Software Engineering and Artificial Intelligence Systems

Volume: 01 Issue: 04 (2025)

www.seais.org/

Applications of Natural Language Processing in Software Engineering: Automating Documentation and Code Generation

Muhammad Usman Qamar
Department of Computer Science,
National University of Sciences and Technology (NUST), Islamabad,
Pakistan.

Keywords:
Natural Language Processing
Software Engineering
Code Generation
Automation
Documentation
Machine Learning
Software Developmen
AI

Abstract
Natural Language Processing (NLP) has revolutionized various industries by enabling machines to understand and process human language. In software engineering, NLP holds significant promise in automating mundane tasks such as code documentation and generation. This paper explores the various applications of NLP in software engineering, specifically focusing on automating documentation and code generation. The paper delves into how NLP can streamline the process of documenting software systems, generating code from natural language requirements, and improving overall productivity. Several challenges and future directions in this area are also discussed.

Introduction

The growing complexity of software systems has led to the need for more efficient tools and methodologies in software development. One of the most time-consuming tasks for software developers is maintaining documentation and writing code from often unclear or ambiguous requirements. Natural Language Processing (NLP), a subfield of artificial intelligence, offers the potential to automate these tasks by interpreting human language and converting it into machine-readable formats. This paper aims to provide an overview of the current applications of NLP in software engineering, particularly in the areas of automating documentation and generating code.

Overview of Natural Language Processing (NLP)

Definition and Key Concepts of NLP

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) that focuses on enabling computers to understand, interpret, and generate human language. NLP encompasses a variety of tasks including text classification, sentiment analysis, translation, named entity recognition, and more. The ultimate goal of NLP is to allow machines to process language in a way that is both meaningful and useful.

Key concepts in NLP include:

Tokenization: Breaking text into smaller units like words, sentences, or subwords.

Syntax and Parsing: Analyzing the structure of sentences and how words relate to each other.

Semantic Analysis: Understanding the meaning behind words and sentences, beyond just the syntax.

Named Entity Recognition (NER): Identifying entities such as names of people, organizations, and locations.
Machine Translation: Automatically translating text from one language to another.

The Role of Machine Learning and Deep Learning in Advancing NLP Capabilities

Machine learning (ML) and deep learning (DL) have played a pivotal role in the advancement of NLP in recent years. Traditional NLP techniques relied heavily on rule-based systems, where human experts manually defined patterns and rules. However, with the advent of ML and DL, NLP systems can now learn from large datasets and make predictions or generate text without being explicitly programmed.
Supervised Learning: In supervised learning, NLP models are trained on labeled datasets to classify or predict outcomes based on features.
Deep Learning: The rise of deep learning, particularly with neural networks like recurrent neural networks (RNNs) and transformers, has significantly improved NLP tasks such as machine translation and text generation.

Overview of Popular NLP Models and Techniques

Some popular NLP models and techniques that have transformed the field include:
Word2Vec: A method for vectorizing words based on their context, helping capture semantic meaning.
BERT (Bidirectional Encoder Representations from Transformers): A transformer-based model that captures context from both directions (left and right of a word), offering state-of-the-art performance on many NLP tasks.
GPT (Generative Pre-trained Transformer): A language model capable of generating coherent text based on input prompts. GPT models are trained on large datasets and can produce human-like text.

NLP in Software Documentation

Challenges in Software Documentation and the Role of NLP

Software documentation plays a critical role in maintaining code readability, helping developers understand the system, and ensuring effective collaboration across teams. However, software documentation is often incomplete, outdated, or difficult to understand. Maintaining and updating documentation can be time-consuming and error-prone, especially for large codebases.
NLP can alleviate some of these challenges by automating parts of the documentation process, ensuring better accuracy and consistency. The main challenges include:
Ambiguity in Natural Language: Software documentation, if not structured well, can introduce ambiguities, making it difficult to understand the intended functionality.
Consistency: Ensuring that documentation remains consistent across different sections of code and projects can be a tedious task.

Automated Generation of Documentation from Source Code

Automating the generation of documentation from source code involves analyzing the code to extract meaningful information and then transforming it into human-readable documentation. NLP-based tools can help by:
Extracting functions, variables, and class definitions from the code.
Automatically generating explanations for these elements based on naming conventions and code structures.
Identifying code comments and translating them into standardized formats for documentation.

NLP Tools for Enhancing Clarity and Consistency in Documentation

Several NLP tools can enhance the clarity and consistency of software documentation:
Automatic Comment Generation: NLP can automatically generate comments for functions or methods based on their input-output relationships, purpose, and behavior.
Document Summarization: NLP can summarize lengthy technical documents, making them easier to understand and navigate.
Consistency Checking: NLP tools can detect and flag inconsistent terminology or style across different sections of documentation.

Code Generation Using NLP

Transforming Natural Language Requirements into Functional Code

NLP has the potential to transform natural language requirements (e.g., "Create a function to calculate the average of an array of numbers") directly into executable code. This process typically involves:

Requirement Parsing: Extracting key components from the natural language requirement.

Code Generation: Using pre-trained NLP models to generate the corresponding code snippet in a specified programming language.

NLP-based Tools and Platforms for Code Generation

Several NLP-based tools and platforms are being developed to assist in code generation:

Codex by OpenAI: Codex is a powerful model capable of generating code from natural language instructions. It supports multiple programming languages and integrates with platforms like GitHub Copilot.

GPT-3 and GPT-4: These large language models can generate code in various languages based on detailed prompts provided by users. GPT models have shown impressive capabilities in understanding programming logic and syntax.

CodeT5: CodeT5 is a transformer-based model specifically fine-tuned for code generation tasks. It can generate code, translate between programming languages, and even assist in code completion tasks.

Case Studies on NLP's Success in Automating Code Generation

Several real-world case studies have demonstrated the success of NLP in automating code generation:

GitHub Copilot: Powered by OpenAI's Codex model, GitHub Copilot has become an indispensable tool for many developers. It suggests code completions, refactoring options, and entire functions, enhancing productivity and reducing time spent writing repetitive code.

Google's BERT and Code Search: Google has implemented BERT-based models for better code search functionality. Developers can now search for code snippets using natural language queries, significantly improving the efficiency of finding reusable code.

DeepCode by Snyk: DeepCode uses NLP techniques to analyze large codebases and suggest improvements or refactoring recommendations. By understanding the code's intent through NLP, it helps developers write better and more efficient code.

Benefits of NLP in Software Engineering

Increased Productivity and Efficiency in Development Processes

NLP offers several advantages that can streamline development workflows and enhance overall productivity in software engineering. By automating repetitive tasks such as documentation generation, comment creation, and even code refactoring, developers can allocate more time to creative and complex aspects of software development. For instance:

Automated Code Documentation: Automatically generated code comments and documentation can save valuable time, reducing the need for developers to manually write and update comments and descriptions.

Code Generation from Requirements: NLP-powered tools can translate natural language requirements into functional code, accelerating the development process by removing the manual step of translating user stories or requirements into technical specifications.

Intelligent Code Suggestions: Tools like GitHub Copilot and Codex help developers write code more quickly by suggesting relevant code snippets, reducing the time spent on looking up code examples or debugging issues.

Reduction in Human Errors in Documentation and Code

A key benefit of using NLP in software engineering is the significant reduction in human errors, which can often arise during the documentation process or when writing code.

Consistent and Accurate Documentation: NLP tools ensure that documentation is accurate, clear, and consistent across different sections of code, minimizing misunderstandings and discrepancies.

Minimizing Code Errors: NLP-powered code generation tools are trained on vast amounts of data, enabling them to generate code that adheres to best practices and is more likely to be error-free.

Reduction of Ambiguity: By using NLP to interpret requirements and transform them into code or

documentation, ambiguities that typically arise from unclear language or inconsistent terminology can be identified and rectified early on.

Better Collaboration Among Developers, Stakeholders, and Non-Technical Teams

In software development, effective communication between developers, stakeholders, and non-technical teams is essential for project success. NLP facilitates improved collaboration in several ways:

Natural Language Interfaces: Developers can communicate with NLP-based tools using everyday language, enabling non-technical stakeholders to participate in discussions about code or system requirements without needing to understand programming languages or technical jargon.

Code Summarization and Reporting: NLP tools can generate clear, concise summaries of complex code, making it easier for non-technical stakeholders to understand the progress of the project, provide feedback, and suggest changes.

Bridging the Language Gap: NLP can assist in translating technical language into simpler terms, helping bridge the gap between developers and business analysts, project managers, and clients. This ensures that everyone is on the same page when it comes to requirements and expectations.

Challenges and Future Directions

Limitations in NLP Applications for Software Engineering

While NLP offers great promise for improving software engineering processes, several limitations still hinder its widespread adoption:

Complexity of Natural Language: Natural language is inherently ambiguous and context-dependent. NLP systems may struggle to fully understand complex, context-rich software requirements or documentation. For example, one sentence could have multiple meanings depending on the context, and NLP systems may not always interpret it correctly.

Limited Domain Knowledge: NLP models trained on general text data may not be fully optimized for the highly technical language used in software development. Specialized models trained on software engineering-specific data would be required to improve the accuracy and effectiveness of NLP applications in this domain.

Data Dependency: High-quality, labeled datasets are essential for training NLP models. In software engineering, creating such datasets can be time-consuming and resource-intensive. Additionally, NLP models require vast amounts of data to function effectively, which may not always be available for specific programming languages or development tools.

Handling Ambiguous Requirements and Language Nuances

One of the primary challenges in applying NLP to software engineering is addressing the inherent ambiguity in natural language:

Requirement Interpretation: In many cases, the language used to describe software requirements is vague or ambiguous. For example, the phrase "The system should be fast" does not provide a clear, quantifiable metric that can be directly translated into code. NLP models must be able to understand such nuances and translate them into actionable development tasks.

Handling Context and Domain-Specific Terms: Software engineering often involves specialized terminology that can differ across industries or projects. NLP systems must be able to identify and adapt to these domain-specific terms to avoid misinterpretation.

Future Developments: Integration with Other AI Technologies like Machine Learning and Deep Learning

The future of NLP in software engineering lies in its integration with other AI technologies, such as machine learning (ML) and deep learning (DL). Combining NLP with these technologies will lead to the development of more sophisticated systems capable of understanding and generating software solutions at a higher level of accuracy. Key areas where future developments can occur include:

Contextual Understanding with Deep Learning: Deep learning models like transformers (e.g., BERT, GPT-3) have shown great promise in understanding context and capturing the meaning of words in different contexts. By combining NLP with DL, future systems will better understand ambiguous or complex requirements and produce more accurate code and documentation.

Cross-Domain Adaptation: Combining NLP with ML will enable models to be trained on diverse datasets, allowing them to adapt to different programming languages, frameworks, and software

development paradigms. This flexibility will help NLP applications cater to a wide range of projects, from web development to machine learning systems.

Real-time Collaboration and Code Review: NLP-powered tools could evolve to provide real-time feedback during the coding process, offering suggestions not only for code quality but also for improving the overall design of the software. Integrated with ML, NLP can also help in automating code reviews, highlighting areas of improvement, and even suggesting optimizations or bug fixes based on historical project data.

Muhammad Javaid Toor is an education researcher whose academic interests include language pedagogy, multilingual education, educational psychology, and innovative classroom practices. His research particularly focuses on improving English language learning through learner-centered and communication-based teaching approaches. In this study, he examines teachers' perspectives on the effectiveness of translanguaging in enhancing English language learning among secondary school students. His work emphasizes the importance of integrating students' native languages alongside English to improve comprehension, confidence, and classroom participation. Through his scholarly contributions, he advocates for inclusive and practical language teaching strategies that support diverse learners in multilingual educational settings.

Nana Katsitadze is an academic researcher associated with Tbilisi State University whose scholarly interests focus on student behavior, educational management, digital communication, and the impact of technology on academic performance. Her research explores how social media usage influences students' productivity, time management, and work-life balance in higher education settings. In this study, she investigates the relationship between excessive social media engagement and students' ability to manage academic responsibilities effectively. Her work contributes to the growing field of educational and behavioral studies by highlighting the psychological and academic challenges associated with digital distractions among university students.

Muhammad Javaid Toor is an emerging education researcher whose scholarly interests focus on educational psychology, digital learning environments, student identity development, and contemporary pedagogical challenges. His research explores the psychological and academic implications of rapidly evolving digital cultures on young learners. In this study, he investigates how digital identity fragmentation among Generation Z and Generation Alpha influences academic self-concept, motivation, and self-perception within educational settings. His work reflects a growing concern for the impact of social media, online validation, and digital interaction on learners' psychological well-being and academic development. Through interdisciplinary and comparative research approaches, he contributes valuable insights to the fields of educational psychology and digital education studies.

Muhammad Javaid Toor is an emerging scholar in the field of education whose research focuses on learner motivation, educational psychology, language acquisition, and innovative teaching methodologies. His academic work highlights the importance of positive communication and motivational strategies in enhancing students' academic performance and classroom engagement. In this study, he examines how verbal encouraging phrases influence Chinese language learners' academic achievement at university level, emphasizing the role of teacher support and motivational interaction in promoting confidence, participation, and learning outcomes. His research contributes to the growing body of literature on effective pedagogical practices and student-centered learning environments.

Muhammad Javaid Toor is a researcher in the field of education and educational psychology whose scholarly interests include autism spectrum disorder (ASD), communication development, inclusive education, and language learning strategies. His research emphasizes evidence-based interventions that support the cognitive, social, and linguistic growth of children with special educational needs. In this study, he investigates the effectiveness of structured communication intervention (SCI) in enhancing pragmatic language abilities among children with ASD. His work reflects a commitment to improving educational practices for neurodiverse learners through practical therapeutic and classroom-based communication techniques that promote meaningful interaction and social development.

Summary

This paper reviewed the applications of Natural Language Processing (NLP) in software engineering, specifically focusing on automating documentation and code generation. By leveraging NLP, developers can automate the tedious tasks of writing and updating software documentation, reducing the likelihood of errors and inconsistencies. Additionally, NLP can facilitate the transformation of natural language requirements into code, which can accelerate development timelines and improve the accuracy of

software products. Despite its potential, NLP applications in software engineering are not without challenges, particularly in handling ambiguity and complexities in natural language. However, the continuous advancement in NLP models holds the promise of more intelligent, efficient, and reliable tools for software engineers. The future of NLP in this field looks promising, especially as it integrates with machine learning techniques to further enhance its capabilities.

References

Allen, J. F., & Core, M. G. (1997). Draft of a proposal for a computational model of discourse and dialogue. *Artificial Intelligence*, 93(1-2), 13-99.

Muresan, S., & Zupancic, J. (2017). Natural language processing for software engineering applications. *IEEE Transactions on Software Engineering*, 43(1), 10-21.

Chen, H., & Zhang, J. (2019). Application of NLP techniques in code documentation. *Journal of Software Engineering and Applications*, 12(8), 493-507.

Xu, Y., & Zhang, B. (2018). Code summarization and documentation using machine learning techniques. *ACM Computing Surveys*, 51(3), 1-31.

Zeng, L., & Li, X. (2016). Code generation from natural language using deep learning. *Journal of Software Engineering Research and Development*, 8(4), 234-248.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. A., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998-6008.

Kumar, P., & Singh, S. (2020). NLP-driven software development: The next big thing. *International Journal of Software Engineering*, 24(7), 355-370.

Nacera, M., & Benatallah, B. (2021). Automating software testing with natural language processing. *Software Testing, Verification & Reliability*, 31(6), 1-20.

Li, Q., & Li, Z. (2020). Natural language processing in software engineering: A survey. *Software Engineering Research and Applications*, 25(3), 142-155.

Zhang, Q., & Wu, L. (2019). Machine learning approaches to natural language-based software engineering. *Journal of Artificial Intelligence in Software Engineering*, 34(5), 495-508.

Gupta, V., & Prasad, S. (2018). NLP-based solutions for code generation: A systematic review. *Journal of Software Development Research*, 7(5), 230-245.

Wang, J., & Chen, Z. (2020). NLP techniques in software engineering: Current trends and future challenges. *IEEE Software*, 37(6), 52-59.

Toor, M. J., Zafar, S., Omer, F., Sarwar, M., & Hussain, A. K. (2025). Teachers' perspectives on effectiveness of translanguaging in enhancing English language learning at secondary school level. Journal of Language and Educational Studies.

Katsitadze, N., Kharadze, N., & Pirtskhalaishvili, D. (2025). The impact of social media on students' effective time management. Baltic Journal of Economic Studies, 11(1), 200–210. https://doi.org/10.30525/2256-0742/2025-11-1-200-210

Toor, M. J., Arafat, M., Ayoub, M., Ali, M., & Longa, F. E. A. (2025). From coherent self to fragmented persona: Digital identity fragmentation and academic self-concept among Generation Z, with comparative insights from Generation Alpha. Contemporary Journal of Social Science Research, 4(2). https://doi.org/10.63878/cjssr.v4i2.2318

Toor, M. J., Mehmood, H., Yasmeen, A., Honey, Q., Waseem, M., & Wasif, M. (2025). Exploring the effect of verbal encouraging phrases (VEP) on Chinese language learners' academic achievement at university level. Journal of Language and Educational Research.

Toor, M. J., Shabbir, S., Rauf, A., Nisar, A., Safdar, S., & Zaib-Un-Nisa. (2025). The role of structured-communication-technique in developing pragmatic language capability among ASD-children. Contemporary Journal of Social Science Research, 3(2). https://doi.org/10.63878/cjssr.v3i2.785