Technical Implementation Details

This section outlines the technical stack, methodologies, and environment utilized for the Email Subject Line Generation capstone project.

1. Programming Language:

- **Python:** The primary programming language used for all stages of the project, including data loading, cleaning, preprocessing, model training, evaluation, and the demonstration application. (Specify version if known, e.g., Python 3.10 or 3.11).

2. Key Libraries & Frameworks:

- **Core NLP & ML:**
    - **Hugging Face transformers:** Used extensively for accessing pre-trained models (AutoModelForSeq2SeqLM), tokenizers (AutoTokenizer), setting up training (Seq2SeqTrainingArguments), and running the fine-tuning process (Seq2SeqTrainer). Chosen model: facebook/bart-large.
    - **Hugging Face datasets:** Utilized for loading CSV data (load_dataset), efficient data manipulation (map, filter), and saving/loading processed datasets (save_to_disk, load_from_disk).
    - **Hugging Face evaluate:** Employed for calculating the ROUGE metric (evaluate.load("rouge")) during model evaluation.
    - **PyTorch:** Served as the backend deep learning framework for the transformers library, handling model computations and GPU acceleration.
- **Data Manipulation & Numerics:**
    - **Pandas:** Used heavily for initial data loading from CSVs, data structuring, manipulation during cleaning and EDA, and handling human evaluation results.
    - **NumPy:** Utilized for numerical operations, particularly during metric calculations and handling NaN values.
- **NLP Utilities:**
    - **NLTK (Natural Language Toolkit):** Used specifically for downloading resources (punkt, punkt_tab) required by the evaluate library for standard ROUGE score calculation (sentence splitting) and optionally for stopword lists during EDA.
- **Web Application:**
    - **Streamlit:** Used to build the interactive web-based demo application (app_vX.py) for showcasing the fine-tuned model.

- **Inter-Annotator Agreement (Optional but Used):**
    - **Scikit-learn:** Specifically sklearn.metrics.cohen_kappa_score was used to calculate pairwise inter-annotator agreement for the human evaluation results.
    - *(Mention krippendorff if you installed and used it)*.
- **Standard Libraries:** Utilized various Python standard libraries including os, sys, re (for regular expressions in cleaning), datetime, logging, glob, json, time, collections (Counter).
- **GUI (Initial Scripts):** tkinter was used in early local script versions for graphical file and folder selection.

3. Core Algorithms & Models:

- **Model Architecture:** The project primarily focused on fine-tuning a pre-trained **Transformer-based Encoder-Decoder model (BART-large)**. An initial experiment with **T5-small** was also conducted for comparison.
- **Training Paradigm: Fine-tuning / Transfer Learning** on the pre-trained BART-large model using the AESLC dataset.
- **Optimization: AdamW Optimizer** (default within the Hugging Face Trainer).
- **Preprocessing:**
    - **Tokenization:** BART Tokenizer (using Byte-Pair Encoding - BPE subwording) via AutoTokenizer.
    - **Input Formatting:** Preparing input_ids, attention_mask, and labels as required by the Seq2Seq model.
    - **Truncation & Padding:** Handled by the tokenizer and DataCollatorForSeq2Seq respectively.
- **Generation Strategy: Beam Search** (used during model.generate in the Streamlit app and potentially during evaluation with predict_with_generate=True).
- **Evaluation Metrics:**
    - **Automated:** ROUGE (ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-Lsum F1-scores).
    - **Human:** Likert scale ratings (e.g., 1-4) for Relevance, Conciseness, and Fluency. Inter-Annotator Agreement (Cohen's Kappa).

4. Development & Execution Platforms:

- **Local Machine:** Used for initial data loading, cleaning, EDA script development, running GUI-based scripts, and potentially running the Streamlit app. (Specify OS, e.g., Windows 11).
- **Google Colab:** Utilized primarily for the computationally intensive **model training phase** due to the availability of free GPU resources. Also used for running preprocessing scripts if needed.
- **Version Control:** GitHub (assumed, specify if used) for code storage and management.

5. Hardware:

- **Local:** Standard CPU.
- **Cloud:** Google Colab GPU (e.g., NVIDIA T4 or K80, specify if known) was essential for fine-tuning the bart-large model in a reasonable timeframe.

6. Other Tools & Services:

- **IDE/Editor:** (Specify what you used, e.g., Visual Studio Code, Google Colab Interface, Jupyter Notebook).
- **Dataset Source:** AESLC Dataset obtained from its public GitHub repository.

This comprehensive overview details the technical choices made throughout the project lifecycle, providing context for the implementation and results.

Same as above – formatted:

2 Technical Implementation Details

This section outlines the technical stack, methodologies, and environment utilized for the Email Subject Line Generation capstone project.
Here is your project's technical overview in a clear, readable table format:

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 stopword 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

Category	Tools/Technologies Used	Details/Notes
	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

4.1 Programming Language:

- **Python:** The primary programming language used for all stages of the project, including data loading, cleaning, preprocessing, model training, evaluation, and the demonstration application. (Specify version if known, e.g., Python 3.10 or 3.11).

4.2 Key Libraries & Frameworks:

- **Core NLP & ML:**
  - **Hugging Face transformers:** Used extensively for accessing pre-trained models (AutoModelForSeq2SeqLM), tokenizers (AutoTokenizer), setting up training (Seq2SeqTrainingArguments), and running the fine-tuning process (Seq2SeqTrainer). Chosen model: facebook/bart-large.
  - **Hugging Face datasets:** Utilized for loading CSV data (load_dataset), efficient data manipulation (map, filter), and saving/loading processed datasets (save_to_disk, load_from_disk).
  - **Hugging Face evaluate:** Employed for calculating the ROUGE metric (evaluate.load("rouge")) during model evaluation.
  - **PyTorch:** Served as the backend deep learning framework for the transformers library, handling model computations and GPU acceleration.
- **Data Manipulation & Numerics:**
  - **Pandas:** Used heavily for initial data loading from CSVs, data structuring, manipulation during cleaning and EDA, and handling human evaluation results.
  - **NumPy:** Utilized for numerical operations, particularly during metric calculations and handling NaN values.
- **NLP Utilities:**
  - **NLTK (Natural Language Toolkit):** Used specifically for downloading resources (punkt, punkt_tab) required by the evaluate library for standard ROUGE score calculation (sentence splitting) and optionally for stopword lists during EDA.
- **Web Application:**
  - **Streamlit:** Used to build the interactive web-based demo application (app_vX.py) for showcasing the fine-tuned model.
- **Inter-Annotator Agreement (Optional but Used):**
  - **Scikit-learn:** Specifically sklearn.metrics.cohen_kappa_score was used to calculate pairwise inter-annotator agreement for the human evaluation results.
  - *krippendorff*
- **Standard Libraries:** Utilized various Python standard libraries including os, sys, re (for regular expressions in cleaning), datetime, logging, glob, json, time, collections (Counter).
- **GUI (Initial Scripts):** tkinter was used in early local script versions for graphical file and folder selection.

4.3 Core Algorithms & Models:

- **Model Architecture:** The project primarily focused on fine-tuning a pre-trained Transformer-based Encoder-Decoder model **(BART-large)**. An initial experiment with **T5-small** was also conducted for comparison.

- **Training Paradigm: Fine-tuning / Transfer Learning** on the pre-trained BART-large model using the AESLC dataset.
- **Optimization: AdamW Optimizer** (default within the Hugging Face Trainer).
- **Preprocessing:**
    - **Tokenization:** BART Tokenizer (using Byte-Pair Encoding - BPE subwording) via AutoTokenizer.
    - **Input Formatting:** Preparing input_ids, attention_mask, and labels as required by the Seq2Seq model.
    - **Truncation & Padding:** Handled by the tokenizer and DataCollatorForSeq2Seq respectively.
- **Generation Strategy: Beam Search** (used during model.generate in the Streamlit app and potentially during evaluation with predict_with_generate=True).
- **Evaluation Metrics:**
    - **Automated:** ROUGE (ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-Lsum F1-scores).
    - **Human:** Likert scale ratings (e.g., 1-4) for Relevance, Conciseness, and Fluency. Inter-Annotator Agreement (Cohen's Kappa).

4.4 Development & Execution Platforms:

- **Local Machine:** Used for initial data loading, cleaning, EDA script development, running GUI-based scripts, and potentially running the Streamlit app. (Specify OS, e.g., Windows 11).
- **Google Colab and Kaggle:** Utilized primarily for the computationally intensive **model training phase** due to the availability of free GPU resources. Also used for running preprocessing scripts if needed.
- **Version Control:** GitHub for code storage and management.

4.5 Hardware:

- **Local:** Standard CPU (or all program executions that did not need GPU)
- **Cloud:** Google Colab GPU (e.g., NVIDIA T4 or K80, specify if known) and Kaggle GPU were essential for fine-tuning the bart-large model in a reasonable timeframe.

4.6 Other Tools & Services:

- **IDE/Editor:** (Kaggle, Google Colab Interface).
- **Dataset Source:** AESLC Dataset obtained from its public GitHub repository.

This comprehensive overview details the technical choices made throughout the project lifecycle, providing context for the implementation and results.