IE University

School of Sciences & Technology

Bachelor of Computer Science and Artificial Intelligence

AI: Chatbots and Recommendation Engines

Code Compass

Navigating Developers to Personalized GitHub exploration and Insights

---

Gabriel de Olaguibel

gdeolaguibel.ieu2021@student.ie.edu

Keti Sulamanidze

ksulamanidze.ieu2021@student.ie.edu

Maud Hovland

mhovland.ieu2021@student.ie.edu

Miranda Drummond

mdrummond.ieu2021@student.ie.edu

Luca Cuneo

lcuneo.ieu2021@student.ie.edu

Felix Gomez-Guillamon

fgomezguilla.ieu2021@student.ie.edu

25.01.2024

Introduction

In the ever-expanding realm of software development, navigating through countless repositories, technologies, and learning resources on platforms like GitHub can be overwhelming. Code Compass aims to simplify this journey, offering a tailored experience for developers and students alike to explore and grow in the GitHub ecosystem.

Motivation

Bridging Gaps in the Digital Learning Landscape

The motivation behind Code Compass is rooted in two primary challenges faced by GitHub users:

1. **Information Overload**: With millions of repositories and a vast array of technologies, finding relevant projects and learning resources on GitHub can be daunting, especially for newcomers.
2. **Personalized Exploration/Learning Needs**: Each developer's learning path is unique, influenced by their current skills, interests, and career aspirations. However, current platforms lack personalized guidance to cater to these individual learning journeys.

Code Compass: A Solution

Addressing these challenges, Code Compass proposes a novel solution: a combination of a recommendation system and a chatbot interface. This dual approach aims to:

- **Curate Personalized Recommendations**: Utilizing user-specific data such as project involvement, coding languages, and collaboration networks, Code Compass will recommend relevant GitHub projects, technologies, and connections.
- **Interactive Learning and Exploration**: The integrated chatbot will serve as an interactive guide, helping users to delve deeper into recommendations, understand technology applications, and navigate GitHub more effectively.

Goals and Objectives

The primary goal of Code Compass is to enhance the GitHub experience for developers and learners by providing personalized guidance and resources. This system aims to simplify the process of finding relevant projects and learning materials, making the path to knowledge and collaboration more accessible and efficient.

Objectives:

1. **Personalized Project Recommendations:**
    a. Functionality: To develop an algorithm that analyzes user profiles and activity to suggest GitHub projects that align with their interests and expertise.
    b. Utility: Enables users to discover projects that are relevant to their skill level and areas of interest, fostering a more engaging and productive GitHub experience.

2. **Skill Development Guidance:**
    a. Functionality: To offer recommendations on programming languages and technologies that users could learn or improve upon, based on their current activities and goals.
    b. Utility: Assists users in identifying and focusing on skill development areas that are most relevant and beneficial for their personal growth and career advancement.
3. **Networking and Community Engagement:**
    a. Functionality: To suggest people to follow and collaborate with, thus enhancing users' professional network within the GitHub community.
    b. Utility: Promotes community engagement and collaboration, opening opportunities for mentorship, knowledge sharing, and collaborative projects.
4. **Interactive Chatbot for Enhanced Exploration:**
    a. Functionality: To integrate a chatbot that assists users in exploring recommendations, answering queries about GitHub repositories, and providing insights on programming languages and technologies.
    b. Utility: Offers an interactive, user-friendly platform for users to deepen their understanding of recommended projects and skills, enhancing the overall learning experience.
5. **Continuous Learning and Adaptation:**
    a. Functionality: To ensure the system continuously learns from user interactions and feedback, adapting its recommendations to changing user preferences and emerging trends in the GitHub ecosystem.
    b. Utility: Keeps the recommendations dynamic and relevant, aligning with the ever-evolving nature of technology and software development.

Project Scope

Scope Inclusions:

1. **User Data Analysis**: The project will utilize GitHub user data, such as repository interactions, programming languages used, and user networks, to generate personalized recommendations.
2. **Recommendation Engine**: Development of algorithms to recommend GitHub projects, technologies to learn, and users to follow, based on individual user data.
3. **Chatbot Integration**: Implementation of a chatbot for interactive queries about recommendations, offering insights and further information.
4. **User Interfac**e: A simple and intuitive interface for users to interact with Code Compass and receive their personalized recommendations.
5. **Feedback Mechanism**: Incorporation of a feedback system to refine and improve recommendation accuracy.

Scope Exclusions:

1. **Non-GitHub Data**: The project will not incorporate data sources outside of GitHub (e.g., LinkedIn, Twitter) for recommendations.
2. **Real-time Data Processing**: While the system will update recommendations regularly, it will not process data in real-time.
3. **Advanced AI Features**: Features like voice recognition or vision functionalities in the chatbot are beyond the scope of this project.

Limitations and Constraints:

1. **Data Privacy**: Adherence to GitHub's data usage policies and user privacy constraints.
2. **API Constraints**: Dependence on the GitHub API's limitations and availability.
3. **Technical Complexity**: Certain sophisticated machine learning models may be beyond the scope due to complexity and resource constraints.

Methodology

Data Preparation

The success of Code Compass hinges on the efficient handling of data, starting from its retrieval to its storage and processing:

1. **Data Retrieval:**
    a. The project will leverage the GitHub API to fetch user-specific data, such as repository interactions, programming languages, and user networks. This data is crucial for powering the recommendation algorithms and chatbot functionalities.
    b. The GitHub API provides a rich source of JSON-type data, ideal for our purposes.
2. **Data Storage**: Considering the nature of the data (JSON format), we propose the following storage methods:
    a. Relational Database (e.g., PostgreSQL): Ideal for structured data, offering robust query capabilities and scalability.
    b. Document-Oriented Database (e.g., MongoDB): Natively supports JSON data, providing flexibility in handling semi-structured data.
    c. Data Lake (e.g., ADLS): Suitable for storing large volumes of unstructured data, with high scalability and integration capabilities for various data analysis tools.
3. **Data Processing**:
    a. **Python** will be the primary programming language used for data processing, due to its extensive support for data analysis and machine learning tasks.
    b. **Pandas**: A powerful Python library, will be utilized for data manipulation and cleaning. It offers efficient tools to transform JSON data into a usable format for our machine learning models.
    c. **Scikit-Learn**: This library will be employed for preprocessing tasks, such as feature extraction and normalization, which are essential for preparing the data for machine learning algorithms.

Machine Learning Algorithms

For the recommendation engine in Code Compass, a variety of machine learning algorithms will be employed to cater to different aspects of personalization. These algorithms will be selected based on their ability to provide meaningful and accurate suggestions to users.

1. **Simple Rule-Based Algorithm:**

   a. **Usage**: As a starting point, a simple rule-based algorithm can be implemented.
   b. **Function**: It can recommend based on straightforward metrics like the most starred or popular repositories, or trending programming languages.
   c. **Level**: This will serve beginner users or provide general suggestions when specific user data is limited.
2. **Content-Based Filtering:**
   a. **Usage**: To provide more personalized recommendations based on user profiles.
   b. **Function**: This algorithm will analyze items (e.g., repositories, languages) a user has interacted with and suggest similar items by comparing content features.
   c. **Level**: Suitable for individualized recommendations, especially when user-specific data is available.
3. **Collaborative Filtering:**
   a. **Usage**: To leverage community preferences and patterns.
   b. **Function**: This method makes recommendations based on similarities between users, suggesting items liked by similar profiles.
   c. **Level**: Ideal for integrating community trends and peer recommendations.
4. **Hybrid Approach:**
   a. **Usage**: Combining both content-based and collaborative filtering.
   b. **Function**: This approach merges the strengths of both methods to provide more accurate and comprehensive recommendations.
   c. **Level**: Advanced level, offering a balanced mix of personalized and community-driven suggestions.
5. **Natural Language Processing (NLP) for Chatbot:**
   a. Usage: For the chatbot component to interpret and respond to user queries.
   b. Function: Implementing NLP models to understand user inputs and provide relevant information, explanations, or further recommendations.
   c. Level: Essential for the interactive element of Code Compass, enhancing user engagement.
6. **Deep Learning Techniques** (if resources allow)**:**
   a. **Usage**: For more sophisticated recommendation systems.
   b. **Function**: Utilizing neural networks to capture complex patterns in user data and preferences.
   c. **Level**: Advanced, potentially offering highly accurate and nuanced recommendations.

Deployment

Considering the academic nature and resource considerations of the Code Compass project, a local deployment strategy will be adopted. This approach allows for effective demonstration and testing while avoiding the complexities and costs associated with cloud-based deployment.

1. **Local Server Setup:**
   a. Utilize a local server environment for hosting the application. This can be set up using software like Apache or Nginx.
   b. Ensure the server is configured to handle the application's requirements, including the necessary Python runtime environment for running the recommendation algorithms and chatbot.

2. **Database Integration:**
    a. The chosen database (whether a relational database like PostgreSQL or a document-oriented database like MongoDB) will be set up locally.
    b. Ensure proper connection and seamless data flow between the database and the application server.
3. **Chatbot Integration:**
    a. The chatbot component will be integrated into the main application. It will run on the local server, allowing users to interact with it through the application interface.
    b. The chatbot's NLP capabilities will be powered by the chosen machine learning libraries and frameworks.
4. **User Interface Accessibility:**
    a. Develop a user-friendly interface accessible through a local web browser.
    b. The interface will allow users to interact with the recommendation system and chatbot, providing a seamless user experience.
5. **Testing and Debugging:**
    a. Rigorous testing will be conducted in the local environment to ensure all components of the application work as intended.
    b. Debugging and troubleshooting will be an ongoing process to refine the application's performance and user experience.

Team Roles and Responsibilities

The successful execution of Code Compass hinges on the distinct yet collaborative roles of our team members. Each member brings a unique set of skills and responsibilities to the project.

1. **Product Manager (PM) - Gabriel de Olaguibel**
    a. Responsibilities:
        i. Serves as the bridge between the technical team and the end-users.
        ii. Understands and interprets customer needs to the team.
        iii. Assists the data team in product development and ensuring alignment with business objectives.
    b. Skills: Strong business acumen, excellent communication abilities, and a foundational understanding of data and AI.
2. **Data Engineer - Gabriel de Olaguibel, Maud Hovland, and Luca Cuneo**
    a. Responsibilities:
        i. Develops and maintains data pipelines.
        ii. Cleans and manages data, ensuring its quality and accessibility for AI applications.
        iii. Provides the foundational data infrastructure for the system.
    b. Skills: Proficiency in SQL, experience with distributed computation, and versatility in multiple programming languages.
3. **Data Scientist - Keti Sulamanidze, Miranda Drummond, Maud Hovland, Luca Cuneo, and Felix Gomez-Guillamon**
    a. Responsibilities:
        i. Develops and experiments with machine learning models.
        ii. Each member focuses on different algorithms and model parameters to diversify the approach.

   b. Skills: Expertise in machine learning and statistics, proficient in Python.
4. **Machine Learning Engineer: Luca Cuneo and Felix Gomez-Guillamon**
   a. Responsibilities:
      1. Deploys machine learning models and manages their versions.
      2. Oversees the integration of models into the production environment.
      3. Sometimes overlaps with the responsibilities of the MLOps Engineer.
   ii. Skills: Knowledge of cloud services, API development, and proficiency in multiple programming languages.
5. **MLOps Engineer: Miranda Drummond and Keti Sulamanidze**
   a. Responsibilities:
      i. Ensures the system's reliability and efficiency through regular testing.
      ii. Maintains the operational stability of the deployed models.
      iii. Sometimes takes on tasks typical of a Machine Learning Engineer.
   b. Skills: Familiarity with cloud technologies, expertise in multiple programming languages.

Development Practices

To ensure the smooth progression and high quality of the Code Compass project, our team will adhere to a set of rigorous development practices. These practices are designed to foster a collaborative, efficient, and iterative development environment.

1. **Iterative/Agile Approach:**
   a. The project will be managed using an agile methodology, emphasizing flexibility, continuous improvement, and regular feedback.
   b. Work will be divided into sprints, with each sprint focusing on specific deliverables and allowing for iterative refinement of the project.
2. **Centralized Code Repository:**
   a. All code and documentation will be hosted on a singular repository within our GitHub group organization.
   b. This centralization ensures consistency and ease of access for all team members.
3. **Branching and Pull Requests:**
   a. Team members are required to commit code to dedicated branches and not directly to the main branch.
   b. Each new feature or fix will be developed on a separate branch and merged via pull requests (PRs).
4. **Code Review and Testing:**
   a. All pull requests must pass automated code tests before being considered for merging.
   b. Every PR requires a review by at least one other team member, ensuring a second set of eyes on each piece of code and maintaining code quality.
5. **Release Management:**
   a. Larger updates and fixes will be bundled into releases.
   b. This approach allows for more stable and controlled updates to the system.
6. **Repository Organization:**
   a. The GitHub repository will be structured according to the model provided by Miguel Fierro's project template repository.

   b. This standardized structure aids in maintaining an organized and easily navigable repository.
7. **Documentation and Comments:**
   a. Comprehensive documentation of code and processes is mandatory.
   b. In-code comments and clear README files will be used to ensure that all aspects of the project are well-understood and easily approachable for all team members.

Timeline and Milestones

January 26th - Project Kickoff
- Task: Finalize project proposal and begin initial planning.
- Milestone: Project proposal submitted and roles assigned.

January 27th - February 5th: Initial Setup and Data Retrieval
- Task: Set up the GitHub repository, begin data retrieval from the GitHub API, and start initial data preprocessing.
- Milestone: Repository set up complete and initial dataset retrieved.

February 6th - February 12th: Exploratory Data Analysis (EDA) and Algorithm Design
- Task: Perform EDA on retrieved data and design basic recommendation algorithms.
- Milestone: EDA report completed and algorithm prototypes designed.

February 13th - February 18th: Development Sprint for Midterm Presentation
- Task: Develop a basic version of the recommendation system and chatbot for the midterm presentation.
- Milestone: Basic functionality of Code Compass ready for demonstration.

February 19th: Midterm Project Presentation
- Task: Present the current progress of Code Compass, including EDA findings and a demo of the initial recommendation system and chatbot.
- Milestone: Successful midterm presentation and feedback incorporation.

February 20th - March 10th: Refinement and Advanced Feature Development
- Task: Based on midterm feedback, refine algorithms, enhance chatbot functionality, and improve system integration.
- Milestone: Advanced features and improvements implemented.

March 11th - March 24th: Testing and MLOps Integration
- Task: Conduct thorough testing of the system, integrate MLOps practices, and start preparing for final deployment.
- Milestone: System tested and MLOps procedures in place.

March 25th - April 2nd: Final Preparations and Documentation
- Task: Finalize all features, complete documentation, and prepare for the final presentation.
- Milestone: Code Compass fully operational and documented.

April 4th: Final Project Submission and Presentation
- Task: Submit the final version of Code Compass and present the completed project.
- Milestone: Successful submission and presentation of Code Compass.