train batch size of 16, a validation batch size of 32, three training epochs, and a decay rate of 0.01.

The outcomes were promising, with a training loss of 0.19 and a validation loss of 0.21 by the final epoch, a SacreBLEU score of 91.34, ROUGE metrics around 95, and a BERTScore of 94.86. Slightly lower loss in training indicates a potential overfitting to the training data and/or limited model resilience.

To diagnose specific areas for improvement, I examined a random sample of 10 pairs of disfluent, original, and rewritten texts. Although the rewrite model performed well in most cases, it struggled with texts that had coreferences or multiple mentions of the same entity.

For e.g.

> **“Disfluent text”:** “what other property of Goldbach's conjecture does number theory focus on, Besides the analytic property of Goldbach's conjecture?”
> **“Original output”:** “Besides the analytic property of Goldbach's conjecture, what other property of Goldbach's conjecture does number theory focus on?”
> **“Rewritten output”:** “Besides the analytic property of Goldbach's conjecture, what other property does number theory focus on?”

Following this, I pursued data augmentation, informed by the original paper’s findings that such methods could mitigate performance degradation. I generated 5,000 augmented questions from the SQuAD dataset by appending prefixes from other contextually related questions followed by natural speech fillers, thereby simulating disfluent speech patterns.

For e.g.

1. **'original':** 'To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?', **'disfluent':** 'What is in front of no actually To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?'

2. **'original':** 'What is the Grotto at Notre Dame?',
   'disfluent': 'What is in front of wait make that What is the Grotto at Notre Dame?'

Training resumed with the same parameters but now included the augmented data. Results showed a marginal decrease in validation loss to 0.21 and a training loss of 0.08, a slight improvement in the SacreBLEU score to 91.44, similar ROUGE metrics, and a slight increase in BERTScore to 95.16.

After initiating a third iteration to refine the hyperparameters from a pre-defined space, aiming to further optimize model performance, I ran five trials of hyperparameter tuning using the Optuna backend. I identified the optimal hyperparameters as a learning rate of 4.42e-5, a train batch size of 16, five training epochs, and a decay rate of 0.087. These settings resulted in the best results with a SacreBLEU score of 92.10, improved ROUGE metrics, and a BERTScore of 95.40.

This marks the end of the question-rewrite model, but I would continue to pursue several promising avenues. Building on the foundation established by the initial and augmented phases of training, my future work would focus on incorporating more sophisticated techniques and expanding the model's capabilities.

1. Exploration of Larger Language Models (LLMs): Given the promising results with smaller models, exploring the potential of LLMs such as Llama 3.1 8B and Mistral 7B becomes a viable next step. These models, known for their deeper learning capabilities and broader contextual understanding, might offer significant improvements, especially in handling complex disfluencies.

2. Advanced Data Augmentation Techniques: While the initial augmentation strategies provided marginal improvements, there's scope for employing more advanced techniques. Techniques such as paraphrasing, and coreference injection could provide a richer, more varied dataset that better mimics real-world disfluencies.

Train and validation results from the best model are given below:

```
# Training the best model
best_trainer.train()
```

[3805/3805 10:52, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Sacrebleu	Rouge1	Rouge2	Rougel	Rougelsum	Bertscore Precision	Bertscore Recall	Bertscore F1
1	0.150400	0.279656	90.931300	96.242100	92.878300	95.426200	95.404300	0.949600	0.944400	0.946900
2	0.034800	0.247288	91.758300	96.386500	93.265500	95.689900	95.681100	0.953200	0.950400	0.951700
3	0.058700	0.241348	91.828000	96.439200	93.412400	95.707900	95.697600	0.951700	0.950800	0.951200
4	0.026900	0.258785	91.983500	96.578000	93.622100	95.899200	95.879800	0.952400	0.952700	0.952500
5	0.043900	0.267818	92.103300	96.681500	93.639100	95.967700	95.946900	0.954400	0.953600	0.954000

Fig1. Training results

```
[32] best_trainer.evaluate(eval_dataset=tokenized_datasets['validation'])
```

[32/32 00:31]

```
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils
  warnings.warn(
Some weights of RobertaModel were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able t
{'eval_loss': 0.26781779527664185,
 'eval_sacrebleu': 92.1033,
 'eval_rouge1': 96.6815,
 'eval_rouge2': 93.6391,
 'eval_rougeL': 95.9677,
 'eval_rougeLsum': 95.9469,
 'eval_BERTScore_precision': 0.9544,
 'eval_BERTScore_recall': 0.9536,
 'eval_BERTScore_f1': 0.954,
 'eval_runtime': 36.0235,
 'eval_samples_per_second': 27.76,
 'eval_steps_per_second': 0.888,
 'epoch': 5.0}
```

Fig1. Validation results