TAKE HOME ASSIGNMENT FOR MACHINE LEARNING ENGINEER
Qode.world

Open-Source LLMs In Personalized Applicant Emails

Candidate's name: Phuc Ho - tel :+84919746992

I. Introduction

A. Background on the problem and objectives

In today's competitive job market, attracting qualified candidates requires personalized outreach that highlights their fit for the role and company. Generic job postings often fail to engage top talent effectively. The objective of this assignment is to develop a system to generate personalized email messages to candidates based on their profile and the job description. These messages should include a company pitch, opportunity narrative explaining why the role is a good fit, highlighting the candidate's relevant skills/experience, and a clear call-to-action to apply or interview. Achieving this level of personalization at scale can significantly increase the pool of applicants and hiring potential.

B. Overview of approach

The code implements a command-line interface that prompts the user to choose the source of the job description (URL or PDF files). It then extracts the text content from the provided sources. Next, the user is prompted to input the paths of the candidate resume files (in PDF format).

The extracted job description and resume texts are then used to generate personalized email messages for each combination of job description and resume. The code utilizes four different language models to generate these emails:

Gemini model: A proprietary model (provided freely by google) accessed through an API.

LLaMa 3 model: An open-source small language model.

LLaMa 2 model: Another open-source small language model, a different version or configuration of LLaMa.

Phi 3 model: Another open-source small language model.

For each job description and resume combination, the code creates a prompt using a helper function prompt_for_email. This prompt is then passed to each of the four language models to generate a personalized email message. The generated emails are saved as separate text files with a naming convention that includes the job description index, resume index, and the model used.

II. Data

A. Job Description

1. Job description provided by Qode

The job description is for the role of Senior Software Engineer at Renn Labs, an AI and tech innovation company. It is in text format and covers:

- Overview of the company and positioning the role's importance
- Key responsibilities like software design, development, code reviews, project planning, DevOps collaboration, performance optimization, mentoring
- Required skills including Node.js, AWS, React.js, databases, Git, problem-solving, communication, project management
- Company background in AI, finance, e-commerce solutions
- Benefits like access to experienced team, AI tech exposure, flexible schedules, competitive pay

2. **Online Job description :**

The job description text was initially available online as an HTML webpage. To extract the core text content from this webpage, the code employed web scraping techniques using the Selenium library. Selenium automates web browser interactions, allowing it to fetch the full rendered HTML of the target page.

However, the raw HTML contained extraneous elements like navigation menus, advertisements, and other website layout noise in addition to the actual job description text. To isolate just the relevant description, the Selenium script simulated selecting all the page text content by sending control+A keyboard commands to the browser.

This selected chunk still contained some residual noise like formatting tags and javascript snippets mixed with the desired natural language description. To further clean and distill only the job description narrative, the selected text was passed as a prompt to the Gemini AI language model.

Gemini leveraged its natural language understanding capabilities to parse through the noisy input, identifying and extracting just the coherent sections pertaining to the job overview, responsibilities, qualifications and company details. It was able to intelligently filter out the irrelevant code debris while stitching together the

fragmented yet pertinent description passages into a single clean textual output.

This refined job description text from Gemini's output was then used as the primary input data for subsequent processing and modeling steps in generating the personalized email content. Gemini's role was crucial in transforming the initially noisy raw web data into a structured, contextualized and usable job description format for downstream tasks.

B. Candidate Resumes

The candidate dataset comprises five individual PDF files containing the resumes of potential job applicants. These PDF resumes preserve the original formatting, layout, and any visual elements employed by the candidates. Typical sections found in resumes, such as personal information, professional summaries, work experience details (company names, job titles, employment dates, responsibilities, and achievements), educational qualifications (degrees, certifications, institutions, and graduation years), skills, and additional sections like projects, publications, or awards, are expected to be present. While the PDF format maintains the visual integrity of the resumes, further processing steps, such as text extraction and parsing, may be required to convert the content into a machine-readable format suitable for analysis and processing by the language models. Nevertheless, this dataset of five candidate resumes in PDF format provides a comprehensive representation of the applicants' backgrounds, experiences, and qualifications for the specific job role under consideration.

III. Models

A. Gemini

Gemini Pro 1.0 Specs:

- Access via online API key
- Input Token Limit: 30,720 tokens
- Output Token Limit: 2,048 tokens
- Model Safety: Automatically applied safety settings; adjustable by developers
- Rate Limit: 60 requests per minute

B. lilama 2

Llama 2 Specs (7B Model):

- Hosted locally using Ollama
- Token Limit: Trained on 2 trillion tokens, supports a context length of 4096 tokens.
- Parameters: 7 billion
- Quantization: 4-bit
- Memory Requirement: At least 8GB of RAM

C. llama3(8B model):

Llama 3 Specs (8B Model):

- Hosted locally using Ollama
- Initial release includes models with 8B and 70B parameters.
- Trained on over 15 trillion tokens of pretraining data.
- 8B model integrates Group Query Attention (GQA) for enhanced inference efficiency.
- Sequences are trained with a length of 8,192 tokens.
- Evaluated across 1,800 human prompts covering 12 use cases.
- Achieves new benchmarks, establishing state-of-the-art performance in evaluations like HumanEval.

D. phi3

Phi-3-small Specifications:

- Hosted locally using Ollama
- Architecture: Decoder-only transformer
- Model Size: 7 billion parameters
- Context Length: 8K
- Vocab Size: 32,064
- Trained on: 3.3 trillion tokens

E. mixtral:8x7b

mixtral:8x7b specification:

- Hosted locally using Ollama
- Model Type: Sparse Mixture-of-Experts (SMoE)
- Total Parameters: 46.7 billion
- Parameters per Token: 12.9 billion
- Context Handling: Gracefully handles a context of 32k tokens
- Supported Languages: English, French, Italian, German, and Spanish
- Performance in Code Generation: Strong
- Finetuning Capability: Can be finetuned into an instruction-following model achieving a score of 8.3 on MT-Bench

IV. Model Comparison & Selection

The device on which experiments were carried out has these specifications:

Hardware Model	Acer Nitro AN515-57
Memory	8,0 GiB
Processor	11th Gen Intel® Core™ i5-11400H @ 2.70GHz × 12
Graphics	NVIDIA Corporation GA107M [GeForce RTX 3050 Ti Mobile] / Mesa Int…
Disk Capacity	512,1 GB

A. Summary of evaluation results across models

1. Evaluation based on specs and benchmarking practice:

Looking at the benchmarking results for common evaluation techniques, it can be observed that larger models tend to perform well across various methods, such as General Ability (Chatbot Arena), Reasoning & Knowledge (MMLU), Reasoning & Knowledge (MT Bench), and Coding (HumanEval). However, it is noteworthy that Llama 3 (8B), a relatively smaller model, exhibited impressive efficiency in reasoning and knowledge tasks, making it potentially useful for extracting information from job descriptions and CVs.

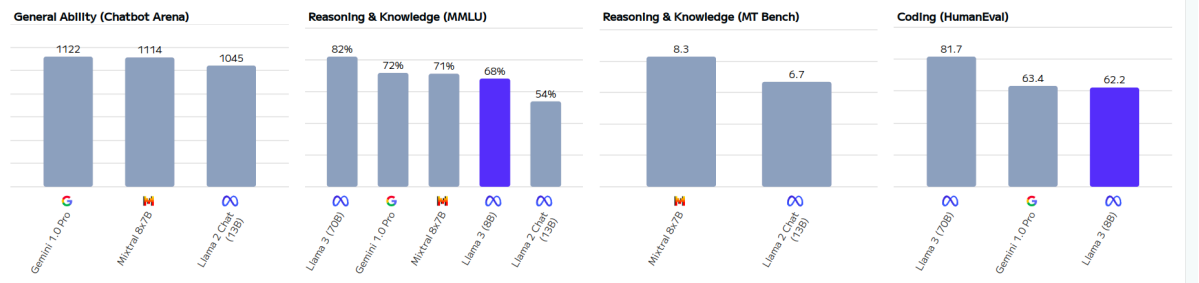

figure 1. Common evaluation techniques result

Regarding the quality of output tokens, Llama 3 (8B) and Mixtral 8x7B showed considerably lower results. However, this can be attributed to their compact size.

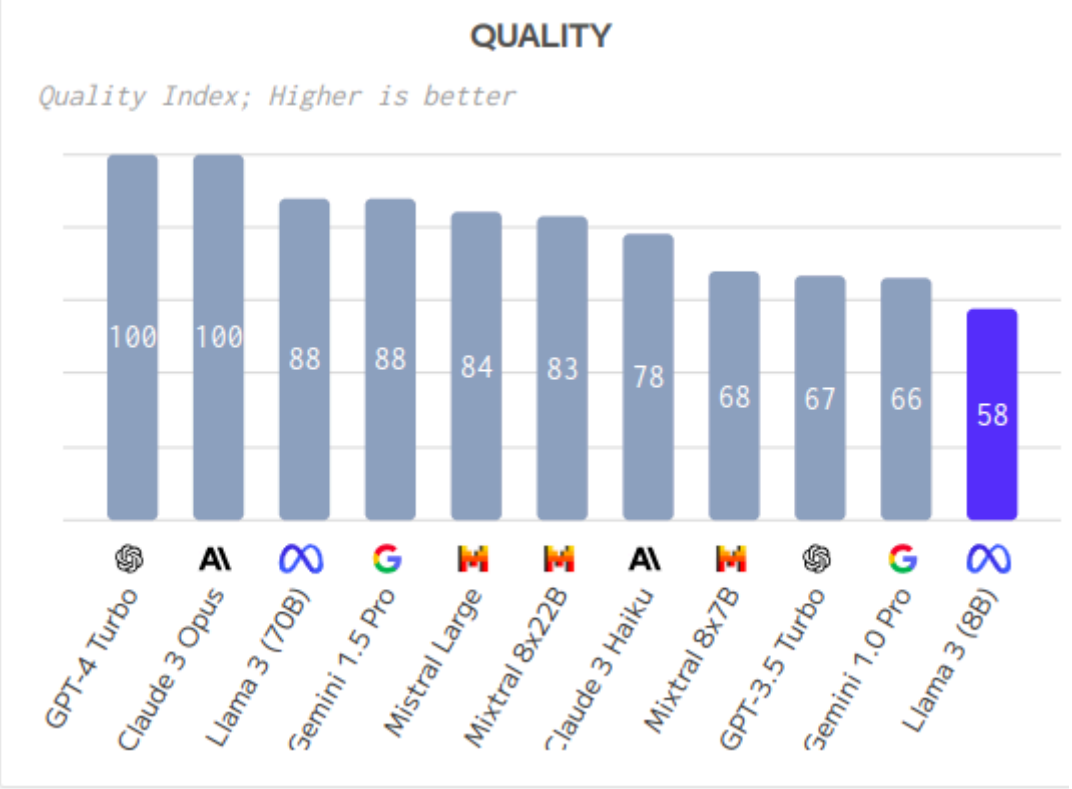

figure 2. LLMs' quality index

In terms of speed, Llama 3 (8B) proved to be the fastest, followed by Mixtral 8x7B and Gemini 1.0 Pro.

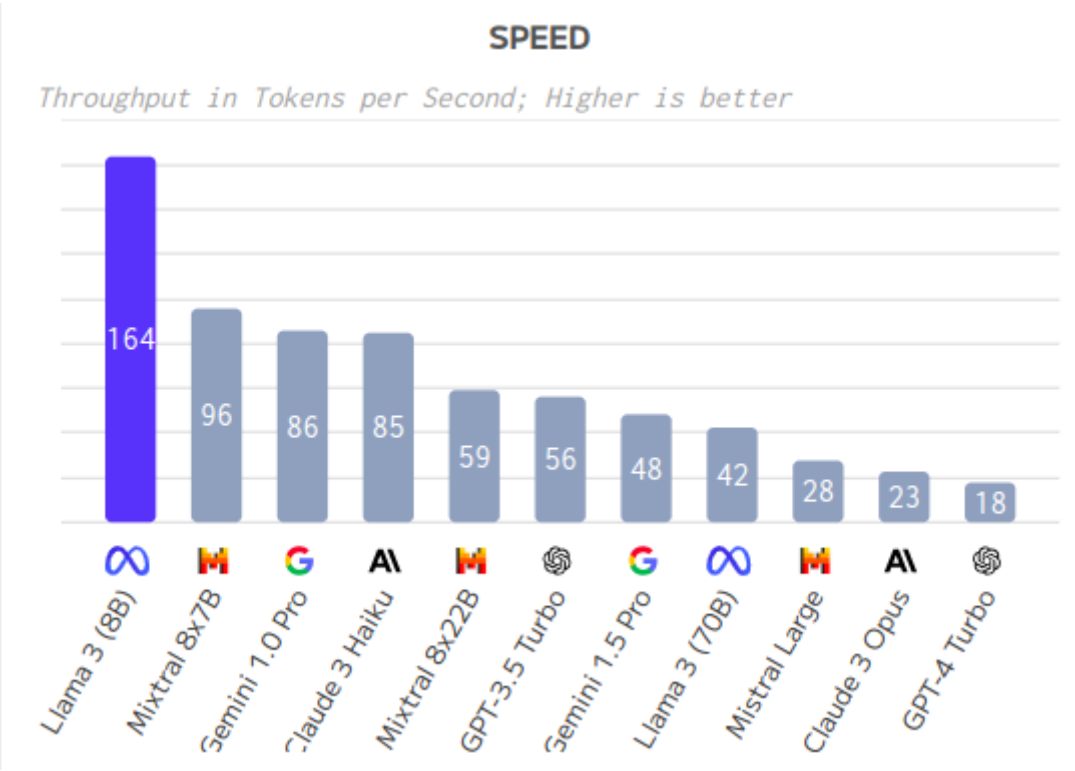

figure 3. LLMs' speed index

2. **Evaluation metrics for personalized email generation:**

   The criteria for evaluation includes:

   - formatting:
     - Address the candidate by name.
     - Company pitch 2-3 lines (based on information from job description and public data of a company on the internet).
     - Opportunity narrative 2-3 lines, briefly explain why this specific opportunity would be a good fit for the candidate's interests and career goals (based on information from the profile and job description).
     - Personalized candidate pitch: highlight the candidate's tenure, skills and experience that are relevant to the job description.
     - Call to action: Include a clear call to action, encouraging the candidate to apply for the position or schedule an interview.
   - The privacy of the model.
   - The speed of generation.
   - The hardware requirements and potential for fine tuning.

3. **The evaluation result**

   Fomating: all models succeeded in addressing candidates by name. For company pitch, opportunity narrative, smaller models with small context windows suffer from the vanishing context of long resumes. They often mistake the companies that are present in long CVs as the hiring company. Whereas Gemini and Mixtral with the context length 1 millions tokens and 32k tokens, gracefully pass this test.

   The privacy of the models: all models aside from Gemini pro 1.0 are hosted locally. Therefore, they ensure the privacy of candidates.

   The speed of Generation: Using API key is the sole reason that makes Gemini perform better than other models, speed wise. However, with better hardware, Llama3 (70b) which is a larger version of Llama3 7b, can outperform it. Mixtal 8x7B also performed with acceptable speed.

The hardware requirements and fine tuning: Gemini does not support fine tuning. other model can be finetuned via ollama. Mixtal requires extensive hardware and during this experiment was pushed onto a kaggle notebook using GPU T100 to be able to function.However, the prospect of being able to finetune mixtal to become a instruction-following system is perfectly aligned with the project's aim

B. Recommendation for most suitable model

Based on the evaluation results, Mixtral 8x7B or LLama3 7B would be a suitable model for generating personalized email messages for job candidates. Here's a breakdown of the reasoning and trade-offs for each model

1. Reasoning & trade-offs

a) Mixtral 8x7B

- Larger model size (46.7B parameters) compared to Llama3 7B, potentially leading to better performance.
- Performed well in formatting tasks, handling long resumes and maintaining context (context length of 32k tokens).
- Demonstrated acceptable speed performance.
- Offers the potential for fine-tuning to become an instruction-following system, aligning with the project's aim.
- Requires extensive hardware resources (GPU T100 used in the experiment).

Trade-off: While Mixtral 8x7B has a larger model size and performs well in the evaluated tasks, its hardware requirements are significant, which may pose a challenge in terms of deployment and scalability.

b) LLama3 7B

- Smaller model compared to Mixtral
- Performed well with medium to short resume and has better generation speed on limited hardware.
- Can be fine tuned.

Trade-off: The compact size of Llama3 7B proves to be an advantage. However the small context size is detrimental to the generation of personalized email.

C. Potential areas for improvement

Better hardware or better consolidation of a resume to make up for the lost of context for smaller models.

V. Implementation Details

A. Code structure & libraries used

The code is structured as a Python script that implements the following main components:

- Command-line Interface: The script prompts the user to choose the source of the job description (URL or PDF files) and input the resume file paths.
- Text Extraction: The extract_text_from_page function uses the Selenium library to extract text content from a given URL, simulating browser interactions. The extract_text_from_pdf function (not shown) is likely responsible for extracting text from PDF files.
- Prompt Generation: The prompt_for_email function creates a prompt for the language models by combining the job description and resume text with a predefined template.
- Language Model Integration: The script integrates four different language models:
- Gemini: Accessed via the google.generativeai library, using a provided API key.
- LLaMa 3, LLaMa 2, and Phi 3: Open-source models hosted locally using the Ollama library.
- Email Generation: For each job description and resume combination, the script generates personalized email content using the integrated language models and saves the generated emails as text files.

The main libraries used in the code are:

- Selenium: A web automation tool used for web scraping and extracting text from web pages.
- google.generativeai: A library provided by Google to access their Generative AI models, including Gemini, through an API.
- Ollama: A library for hosting and using open-source Large Language Models (LLMs) like LLaMa and Phi locally.
- langchain_community.llms: A library used to invoke the Ollama models and generate responses.

B. Maintainability & scalability considerations

- Error Handling: Implementing more robust exception handling and logging mechanisms could aid in debugging and troubleshooting.
- Configuration Management: Moving configurations like API keys and safety settings to separate files or using environment variables could improve portability and ease of management.
- Parallelization: As the number of job descriptions and resumes increases, parallelizing the email generation process using multiprocessing or multithreading could improve performance and scalability.

VI. Challenges & Limitations

Throughout this task, I grappled with several obstacles that challenged the implementation and execution of the proposed system. Firstly, I encountered difficulties in directly extracting job descriptions from HTML source code, necessitating the use of Selenium to retrieve the content from websites. This added an extra layer of complexity and potential points of failure in the data acquisition process. Moreover, the limited hardware resources at my disposal hindered the efficient execution of larger language models like Mixtral 8x7B, prompting me to leverage online notebooks to access the required computational power. Another hurdle arose with prompt engineering, as failing to specify the desired output format resulted in some models generating email content in fragmented or bullet-pointed structures rather than cohesive paragraphs. Time constraints further exacerbated these challenges, preventing me from fine-tuning the models before the submission deadline, which could have potentially improved their performance and output quality. Despite these obstacles, the task provided valuable insights into the practical considerations and trade-offs involved in developing a system for generating personalized email content at scale using language models.

VII. Conclusion

This assignment presented an interesting business case for me to consider, pushing meaningful changes using AI technology. The workflow I proposed, which involves generating personalized email messages for job candidates based on their resumes and the job descriptions, has the potential to minimize workloads for HR professionals if implemented correctly. By leveraging the capabilities of large language models, such as Gemini, LLaMa, and Phi, the system can craft tailored outreach messages that highlight the candidate's fit for the role and company, potentially increasing engagement and attracting top talent more effectively. However, the implementation process also revealed various challenges and limitations that need to be addressed to ensure the system's effectiveness, scalability, and real-world applicability. These challenges ranged from data acquisition complexities, hardware constraints, and prompt engineering difficulties to issues related to model interpretability, continuous monitoring, and system maintenance.

Overcoming these hurdles through improved data handling techniques, optimized resource utilization, enhanced prompt design, and robust system architecture would be crucial for realizing the full potential of this AI-driven solution and driving meaningful changes in the HR and recruitment domains.