Generative AI for Outreach Messages

Introduction

This project aimed to develop an automated data pipeline to generate personalized messages for specific target users. The pipeline integrates several key components, including web scrapers for job postings, retrieval-augmented generation (RAG) using large language models (LLMs) such as ChatGPT-3.5 and Llama3, and evaluation of the generated messages using BERT embeddings and LLMs. This report details the steps taken, the rationale behind our choices, and the results obtained.

Data Product

The data product is a data pipeline to enhance AI-generated message personalization. Our approach integrates data from user profiles, product information, and job postings scraped from job boards. We employed a three-component data pipeline (Fig. 1):

1. A web scraper for collecting job posting data from external sources.
2. An LLM-powered (GPT-3.5/Llama3) model built on the LangChain framework for message generation [1].
3. An evaluation model using BERT embeddings and LLM (GPT-3.5).

Fig 1. Data product pipeline

1. **Document Loading**:
   - Input: "Job postings"
   - The process begins with loading job postings, which are represented as documents in the system.
2. **Splitting**:
   - The documents are split into smaller, manageable sections or chunks. This step ensures that the data can be efficiently processed in subsequent steps.
3. **Storage**:
   - The split documents are stored in a **Vectorstore**. This storage system likely indexes the documents, allowing for quick and efficient retrieval based on queries.
4. **Retrieval**:
   - A query is made to the system, asking for information such as "Sales positions that can be enhanced by 'sender product'."
   - Relevant splits (document sections) are retrieved from the Vectorstore based on the query.
5. **Output**:
   - A language model (such as **ChatGPT-3.5** or **Llama3**) processes the relevant splits to generate personalized messages or responses.
   - These responses are tailored to the query and are output by the system.
6. **Fact Check**:
   - Another instance of the language model (**ChatGPT-3.5**) is used to fact-check the personalized messages.
   - This step ensures that the information provided is accurate and reliable.
7. **Evaluation**:
   - The responses are further evaluated, possibly using another model like **BERT** along with the language model.
   - This step is crucial to assess the quality and relevance of the output, making any necessary adjustments before final delivery.

These steps describe a pipeline where job postings are processed, queried, and evaluated to produce personalized and accurate information.

Development Process

Data collection

To collect data, we developed and utilized a web scraper focused on publicly available information from companies listed on job board services. After running the scrapers, you will have 5 new features added to the dataframe:

Feature Name	Data Type	Explanation
career_page_url	string	job board of receiver companies' official websites
company_url	string	page of receiver companies
position_count	int	number of job openings as of last scraping
position_description	dictionary	job title and job description of the job openings
last_time_scraped_job	timestamp	last time we ran the scrapers

Table 1. New features added to the database

Hybrid model for Retrieval-Augmented Generate (RAG):

Initially, we used RAG retrieval on all features, including sender product information, receiver's first name, company name, job title, job description, company description, professional summary, headline, post, location, and the company's hiring page [2]. However, we realized that the sheer number of features overwhelmed the model, leading to inaccuracies in placing the right names and details. To solve this, we streamlined our approach by fixing certain features like the receiver's first name and company name in our prompt and focusing retrieval solely on the hiring page. Since some companies have over 200 job posts, we aimed for RAG to find the most relevant job post related to our product. This hybrid method significantly improved our RAG retrieval efficiency.

Fig 2. Retrieval function on company job posting information.

Retrieval via Maximal Marginal Relevance (MMR)

In the retrieval part of our RAG system, we initially used semantic similarity to select examples. However, this approach often returned repetitive or very similar documents. To address this, we ultimately chose Max Marginal Relevance (MMR) as our search type for retrieval (Fig. 2) [3].

The similarity method selects examples based on their semantic similarity to the input, using cosine similarity to find the most relevant examples. While effective in identifying closely related documents, this method can lead to redundancy. In contrast, the MMR method strikes a balance between relevance and diversity. It selects examples that are not only similar to the input but also distinct from each other.

In the project, the similarity method provided the most relevant five job postings, but they were often repetitive and missed out on other crucial aspects. However, the MMR method gave us a completely different set of job postings. Therefore, we switched to MMR because it balances relevance and diversity. MMR ensures that the selected documents are not only relevant but also varied, providing a richer, more comprehensive information base for generating responses.

Generation Model

In our search for the best LLMs for text generation, we compared OpenAI's ChatGPT 3.5, Meta's LLama-3, and Hugging Face's MPT-7B, GPT-neo-2.7B, and MPT-1.3B. ChatGPT 3.5 and LLama-3 excelled, offering high-quality, nuanced text and easy integration with LangChain [4-8]. ChatGPT 3.5 posed privacy concerns and was costly to scale, while LLama-3, despite being computationally demanding and less documented, could run locally, solving privacy issues. MPT-7B and GPT-neo-2.7B were impractical due to their high GPU requirements. Ultimately, LLama-3 and ChatGPT 3.5 were favored. Users can choose LLama-3 for local processing of sensitive data, and ChatGPT 3.5 for faster runtime and lower computational costs.

Prompt Engineering

To define an effective prompt engineering strategy for LLMs, we started by manually retrieving important features and inputting them into LLMs to observe responses and iteratively refine the prompts. This process helped us understand how the models processed information and allowed us to enhance the prompts accordingly.

We integrated our refined prompts into the LangChain framework, summarizing enriched CSV data, identifying columns for the receiver and sender, and defining the message's purpose. For example, a prompt specified: "Create a personalized message for the receiver from the sender based on the following details, for a specified purpose, within a set word count." Using the GPT-3.5 API or LLama-3, we tested various inputs to ensure the generated text met our expectations. Structured prompts with sections for parameters, goals, step-by-step instructions, and desired outputs produced the best results (fig. 3).

Fig 3. Current prompt structure.

In order to meet the word limit of 300 characters, we have also considered adding the word limit in the prompt above. However, the result is compromised because the prompt has too many requirements, and sometimes the model misses this information. The limited word count also comprises the quality of the message making it very generic. To solve, we use a second LLM model with the same structure but only change the prompt to make the message more concise (Figure 4). Combining these two RAGs together, we have the current generate and compress message function.

Fig 4. Prompt for compressing the message.

Fact-Checking Mechanism

To enhance the accuracy and authenticity of personalized messages, we implemented a comprehensive fact-checking mechanism tailored to enhance the accuracy and authenticity of personalized messages generated for business outreach. Utilizing the ChatGPT-3.5 model combined with RAG, our methodology mirrors the workflow of our generation model but incorporates a tailored prompt. This prompt specifically requests verification for each sentence in the generated messages, ensuring each claim is cross-referenced against a structured dataset containing company-specific information such as job titles, company descriptions, relevant professional summaries, and company hiring posts (Fig. 5).

Fig 5. Fact check prompt.

We also experimented with using Llama-3 for this task but found it to be slower and less effective compared to ChatGPT-3.5. Llama-3 struggled to understand the task fully, failing to dissect the sentence and identify sources from the features accurately. While ChatGPT-3.5 generally worked well, it could only partially capture the sources. When information was derived from multiple features, it often captured only the most obvious one. Future work could enhance this part by refining the prompt further and potentially fine-tuning the GPT model with specific features to improve source identification and accuracy.

Evaluation model using BERT and LLMs

To evaluate the quality of generated text messages and predict their potential success with recipients, we tried various machine learning methods such as Logistic Regression (LR), Support Vector Machines (SVM), and XGBoost [9-11]. However, without ground truth for the generated messages, these methods proved ineffective.

Instead, we used BERT embeddings and LLMs for our evaluations. BERT embeddings capture the context within their vectors, allowing us to measure message relevance quantitatively by computing cosine similarity between context and message vectors. This helps determine how closely messages align with the context, providing nuanced insights into their relationships. However, this method requires significant computational resources.

LLMs, such as ChatGPT-3.5, enable large-scale automated evaluations of message relevance, alignment, and personalization by scoring messages on a scale of 0 to 10 across these criteria. This automation reduces the need for costly human judgments and provides qualitative insights, though human evaluation may still be necessary for capturing nuanced personalization.

Results

Personalized Message Result

In this section, we will compare example messages generated by both Llama3 and ChatGPT-3.5 models.

Messages generated by both LLama-3 and ChatGPT-3.5 are similar in content and tone, effectively conveying the value proposition of the GenAI solution by focusing on automation and efficiency improvements in the hiring process. This indicates that our model settings and prompts successfully guided the models to highlight key information among all the features provided. The use of emojis and engaging ending phrases in both messages helps attract attention and generate interest. Overall, both messages utilized the key features correctly, communicated the intended benefits effectively, and included elements that made the message exciting and engaging.

The fact-check results demonstrated that the claims made in the generated messages were substantiated by reliable sources. Each statement in the generated message was verified against information from the sender company and receiver's information, enhancing the credibility of the generated messages. This validation ensures that the information presented is accurate and trustworthy.

Evaluation Result

To measure the quality of generated messages, we used BERT embeddings and LLMs [12, 13]. The evaluation results are promising when we compare them with the original messages. The evaluation results for Llama-3 are not shown due to poor performance.

LLMs Evaluation: We utilized GPT-3.5 to evaluate messages generated by GPT-3.5 and Llama-3 against original messages. The metrics we used are relevance to job posts, alignment with company values, connection and personalization. We found scores were higher for GPT-3.5 and Llama-3 messages compared to original messages. In Fig 6, box plots indicated a narrower interquartile range (IQR) for generated messages, suggesting more consistent quality.

Fig 6. Evaluation results of LLM approach (The very narrow IQR in the red box plot is due to the randomness of LLMs. Repeated experiments may address this issue.)

BERT Embedding Evaluation: We employed cosine similarity to assess message alignment with job posting context. The compared categories are original message vs. context, GPT-3.5 message vs. context, Llama-3 message vs. context, GPT-3.5 message vs. original message, and Llama-3 message vs. original message. In Fig. 7, we find that ChatGPT-3.5 and Llama3 messages closely matched job contexts better than original messages, achieving a higher cosine similarity score, indicating strong fidelity and improved relevance. Additionally, ChatGPT-3.5 and Llama3 messages closely resembled original messages, achieving a cosine similarity score above 0.90.

Fig 7. Evaluation results of BERT embedding approach.

Conclusion

The automated data pipeline successfully generated personalized messages for specific target audiences by integrating web scraping, RAG, and prompt engineering with robust evaluation methods. The structured and interactive approach ensured high-quality, relevant, and personalized messages, enhancing the overall effectiveness of communication strategies. Future developers can build on this foundation, exploring improvements in data collection methods and further refining prompt engineering techniques.

References:

[1] LangChain. (n.d.). LangChain documentation and resources. Retrieved from LangChain Website

[2] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[3] Carbonell, J., & Goldstein, J. (1998). The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 335-336.

[4] OpenAI. (2024, April 29). GPT-3.5 Turbo fine-tuning and API updates. Retrieved from OpenAI

[5] Meta AI. (n.d.). LLaMA: Open and Efficient Foundation Language Models. Retrieved from Meta AI

[6] MosaicML. (n.d.). Introducing MPT-7B. Retrieved from MosaicML

[7] Hugging Face. (n.d.). GPT-Neo. Retrieved from Hugging Face

[8] MosaicML. (n.d.). Introducing MPT-1.3B. Retrieved from MosaicML

[9] Hosmer, D. W., & Lemeshow, S. (2000). Logistic Regression. In Applied Logistic Regression (2nd ed.). John Wiley & Sons, Inc.

[10] Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273-297.

[11] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

[12] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT.

[13] Wang, Y., Jiang, J., Zhang, M., Li, C., Liang, Y., Mei, Q., & Bendersky, M. (2023). Automated evaluation of personalized text generation using large language models. arXiv preprint.