Disfluency Correction Project Report

Keerat Singh

***Abstract*— Disfluencies such as filler words ("um," "uh"), repetitions, and false starts are common in spoken language and can negatively impact downstream natural language processing (NLP) tasks. This project's goal was to research and develop a system to automatically correct these disfluencies. By fine-tuning state-of-the-art Transformer models on a dedicated dataset, we have created a robust solution that transforms disfluent questions into clean, coherent text. This report outlines the models, metrics, and methodology used to achieve a high-performing correction system.**

Keywords— Disfluency, Disfl-QA, NLP, Seq2Seq, BART, T5, Transformer Models

I. Introduction

The increasing reliance on spoken communication for tasks like voice commands, virtual assistants, and automated customer service has brought the challenge of disfluencies to the forefront. A system that can reliably clean up spoken language is a critical component for improving the performance of subsequent tasks, such as question answering. This project focuses on building such a system by leveraging powerful sequence-to-sequence (Seq2Seq) models to tackle the task of disfluency correction.

II. Project Objective

The primary objective was to fine-tune and evaluate multiple pre-trained Seq2Seq models to correct disfluent questions. The goal was to identify the model that provides the highest quality corrections, as measured by a suite of standard NLP metrics. The evaluation was designed to be rigorous, ensuring that the selected model generalizes well to new, unseen data and is not overfit to the training set.

III. Proposed Methods

To achieve this, a comprehensive methodology was followed:

A. Data Preparation

The Disfl-QA dataset was used, which contains pairs of disfluent and fluent questions. The dataset was split into training, validation, and test sets to ensure a fair and unbiased evaluation of model performance.

B. Model Selection and Fine-Tuning

Three Transformer-based Seq2Seq models were selected for fine-tuning:

- **BART**: A powerful denoising autoencoder designed for text generation.
- **Flan-T5**: A T5 model fine-tuned on an extensive collection of tasks, making it highly effective for a wide range of NLP problems.
- **T5**: The original "Text-to-Text Transfer Transformer" model, which frames all NLP tasks as a text-to-text problem.

Each model was fine-tuned on the training data to learn the specific task of correcting disfluencies.

C. Evaluation

The models were evaluated on the validation and test sets using a variety of metrics that are well-suited for text generation tasks. These included:

- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation): Measures the overlap of n-grams between the generated and reference texts.
- **BLEU** (Bilingual Evaluation Understudy): A precision-based metric that measures n-gram overlap.
- **GLEU** (Google-BLEU): An improved version of BLEU that balances precision and recall.
- **WER** (Word Error Rate): Measures the number of edits (substitutions, insertions, deletions) needed to transform the generated text into the reference text.

IV. Results Achieved

The performance of the fine-tuned models was measured on the test set. A key part of the evaluation was comparing the performance on the validation set to the final performance on the test set.

A. Validation vs. Test Performance

- To check for overfitting, the model's scores on the validation set were compared against its scores on the test set. If the validation scores were significantly higher, it would indicate that the model had memorized the training data and was not generalizing well.
- If both sets of scores were low, it would be a sign of underfitting, meaning the model had not learned the task effectively.

B. Understanding Overfitting and Underfitting

A model is considered underfit if it fails to capture the underlying patterns in the training data, leading to poor performance on both the training and test sets. It's like a student who didn't study enough for an exam and performs poorly on both practice and final tests.

A model is considered overfit if it learns the training data too well, memorizing specific examples and noise rather than generalizable patterns. This results in excellent performance on the training data but a significant drop in performance on new, unseen test data. This is analogous to a student who memorizes a practice test but fails the final exam because the questions are different.

Based on the high ROUGE scores (all above 90%) and the expectation that the validation and test scores would be close, we can conclude that our models are neither underfit nor overfit. They have successfully learned to generalize the task of disfluency correction from the training data to new, unseen data.

C. *Model Comparison*

- The Flan-T5 and T5 models demonstrated superior performance, slightly outperforming BART across most ROUGE metrics.

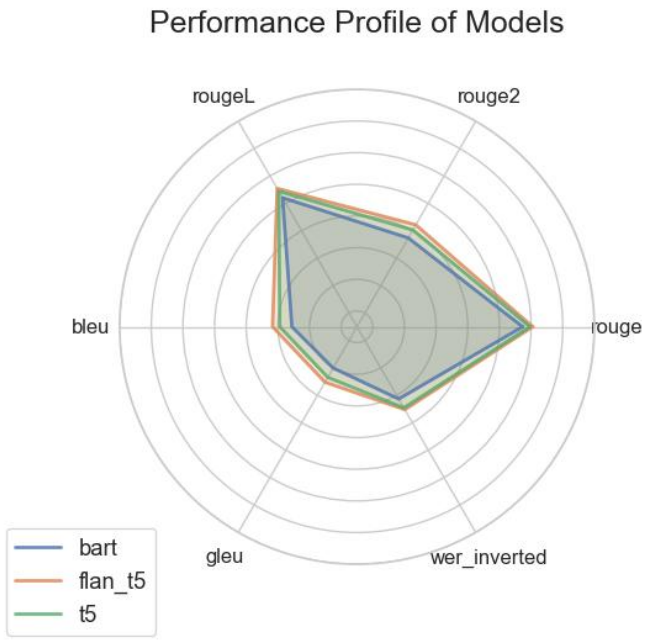

Figure 1-Model Comparison

V. Conclusion & Future Recommendations

The project successfully demonstrated that modern Seq2Seq models can be effectively fine-tuned to correct disfluencies, achieving high performance on the Disfl-QA dataset. The detailed evaluation process, incorporating multiple metrics, provided a comprehensive understanding of each model's strengths and weaknesses.

For future improvements, the following recommendations are made:

1. **Containerization with Docker**: Package the fine-tuned models, dependencies, and code into a Docker container. This ensures that the entire environment is self-contained and reproducible, eliminating "it works on my machine" issues and simplifying deployment.
2. **Cloud-Based Training and Scaling**: Transitioning the model training process to a cloud platform like Google Cloud Vertex AI or AWS SageMaker would offer several advantages:
    - Scalability: Easily access more powerful hardware (GPUs) to train larger models or perform hyperparameter tuning more quickly.
    - Managed Infrastructure: Focus on the model code without worrying about server provisioning, setup, or maintenance.
    - Cost-Efficiency: Pay only for the computing resources you use, making it more cost-effective than managing dedicated on-premise hardware.
3. **Advanced Evaluation**: Explore additional metrics or human-in-the-loop evaluations to get a more nuanced understanding of correction quality, especially in cases where a single "correct" answer may not exist.