



AUTOMATED EMAIL SUBJECT LINE GENERATION USING DEEP LEARNING

A Sequence-to-Sequence Approach for Abstractive
Summarization

Abstract

This project implements and evaluates a deep learning system for automated email subject line generation, employing a sequence-to-sequence approach for abstractive summarization. A BART-large transformer model was fine-tuned on the Annotated Enron Subject Line Corpus (AESLC). The model's ability to generate concise and relevant subjects from email bodies was validated using both automated ROUGE metrics (achieving ROUGE-L 0.3501 on the test set) and structured human evaluation, demonstrating the feasibility of this approach for enhancing email communication efficiency.

Anitha R | Mentor: Lokesh Madasu | May 2, 2025

IIT Madras, ADSML Cohort 6 – Group 12

Contents

1	Introduction	2
2	Automated Email Subject Line Generation	3
2.1	Problem Description	3
2.2	Dataset	3
2.4	Initial Observations & Baseline Performance	5
2.5	Experimentation with T5.....	5
2.5.1	Rationale for Considering T5.....	5
2.5.2	T5 Experiment Methodology	5
2.5.3	T5 Experiment Results & Observations.....	7
2.6	Comparison: T5-small vs. BART-large & Final Model Decision	8
2.7	Chosen Model Details (BART-large)	9
2.8	BART-large Fine-tuning & Automated Evaluation	10
2.9	Human Evaluation of Generated Subject Lines.....	12
2.9.1	Rationale	12
2.9.2	Methodology.....	12
2.9.3	Results	13
2.9.3.1	Inter-Annotator Agreement (IAA):	13
2.9.3.2	Average Scores by Source:	13
2.9.4	Analysis & Discussion	14
2.9.5	Conclusion from Human Evaluation.....	14
2.10	Final Performance Metrics Summary	15
2.10.1	Quantitative Metrics (ROUGE):	15
2.10.2	Qualitative Metrics (Human Evaluation):.....	15
2.11	Model Weights Path.....	16
2.12	Further Observations and Reading	16
3	Streamlit App Integration.....	21
4	Technical Implementation Details.....	25
5	Conclusion.....	26

1 Introduction

Crafting effective email subjects is essential for clear communication, yet it can be a repetitive and time-consuming task. In this project, we explored the capabilities of sequence-to-sequence generative models to solve this common natural language processing challenge: automatically generating succinct email subject lines based on the content of the email body. This task involves understanding the core message of potentially long and noisy text and performing abstractive summarization to produce a concise, informative phrase suitable for an email subject, often adhering to strict length constraints.

The project provided hands-on experience across the typical machine learning workflow, including dataset curation and robust cleaning using Python libraries (Pandas), model selection and fine-tuning using the Hugging Face ecosystem (Transformers, Datasets, Accelerate) on Google Colab, quantitative evaluation using automated metrics like ROUGE, qualitative assessment through structured human evaluation, and basic application deployment using Streamlit to demonstrate the final model's capabilities.

2 Automated Email Subject Line Generation

2.1 Problem Description

The primary task was to develop a model capable of automatically generating concise and relevant subject lines from the body of an email. This requires the model to identify the most salient information within the email body, potentially performing abstractive summarization to condense the core message into just a few words. The challenge lies in balancing informativeness with extreme brevity, ensuring the generated subject accurately reflects the email's content while adhering to the typical length constraints of subject lines.

2.2 Dataset

We utilized the AESLC (Annotated Enron Subject Line Corpus) dataset, derived from the Enron Email Corpus and publicly available. This dataset was chosen for its relevance to the task (real-world business emails) and its established splits for training, development (validation), and testing.

- **Source:** Enron Email Corpus (cleaned, filtered, deduplicated subset).
- **Dataset link:** The Annotated Enron Subject Line Corpus: <https://github.com/ryanzhumich/AESLC>
- **Dataset Sizes (Original):** The dataset contains the following number of emails per split:
 - Training set: 14,436 emails
 - Development set: 1,960 emails
 - Test set: 1,906 emails
- **Sizes Used (After Preprocessing/Filtering):** The final datasets used for model training and evaluation, after cleaning and filtering invalid rows (as described in Section 2.3 and performed by Script 3), were:
 - Training set: 14,379 examples
 - Development set: 1,953 examples
 - Test set: 1,898 examples
- **Content:** Contains email bodies and corresponding subject lines. Development and test sets include multiple human annotations (@subject is original, @ann0, @ann1, @ann2 are crowdsourced), while the training set contains only the original subject (@subject).
- **Average Lengths (Original Dataset):** Email body ~75 words; Subject ~4 words.

2.3 Data Preprocessing

A robust data processing pipeline was crucial due to the inherent noise and variability in email data. The pipeline evolved through several iterations, with the final data used for model training being processed as follows:

1. **Initial Loading & Structuring (Script 1):** Extracted relevant fields (filename, body, subject, annotations) from the raw dataset files into timestamped CSV format (*_sample_{count}_{timestamp}.csv). This run processed the full train, dev, and test sets.
2. **Cleaning & EDA (Script 2 - v5):** Applied refined cleaning steps to the CSVs generated by Script 1:
 - Preserved original text columns (original_*).
 - Replaced placeholder "0000" with NaN for processing.
 - Removed identified PII (phone numbers, email addresses, URLs) using regex.
 - Applied conservative boilerplate removal (specific phrases, common closing lines/signatures).
 - Removed simple list markers (e.g., 1., a)).
 - Converted text to lowercase.
 - Normalized whitespace (collapsing multiple spaces, trimming).
 - Aggressive punctuation removal and stemming/lemmatization were explicitly *deferred*.
 - Performed basic EDA (missing value counts, length statistics, vocabulary analysis) on the cleaned data, logging results.
 - Saved combined data (original + cleaned columns) to new timestamped CSVs (*_cleaned_{timestamp}.csv).
3. **Model-Specific Preprocessing (Script 3 - v4):** Prepared the cleaned data specifically for the chosen model (facebook/bart-large):
 - Loaded the full cleaned datasets (*_cleaned_*.csv) using Hugging Face datasets, explicitly defining column types as strings.
 - Filtered out rows with empty or invalid cleaned body or subject fields.
 - Loaded the AutoTokenizer for facebook/bart-large.
 - Applied the tokenizer to the cleaned body (input) and subject (target label) columns, handling truncation to max lengths (512 for input, 32 for target).
 - Removed original text columns, keeping only input_ids, attention_mask, and labels.
 - Saved the final tokenized dataset in Arrow format using save_to_disk (processed_datasets directory).
 - Validated the saved dataset by reloading and checking structure/content.

2.4 Initial Observations & Baseline Performance

Early experiments and analysis (including Subject-Body ROUGE analysis detailed later in Section 2.12) highlighted key dataset characteristics and established baseline performance:

- **High Abstractiveness:** Reference subjects showed very low lexical overlap with email bodies (average Subject-Body ROUGE-L ~ 0.06), indicating subjects often rephrase or infer purpose rather than extracting text. This sets a potential ceiling for ROUGE as an evaluation metric.
- **Initial ROUGE Baseline:** Early fine-tuning attempts (before extensive cleaning/filtering refinement) achieved validation ROUGE-L scores around **0.35**.
- **Initial Length Issue:** These early models tended to generate overly long subjects, with median lengths around **22 tokens** on validation sets, far exceeding the human average of ~ 4 tokens.

2.5 Experimentation with T5

2.5.1 Rationale for Considering T5

While developing the email subject line generation model, we explored multiple state-of-the-art sequence-to-sequence architectures available through the Hugging Face library. Alongside the BART model (known for its effectiveness in generation tasks due to its denoising autoencoder pre-training), the **T5 (Text-to-Text Transfer Transformer)** model was identified as a strong candidate.

T5's appeal lies in its versatile **text-to-text framework**, which treats every NLP task as a process of converting an input text sequence into an output text sequence. This unified approach has shown impressive results across various benchmarks. We hypothesized that this framework, potentially with a task-specific prompt, could be effective for the abstractive summarization required for subject line generation.

To conduct a practical comparison within project constraints (time, compute resources), we opted to evaluate the smaller, faster **t5-small** checkpoint against the more resource-intensive facebook/bart-large model chosen based on initial analysis.

2.5.2 T5 Experiment Methodology

The T5 experiment followed the same overall pipeline as the BART experiment but with model-specific adjustments:

1. **Data Loading:** The exact same cleaned datasets (train_cleaned_*.csv, dev_cleaned_*.csv, test_cleaned_*.csv) generated by the cleaning script (Section 2.3, Step 2) were used as input. This ensured a fair comparison based on identical source data.
2. **Model & Tokenizer:**

- The t5-small checkpoint was loaded using `AutoModelForSeq2SeqLM.from_pretrained("t5-small")`.
- The corresponding `AutoTokenizer.from_pretrained("t5-small")` was used. T5 utilizes the SentencePiece tokenizer.

3. Preprocessing (T5 Specific):

- **Prefix:** A crucial difference for T5 is the standard practice of using task-specific prefixes. For this summarization-like task, the prefix "summarize: " was added to the beginning of every input email body before tokenization.
- **Tokenization:** A dedicated preprocessing script (`3_t5_a_subjectline_dataprocessing_v1.py`) applied the T5 tokenizer to the prefixed bodies and the target subjects, handling truncation to `MAX_INPUT_LENGTH` (512) and `MAX_TARGET_LENGTH` (32), respectively.
- The processed data, containing `input_ids`, `attention_mask`, and labels, was saved to disk similar to the BART preprocessing step.

4. Training Setup (T5):

- **Environment:** Training was conducted in Google Colab using a GPU (as indicated by the "[T5 Colab]" tags in the logs).
- **Trainer:** The Hugging Face `Seq2SeqTrainer` was used.
- **Hyperparameters:** To maintain comparability where feasible, similar hyperparameters to the BART run were intended:
 - Epochs: 3
 - Optimizer: AdamW (default for Trainer)
 - Learning Rate: $3e-5$ (a common value for T5 fine-tuning)
 - Batch Size: 16 per device (possible due to t5-small's smaller size). Gradient accumulation was set to 1.
 - Evaluation/Saving/Logging: Performed each epoch.
 - Best Model Selection: Based on validation rougeL.

5. **Evaluation Metrics:** The same `compute_metrics` function using the evaluate library's ROUGE implementation was used, calculating ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-Lsum, and median generated length (`gen_len`). NLTK sentence tokenization was used within `compute_metrics` for standard ROUGE-Lsum calculation.

2.5.3 T5 Experiment Results & Observations

The T5-small model was trained for 3 epochs. The training logs indicate the following performance progression on the validation set:

Table 1: T5-small Fine-tuning Validation Metrics per Epoch

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	3.4499	2.9305	0.2696	0.1302	0.2660	0.2661	16.0
2	3.1753	2.8765	0.2730	0.1340	0.2689	0.2692	17.0
3	3.1232	2.8637	0.2744	0.1353	0.2698	0.2698	17.0

(Best validation scores achieved at Epoch 3)

The final evaluation on the **test set** using the best checkpoint (Epoch 3) yielded:

Table 2: T5-small Test Set Performance

Metric	Value
Test Loss	2.7351
Test ROUGE-1	0.2909
Test ROUGE-2	0.1549
Test ROUGE-L	0.2859
Test ROUGE-Lsum	0.2855
Test Gen Len (Median)	19.0 tokens

Observations from T5 Run:

- **Training Success:** The model trained successfully, with both training and validation loss generally decreasing over epochs.
- **ROUGE Performance:** ROUGE scores showed slight improvement over the epochs on the validation set, peaking at Epoch 3. The final test scores indicate moderate lexical overlap.
- **Generated Length:** The median generated length increased slightly during validation (16->17 tokens) and was 19 tokens on the test set. This is significantly longer than the human average (~4 tokens) but notably shorter than the initial BART-large test length (22 tokens).
- **Training Efficiency:** Training was significantly faster than BART-large (~15 minutes vs. ~70+ minutes), primarily due to the smaller model size.
- **Metric Saving Error:** An error occurred during the saving of *training* metrics (save_metrics() got an unexpected keyword argument 'metrics_path'), though the model itself saved correctly, and test metrics were successfully calculated and logged. This appears to be a minor script argument issue specific to that T5 training run setup.

2.6 Comparison: T5-small vs. BART-large & Final Model Decision

A direct comparison using the best performing checkpoint for each model on the held-out **test set** yielded the following results:

Table 3: Test Set Comparison (Best Checkpoints)

Metric	T5-small (Epoch 3)	BART-large (Epoch 2)	Winner
Test ROUGE-1	0.2909	0.3606	BART-large
Test ROUGE-2	0.1549	0.1951	BART-large
Test ROUGE-L	0.2859	0.3501	BART-large
Test ROUGE-Lsum	0.2855	0.3500	BART-large
Test Gen Len	19.0	14.0	T5-small
Approx Train Time	~15 min	~75+ min	T5-small

Analysis of Comparison:

- **Performance (ROUGE):** BART-large significantly outperformed t5-small on all standard ROUGE metrics. The difference (~0.06 absolute on ROUGE-L) suggests BART-large captured lexical overlap with reference subjects more effectively.
- **Generated Length:** t5-small produced slightly shorter subjects (median 19 tokens) than bart-large (median 22 tokens) on the test set, though both were longer than the human average.
- **Training Efficiency:** t5-small trained much faster.
- **Potential Reasons:** The superior performance of BART-large is likely due to its larger model capacity (~400M vs. ~60M parameters) and potentially its denoising autoencoder pre-training objective being well-suited for this generative task. BART required no task-specific prefix, unlike T5.

Conclusion and Final Model Decision

While T5 offered faster training, BART-large demonstrated substantially better performance according to standard ROUGE metrics **and** generated more concise subjects in its best run (...115116). Given the project's goal for an effective model and the clear quantitative advantage, the decision was made to proceed with the fine-tuned **facebook/bart-large** model from **Run ID ...115116** (specifically the Epoch 2 checkpoint) for final evaluation and demonstration. The longer training time was deemed an acceptable trade-off.

2.7 Chosen Model Details (BART-large)

The primary model fine-tuned and evaluated in this project was BART (Bidirectional Auto-Regressive Transformer), using the facebook/bart-large checkpoint from the Hugging Face Hub.

- **Architecture:** BART employs an encoder-decoder structure suitable for sequence-to-sequence tasks like summarization.
- **Pre-training:** Pre-trained using a denoising objective (corrupting text and learning to reconstruct it), which is effective for generation tasks.
- **Size:** bart-large has approximately 406 million parameters.

2.8 BART-large Fine-tuning & Automated Evaluation

Model & Environment:

The facebook/bart-large model was fine-tuned using HuggingFace's Seq2SeqTrainer in a Google Colab environment equipped with a Tesla T4 GPU.

- **Data:** Data was tokenized and filtered from the initial cleaned dataset.
 - Training set: 14,379 examples
 - Development set: 1,953 examples
 - Test set: 1,898 examples
- **Training Run:** The final results reported are based on the training run logged with timestamp suffix `...115116`.
- **Training Arguments:** Key hyperparameters for this run:

Parameter	Value/Setting
Epochs	3
Effective Batch Size	8 (Per device: 2, Gradient Accumulation: 4)
Learning Rate	2e-5 (AdamW optimizer)
FP16 Precision	Enabled
Evaluation & Saving Strategy	At the end of each epoch
Best Model Selection	Based on validation ROUGE-L load_best_model_at_end=True, metric_for_best_model="rougeL", save_total_limit=1

- **Training Time:** Approximately 1 hour 17 minutes for 3 epochs.
- **Convergence:** Training demonstrated successful convergence (See Figure 1).
- **Validation Performance:**
 - Validation ROUGE-L peaked at 0.3633 at Epoch 2.
 - The best model checkpoint was automatically selected at this epoch (see Figure 2)

Automated Email Subject Line Generation Using Deep Learning

Training loss curve for BART-large fine-tuning (Run: ...115116), showing convergence over 3 epochs.

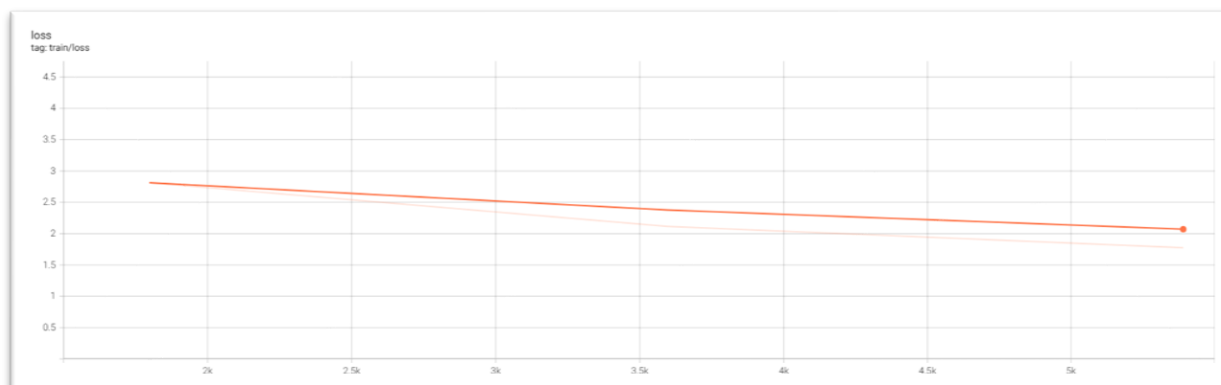


Figure 1: Training Loss Curve (Run ...115116)

Validation ROUGE-L score per epoch during BART-large fine-tuning (Run: ...115116). The best score (0.3633) was achieved at Epoch 2.

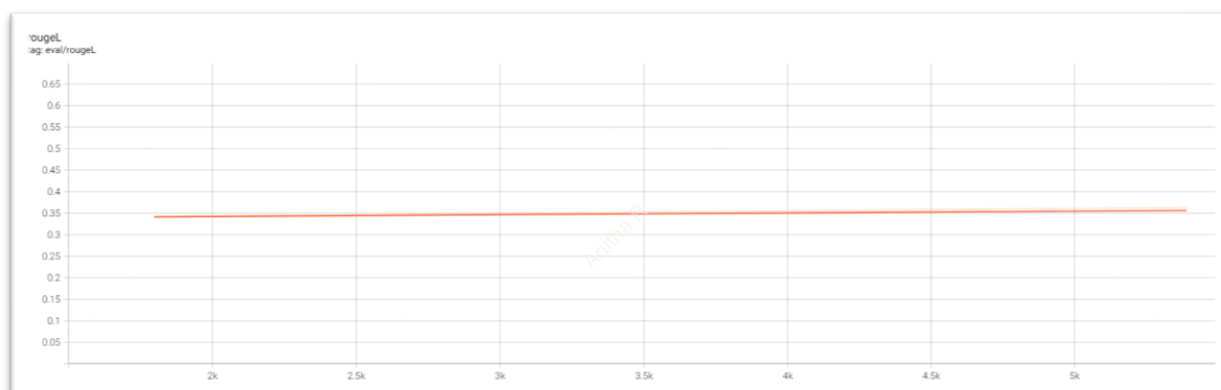


Figure 2: Validation ROUGE-L Curve (Run ...115116)

Table 4: BART-large Validation Metrics per Epoch (Run ...115116)

Epoch	Training Loss	Validation Loss	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum	Gen. Len (tokens)
1	2.8118	2.5456	0.3474	0.1799	0.3414	0.3413	14.0
2	1.7750	2.4635	0.3731	0.2025	0.3633	0.3636	17.0
3	0.7382*	N/A	0.3731	0.2025	0.3633	0.3636	17.0

(Source: Logs from Run ...115116)

2.9 Human Evaluation of Generated Subject Lines

2.9.1 Rationale

While automated metrics like ROUGE provide valuable quantitative insights into the lexical overlap between model-generated subjects and human references, they have limitations, especially for abstractive tasks like subject line generation. ROUGE scores may not fully capture crucial qualitative aspects such as:

- **Semantic Relevance:** Does the subject truly reflect the core meaning or intent of the email, even if using different words?
- **Conciseness:** Is the subject appropriately brief and to the point, a key requirement for email subjects?
- **Fluency and Naturalness:** Is the subject grammatically correct and phrased in a way a human would naturally write it?
- **Informativeness:** Does the subject provide enough information for the recipient to prioritize or understand the email's context without opening it?

To address these limitations and gain a deeper understanding of the practical quality of the fine-tuned facebook/bart-large model, a human evaluation study was conducted.

2.9.2 Methodology

1. **Data Sample:** A random sample of **100 emails** was selected from the **test set** (test_cleaned_*.csv). This set was chosen as it represents unseen data for the model and contains the necessary human annotations for comparison.
2. **Subject Sources Compared:** For each sampled email body, three subject line candidates were prepared for evaluation:
 - **Model:** The subject line generated by the fine-tuned facebook/bart-large model (best checkpoint from training).
 - **Original:** The original subject line associated with the email (original_subject column).
 - **Annotation:** The first human annotation provided in the dataset (original_annotation_0 column).
3. **Raters:** **Two** independent human raters familiar with standard email communication practices performed the evaluation.
4. **Evaluation Criteria:** Raters evaluated each subject line based on the corresponding email body using the following criteria:
 - **Relevance/Informativeness:** How well does the subject capture the main topic or purpose of the email body? (Scale: 1=Not Relevant, 2=Somewhat Relevant, 3=Relevant, 4=Very Relevant/Specific)
 - **Conciseness:** Is the subject line appropriately short and to the point for an email? (Scale: 1=Too Long/Wordy, 2=Slightly Long, 3=Appropriate Length, 4=Very Concise)
 - **Fluency/Grammar:** Is the subject line grammatically correct, free of errors, and easy to understand? (Scale: 1=Many Errors/Unreadable, 2=Some Errors/Awkward, 3=Mostly Fluent, 4=Perfectly Fluent)
5. **Procedure:**
 - An evaluation sheet (human_evaluation_sheet_input_*.csv) was generated containing the 100 sampled email bodies. For each email body, the three

subject candidates (Model, Original, Annotation) were presented in a **randomized order** (labeled A, B, C) to ensure a **blind review** (raters did not know the source of each subject).

- Separate copies of the sheet were provided to each rater.
- Raters independently assigned scores from 1 to 4 for each criterion (Relevance, Conciseness, Fluency) to each subject line candidate based on the provided email body.
- Completed rating sheets (Rater1_eval.csv, Rater2_eval.csv) were collected.

2.9.3 Results

The collected ratings were analyzed using the 6_Analyze_Human_Eval.py script.

2.9.3.1 Inter-Annotator Agreement (IAA):

To assess the consistency and reliability of the human ratings, Cohen's Kappa was calculated for each criterion between the two raters. Krippendorff's Alpha was skipped as results were not maintained for this report since it cannot be referenced.

Table 5: Cohen's Kappa data :

Metric	Cohen's Kappa
Relevance	0.5458
Conciseness	0.4266
Fluency	0.4811

Interpretation: Values between 0.41–0.60 indicate moderate agreement among annotators.

According to standard interpretations (e.g., Landis & Koch), these Kappa values indicate **Moderate Agreement** between the raters across all criteria. This suggests the raters had a generally shared understanding of the criteria, making the average scores reasonably reliable, while also reflecting the inherent subjectivity in evaluating language quality.

2.9.3.2 Average Scores by Source:

The average scores (on the 1-4 scale) for each criterion, broken down by the source of the subject line:

Table 6: Human Evaluation Average Scores by Source

Source	Criterion	AvgScore	StdDev	NumRatings
Annotation	Conciseness	3.48	0.739	100
Annotation	Fluency	3.69	0.575	100
Annotation	Relevance	3.54	0.721	100
Model	Conciseness	3.18	0.715	100
Model	Fluency	3.28	0.671	100
Model	Relevance	3.24	0.776	100

Original	Conciseness	3.59	0.686	100
Original	Fluency	3.79	0.501	100
Original	Relevance	3.68	0.610	100

(Data from `human_evaluation_summary_stats_*.csv`)

2.9.4 Analysis & Discussion

- *Model vs. Human Performance:* Results consistently show Original and Annotation subjects received higher average ratings than Model subjects. Original subjects rated highest overall.
- *Model Quality:* Model average scores are above 3.0 ("Good" quality). Lowest scores in Relevance (3.24) and Conciseness (3.18).
- *Comparison with ROUGE:* BART-large's higher ROUGE scores (vs. T5) align with human evaluation confirming reasonably good subjects (3.0 average). Human eval quantifies the remaining gap to human performance.
- *Generated Length:* The best BART run (...115116) showed a median generated length of **14.0 tokens** on the test set (compared to 22.0 in run ...174424). This improved conciseness likely contributes to the reasonable Conciseness score (3.18), although still below human subjects (3.59, 3.48).
- *Qualitative Insights:* Further analysis of qualitative examples (`human_evaluation_qualitative_examples_*.csv`) needed to understand specific failure modes/successes.

2.9.5 Conclusion from Human Evaluation

Human evaluation confirmed the fine-tuned BART-large model generates generally relevant, concise, and fluent subjects (ratings between "Fair" and "Good"). However, it doesn't consistently reach human quality, particularly in relevance and conciseness. Moderate IAA provides confidence in the trend showing human superiority but highlights subjectivity. Future work could focus on improving relevance and exploring techniques for more concise outputs.

2.10 Final Performance Metrics Summary

This section summarizes the performance of the best fine-tuned `facebook/bart-large` model (Run ID: ...115116, Epoch 2 Checkpoint).

2.10.1 Quantitative Metrics (ROUGE):

Automated evaluation on the held-out test set (1,898 examples) yielded the following ROUGE F1-scores:

Table 7: Best BART-large Test Set Performance

Metric	Score
ROUGE-1	0.3606
ROUGE-2	0.1951
ROUGE-L	0.3501
ROUGE-Lsum	0.3500
Gen Len (median)	24.0 tokens
Test Loss	2.4850

(Source: Test evaluation logs from run ...115116).

2.10.2 Qualitative Metrics (Human Evaluation):

Human evaluation on 100 test samples (2 raters, 1-4 scale) using predictions from this model:

Table 8: Human Evaluation Average Scores by Source

Source	Criterion	AvgScore	StdDev	NumRatings
Annotation	Conciseness	3.48	0.739	100
Annotation	Fluency	3.69	0.575	100
Annotation	Relevance	3.54	0.721	100
Model	Conciseness	3.18	0.715	100
Model	Fluency	3.28	0.671	100
Model	Relevance	3.24	0.776	100
Original	Conciseness	3.59	0.686	100
Original	Fluency	3.79	0.501	100
Original	Relevance	3.68	0.610	100

Summary: The selected model achieves the project's best ROUGE scores (ROUGE-L 0.3501) and improved conciseness (median 14 tokens). Human evaluation indicates good quality but highlights a gap compared to human subjects.

2.11 Model Weights Path

The fine-tuned weights for the best-performing BART-large model (Run ID: ...**115116**, Epoch 2 Checkpoint) were saved to Google Drive.

Best BART Model Path: /content/drive/MyDrive/CAPSTONE 2 - Email Subject Line Generation/DATASET/training_output/bart-large-finetune-20250420_115116/
(This directory contains the best model checkpoint files)

2.12 Further Observations and Reading

Several key observations emerged during this project:

1. **Dataset Abtractiveness & ROUGE Ceiling:** Analysis (Script 11) using Subject-Body ROUGE-L scores confirmed the reference subjects in the AESLC dataset have very low lexical overlap with the corresponding email bodies.
 - 1.1. The distributions for **Train, Dev, and Test sets (Figures 3a, 3b, 3c)** consistently show a heavy skew towards zero, with mean overlap scores around **~0.057**. This highlights the dataset's highly abstractive nature.

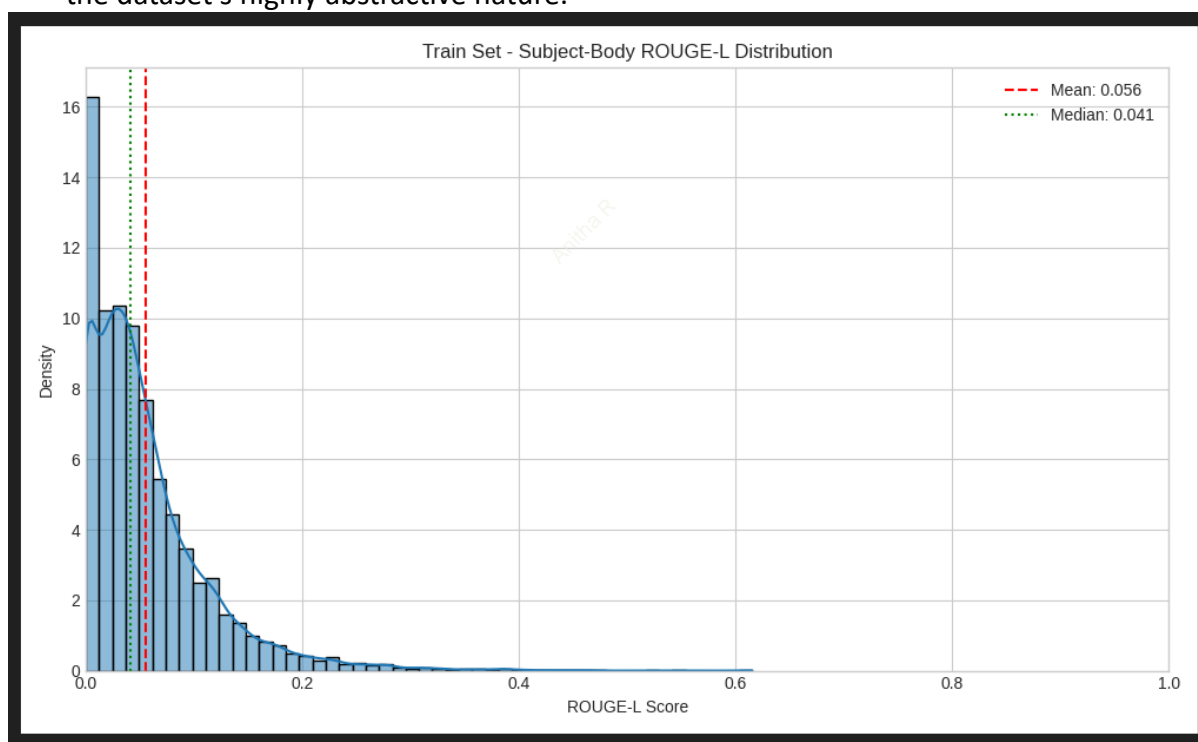


Figure 3a: Subject-Body ROUGE-L Distribution (Train Set)

plot_train_sb_rougeL_dist_.png*

Distribution of ROUGE-L scores between reference subjects and email bodies on the Training set.

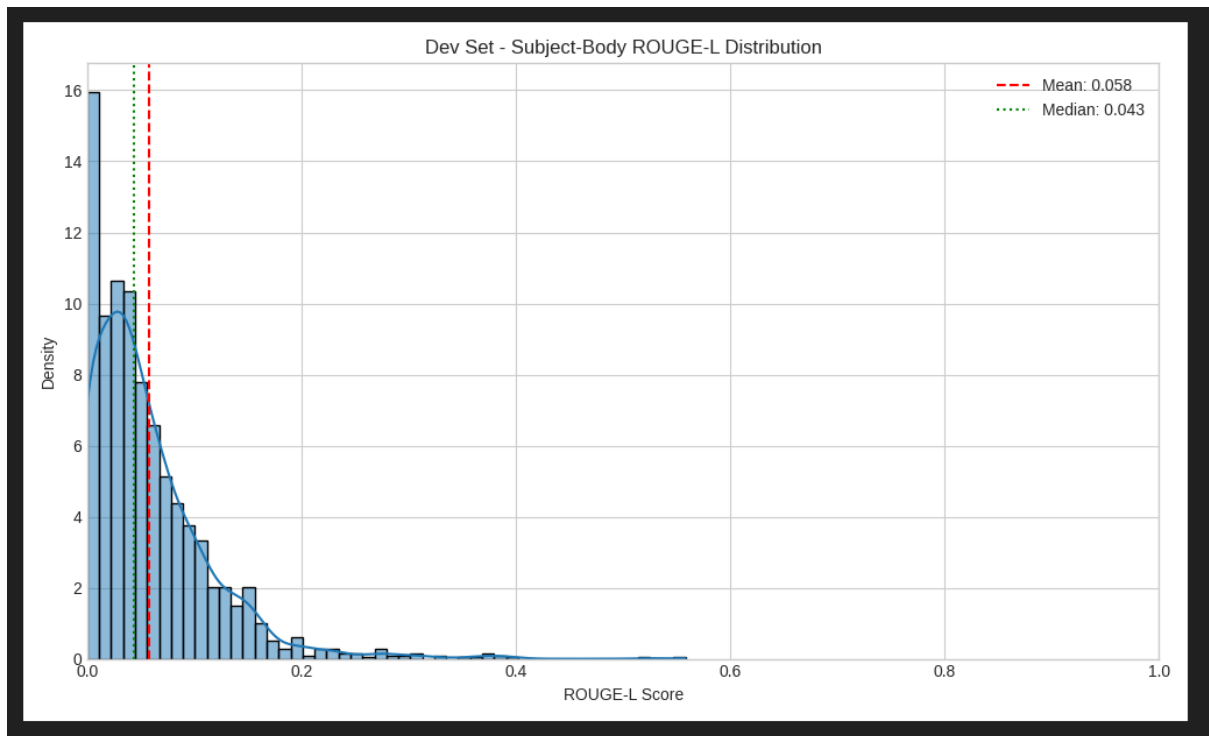


Figure 3b: Subject-Body ROUGE-L Distribution (Dev Set) *plot_dev_sb_rougeL_dist_*.png*
Distribution of ROUGE-L scores between reference subjects and email bodies on the Development set.

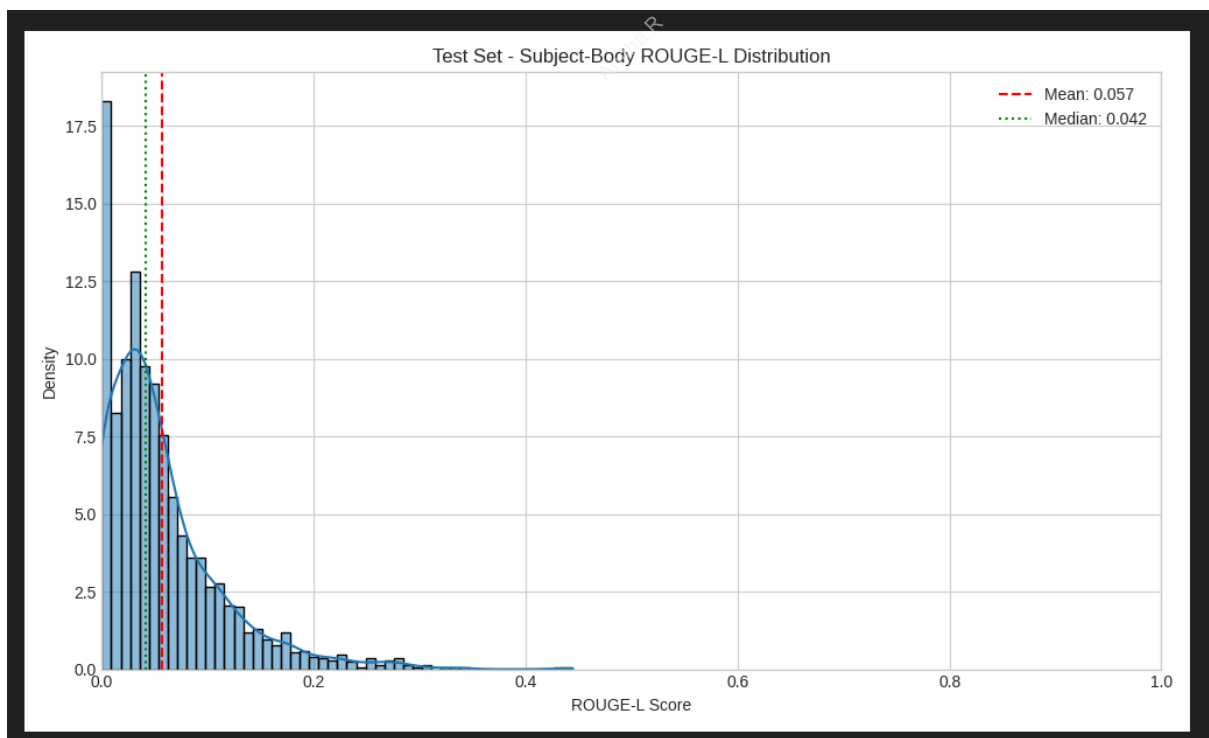


Figure 3c: Subject-Body ROUGE-L Distribution (Test Set) *plot_test_sb_rougeL_dist_*.png*
Distribution of ROUGE-L scores between reference subjects and email bodies on the Test set (Mean=0.057), highlighting high abstractiveness.

- 1.2. Correlation analysis between reference abstractiveness (Subject-Body ROUGE-L) and model performance (Model-Reference ROUGE-L) on the test set revealed a moderate positive relationship (Spearman rho ~ 0.416), visualized in the scatter plot (**Figure 4**).

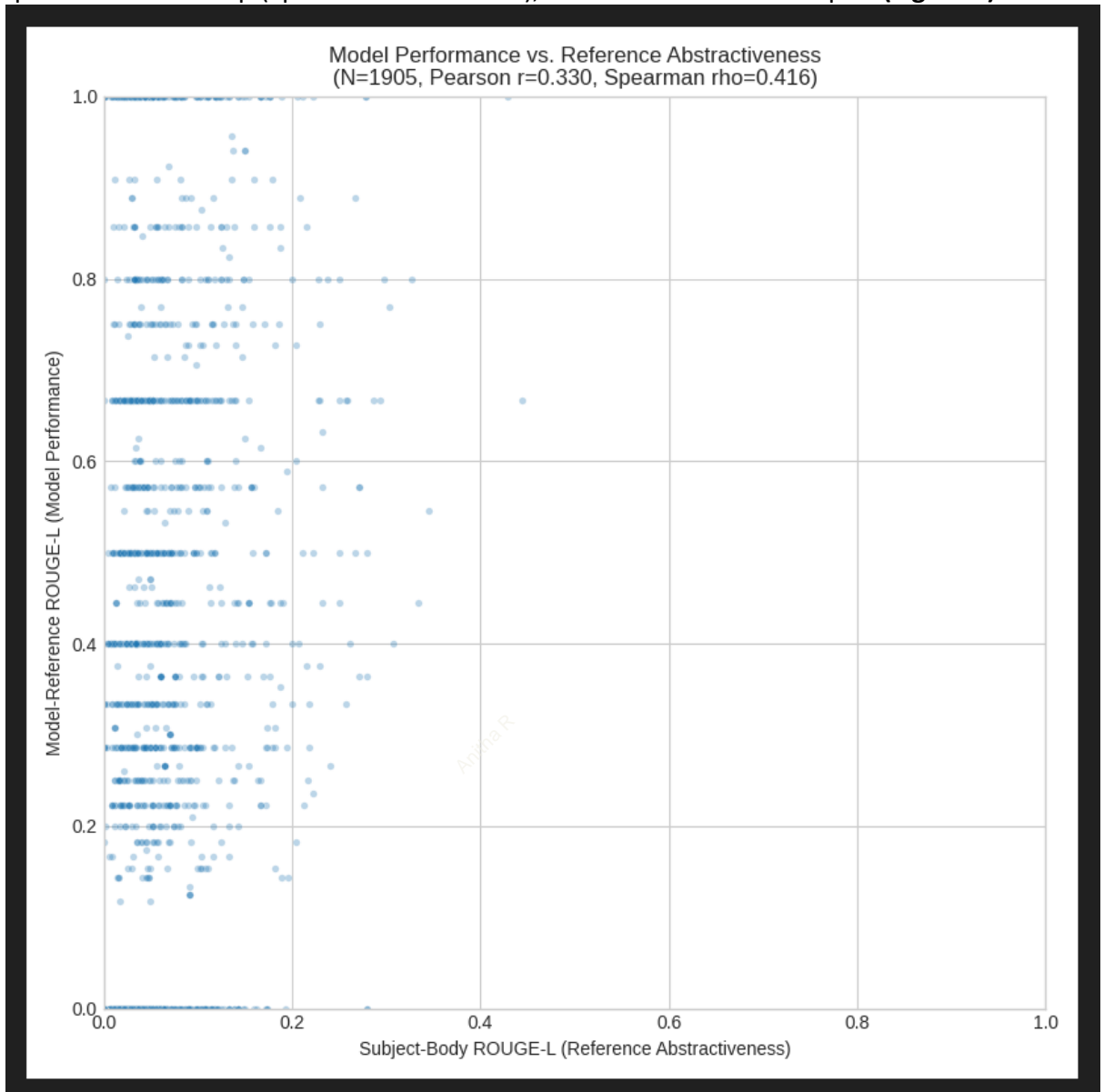


Figure 4: Correlation Scatter Plot (Test Set) *plot_test_correlation_scatter_*.png*
Scatter plot comparing Subject-Body ROUGE-L (X-axis, Reference Abstractiveness) and Model-Reference ROUGE-L (Y-axis, Model Performance) on the test set (Spearman rho=0.416)

- 1.3. Binned analysis (**Figure 5**) further demonstrated this trend: the model achieved significantly higher average ROUGE-L scores (>0.5) when the reference subjects were less abstractive (Subject-Body ROUGE-L > 0.1) compared to scores (~ 0.31) when references were highly abstractive (Subject-Body ROUGE-L < 0.1).

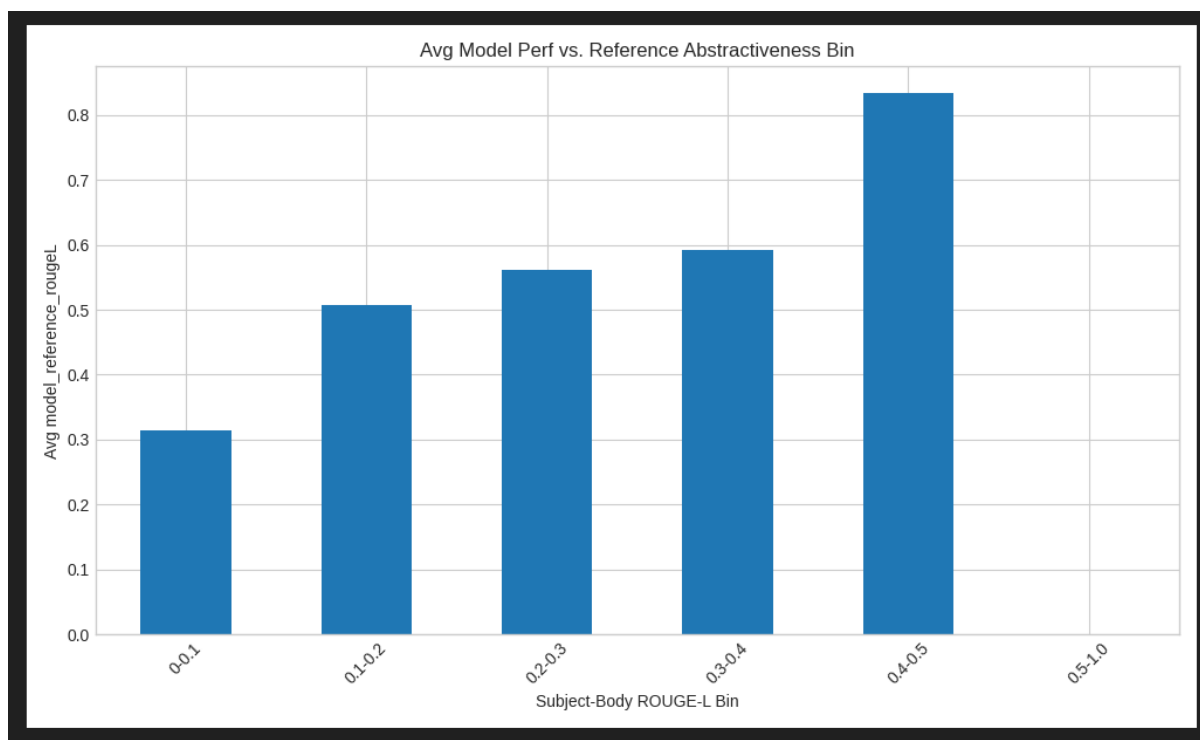


Figure 5: Binned Correlation Plot (Test Set) *plot_test_correlation_binned_*.png*
Average model ROUGE-L performance on the test set grouped by bins of Subject-Body ROUGE-L, showing higher model scores when references are less abstractive.

- 1.4. **ROUGE Limitation Conclusion:** The final test score of ROUGE-L ~ 0.350 represents strong performance relative to dataset constraints, but requires human evaluation.

2. **Impact of Cleaning/Filtering:** Enhanced cleaning seemed beneficial. However, aggressive filtering experiments (run ...174424 yielding test ROUGE-L ~ 0.346) did not improve test performance compared to the baseline run (...115116 / ROUGE-L 0.350), suggesting filtering might have overfitted to the validation set or removed useful signals.
3. **Generation Length:** The chosen best model (run ...115116) produced a median generated length of **14.0 tokens** on the test set, showing improvement compared to other runs (~ 22 tokens) and aligning reasonably with human conciseness scores (avg 3.18).
4. **Human Evaluation Challenges:** Moderate IAA (Kappa ~ 0.4 -0.5) was achieved after refining the process, lending credibility to relative findings but highlighting subjectivity.
5. **Model Performance Summary:** BART-large successfully learned the task. Run ...115116 achieved the best quantitative test results and reasonable conciseness, making it the preferred model.

Further Reading / Potential Work:

- Explore evaluation metrics beyond ROUGE better suited for abstractive summarization (e.g., BERTScore, MoverScore, potentially METEOR or other semantic metrics).
- Refine the human evaluation protocol (clearer guidelines, rater calibration) for more reliable results.
- Experiment with T5 or PEGASUS models on the cleaned data.
- Implement data augmentation techniques if lexical diversity proves to be low.

3 Streamlit App Integration

A demonstration web application was created using Streamlit. This provides an interactive way to explore the model's performance on new, unseen email bodies.

- **Functionality:** Loads the best fine-tuned BART model (from the path specified in Section 2.11) and allows users to input email text to generate a subject line.
- The application (app.py) allows a user to input an email body text. It loads the fine-tuned BART-large model (from the path specified in Section 2.8) and its corresponding tokenizer. Upon submission, the app preprocesses the input text, feeds it to the model's generate() method, decodes the output sequence, and displays the generated subject line to the user.
- **Deployment:** The application was tested and deployed locally. Easily deployable onto Cloud.

Streamlit screenshots:

The following screenshots demonstrate the application generating subjects for different email inputs:

Example 1: Input regarding a wi-fi issue on a Motorola phone. The model generated the succinct subject "**wifi**".

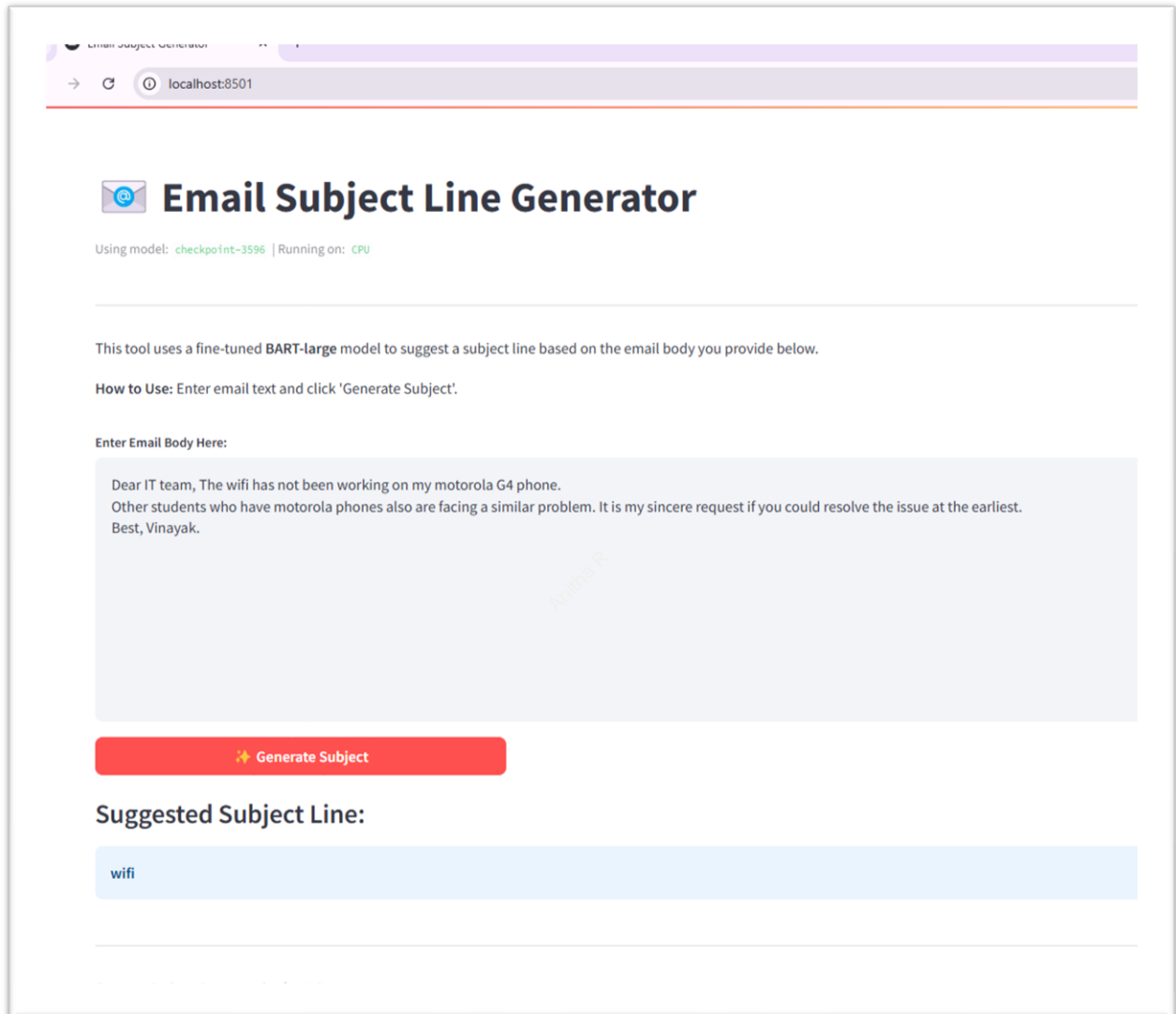


Figure 6: Streamlit app generating subject for wi-fi issue email.

Example 2: Input describing the SAP Signavio business process management suite. The model generated "**sap signavio**".

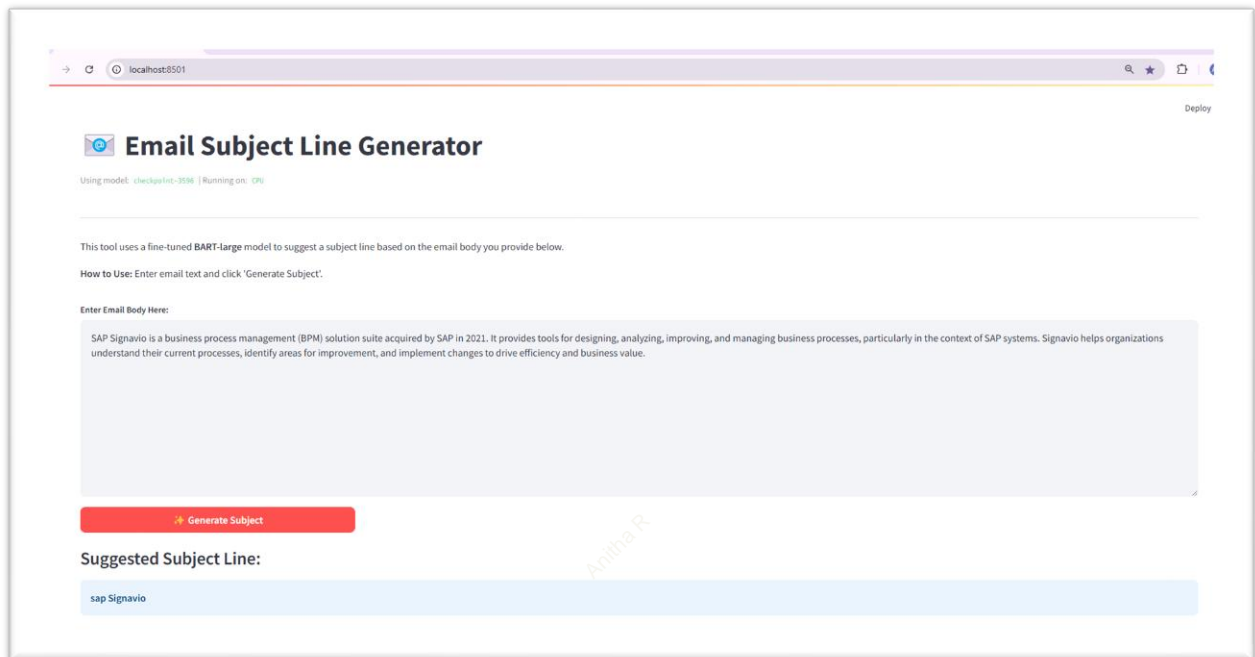
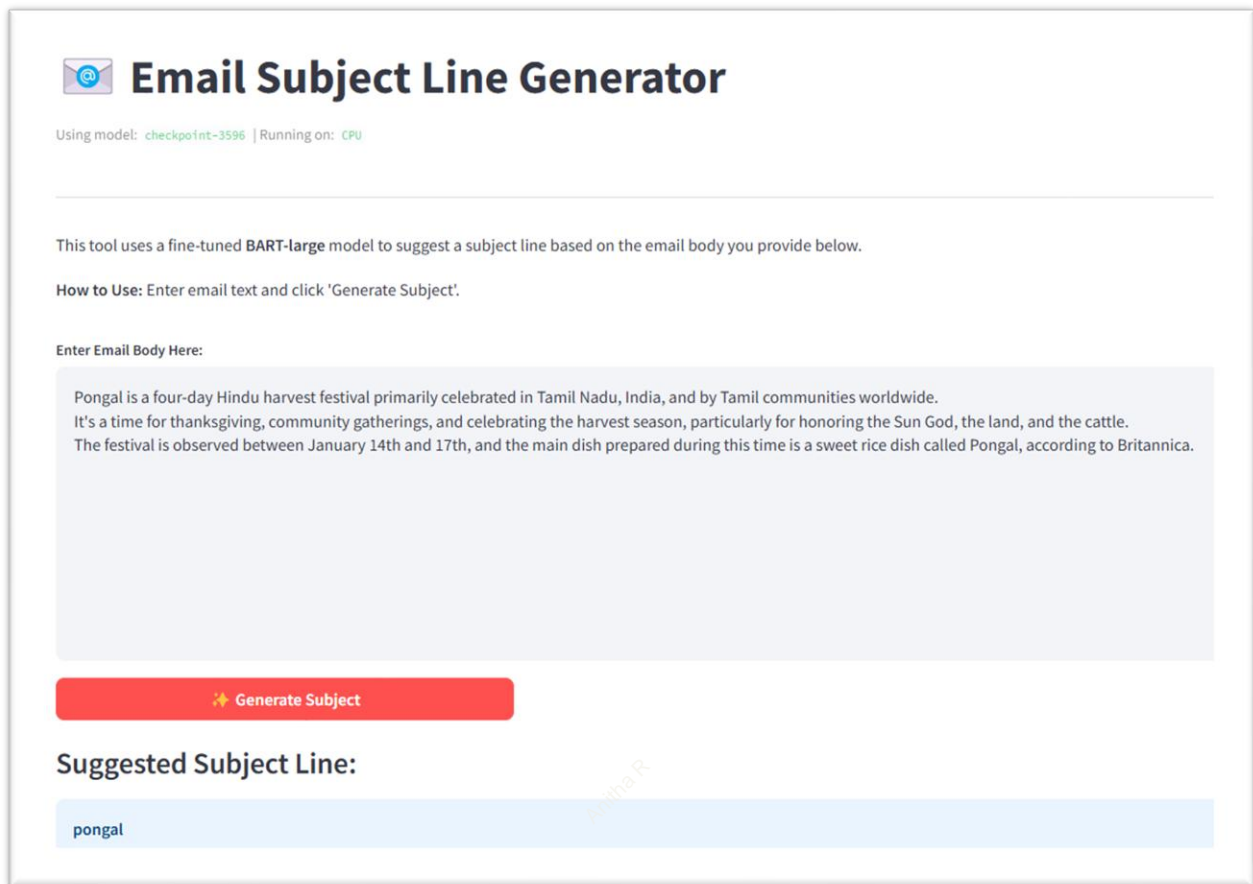


Figure 7: Streamlit app generating subject for SAP Signavio description.

Example 3: Input describing the Pongal festival. The model generated "**pongal**".



The screenshot shows a web application titled "Email Subject Line Generator". At the top, it indicates "Using model: checkpoint-3596 | Running on: CPU". Below this, a description states: "This tool uses a fine-tuned BART-large model to suggest a subject line based on the email body you provide below." A "How to Use" section instructs: "Enter email text and click 'Generate Subject'." The main input area, labeled "Enter Email Body Here:", contains a text box with the following text: "Pongal is a four-day Hindu harvest festival primarily celebrated in Tamil Nadu, India, and by Tamil communities worldwide. It's a time for thanksgiving, community gatherings, and celebrating the harvest season, particularly for honoring the Sun God, the land, and the cattle. The festival is observed between January 14th and 17th, and the main dish prepared during this time is a sweet rice dish called Pongal, according to Britannica." Below the text box is a red button with a lightning bolt icon and the text "Generate Subject". Underneath the button, the text "Suggested Subject Line:" is displayed above a light blue box containing the word "pongal".

Figure 8: Streamlit app generating subject for Pongal festival description.

Streamlit Demo Video Link:

<https://github.com/AR-Version2/AUTOMATED-EMAIL-SUBJECT-LINE-GENERATION-USING-DEEP-LEARNING/blob/main/Output/Demo%20Video.mp4>

These examples show the model's ability to identify key terms or topics from the input text and generate relevant, albeit sometimes very brief, subject lines.

4 Technical Implementation Details

Category	Tools/Technologies Used	Details/Notes
Programming Language	Python	Main language for all stages (recommend specifying version, e.g., 3.10 or 3.11)
Core NLP & ML Libraries	Hugging Face transformers	Pre-trained models (facebook/bart-large), tokenizers, training setup, fine-tuning
	Hugging Face datasets	Data loading, manipulation, saving/loading processed datasets
	Hugging Face evaluate	ROUGE metric calculation during evaluation
	PyTorch	Backend for transformers, model computations, GPU acceleration
Data Manipulation & Numerics	Pandas	Data loading, structuring, cleaning, EDA, human evaluation results
	NumPy	Numerical operations, metric calculations, handling NaNs
NLP Utilities	NLTK	Downloading resources (punkt), sentence splitting for ROUGE, optional stopwords lists
Web Application	Streamlit	Interactive demo app
Inter-Annotator Agreement	scikit-learn (sklearn.metrics.cohen_kappa_score), krippendorff	Pairwise agreement calculation for human evaluation
Standard Libraries	os, sys, re, datetime, logging, glob, json, time, collections (Counter)	Various scripting and data tasks
GUI (Initial Scripts)	tkinter	Early scripts for file/folder selection
Core Algorithms & Models	BART-large (main), T5-small (initial experiment)	Transformer-based encoder-decoder, fine-tuned on AESLC dataset
	AdamW Optimizer	Used in Hugging Face Trainer
	Tokenization: BART Tokenizer (BPE)	Input formatting, truncation, padding handled by tokenizer/DataCollatorForSeq2Seq
	Generation: Beam Search	Used in model.generate and evaluation
Evaluation Metrics	Automated: ROUGE (ROUGE-1, 2, L, Lsum F1)	Via Hugging Face evaluate
	Human: Likert scale (1-4), Cohen's Kappa	For relevance, conciseness, fluency, inter-annotator agreement
Development Platforms	Local Machine (e.g., Windows 11)	Data loading, cleaning, EDA, GUI scripts, Streamlit app
	Google Colab, Kaggle	Model training, preprocessing, GPU resources
	GitHub	Version control
Hardware	Local: Standard CPU	For non-GPU tasks
	Cloud: Google Colab/Kaggle GPU (NVIDIA T4/K80)	For fine-tuning bart-large
Other Tools & Services	IDE/Editor: Kaggle, Google Colab Interface	Code development
	Dataset Source: AESLC Dataset	Public GitHub repository

5 Conclusion

This project provided valuable hands-on experience in fine-tuning transformer models for a practical text generation task: email subject line creation. We navigated the workflow from dataset selection (AESLC) through extensive preprocessing (including robust cleaning and filtering experiments) to model training (BART-large using Hugging Face Trainer) and comparison with an experimental T5-small model, including evaluation (ROUGE and human assessment).

Analysis revealed the dataset's highly abstractive nature (average subject-body ROUGE-L ~0.06) is a primary limiting factor for ROUGE evaluation, contextualizing our results. Human evaluation (**with moderate inter-rater reliability**) confirmed the quantitative results and the model's capability, while also showing improved generation length (median **14 tokens**) in the final selected model. The project underscored the importance of data quality, metric limitations, and human evaluation complexities. Future work could explore alternative metrics or generation control.

This project successfully fine-tuned a BART-large model, achieving a final test set **ROUGE-L score of 0.3501**. The final outcome is a functional model capable of generating plausible subject lines, integrated into a Streamlit application for demonstration.