Project requirement analysis:

This document breaks down the project based on the provided description, reference paper, dataset details, and your EDA slides.

1. Understanding of the Project

- **Core Task:** The goal is to build a system that automatically generates a concise and informative email subject line given the body of an email. This is framed as a **highly abstractive text summarization** task with a very high compression ratio (average 75 body words to 4 subject words).
- **Dataset:** You'll be using the Annotated Enron Subject Line Corpus (AESLC), derived from the Enron email dataset. Key features include:
    - Pre-cleaned and filtered emails (boilerplate removed, short/redundant emails excluded, deduplicated).
    - Standard Train/Dev/Test splits.
    - **Crucially:** The Dev and Test sets contain **three human-written subject lines** per email (@subject seems to be the original, @ann0, @ann1, @ann2 are the crowdsourced ones). This is vital for robust evaluation, acknowledging subject line variability.
- **Challenges (Highlighted in Paper & Slides):**
    - **Extreme Abstraction & Compression:** Unlike news headlines which might extract key phrases, email subjects often require synthesizing the core *purpose* or *topic* in very few words, sometimes using terms not explicitly in the body.
    - **Implicit Shared Context:** Real-world email subjects often rely on context between sender and recipient (e.g., prior conversation, shared project knowledge) which is *not* available in the dataset text alone. This makes generating context-dependent subjects like "Meeting Update" difficult without that external knowledge. The model must rely solely on the email body.
    - **Subject Variability:** There's often no single "correct" subject line. The multiple annotations help account for this during evaluation.
- **Methodology (Inspired by Paper, adapted for Project):** The reference paper proposes a complex two-stage extractor-abstractor model with RL and a custom quality metric (ESQE). For a course project, a more practical approach would be to leverage **pre-trained sequence-to-sequence (Seq2Seq) models** (like BART, T5,

PEGASUS) from Hugging Face and **fine-tune** them on the AESLC dataset for the summarization task.

- **Evaluation:** Measuring success involves:
    - **Automatic Metrics:** Standard summarization metrics like ROUGE (R-1, R-2, R-L), potentially BLEU and METEOR. These should be calculated against the multiple reference subjects in the dev/test sets.
    - **(Potentially) Custom Metrics:** The paper's ESQE metric highlights the limitation of standard metrics. While replicating ESQE might be out of scope, understanding *why* it was created (better correlation with human judgment for this specific task) is important.
    - **Qualitative Analysis:** Manually inspecting generated subjects to see if they make sense, are relevant, and capture the email's essence.
- **Deployment:** The final model should be accessible via an API (FastAPI) and potentially a simple web interface (Streamlit), deployed on a cloud platform (Heroku/GCP).
- **EDA Importance:** Your slides correctly outline the need for thorough EDA to understand data characteristics (lengths, vocabulary, annotations) before modeling.

2. Proposed Approach

Here's a step-by-step plan incorporating your EDA ideas and the project requirements:

Phase 1: Setup and Data Exploration (EDA)

1. **Environment Setup:** Create a Python environment (e.g., Conda, venv). Install necessary libraries: torch or tensorflow, transformers, datasets, nltk, scikit-learn, pandas, matplotlib, seaborn, wandb, fastapi, uvicorn, streamlit.
2. **Data Loading:**
    - Download the AESLC dataset from the GitHub link.
    - Write Python code to parse the files. Pay attention to the structure mentioned in your slides (page 4):
        - **Training files:** Contain Email Body, @subject, Subject Line.
        - **Dev/Test files:** Contain Email Body, @subject, Subject Line, @ann0, Annotation 0, @ann1, Annotation 1, @ann2, Annotation 2.
    - Load the data into a suitable structure (e.g., Pandas DataFrames, Hugging Face Dataset objects). Store the body, the original subject (@subject), and all available annotations.

3. **Exploratory Data Analysis (EDA - Implementing your slides):**
    - **Basic Stats:** Calculate file counts per split, verify against documentation.
    - **Data Cleaning (as needed):** Apply steps from slide 14 (text decoding, lowercasing, whitespace). Be cautious with punctuation/special chars initially – analyze if they appear meaningfully in subject lines. Check for any remaining boilerplate/signatures missed by the original paper's preprocessing. Handle missing data if any.
    - **Length Analysis:** Calculate and plot distributions (histograms, box plots) of email body length (words/tokens) and subject line length (words/tokens) for all splits. Calculate averages (confirm ~75 words body, ~4 words subject).
    - **Vocabulary Analysis:** Tokenize text (using NLTK or a basic split initially). Find and plot most frequent words/n-grams in bodies and subjects. Generate word clouds.
    - **Annotation Analysis (Dev/Test):**
        - Analyze length distribution of annotations.
        - Calculate vocabulary overlap *between* the multiple annotations for the same email (e.g., using Jaccard similarity or ROUGE scores between ann0, ann1, ann2). This quantifies "Annotation Diversity".
        - Calculate keyword overlap between the email body and each of its subject lines/annotations.
    - **Preprocessing Exploration (Slide 15):** Experiment with stop word removal, stemming/lemmatization on *copies* of the data to see the impact, but the final preprocessing for the Seq2Seq model will likely just involve tokenizer-specific steps.
    - **Summarize EDA:** Document findings, focusing on characteristics relevant to modeling (lengths, vocabulary, potential challenges).

Phase 2: Data Preprocessing for Modeling

1. **Select Tokenizer:** Choose a tokenizer corresponding to the pre-trained Seq2Seq model you'll use (e.g., BartTokenizer, T5Tokenizer from Hugging Face).
2. **Preprocessing Pipeline:**
    - Define a function that takes an email body and subject(s) as input.
    - Apply necessary cleaning (lowercasing, etc. decided during EDA).

- Tokenize the email body (input text) and the target subject line(s) using the chosen tokenizer. Ensure you handle truncation/padding appropriately for model input.
- Format the data as required by the Hugging Face Trainer or your custom training loop (e.g., dictionaries with input_ids, attention_mask, labels).

3. **Apply to Splits:** Process the train, validation (dev), and test sets using this pipeline. Remember to handle the multiple reference subjects for validation/testing.

Phase 3: Model Training and Tuning

1. **Select Model:** Choose a pre-trained Seq2Seq model from Hugging Face suitable for summarization (e.g., google/pegasus-xsum, facebook/bart-large-cnn, t5-small, t5-base). Start with a smaller model for faster iteration.
2. **Define Training Setup:**
   - Use the Hugging Face Trainer API for simplicity or write a custom training loop in PyTorch/TensorFlow.
   - Configure training arguments (TrainingArguments): output directory, epochs, batch size, learning rate, weight decay, evaluation strategy (evaluate per epoch), save strategy.
   - **Integrate WandB:** Set up Weights & Biases for experiment tracking, logging metrics (loss, ROUGE), and visualizing results.
3. **Train/Fine-tune:** Train the model on the preprocessed AESLC train set, using the dev set for validation. The target during training will be the original subject line associated with @subject.
4. **Hyperparameter Tuning (Optional but Recommended):** Use the validation set results to tune hyperparameters (learning rate, batch size, number of epochs). Tools like Optuna or Ray Tune integrated with Trainer can help.

Phase 4: Evaluation

1. **Generate Predictions:** Use the fine-tuned model to generate subject lines for the test set email bodies.
2. **Automatic Metrics:**
   - Implement ROUGE score calculation (R-1, R-2, R-L) using a library like evaluate or rouge-score.
   - **Multi-Reference Evaluation:** For each email in the test set, calculate the ROUGE score of the generated subject against *each* of the available human

references (@subject, @ann0, @ann1, @ann2). Report the average or the maximum score achieved across references.

- Calculate other metrics like BLEU/METEOR if desired, again using multiple references.

3. **Qualitative Analysis:** Manually examine a sample of generated subjects from the test set. Compare them to the email body and the human references. Note patterns of success and failure (e.g., generic outputs, missed key info, factual errors, good conciseness).
4. **Compare to Baselines (Optional):** If time permits, implement a simple baseline (e.g., extract first sentence) and compare its ROUGE scores.

Phase 5: Deployment

1. **Save Model:** Save the fine-tuned model and tokenizer.
2. **Build API (FastAPI):**
   - Create a FastAPI application.
   - Define a POST endpoint (e.g., /generate-subject) that accepts JSON input containing the email body text.
   - Load the saved model and tokenizer within the API.
   - The endpoint should preprocess the input text, feed it to the model, decode the generated subject line, and return it as JSON.
3. **Build UI (Streamlit):**
   - Create a Streamlit application.
   - Add a text area for users to paste an email body.
   - Add a button ("Generate Subject").
   - When clicked, the Streamlit app should either call the FastAPI endpoint or load and run the model directly to display the generated subject line.
4. **Deploy:** Package the API/Streamlit app (using Docker is recommended) and deploy it to Heroku or Google Cloud Platform (e.g., Cloud Run).

Phase 6: Reporting

1. **Document:** Write a report detailing the project: Introduction, EDA findings, Preprocessing steps, Model selection and training details, Evaluation results (quantitative metrics and qualitative examples), Deployment description, Challenges faced, and Conclusion/Future Work.