Hiring Assignment

AI Research Engineer Assessment: Research Innovation Lab

We have designed this assessment to evaluate your ability to approach complex problems in artificial intelligence research and implementation. The assessment consists of **two distinct challenges**, each targeting a specific area of expertise: **text normalization** and **cover/similar song recognition using sound data**.

The assessment consists of **two parts**, each targeting different aspects of AI research and problem-solving. The candidate has the flexibility to choose one of the following tasks to perform a **full implementation**, while the other will require a **detailed description of the approach** that could be followed to address the problem.

Candidate's Choice

You must select **one task** to complete as a **full implementation**, where your solution will be evaluated on functionality, accuracy, and scalability. For the other task, you are expected to provide a **comprehensive description** of your approach, explaining your reasoning and methodology in depth.

What We're Evaluating:

- **For the full implementation**:
  - Code quality and efficiency.
  - Robustness of the solution.
  - Innovation and scalability.
  - Clarity and completeness of documentation.
- **For the description**:
  - Depth of understanding of the problem.
  - Creativity in problem-solving.
  - Feasibility of the proposed approach.

Submission Guidelines

- **For the full implementation**: Submit a working prototype (Python script or notebook), documentation, and any required data or dependencies.
- **For the description**: Submit a well-structured write-up (PDF) that includes a clear breakdown of your approach, techniques, and considerations.

We encourage you to showcase your strengths in tackling one task while demonstrating your strategic thinking and technical knowledge for the other.

A. Hiring Assessment: Text Normalization

Objective:

The goal of this task is to evaluate the candidate's ability to design and implement normalization techniques for textual data. The candidate will work with a provided dataset containing raw information about composition writers and the normalized one. In the music industry, the data may come with redundant or missing information that's why a solution should be proposed to identify and normalize the given text without losing information. The solution should be generalizable to unseen data while demonstrating proficiency in handling complex and potentially redundant text information. The dataset can be found here.

Goal

Having the raw composition writer's information as it has been given, propose a solution that generally can normalize the given text, removing redundant information, and keeping only the writer names in the output.

Example1:

RAW TEXT: **<Unknown>/Wright, Justyce Kaseem**

Normalized Text: **Justyce Kaseem Wright**

Example 2:

RAW TEXT: **Pixouu/Abdou Gambetta/Copyright Control**

Normalized Text: **Pixouu/Abdou Gambetta**

Example 3:

RAW TEXT: **Mike Hoyer/JERRY CHESNUT/SONY/ATV MUSIC PUBLISHING (UK) LIMITED**

Normalized Text: **JERRY CHESNUT/Mike Hoyer**

Dataset Overview:

Dataset Description: The dataset provided is a CSV file containing raw compositions from writers' texts and the normalized version. Below is an overview of the dataset columns:

- `raw_comp_writers_text` : Original text containing composer and writer information.
- `CLEAN_TEXT` : The clean normalized text after removing redundant information.

Assessment Instructions:

1. **Understand the Dataset**
   Carefully review the dataset to understand the structure and types of information it contains. Pay attention to the redundancies in the data.
2. **Design a General Solution**
   Develop a solution to normalize the raw text in a generalizable manner. Your approach should:
   - Address redundancies in the dataset.
   - Merge and filter names or information where appropriate.
   - Preserve the structure and relationships within the data.
3. **Proposed Techniques**
   You may utilize a combination of the following:
   - Large Language Models (LLMs) for text understanding and transformation.
   - Rule-based techniques for explicit pattern matching or heuristics.
   - Deep learning or machine learning models for predicting normalized structures.
   - Any other methods you consider suitable for the task.
4. **Evaluation & Discussion**
   - Demonstrate the solution's applicability.
   - Clearly explain the logic or models used.
   - Highlight any assumptions or limitations.
5. **Deliverables**
   - A detailed explanation of your normalization approach.
   - A Python-based implementation (notebook or script) of your normalization solution.
   - Example outputs demonstrating how your solution processes the dataset.
   - Suggestions for improving or scaling the solution.
6. **Submission Guidelines**
   - Upload your completed assessment and share a link to your repository.
     - Github link

- Ensure that all code is well-documented.
- Include a brief report summarizing your approach, findings, and recommendations.

Assessment Notes:

- Candidates are encouraged to use external libraries or tools for NLP (e.g., NLTK, SpaCy, HuggingFace, or PyTorch)
- Explicitly mention and justify the trade-offs of the selected techniques.
- The dataset contains non-Latin characters—your solution should handle these seamlessly.

B. Hiring Assessment: Cover Song Similarity

Objective:

This task focuses on the candidate's ability to develop and apply a similarity metric that quantifies the relationship between original songs and their cover versions. Candidates are asked to design or implement a model that measures how similar two musical tracks are. The candidate will be evaluated on their approach to defining and learning the metric, as well as their ability to interpret the results and assess the effectiveness of their model in capturing the nuanced differences between an original and its cover. This task tests both technical and analytical skills in music data analysis and machine learning.

Goal

Given a dataset consisting of cover song cliques, design a similarity metric that measures the resemblance between two songs. For songs with a cover relation the metric should be small while for songs without a cover relation it should be large.

Dataset Overview:

Dataset Description: The dataset that you will use for this assignment is Da-TACOS (MTG/da-tacos). It contains metadata such as title, artist, etc., as well as pre-extracted features such as chroma and hpcp, which you could use to develop your similarity metric.

Assessment Instructions:

1. **Understand the Dataset**
   Carefully review the dataset to understand the structure and types of information it contains. Pay attention to the redundancies in the data.
2. **Design a General Solution**
   Develop a solution for a similarity metric to minimize distance between cover songs and maximize it otherwise. Your approach should:
   - Address potential issues with the dataset.
   - Address the pros and cons of your choice of features.
   - Address the pros and cons of your choice of methodology.
3. **Proposed Techniques**
   You are free to utilize any of the following:
   - Simple machine learning methodologies such as clustering.
   - Deep learning models for predicting the distance between two potential covers.
   - Any other methods you consider suitable for the task.
4. **Evaluation & Discussion**
   - Demonstrate the solution's applicability.
   - Clearly explain the logic or models used.
   - Highlight any assumptions or limitations.
5. **Deliverables**
   - A detailed explanation of your approach.
   - A Python-based implementation (notebook or script) of your solution.

    - Example outputs demonstrating how your solution performs.
    - Suggestions for improving or scaling the solution.
6. **Submission Guidelines**
    - Upload your completed assessment and share a link to your repository.
        - Github link
    - Ensure that all code is well-documented.
    - Include a brief report summarizing your approach, findings, and recommendations.

Assessment Notes:

- Candidates are encouraged to use external libraries or tools for Machine Learning or Deep Learning (e.g. XGBoost, Scikit-learn, HuggingFace, or PyTorch)
- Explicitly mention and justify the trade-offs of the selected techniques.