**Generating Personalized Emails with GPT-2**

Group O: Yun Cao, Yue Liu, Muyang Cheng, Nan Chen

Github: [CS6120NLP/Final at main · awakn123/CS6120NLP (github.com)](https://github.com/awakn123/CS6120NLP/tree/main/Final)

Hugging Face: [EmailGeneration - a Hugging Face Space by 24NLPGroupO](https://huggingface.co/spaces/24NLPGroupO/EmailGeneration)

# Introduction

The primary objective of this project is to develop a sophisticated communication tool designed to enhance user engagement by generating personalized emails. This tool leverages the capabilities of artificial intelligence to tailor messages based on individual user inputs, thereby facilitating more meaningful connections.

To achieve this, we employed the [postbot/distilgpt2-emailgen-V2](https://huggingface.co/postbot/distilgpt2-emailgen-V2) model, a variant of the GPT-2 architecture, which is renowned for its efficiency in text generation. The scope of our study encompasses the training of this model using the [LightTai/personalized-email](https://huggingface.co/datasets/LightTai/personalized-email) dataset, aimed at optimizing the quality and relevance of the generated content.

Throughout this project, we utilized a suite of programming libraries including Transformers, datasets, torch, scikit-learn, and accelerate, each playing a crucial role in enhancing the model's performance and expediting the development process. This report will detail the methodologies employed, discuss the model's performance, and explore potential applications of the tool in professional networking contexts.

By bridging the gap between technological capability and user-centric communication, this project seeks to set a new standard for automated yet personal interactions on professional platforms.

# Analysis

## 2.1 Data Analysis

For data analysis, we used four input, product, gender, profession, and hobby to generate an email. We used four NLP methodology and techniques employed. First one is tokenization and text processing, we used the AutoTokenizer from the transformers library to tokenize input data, which is crucial for processing text in a format suitable for NLP models. Second one is dataset handling, we employed the datasets library to load and manage datasets, converting datasets to Pandas tables for easier handling and then back to the dataset format after preprocessing for training. Third one is neural network modeling, we used AutoModelForCausalLM from the transformers library to load a pretrained causal language model specifically designed for generating sequences of text. And the last one is evaluation metrics, we implemented evaluation metrics using the evaluate library (specifically the rouge metric), which is widely used to assess the quality of text generation against reference texts. This is critical for understanding model performance in generative tasks.

## 2.2 Model analysis

The following steps explain how we transfer dataset to model tokenize. First step is loading pre-trained model tokenizer, an AutoTokenizer is loaded with a specified pre-trained model checkpoint. The tokenizer is a crucial tool that converts text data into a format that can be processed by machine learning models. Second step is to use a tokenization function, the tokenize function is defined to tokenize the inputs. It takes examples from the dataset and pulls the product, gender, profession, and hobby columns. These columns are tokenized, meaning they are converted into a sequence of numbers that represent the text in a way the model can understand. Third step is processing the dataset, the dataset is split into a training set and a test set with a 80/20 split. The original data frame is then converted into a dataset object, which is a more efficient format for handling datasets in machine learning workflows. The last step is tokenizing the dataset, the actual tokenization process happens where the training and test sets are mapped with the tokenize function. This means that all the text data in these datasets is converted into tokens.

To provide a more detailed explanation of the code. First, in data preparation phase, the code starts by installs essential Python libraries and loads a personalized email dataset from Hugging Face, converting it to a pandas DataFrame. This format simplifies data manipulation and analysis. Second, in NLP model setup phase, it utilizes a pre-trained model specifically for email generation, employing AutoTokenizer for preparing the input data. The dataset is split into training and testing subsets, and a custom tokenization function processes these inputs for model training. Third, in training and evaluation phase, the model is trained with specified arguments, such as learning rate and epoch count, using the Trainer class from the transformers library. Post-training, the model’s performance is evaluated using rouge metrics and perplexity calculations. Fourth, in text generation and cleaning phase, a text generation pipeline is established with the trained model to produce emails based on user attributes like product type and hobby. The code includes functions to clean the generated text, removing extraneous elements to enhance clarity and relevance. In the end, this streamlined code demonstrates the use of advanced NLP techniques for automating the generation of contextually relevant, personalized emails, highlighting the capabilities of modern NLP technologies.

## 2.3 Evaluation

In this section, we were trying to find an appropriate evaluation metrics for our project. The primary test data utilized in this analysis includes a Target Email, an Email Generated by GPT 3.5, a Target Email being translated twice, Email Title, and a character "X". Our evaluation metrics consist of BLEU, BERT, and ROUGE scores.

The test data comprises:

1. **Target Email:** The original email used as the standard for comparison.
2. **Email Generated by GPT 3.5:** An email created by the GPT 3.5 model.
3. **Target Email Translated by DEEPL:** The target email being translated to Chinese and back by the DEEPL service.
4. **Email Title:** The subject line of the target email.
5. **X:** A control variable representing non-relevant text.

The following metrics were used:

* **BLEU (Bilingual Evaluation Understudy):** Measures the correspondence of phrases between the generated and the target texts.
* **BERT (Bidirectional Encoder Representations from Transformers):** Evaluates the semantic similarity of text representations.
* **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Focuses on the overlap of n-grams between the compared texts.

The evaluation results are summarized in the table below:

| **Data Type** | **BLEU** | **BERT** | **ROUGE** |
| --- | --- | --- | --- |
| Target | 1.0 | 1.0 | 1.0 |
| GPT3.5 | 0.052389 | 0.843668 | 0.316623 |
| Translated Twice | 0.564568 | 0.96876 | 0.819048 |
| Title | 0.000041 | 0.954038 | 0.140351 |
| X | 0.0 | 0.774057 | 0.0 |

* **BLEU Score Analysis:** BLEU scores followed our expectations in terms of ordering the data types by relevance. However, the numerical differences between scores are minimal, lacking a clear linear pattern, which led us to dismiss BLEU as a primary evaluation metric.
* **BERT Score Analysis:** Despite BERT scores indicating a high semantic similarity across most text types, including the non-relevant "X" text (score of 0.774057), this metric did not effectively differentiate between the nuances of the text types involved. Consequently, BERT was ruled out as a useful metric for this evaluation.
* **ROUGE Score Analysis:** ROUGE scores proved the most effective in capturing the qualitative differences among the texts. The scores aligned well with our expectations both in terms of order and numerical differentiation, making ROUGE the chosen metric for evaluating our project.

Based on the comparative analysis of the metrics, the **ROUGE score** was selected as the most suitable evaluation metric for our project. It consistently reflected the expected trends and displayed significant numerical distinctions appropriate for evaluating the performance of the generated and translated emails.

# Spaces

In this project, an application was created using Space and Streamlit to leverage a trained machine-learning model. The application provides an interactive user interface where users can input their product, gender, profession, or hobby, and receive an email generated by the model.

The application workflow is as follows:

Start the Application: Upon starting the application, an input container and a “clear” button are displayed.

User Input: Users input their product, gender, profession, or hobby in the input container and click the “submit” button.

Processing: The application takes a few minutes to process the input.

Output: An output container is displayed on the right, showing both the user input and the email generated by the model.

Clear Output: Users can click the “clear” button to clear the output container and retry the process.

Several valuable lessons were learned during this project:

Interactive User Interface: The use of Streamlit to create an interactive user interface for the model allowed users to interact with the model in real-time. This provided a user-friendly way to input data and view the results.

Model Training and Application: Experience was gained in training a machine learning model and applying it in a real-world application. This is a valuable skill in the field of data science and machine learning.

Iterative Development: The ability to clear the output and retry suggests an iterative approach to development, allowing for continuous improvement and adjustment based on user feedback or new data.

Despite the successes of the project, there were also some limitations:

Processing Time: The model takes over three minutes to run in the application. This could potentially lead to a less than optimal user experience, especially if users are expecting quick results.

Data Formatting: Ensuring the output is well-formatted and easy to understand is crucial for the end-user experience. The generated email does not have a perfect format, indicating that there might be room for improvement in the data preprocessing or postprocessing stages.

Overall, this project was a valuable learning experience in creating an interactive application to leverage a machine-learning model. Despite some limitations, the project demonstrated the potential of using such models in real-world applications. Future work will focus on improving the processing time and data formatting to enhance the user experience.

# Findings

This section delves into the results derived from various testing phases of our final project, focusing on the training period, manual testing, and the overall conclusions drawn from the evaluation data.

## 4.1. Training Data Analysis

Throughout the training period, several metrics were monitored to evaluate the performance improvements of our model across multiple epochs. The primary focus was on the evaluation loss and ROUGE scores as indicators of the model's ability to generate text closely matching the target email.

The final perplexity is 14.17, and there is a significant reduction in evaluation loss from 4.3 to 2.65, suggesting the model was learning effectively. However, the ROUGE scores remained low, around 0.04, highlighting a weak correlation with the target content. The lack of improvement in ROUGE scores points to potential issues either with the model's suitability for this specific text generation task or with the dataset's alignment with our objectives.

## 4.2. Manual Testing Results

To further assess the model's practical performance, we manually tested it by generating 12 emails and rating them on three aspects: relevance, format accuracy, and utility.

Manual Testing Scores:

• Relevance to features: 0.25

• Accuracy as an email: 0.42

• Usefulness: 0.25

These scores reveal that the model occasionally produced relevant and potentially useful content but was generally inconsistent and unreliable for practical deployment.

## 4.3. Conclusions from the Findings

• Practical Value: The current model lacks practical value for real-world applications, primarily due to its inability to consistently generate high-quality emails.

• Decreasing Effectiveness: Despite a reduction in evaluation loss, the declining ROUGE scores during training suggest diminishing effectiveness, indicating that increasing dataset size alone may not necessarily lead to better performance.

• Need for Alternatives: Considering the current model's limitations, there is a strong case for exploring alternative models or making substantial adjustments to the training process or data handling.

Based on these findings, further actions could include reevaluating the dataset, considering a different model architecture, or making critical adjustments to the training regimen to enhance the quality and reliability of the generated text.

# Weaknesses and Shortcomings

**Model Shortcoming**: During the evaluation phase, several issues were identified with the output of the model. Firstly, the content of the generated emails often did not align with the intended output, suggesting a discrepancy between user expectations and the model's capabilities. Additionally, there is a significant challenge in preventing the generation of spam or phishing-like emails. Our analysis indicates that further training does not significantly mitigate these issues, which we attribute to two primary factors:

1. Model Limitations: The inherent constraints of the model may not fully capture the nuances required for our specific application. Its architecture, while efficient for general purposes, might lack the sophistication needed to discern and replicate the subtleties of personalized communication.
2. Dataset Limitations: The dataset we chose was too small to effectively train the model to achieve the high standards of personalization and accuracy we aimed for. A larger and more diverse dataset could potentially improve the model's ability to generate emails that are both relevant and free of spam or phishing elements.

**Processing Time Limitation**: Another significant concern is the processing time required by the model. Currently, it takes over three minutes to generate an email when integrated into the application. This delay is suboptimal and could detract from the user experience, particularly for those expecting immediate responses. This processing lag needs addressing to enhance usability and maintain engagement.

**Email Format Biases**: The format of the generated emails is another area requiring improvement. Ensuring that the output is well-structured and easy to understand is essential for user satisfaction. Currently, the emails produced do not consistently meet these standards, which could confuse users or diminish the perceived quality of the application. This formatting issue underscores the need for refined data preprocessing or post-processing steps that could better prepare the model's outputs for practical use.