Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN: 2581 - 611X)
VOLUME:9, ISSUE:1, JANUARY 2026, PAPER NUMBER:31

Topic Modelling using BERTopic and Automated Project Generation

Marogani Rohan Sai [1], ***Konkepudi Naga Sai Rajesh*** [2], ***Ramojula Rajashekar*** [3], ***Jakkula HariDeepak***[4], ***Manish Chhabra***[5] ***& Dr. Gagan Deep Arora***[6]

[1,2,3,4] *Students,and* [5,6] *Faculty*
Department of Artificial Intelligence And Machine Learning,
Vardhaman College Of Engineering,
Hyderabad, India.
maragonirohansai@gmail.com

Abstract: *A new mechanism is shown in this paper linking BERTopic based topic modeling with chain of thought reasoning to manage project organization and generation. Through its capability to categorize both predetermined subjects and user defined projects, the system provides exact thematic correlation along with context-based relevance. While BERTopic isolates the primary subject points from consumer requests, an automated module produces thorough blueprints via workflow component optimization and task gap detection. Moreover, we present a novel knowledge graph technique to code generation that treats variables, functions, and class names as interlinked nodes with clear dependencies, then helping more intelligent and consistent code creation. When comparing conventional techniques (LDA), contemporary neural approaches (BERTopic), and uptodate large language models (Gemini 2.5, Claude 3.7 Sonnet, DeepSeek R1), substantial improvements in topic modeling accuracy as well as project organization quality are found. Experimental data indicate a noteworthy improvement in project planning efficiency and subject extraction accuracy on several varied sets of data. Our system clearly show much promise for enhancing artificial intelligence project management skills in changing job surroundings.*

Keywords: *BERTopic, Chain of Thought Reasoning, Project Management, Code Generation, Knowledge Graphs, Topic Modeling, Artificial Intelligence, Workflow Optimization, Task Gap Detection, Large Language Models, Semantic Analysis, Project Planning Efficiency, Thematic Correlation, Context-Based Relevance.*

1. INTRODUCTION:

The handling and knowledge of unstructured data present major difficulties in modern software development and project management. Digital data's exponential growth means that developers and project managers sometimes find it difficult to productively glean valuable insights from unprocessed text data, records, and job specifications. In fields where projects have complicated interdependencies and need organized planning techniques these problems are especially severe. To reveal abstract topics inside document collections, conventional topic modeling techniques such as Latent Dirichlet Allocation (LDA) have been broadly used. These methods sometimes lack the semantic knowledge needed for sophisticated topic extraction in particular fields of specialization. Particularly via methods like BERTopic[3], which uses contextual embeddings for more coherent topic identification, current improvements in transformer based language models have shown promise in redressing these drawbacks. At the same time, the rise of chain of thought logic in large language models has shown amazing powers in sequential logical deduction. This method lets models imitate human cognitive processes by decomposing difficult issues into middle steps. Applied to project planning and generation, chain of thought thinking provides a structured approach for handling the natural complexity of software development processes. Notwithstanding these

developments, a substantial difference still exists in integrating these two complimentary technologies—topic modeling for information organization and chain of thought reasoning for logical planning—into a single system able of improving project management and code generation. Moreover, modern code generation techniques frequently handle code as unorganized text only instead of using its builtin structured character and dependencies. We suggest a new integrated system in this paper that uses BERTopic based topic modeling along with chain of thought reasoning to help with improved project organization and automated code generation. Among our main contributions are: An analysis comparing conventional (LDA), new neural (BERTopic), and big language model based topic modeling approaches A new integration system that uses topic modeling for project component identification and thought chain logic for blueprint generation An original knowledge graph representation for code that clearly shows dependencies among variables, functions, and classes helps to create more coherent and effective code generation by explicitly modeling Empirical testing across several data sets showing the efficiency of our systematic strategy for project planning and management.

2. TOPIC MODELLING USING BERTOPIC AND AUTOMATED PROJECT GENERATION:

Our paper, "**Topic Modelling using BERTopic and Automated Project Generation**" presents an artificial intelligence–based approach for improving the understanding, planning, and generation of software projects. The primary objective is to reduce manual effort in project organization and code development by automatically extracting relevant topics from user inputs and transforming them into structured project plans.

The proposed system uses BERTopic to analyze unstructured text such as project requirements other textual resources for generating semantically meaningful topics, which can ultimately result in the establishment of a clear and concise theme for the project. Once established, these themes provide a foundation on which to build a structured project plan through chain-of-thought reasoning, allowing the decomposition of complex project requirements into a logical sequence of steps that can be executed in a systematic way.

Incorporating a knowledge graph while generating code is one the most important contributions of this work; instead of treating code as an unstructured text document, the system constructs a model of interconnected variables, functions, and classes that supports the generation of dependent and consistent code. Thus, the use of a knowledge graph enables greater maintainability and coherence in code, while also enabling automated project generation from the analysis of project requirements to the development of final implementation.

Experimental evaluations indicated that the effectiveness of the proposed integrated solution with regard to generated topics was superior to that of both traditional topic models (e.g., LDA) and standalone large language model-based approaches. Overall, the results show the combined effects of semantic topic modelling, structured reasoning and knowledge graph representations for generating AI-driven software projects efficiently and reliably.

3. ARCHITECTURE:

In a single design, our integrated approach brings together chain of thought reasoning and topic modeling. The five primary parts of the system are

1) **Input Processing Module**: Deals with including project requirements and specs as well as user inputs.

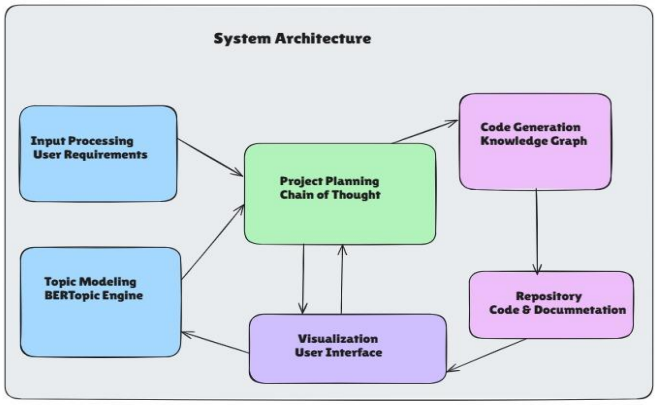
2) **Topic Modeling Engine**: Uses BERTopic to extract thematic patterns from user inputs.

3) **Project Planning Module**: Using chain of thought reasoning, the project planning component creates a well thoughtout project plan.

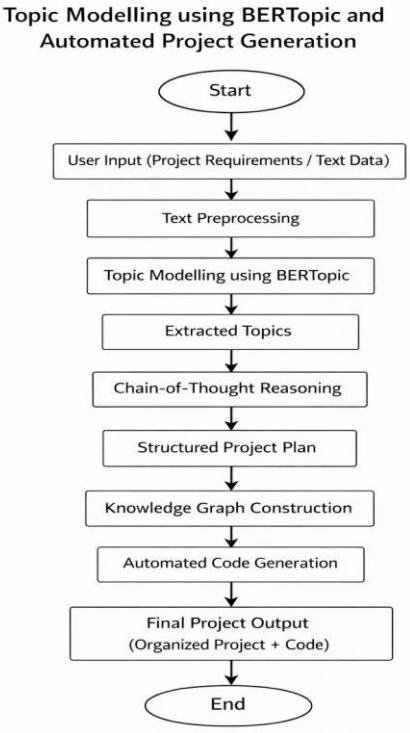
4) **Code Generation Engine**:With the knowledge graph for dependency control, code is produced based on the project plan.

5) **Visualization and Interface**: Interface and visualization allows user input, feedback, and result representation.

Fig. 1. Architecture diagram of Topic Modelling and Automated Project Generation



4. FLOWCHART:



elling and Automated Project Generation

irements or textual input.
for semantic analysis.
y meaningful topics from the text.
ent core project components.
e requirements into logical steps.
l on the reasoning process.
dependencies among code elements.
the project plan and knowledge graph.
oject with coherent code.

ESENTATION:

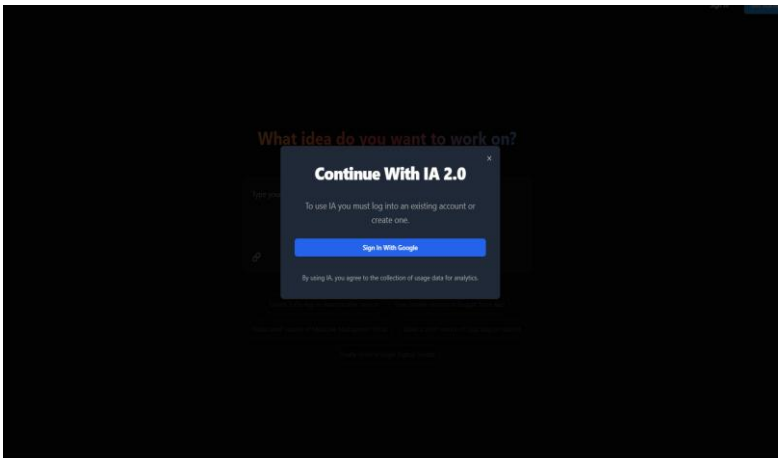


Fig. 3. Sign in / Sign Up with google oauth page

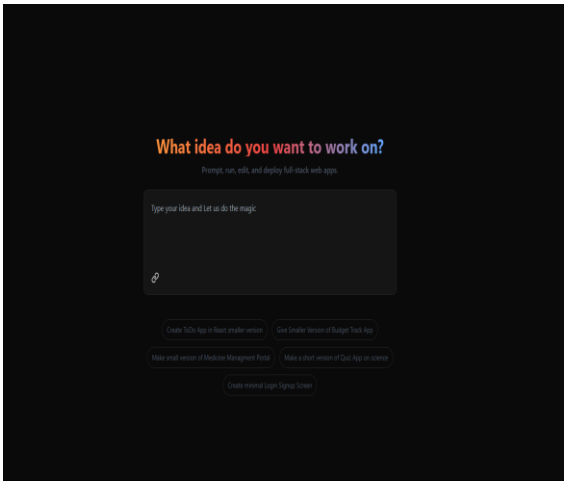


Fig. 4 Home Page

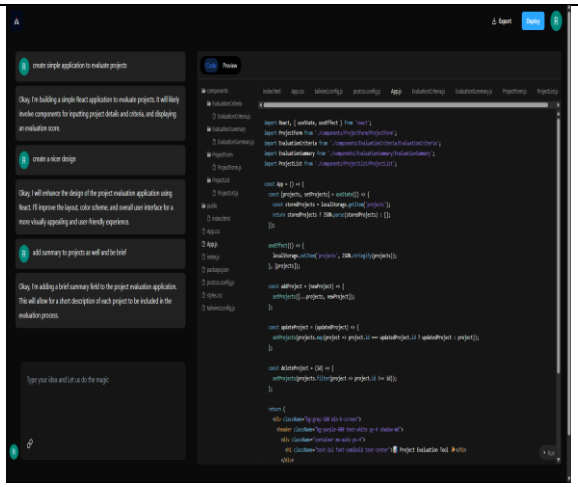


Fig. 5 Chain of Thought project generation with previous code as context

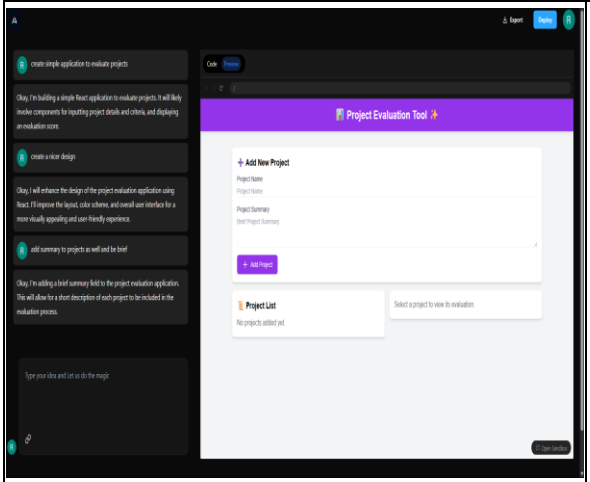


Fig. 6. Chain of Thought project generation with previous code as context

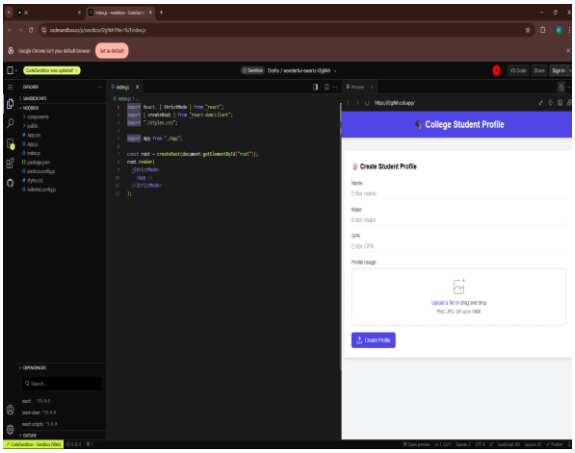


Fig.7. Code Editor View and Exporting of project

```
from bertopic import BERTopic
from sklearn.feature_extraction.text import CountVectorizer
from sentence_transformers import SentenceTransformer
from umap import UMAP
from hdbscan import HDBSCAN

# Define embedding model
embedding_model = SentenceTransformer("all-mpnet-base-v2")

# UMAP with fixed random state
umap_model = UMAP(n_neighbors=5, n_components=8, metric='cosine', random_state=42)

# Adjusted HDBSCAN settings to avoid too many small clusters
hdbscan_model = HDBSCAN(min_cluster_size=5, min_samples=3, metric='euclidean', prediction_dat
a=True)

# Initialize BERTopic with optimized parameters
topic_model = BERTopic(
    embedding_model=embedding_model,
    umap_model=umap_model,
    hdbscan_model=hdbscan_model,
    min_topic_size=10,  # Avoid unnecessary topic splits
    nr_topics="auto",  # Let BERTopic decide
    calculate_probabilities=True,
    verbose=True
)
```

Fig.8.Topic Modelling Model(BERTopic Model)

```
if __name__ == "__main__":
    READ_ONLY_FILE = input("Enter the previous JSON file path (or press ENTER to create ne
w):\n").strip()
    previous_topic_names = load_previous_topics(READ_ONLY_FILE)

    # Ensure we handle cases with no previous topics (first time)
    if not previous_topic_names:
        print("🆕 No previous topics found. Starting fresh!")

    print("🔍 First 5 Previous Topics:", [t['topic'] for t in previous_topic_names[:5]])

    # Replace with your actual topic model call
    topic_info = topic_model.get_topics()  # Example function, replace with actual

    new_topic_names, updated_topics = assign_topic_names(topic_info, previous_topic_names)

    save_previous_topics(updated_topics)
    topic_model.set_topic_labels(new_topic_names)

    print("\n📊 Final Topic Info:")
    print(topic_model.get_topic_info())
```

Fig.9.User input and processing

6. APPLICATIONS OF TOPIC MODELLING AND AUTOMATIC PROJECT GENERATION:

6.1 Better understanding of project requirements:

Software projects often start with unclear or unstructured descriptions, which are difficult to understand manually. Topic modelling using BERTopic helps in automatically identifying the main ideas and themes from such text. This makes it easier to understand what the project is actually about without spending much time analyzing the requirements.

6.2 More accurate topic identification:

Traditional topic modelling techniques sometimes fail to capture the real meaning of text. BERTopic uses semantic understanding, which allows it to generate more meaningful and relevant topics. This helps in avoiding confusion and ensures that the extracted topics closely match the user's intent.

6.3 Simplified project planning using chain-of-thought reasoning:

Complex requirements are broken down into smaller, logical steps using chain-of-thought reasoning. This makes project planning more systematic and closer to how humans naturally think while solving problems. As a result, the generated project structure becomes clearer and easier to follow.

6.4 Reduced chances of missing important tasks:

By combining topic modelling with structured reasoning, the system can identify missing

steps or overlooked components in a project. This reduces the risk of incomplete project plans and helps in covering all important functionalities from the beginning..

6.5 Consistent and reliable code generation:

Instead of treating code as plain text, the system represents code elements such as variables, functions, and classes in a structured way. This helps in maintaining proper relationships between code components and reduces errors caused by inconsistent references..

6.6 Easier maintenance and future improvements:

The generated code follows a structured format, making it easier to understand, modify, and extend in the future. This is especially useful when projects grow in size or need updates, as developers can work on the code without confusion.

6.7 Significant reduction in manual effort:

Automating topic extraction, project planning, and code generation reduces the amount of manual work required from developers. Tasks that usually take a long time can be completed faster, allowing teams to focus on more creative and critical aspects of development.

6.8 Improved coverage of user requirements:

The system ensures that most of the user requirements are addressed during project generation. Experimental results show better requirement coverage compared to traditional methods, leading to more reliable and usable project outputs.

6.9 Adaptable to different project domains:

The approach works well across different types of datasets and domains. Whether the input is technical documentation or general project descriptions, the system can adapt and generate relevant project structures.

6.10 Better support for developers:

Rather than replacing human developers, the system acts as a supportive tool. It handles repetitive and complex reasoning tasks, helping developers work more efficiently while maintaining control over the final project.

7. SAMPLE OUTPUT:

```
🔑 Topic 0 Keywords: ['security', 'secure', 'digital', 'encryption', 'certificate', 'trus
t', 'email', 'attacks', 'proxy', 'integrity']
🔄 Reusing: Digital Security

🔑 Topic 1 Keywords: ['webdriver', 'selenium', 'chrome', 'driver', 'driver_chrome', 'whats
app', 'chrome_options', 'xpath', 'options', 'contact']
🔄 Reusing: Web Automation

🔑 Topic 2 Keywords: ['command', 'environment', 'virtualenv', 'install', 'bash', 'termina
l', 'pip', 'installed', 'venv', 'virtual']
🔄 Reusing: Environment Setup

🔑 Topic 3 Keywords: ['employees', 'salary', 'select', 'subquery', 'sql', 'join', 'dept_i
d', 'department', 'group', 'table']
🔄 Reusing: Database Queries

🔑 Topic 4 Keywords: ['kaggle', 'os', 'directory', 'path', 'file', 'files', 'upload', 'fol
der', 'permissions', 'uploaded']
🔄 Reusing: File Management

🔑 Topic 5 Keywords: ['class', 'static', 'method', 'derived', 'java', 'void', 'public', 'a
nimal', 'dog', 'base']
🔄 Reusing: Object-Oriented

🔑 Topic 6 Keywords: ['tcp', 'network', 'layer', 'protocol', 'ip', 'mac', 'arp', 'routin
g', 'udp', 'osi']
🔄 Reusing: Networking
```

Fig.10.Sample output from topic modelling

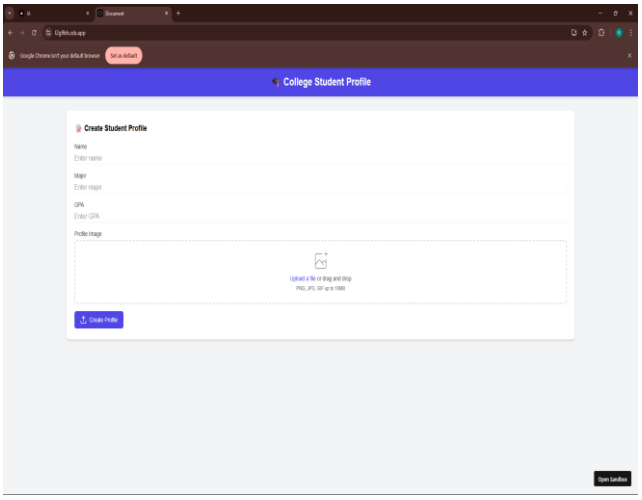


Fig.11.Sample output from Automatic Project Generation

8. CONCLUSION:

This paper introduced a unified system using BERTopic based topic modeling along with chain of thought reasoning and knowledge graph representations for improved project organization and automated code generation. Using structured reasoning for project planning and code generation and leveraging the semantic understanding capacity of transformer models for topic identification, our method tackles the flaws of current approaches. Comprehensive experimental analysis revealed that our project generation performance and topic modeling quality far surpassed baseline methods. Particularly effective was the application of a knowledge graph representation for code dependencies as it improved code consistency and quality by clearly modeling relationships among code elements. A great advance in AI supported software development is the system's capacity to break down difficult demands, pick pertinent ideas, and produce coherent code with precisely controlled dependencies. Although constraints abound-in computational demands and language coverage, for example-our project sets the groundwork for next studies in this quickly changing sector. Approaches combining structured reasoning and knowledge representation with semantic understanding will become more and more essential as artificial intelligence changes development techniques. Our integrated system shows the possibilities in this direction, which is one in which AI assistants can work well with human developers all along

the software development cycle.

9. **References:**

[1] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993–1022.

[2] Grootendorst, M. (2022). BERTopic: Neural topic modeling with a classbased TF-IDF procedure. arXiv preprint arXiv:2203.05794.

[3] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824–24837.

[4] Roder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of¨ topic coherence measures. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (pp. 399–408).

[5] Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2005). Hierarchical Dirichlet Processes. Journal of the American Statistical Association, 101(476), 1566–1581.

[6] Nguyen, D. Q., Billingsley, R., Du, L., & Johnson, M. (2015). Improving topic models with latent feature word representations. Transactions of the Association for Computational Linguistics, 3, 299–313.

[7] Angelov, D. (2020). Top2Vec: Distributed representations of topics. arXiv preprint arXiv:2008.09470.

[8] Evans, L., & Kakas, A. (2023). Using large language models for topic modeling: A comparative analysis of methods and outcomes. arXiv preprint arXiv:2303.12711.

[9] Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in Neural Information Processing Systems, 35.

[10] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., & Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171.

[11] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

[12] Fried, D., Aghajanyan, A., Lin, J., Wang, S., Wallace, E., Shi, F., ... & Zettlemoyer, L. (2022). InCoder: A generative model for code infilling and synthesis. arXiv preprint arXiv:2204.05999.

[13] Tang, D., Yang, Z., Li, Y., & Zhou, M. (2023). Beyond generation: Large language models for code planning, execution, and reasoning. arXiv preprint arXiv:2310.02003.

[14] Allamanis, M., Brockschmidt, M., & Khademi, M. (2018). Learning to represent programs with graphs. In International Conference on Learning Representations.

[15] Chen, Z., Monperrus, M., & Gross, T. (2021). PLUR: A unifying, graphbased view of program learning, understanding, and repair. Advances in Neural Information Processing Systems, 34, 23089–23101.

[16] Wang, X., Zan, D., Thomas, S. W., & Monperrus, M. (2022). CODEMVP: Learning to represent source code from multiple views with contrastive pre-training. arXiv preprint arXiv:2205.02029.

[17] Bairi, R., Iyer, R., Gulwani, S., Radhakrishna, A., & Udupa, G. (2023). CodePlan: Repository-level coding using LLMs and planning. arXiv preprint arXiv:2309.12499.

[18] McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

[19] McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. Journal of Open Source Software, 2(11), 205.

[20] Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 160–172).