Reforge

Natalie Stricker
Fakultät Elektrotechnik, Medien und Informatik
Ostbayerische Hochschule Amberg-Weiden
Amberg, Deutschland

I. Introduction

The chapter discusses the increasing significance of artificial intelligence in various domains, such as personalized product recommendations, autonomous vehicles, and disease diagnosis. It explores the benefits and concerns, including the potential impact on traditional roles like taxi drivers. The text stresses the importance of AI as a supportive tool that must be critically evaluated due to risks like misinformation and biases. Additionally, it highlights the automation potential for repetitive tasks, especially in academia and technical fields. The focus is on developing a web-based report generator named ”Reforge” to streamline the process of analyzing and summarizing technical reports. The chapter outlines the structure of the thesis, covering theoretical foundations, research overview, design concepts, implementation details, results, future enhancements, and concluding remarks.

II. Basics

Chapter ”Fundamentals” explains the necessary basics for automating text generation in this work. The focus is on understanding what a model is and how it functions. A model is a large language model capable of understanding and generating texts. Models are trained on large text datasets like books, articles, and websites to learn words, expressions, and sentence structures to maintain coherence and relevance. The work uses an OpenAI model known for its capabilities in integrating with applications. Different model sizes have varying levels of accuracy and costs, with the work using the -3.5-Turbo model for its balance of performance and cost-effectiveness. The text units for models are tokens, and costs are based on token usage. Prompts guide model outputs, prompting for specific responses such as length or format. Regular expressions are also discussed for pattern recognition in text manipulation. The section also covers web design basics for user-friendly interfaces, including responsive design and visual hierarchy principles. TypeScript is chosen as the programming language for its additional features like typing and object-oriented programming, enhancing code reliability and maintenance. React, a frontend framework, aids in building single-page applications dynamically loading content. Communication between frontend and backend is facilitated by APIs, with RESTful APIs using HTTP protocols for resource management through CRUD operations. The chapter emphasizes the importance of self-descriptive data for proper handling of API requests.

III. State of research and related work

This section provides an overview of current research on technologies essential for understanding and implementing the project, including text summarization, LaTeX, and web development. It discusses different types of summarization methods such as extractive and abstractive, highlighting the challenges and capabilities of each. The potential of AI in generating summaries is explored, noting the need for human verification due to issues like hallucinations in the output. The text also touches on the security concerns related to AI models like prompts for AI text generation. LaTeX is described as a powerful typesetting system widely used in academic circles for document creation. Lastly, the advancements in web development, particularly with JavaScript frameworks like React, are discussed, along with related web applications that generate summaries similar to the project at hand. The project aims to offer an innovative solution for creating technical reports from LaTeX or other document formats, providing flexibility for users in customizing outputs.

IV. Conception and design of Reforge

The text discusses the essential step of requirement analysis in software project development using Kleuker’s model. Functional requirements are classified with modal verbs like ”must,” ”should,” and ”will.” Acceptance criteria are defined for each requirement to ensure proper implementation. The project involves functions for text extraction, summarization, and conversion, with specific criteria for each task. The application utilizes OpenAI models for text summarization due to their precision and ease of integration. The system architecture includes a frontend, backend, and external systems like OpenAI and DeepL for text processing and translation. Data storage is temporary, managed by middleware like Multer. The upload process is illustrated through a sequence diagram showing interactions between components.

V. Implementation of Reforge

The backend of Reforge is responsible for processing uploaded files and generating technical reports. The communication between the frontend and backend is detailed, involving RESTful API calls for data exchange and report download. The frontend triggers the backend process by uploading a LaTeX-ZIP folder, filling in parameters, and clicking a button. The backend processes the file, updates the response header, and sends the report back to the frontend. Important libraries used in backend development include ADM-ZIP for ZIP file