Harvard John A. Paulson
School of Engineering
and Applied Sciences

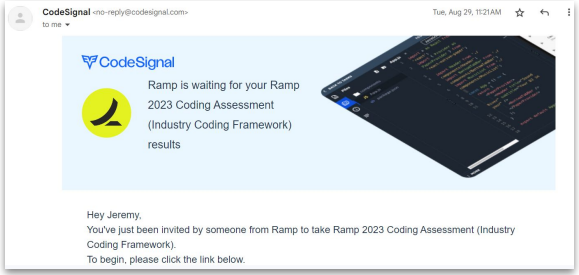
WHERE
SCIENCE
AND
ENGINEERING
CONVERGE

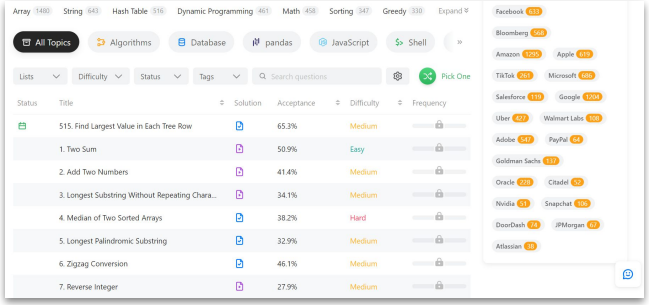
Mockmate: AI Powered Technical Job Interview Preparation

Rudra Barua, Andrew Sima, Jeremy Zhang
AC215, Fall 2023

Motivation

- Inefficient traditional interview prep
  - Requires domain expertise in interviewer
  - Inconsistent feedback
  - Misses nuances of specific roles or industries
  - Easy to over-prepare, under-prepare, or prepare for the wrong skills
- Provide immediate, actionable, and unbiased feedback
- Identify weaknesses in specific areas
- Breakthroughs in LLMs allow for personalization
- We’re graduating seniors that need jobs





Problem Statement

- **Problem definition:**
  - Provide an automated yet personalized interview preparation experience for prospective software engineers

- **Project goals:**
  1. Generate technical questions relevant to software engineering
  2. Evaluate candidate responses in real-time
     - Offer constructive insights on coding efficiency and response quality
  3. Provide insights into candidate's weaknesses and strengths

Data and Preprocessing

- **Source:** Original questions from Leetcode, scraped into a Kaggle dataset here.
  - ~2360 problem statements and sample solutions (in plaintext and a handful of programming languages)
- **Preprocessing:**
  - Separated problem components with Python solutions
  - Created custom PyTorch dataset that guarantees fixed-length token chunks
  - Separated data into training and validation sets to train the model


```
▼ 0:
    id:                "1"
    slug:              "two-sum"
    difficulty:        "Easy"
  ▶ cpp_sol:           "```cpp\n#include <vector…   return {};\n}\n```\n"
  ▶ java_sol:          '```java\nimport java.uti…m solution");\n}\n```\n'
  ▼ python_sol:        "```python\ndef twoSum(nums, target):\n    map = {}\n
                       = i\n    return []\n```\n"
  ▶ javascript_sol:    "```javascript\nfunction …   return [];\n}\n```\n"
  ▶ explanation:       "The algorithm leverages …lexity of O(n) as well."
  ▼ question:          "Given an array of integers `nums` and an integer `target
                       may not use the _same_ element twice.\n\nYou can return t
    ▼ examples:
      ▶ 0:             "**Example 1:**\n\n**Inpu…, we return \\[0, 1\\]."
      ▶ 1:             "**Example 2:**\n\n**Inpu…\n**Output:** \\[1,2\\]"
      ▶ 2:             "**Example 3:**\n\n**Inpu…\n**Output:** \\[0,1\\]"
  ▶ constraints:       "*   `2 <= nums.length <=… valid answer exists.**"
  ▶ followup:          "Can you come up with an …O(n2)` time complexity?"
▶ 1:                   {…}
▶ 2:                   {…}
▼ 3:
    id:                "4"
    slug:              "median-of-two-sorted-arrays"
    difficulty:        "Hard"
  ▶ cpp_sol:           "```cpp\ndouble findMedia…    return 0;\n}\n```\n"
  ▶ java_sol:          "```java\npublic double f…    return 0;\n}\n```\n"
  ▶ python_sol:        "```python\ndef findMedia…  \n    return 0\n```\n"
  ▶ javascript_sol:    "```javascript\nfunction …  return 0;\n}\n```\n\n"
  ▶ explanation:       "1. Choose the smaller ar…ged array, even or odd."
  ▶ question:          "Given two sorted arrays …ould be `O(log (m+n))`."
  ▶ examples:          […]
  ▶ constraints:       "*   `nums1.length == m`\…s1[i], nums2[i] <= 106`"
    followup:          null
```

Model Configuration & Training

- **Base Model:** Initialized pre-trained LLMa model with Meta's `open_llama_3b_v2`

- **Low-Rank Adaption (LoRA):** Adapted the model using LoRa, a method designed to fine-tune models more effectively
  - Factorizes new parameters and reuses pre-trained ones

- **Optimized Loading:** Loaded in models in quantized format
  - Quantized model weights to reduce memory consumption

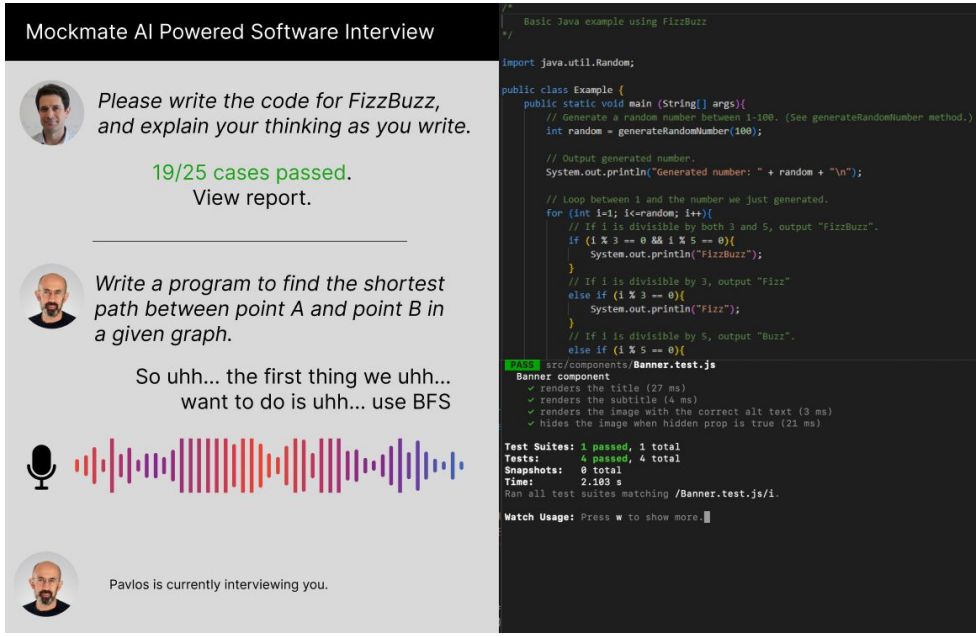
- **Regularization:** Set maximum token generation limit per query
  - Prevents overly verbose outputs and overfitting on certain query patterns

Model Querying

- Created CLI for continuous input queries for the model
- Implemented tokenizer to format user text
- Converted tokenized inputs to PyTorch tensors to CUDA
  - GPU acceleration in response generation
  - Produce tokenized response tensors

Future Work

- This week: finalize, optimize, and refine AI algorithms
- M5 (11/14): Create APIs, fullstack and frontend development
- M6 (12/12): Scaling and production deployment



Harvard John A. Paulson
School of Engineering
and Applied Sciences

Thank you!