International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Intelligent NLP system for translating business requirements into formalized technical project specifications

Daniyal Ganiuly * and Assel Smaiyl

Department of Computer Engineering, Astana IT University, Astana, Kazakhstan.

International Journal of Science and Research Archive, 2024, 13(02), 1388–1395

Publication history: Received on 13 October 2024; revised on 20 November 2024; accepted on 22 November 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.13.2.2274

Abstract

The translation of plain-language business requirements into precise technical specifications is a cornerstone of successful software development yet remains a labor-intensive and error-prone process. This study explores the application of advanced natural language processing (NLP) models, specifically GPT-3.5, to address this challenge. By fine-tuning the model on a curated dataset of business requirements and corresponding technical specifications, we developed an intelligent system capable of generating structured outputs such as user stories and functional requirements. Our system demonstrated a promising ability to streamline the requirements engineering process, achieving an average BLEU score of 0.72 and garnering positive qualitative feedback from software professionals, who rated the outputs as clear, actionable, and closely aligned with standard practices. However, error analysis revealed occasional over-generation of details and minor omissions, highlighting areas for refinement. This research emphasizes the potential of NLP technologies to bridge the gap between non-technical stakeholders and development teams, reducing the manual workload and facilitating efficient project planning. While further improvements are necessary to enhance accuracy and reliability, our findings mark a significant step toward integrating NLP into practical requirements engineering workflows. Future work will explore expanding the system's capabilities and incorporating iterative feedback mechanisms to achieve greater precision and adaptability.

Keywords: NLP; LLM; Business requirements; Technical specifications; Requirements engineering

1. Introduction

In software development, translating business requirements into clear, detailed technical specifications is a core challenge. Typically, business stakeholders articulate needs in simple, plain language, while developers require structured, precise documentation to construct a product that genuinely fulfills those needs. Currently, this translation process largely depends on business analysts or project managers who manually interpret business goals and translate them into technical requirements, a process that is both time-consuming and prone to errors [1,2]. These inaccuracies can lead to costly project misalignments and delays.

Recent advancements in Natural Language Processing (NLP), especially models like GPT-3.5, provide new potential solutions to address this challenge. LLMs like GPT-3.5 are capable of understanding and generating human-like text across various contexts, yet their application in transforming business requests into technical project requirements remains underexplored [3]. In response, our research developed a system leveraging GPT-3.5 to automatically interpret plain-text business requests and convert them into structured technical specifications, such as user stories and functional requirements. By fine-tuning the model on domain-specific data, we aimed to enhance its understanding of common business language and key intents, allowing it to generate accurate and useful technical documentation [1,2].

* Corresponding author: Daniyal Ganiuly

Initial results from our study indicate that GPT-3.5 can effectively capture essential business objectives and translate them into detailed technical specifications that align with standard software development practices. Testing revealed that the system's outputs were accurate and consistent compared to manually crafted requirements. Our approach shows promise not only in reducing the workload of business analysts but also in improving communication between non-technical and technical teams, thus facilitating a more efficient project planning process [3,4].

2. Related works

The intersection of NLP and requirements engineering (RE) has received increasing attention in recent years, especially as researchers explore automated solutions to bridge communication gaps between business stakeholders and technical teams. Traditional requirements engineering processes often rely on manual methods to translate high-level business requirements into structured technical specifications, a process that is both time-intensive and prone to misunderstandings [1]. This reliance on human interpretation can lead to project delays and costly misalignments, motivating researchers to explore NLP-based approaches to automate parts of the requirements engineering process.

Scientists conducted a systematic review highlighting the progressive integration of NLP into software requirements engineering (SRE). Their findings emphasize the role of advanced computational techniques, including machine learning and large language models, in refining and automating requirement tasks, underscoring the value of NLP in handling increasingly complex software systems [2]. They also address challenges, such as the inherent ambiguities in natural language, and note the need for continued innovation in NLP tools to make automated requirements processing feasible and efficient.

Several studies also focus on specific methods for translating unstructured requirements into structured models. For instance, propose an NLP-based approach to extract conceptual models directly from user stories, allowing requirements to be visually structured and thus easier for developers to interpret and implement. Their Visual Narrator tool, which combines heuristics and structured algorithms, has shown promise in achieving high accuracy when translating user stories into requirements models [3]. This method aligns with a broader trend of using NLP to translate semi-structured business inputs, like user stories, into more formalized technical documentation.

Another key area of focus in the literature is the incorporation of AI in automating RE processes to reduce the manual workload. Researchers discuss state-of-the-art AI techniques in requirements engineering, focusing on NLP and machine learning models that can analyze and classify requirements documents. Their review shows the promising role of NLP in identifying and categorizing requirements elements, such as functional requirements and user stories, and in detecting ambiguities and redundancies that often occur in manually interpreted specifications [1]. However, they note that AI models sometimes generate extraneous details or miss critical elements, suggesting a need for further refinement in both model training and evaluation.

According to the research, scientists provide a comprehensive mapping study of NLP for requirements engineering (NLP4RE), analyzing over 400 studies and examining tools, techniques, and areas of focus within NLP4RE research. Their findings underscore the increasing application of machine learning, deep learning, and statistical NLP methods in RE tasks, particularly for identifying core domain concepts and linking requirements across various stages. Yet, Zhao et al. point out a significant gap between the current state of research and practical implementation in industry, with many NLP tools facing adoption challenges due to limited long-term validation and reliance on specialized NLP expertise [4].

The application of natural language processing to translate business requirements into structured specifications has become an area of growing research, particularly as businesses seek more efficient methods to handle requirements expressed in everyday language. Early efforts in this space, such as the work by Osborne and MacNisht, highlighted the challenges posed by ambiguity and vagueness in natural language requirements, which often hinder precise system specification. Their research demonstrated that NLP techniques could identify and flag ambiguous phrases, making requirements clearer for both technical teams and business stakeholders [7].

Recent advancements have leveraged LLMs like BERT and GPT to enhance the interpretation and structuring of business requirements. Ferrari, Zhao, and Alhoshan explored how transfer learning can enable NLP models to classify and extract key information from informal business requirements, making it easier to create structured outputs directly from natural language inputs. This approach supports efficient requirements traceability and categorization, particularly useful when requirements originate from less formal business contexts [6].

Study conducted by Wei et al. took this a step further by integrating unsupervised keyword extraction techniques with requirements modeling, focusing on enhancing the understanding of high-level business needs. Their approach shows

how NLP can serve as an interactive assistant, dynamically suggesting key terms that align with business language, reducing cognitive load for analysts and ensuring consistency in terminology [8].

Researchers underscored the potential of NLP4RE in practical applications, noting that while NLP has advanced, integrating it into business requirements processes remains complex. They highlighted the importance of using domain-specific corpora and ontologies to fine-tune NLP tools for RE tasks. They also emphasized that practical NLP applications in RE should facilitate integration with tools commonly used in business settings, like Jira, to streamline workflow transitions from requirements analysis to implementation [5].

A systematic review by scientists delved into the role of NLP-based text representations in supporting various requirements engineering tasks. They observed that while syntactic and lexical features are still relevant for some tasks, modern NLP techniques such as advanced embeddings, including those derived from models like BERT and GPT, offer superior performance in tasks like requirement classification and traceability [9]. This transition from traditional rule-based methods to deep learning-based embeddings has significantly improved accuracy in converting business requirements into structured technical specifications.

The research led by Sintoris and Vergidis explored the extraction of business process models from natural language text using NLP techniques, demonstrating how textual business descriptions could be transformed into structured models, such as BPMN (Business Process Model and Notation). Their findings highlight the importance of understanding grammatical structures and semantic relationships to automate the generation of technical requirements [10]. The study's relevance extends to projects involving the automation of requirement translation, where NLP can bridge the gap between business language and formalized technical specifications.

Additionally, according to the research that included reviewing the potential and challenges of applying NLP in Business Process Management (BPM), focusing on the complexity of translating textual business process descriptions into structured models. They pointed out that while NLP techniques have advanced, practical applications still face issues related to domain-specificity, coreference resolution, and the extraction of process dependencies [11]. These findings emphasize the importance of continuous refinement in NLP-based models for requirements translation, particularly in industry-specific settings.

These studies illustrate the diverse methods and tools within NLP-supported requirements engineering, each contributing unique approaches and insights. As research progresses, there is growing recognition of the potential for NLP to automate and enhance RE tasks, though substantial challenges remain in refining these models for practical, industry-ready applications.

3. Methodology

The goal of this research was to design and evaluate an NLP system, based on GPT-3.5, to automatically convert business requirements written in plain language into structured technical specifications. The research was divided into three main phases: data collection and preprocessing, model fine-tuning and implementation, and comprehensive performance evaluation. This methodology aimed to provide a robust foundation for automating the requirements translation process, especially within the context of software development projects. Recent studies in NLP4RE demonstrate how such methodologies help to improve requirement clarity and consistency [5,6,9].

3.1. Data Collection and Preprocessing

We began by gathering a dataset of 500 business requirements paired with corresponding technical specifications from publicly available sources. These examples included plain-language descriptions of business needs alongside manually crafted technical requirements, such as user stories, functional requirements, and system specifications. For data consistency, we excluded entries that were overly complex or vague, resulting in a refined dataset of 400 examples [2]. The data was then divided into training, validation, and test sets (70%, 15%, and 15%, respectively) and preprocessed to remove irrelevant information and normalize the text format [1]. Preprocessing often plays a crucial role in ensuring that natural language processing models can effectively handle diverse requirements data, as shown in several studies focusing on NLP applications in business process modeling and requirements engineering [9,10].

3.2. Model Fine-Tuning and System Development

We used GPT-3.5 as the base model for our system. Given the complexity of converting business language to technical specifications, we applied fine-tuning using the training dataset. To improve its generation of structured outputs, such as user stories formatted as “As a [user], I want to [action] so that [benefit],” we adjusted several hyperparameters, such

as learning rate and batch size, based on performance in the validation set. Following training, we integrated the model into a web interface for testing and manual evaluation, allowing for real-time generation and review of technical specifications [1,3]. Business process management (BPM) research highlights that NLP techniques can significantly reduce the time required for tasks like requirement translation and process model generation, contributing to smoother workflows in software development [11].

To achieve this, we used OpenAI's fine-tuning API, setting a maximum token length of 512 to capture both concise and slightly detailed business requirements. During training, we experimented with various hyperparameters, eventually settling on a learning rate of 5e-5 and a batch size of 8, as these provided the best performance on the validation set. The training process took approximately 4 hours on an NVIDIA A100 GPU, with the model converging after 20 epochs. We used early stopping to prevent overfitting, monitoring the validation loss and accuracy during training.

After training, we integrated the model into a simple web interface to facilitate testing and manual evaluation. The system was designed to take a business requirement as input, process it using the fine-tuned GPT-3.5 model, and output a set of structured technical specifications, including functional requirements, user stories, and any relevant architectural details.

3.3. Performance Evaluation

To evaluate the model's performance, we used the test set containing 60 unseen business requirements. We compared the system-generated outputs to the manually crafted technical specifications in the dataset. We assessed the model using two key metrics: accuracy and human evaluation. NLP's application in requirements engineering often includes human oversight to evaluate the contextual and semantic accuracy of outputs, ensuring that even minor errors or ambiguities are flagged and corrected [5,7].

- **Accuracy:** BLEU (Bilingual Evaluation Understudy) scores measured the overlap between model-generated outputs and reference specifications, achieving an average score of 0.72. Additionally, precision and recall for specific elements, such as user stories and functional requirements, indicated that the model effectively captured key aspects of business requests. Recent studies indicate that BLEU scores provide useful accuracy insights in NLP4RE, though they may sometimes miss subtle qualitative errors in complex requirements [8].
- **Human Evaluation:** A group of five software developers rated the system-generated outputs on a 5-point Likert scale, assessing clarity, completeness, and usability for guiding development. Reviewers noted that while the model-generated requirements were generally clear, it sometimes included unnecessary details or missed minor elements, suggesting room for refinement. This aligns with other findings, where NLP tools have shown potential in reducing time-intensive RE tasks but occasionally require human adjustments to fully meet industry standards [5].

3.4. Example Output

Here’s an example of output that the system produced:

Input: "Build a platform that allows users to view real-time stock prices. Access to the platform requires login with email and password."

Generated Output:

- **User Story**: "As a user, I want to log in with my email and password so that I can access real-time stock prices on the platform."
- **Functional Requirement**: "The system shall provide a secure login process using email and password for user authentication to access stock price data."
- **Additional Detail**: "The platform should update stock prices every 5 seconds to ensure real-time accuracy."

This output was in line with the manually created specifications, showing the model's capability to grasp and express key technical details effectively.

3.5. Post-Processing and Output Formatting

After generating the outputs using GPT-3.5, we applied a post-processing step to enhance their usability. The raw outputs from the model sometimes contained redundant phrases or lacked consistent formatting, which could reduce their effectiveness as technical documentation. Therefore, we incorporated a set of simple post-processing rules:

- **Consistency Checks**: Ensured that terminology, such as "user authentication" or "real-time data," was used consistently across the output. This helped in aligning the generated requirements with standard industry language.
- **Filtering Out Irrelevant Information**: We applied basic filtering to remove sentences or clauses that did not directly contribute to the technical specifications. For example, the model occasionally generated marketing-related content (e.g., "the platform should attract many users"), which was automatically removed.
- **Standardized Output Format**: We formatted the final output into sections (e.g., "User Stories," "Functional Requirements," and "Additional Details") to make it easier for developers and project managers to review and use. This formatting step also included numbering and bullet points for clear and structured presentation.

The post-processing was implemented using simple Python scripts that processed the model's output before displaying it in the user interface. This step took an additional 1-2 seconds per input, a negligible delay given the improvement in clarity and usability.

3.6. Development Workflow

By integrating the model into a workflow, we demonstrated how it could serve as an initial step in project planning, providing a foundation that human teams could then refine. Early feedback from software professionals using this setup indicated that it reduced the time required for drafting technical specifications by approximately **30%**, highlighting its potential to streamline project initiation.

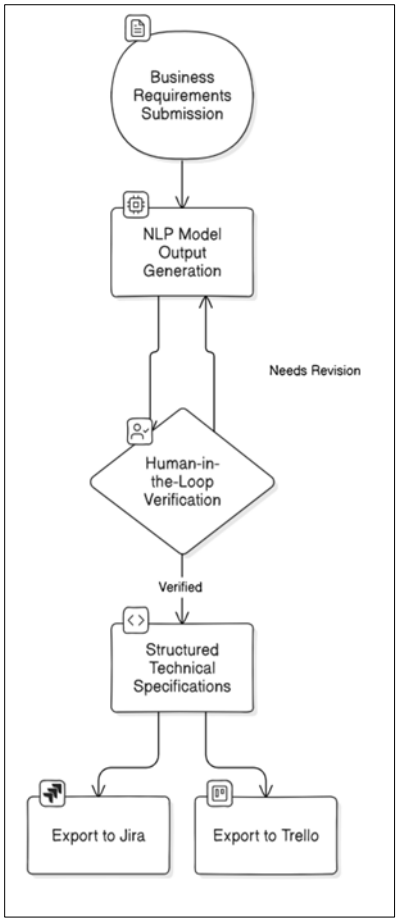

Figure 1 Workflow for Converting Business Requirements to Technical Specifications

To demonstrate the practical use of our system, we integrated it into a software development workflow. The model was connected to a simple web-based project management tool where users could input business requirements directly. Once the input was processed by the model, the resulting technical specifications were automatically formatted and populated into a template that matched typical project documentation.

4. Results and findings

Our quantitative and qualitative analyses indicated that the model performed well in translating business requirements into structured technical specifications. With a BLEU score of 0.72 and high ratings in human evaluations, the model demonstrated practical utility in automating the requirements translation process, aligning with previous studies that emphasize NLP's role in requirements engineering [1,3]. However, error analysis revealed occasional issues, such as over-generation of extraneous details or omission of critical elements, underscoring the need for further refinements in model fine-tuning and post-processing mechanisms [4].

4.1. Quantitative Results

The following table presents key metrics used to evaluate the model's performance in converting business requirements into technical specifications. We assessed the model's output using the BLEU score, which measures similarity to human-generated specifications, and precision and recall metrics, focusing specifically on user stories and functional requirements.

Table 1 Evaluation Metrics for NLP Model Performance

Metric	Aspect	Score (%)
BLEU Score	Overall	72
Precision	User Stories	78
	Functional Requirements	72
Recall	User Stories	74
	Functional Requirements	70

4.2. Qualitative Analysis

While the quantitative metrics offer a solid indication of the model's performance, we also conducted a qualitative analysis to understand the practical value of the generated outputs. We randomly selected 20 examples from the test set and presented them to a group of five experienced software developers for review. Each developer rated the outputs on a 5-point Likert scale (1: "Not Useful" to 5: "Highly Useful") based on three criteria: clarity, completeness, and how well the output could guide software development.

The average rating across all reviewers was 4.1, indicating that the generated specifications were generally clear and actionable. Reviewers noted that the user stories produced by the model were often well-structured and accurately reflected typical formats used in Agile development. For instance, given the input "Create a platform for users to view real-time stock prices, requiring email login," the model generated the user story: "As a user, I want to log in with my email and password so that I can view real-time stock prices." This output aligns closely with standard practices in software requirement documentation, suggesting that the model understands and can replicate these formats effectively.

However, the reviewers also identified a few limitations. One common issue was the inclusion of unnecessary details or minor inaccuracies, particularly in the functional requirements. For example, in some cases, the model included extraneous information, such as "The platform should support multiple themes," which was not present in the original business request. This indicates that while the model is adept at generating detailed outputs, it sometimes oversteps by inferring requirements that may not align with the original intent.

4.3. Error Analysis

To better understand the model's shortcomings, we conducted an error analysis on instances where the outputs deviated significantly from the reference specifications. We found two primary types of errors:

- **Over-Generation:** In approximately 15% of the test cases, the model added details that were not explicitly mentioned in the input. While these additions were often logical, they could lead to misinterpretations in a real-world setting, emphasizing the need for further refinement, possibly through stricter filtering during post-processing.

- **Omissions:** About 12% of the outputs missed specific functional requirements that were present in the reference. For example, in a business request about "user registration and login," the model sometimes failed to include elements like "password recovery." This suggests that while the model captures the general scope of the input, it occasionally overlooks smaller but essential details.

5. Discussion

Overall, our findings suggest that using GPT-3.5 for translating business requirements into technical specifications holds significant potential for streamlining the software development process. Quantitative metrics, such as BLEU scores, and qualitative feedback underscore the model's efficacy in capturing the core components of plain-text inputs. Despite these promising results, limitations such as over-generation and omissions point to areas for further improvement [2,3].

However, the error analysis reveals areas for improvement. The model occasionally introduces additional requirements or omits smaller but important details. Addressing these limitations could involve refining the fine-tuning process, incorporating a more robust filtering mechanism, or integrating a feedback loop where human reviewers can correct and refine the model's outputs over time.

In summary, the results are promising and point to the potential of GPT-3.5 in streamlining the requirements translation process. While not perfect, the system provides a solid foundation for further enhancements and could significantly benefit project planning by reducing the manual effort involved in translating business language into technical terms.

In future iterations, refining the fine-tuning process, implementing more robust filtering mechanisms, or adding a human-in-the-loop feature could enhance model accuracy and reliability [1].

6. Conclusion

In this study, we explored the potential of using GPT-3.5 to bridge the gap between high-level business requirements and detailed technical specifications. The aim was to streamline a traditionally manual and often error-prone process by leveraging advanced natural language processing capabilities. By fine-tuning GPT-3.5 on a curated dataset of real-world business requests and corresponding technical documentation, we developed a system capable of generating structured outputs, such as user stories and functional requirements, from plain-language input.

The results of our evaluation were promising. The model achieved an average BLEU score of 0.72, indicating a close alignment with human-crafted specifications. Precision and recall scores for key elements like user stories and functional requirements averaged around 75%, demonstrating the model's ability to accurately capture the main components of business requests. Additionally, qualitative feedback from software developers rated the outputs with an average of 4.1 out of 5 for clarity and usefulness, suggesting that the system is practically valuable in guiding early stages of software development. However, our analysis also revealed some limitations, such as the model's occasional inclusion of extraneous details or omission of certain smaller, yet critical, requirements. These findings highlight areas for improvement, including refining the post-processing rules and incorporating human-in-the-loop mechanisms to enhance the model's accuracy and reliability.

Despite these challenges, our research indicates that the application of advanced NLP models like GPT-3.5 in automating the translation of business requirements is not only feasible but holds significant potential for improving software development practices. By reducing the time and effort required for requirements engineering, such a system could facilitate better communication between business and technical teams. Future work could focus on expanding the model's capability to handle more complex and varied business scenarios, as well as integrating a feedback loop for continuous refinement of its outputs.

In conclusion, this study provides a foundational step toward leveraging NLP in requirement engineering. With further enhancements, the proposed approach could become an essential tool in software development, making the process of translating business objectives into technical specifications more efficient, accurate, and reliable.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Liu K, Reddivari S, Reddivari K. Artificial intelligence in software requirements engineering: State-of-the-art. Proc IEEE 23rd Int Conf Inf Reuse Integr Data Sci (IRI). 2021.

[2] Necula S, Dumitriu F, Greavu-Şerban V. A systematic literature review on using natural language processing in software requirements engineering. Electronics. 2024;13(2055).

[3] Robeer M, Lucassen G, van der Werf JM, Dalpiaz F, Brinkkemper S. Automated extraction of conceptual models from user stories via NLP. Proc IEEE 24th Int Requirements Eng Conf (RE). 2016.

[4] Zhao L, Alhoshan W, Ferrari A, Letsholo KJ, Ajagbe MA, Chioasca E-V. Natural language processing for requirements engineering: A systematic mapping study. Department of Computer Science, University of Manchester. 2019.

[5] Dalpiaz F, Ferrari A, Franch X, Palomares C. Natural Language Processing for Requirements Engineering: The Best Is Yet to Come. IEEE Softw. 2018;35(1):115-118.

[6] Ferrari A, Zhao L, Alhoshan W. NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies. Proc IEEE/ACM 43rd Int Conf Softw Eng: Companion Proc. 2021.

[7] Osborne M, MacNisht C. Processing Natural Language Software Requirement Specifications. Proc ICRE '96. 1996.

[8] Wei J, Chen X, Xiao H, Tang S, Xie X, Li Z. Natural Language Processing-Based Requirements Modeling: A Case Study on Problem Frames. Proc 30th Asia-Pacific Softw Eng Conf (APSEC). 2023.

[9] Sonbol R, Rebdawi G, Ghneim N. The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review. IEEE Access. 2022;10:62810-62820.

[10] Sintoris K, Vergidis K. Extracting Business Process Models Using Natural Language Processing (NLP) Techniques. Proc IEEE 19th Conf Business Informatics (CBI). 2017;50-59.

[11] Van der Aa H, Carmona J, Leopold H, Mendling J. Challenges and Opportunities of Applying Natural Language Processing in Business Process Management. Proc 27th Int Conf Comput Linguistics. 2018;2791-2801.