23rd Annual Conference on Systems Engineering Research
Intelligent Digital Twin-enabled Systems Engineering for 21st Century Sociotechnical Systems

Editors: Dr. Mathew Amissah[1], Dr. Abbas Zaidi[1], Dr. Ali K. Raz[1], Dr. John Shortle[1], Dr. Azad M. Madni[2], Dr. Dinesh Verma[3],
[1]George Mason University,[2]University of Southern California, [3]Stevens Institute of Technology,

Arlington, VA, USA April 6-9, 2026

Exploring Similarity Assessments of LLM-generated Requirements

Ramón María García Alarcia[a,b], Alessandro Golkar[a], Alejandro Salado[b]
[a]Department of Aerospace and Geodesy, Technical University of Munich, ramon.garcia-alarcia@tum.de
[b]Department of Systems and Industrial Engineering, The University of Arizona, alejandrosalado@arizona.edu

Abstract

A potential application of LLMs in SE is the production of sets of requirements for a particular problem. In this paper, we survey how researchers evaluate the quality of requirements generated by LLMs and assess the adequacy of such approaches. Particularly, we show that metrics traditionally employed to evaluate the similarity of LLM-generated artifacts against ground truths do not perform well for requirements. The underlying reason is that similarity ought not to be achieved on the textual output but on the problem space captured by the requirements.

Keywords: requirements; large language models (LLM); CoPilot; semantic similarity; cosine similarity

Nomenclature

R	requirement, expressed in natural language
S	set of technical solutions that are associated with a requirement
E	vector embedding of a requirement

1. Introduction

The emergence of Large Language Models (LLMs) is reshaping the way in which activities are conducted in a wide variety of industries, and Systems Engineering (SE) is not an exception. The literature in the last couple of years shows significant interest in using LLMs for SE, including Requirements Engineering (RE) [1] [2] [3], and tool vendors have begun to integrate LLMs for RE and system modelling in their products and services [4] [5].

The applications of LLMs for RE are varied and range from the creation or transformation of requirements (e.g., elicitation from stakeholders, retrieval from past documentation, structuration in documents or models) [6], to their analysis (classification, prioritization, tracing to artifacts, relation extraction, quality evaluation) [7] and verification [8]. Due to the stochastic nature of LLMs, the lack of decision explainability, possibility of hallucination, inability to deal with numbers properly, and a lower knowledge base for underrepresented technical fields [9], understanding the accuracy (the goodness) of the requirements generated by LLMs seems important.

The valuation of SE artifacts generated by LLMs often relies on classical metrics from the Natural Language Processing (NLP) field, particularly on the usage of semantic similarity metrics with text embeddings [10]. These metrics establish similarity between texts in a numerical scale, typically from 0 or

-1 (indicating no similarity) to 1 (indicating maximum similarity). If a ground truth or a gold standard text exists, the model-generated text can be compared to the standard and scored numerically with these metrics. The appeal of semantic similarity metrics lies in their conceptual simplicity and their ready-to-use implementation in popular Machine Learning (ML) and NLP code libraries. This is likely why they have become one of the go-to methods for evaluation of LLM-produced text.

However, we argue that the goal when producing requirements is not to achieve textual similarity but similarity in the problem spaces that requirements capture [11] [12]. That is, two sets of requirements R_1 and R_2 are similar when they produce, respectively, equivalent spaces of solutions S_1 and S_2; that is, every solution in S1 exists in S2 and vice versa. We suspect that this interpretation of similarity is not respected by semantic similarity metrics and, therefore, using such metrics to evaluate the performance of an LLM to provide requirements does not yield an accurate evaluation of the LLM's performance.

To investigate this, we performed an experiment in which, for given sets of requirements that had known similarity or lack thereof in the problem spaces they capture, we evaluated the level of similarity assessed by a widely used semantic metric.

2. Background

2.1. Quality or accuracy measurement of requirements

The literature reveals three main approaches to evaluating requirements generated by LLMs (ref. Table 1): (1) (human) expert evaluations, (2) lexical similarity metrics, and (3) semantic similarity metrics. Approaches that leverage similarity metrics are automated and require a gold standard or ground truth requirement to which the evaluated requirement is compared.

Reference	Category	Description	Evaluation method
[13]	Manual	Human-AI methodology to generate new system requirements from a set of initial requirements, leveraging GPT-1 and GPT-2	Human, assessing requirements as relevant or not relevant
[14]	Manual	Comparing the quality of requirements created by ChatGPT to those created by human experts	Human, 5 experts assessing requirements in six qualities from 0 to 10
[15]	Manual	Framework leveraging ChatGPT to produce stakeholder, functional, and quality requirements; and as system specifications	Human, expert evaluation on binary quality questions
[16]	Manual	Pipeline of LLMs (GPT API) and prompts to generate safety requirements for autonomous driving from a list of updated items	Human, experts answering 10 binary questions
[17]	Manual	Generating a Software Requirements Specification with a GPT-4 LLM	Human, grading 8 different qualities from 0 to 5
[18]	Manual	Agentic framework with different LLMs for Requirements Engineering tasks including elicitation and specification	Human, 3 master's students score 3 qualities from 0 (bad) to 2 (good)
[19]	Automated & Manual	Summarizing requirements from contractual obligations with GPT-3, including model fine-tuning	Lexical similarity metrics: ROUGE Human requirement grading on 5 metrics from 1 to 5
[20]	Automated	Recommending the next token to complete a requirement description, leveraging GPT-2	Lexical similarity metrics: ROUGE
[21]	Automated	Integrating generic systems engineering models with LLMs to improve natural language requirements generation	**Semantic similarity metrics: cosine similarity with word embeddings**
[22]	Automated	Identifying conflicts between requirements in System Requirements Specifications, using sentence embeddings	**Semantic similarity metrics: cosine similarity with BERT and TFIDF embeddings**

[23]	Automated	Using process models and LLMs as inputs to the requirements engineering process	**Semantic similarity metrics: cosine similarity with BERT embeddings**
[24]	Automated	Framework to integrate semantic textual similarity with requirements and system models	**Semantic similarity metrics: cosine similarity with all-MiniLM-L6-v2 embeddings**
[25]	Automated	Using LLMs and system models to generate System Requirements Specifications, Product Breakdown Structures and Interface Control Documents	**Semantic similarity metrics: cosine similarity with word embeddings**

Table 1: Review of the literature: works about AI-based requirement creation or transformation, and methods used for evaluation.

Human expert evaluations pose some threats to validity, as research has shown variability in the opinion of experts when evaluating SE artifacts, often away from the ground truth [26] [27].

Lexical similarity metrics consider the frequency and sometimes the position with which words appear in two different sentences [28] [29]. These metrics are deemed at evaluating syntax correctness, looking at tokens and their expected arrangement, and they may only capture semantics or text meaning incidentally.

Semantic similarity metrics assess how closely two texts convey the same meaning. According to existing literature, researchers typically transform text into vector embeddings and calculate cosine similarity [30]. In this paper, we examine the potential limitations of using semantic similarity metrics for comparing requirements (i.e., those generated by large language models (LLMs) versus gold standard requirements). These limitations have not been addressed in the literature yet.

2.2. Semantic similarity measurements

To assess the semantic similarity between two texts (in our case, requirements), modern NLP practice compares their vector embeddings [31]. These embeddings are multi-dimensional vectors of float values that encode the meaning of a word or a sentence in natural language. For sentences, they are commonly generated using a transformer-based neural network, which has been trained to identify the semantic relations of words given their contextual position in the sentence. Given a system requirement of an arbitrary length expressed in English, R, this method produces a vector embedding of fixed size n, $\overrightarrow{E_R} = [e_0, \dots, e_n]$, which is an output of the selected transformer neural network, $\overrightarrow{E_R} = model(R)$. Typically, vector embeddings are compared using similarity or distance metrics such as cosine similarity, cosine distance, or Euclidean distance [31] [32].

3. Methodology

3.1. Experiment design

We have created 7 sets of requirements that represent identical problem spaces (albeit with different formulation), completely different problem spaces (albeit with very similar formulations), and slightly different problem spaces (with slightly different formulations). The requirements are simple enough that determining the differences in the problem space within each set is straightforward. The semantic similarity of each set is determined using cosine similarity and the result compared against the known differences in the problem spaces they capture.

Seven cases are studied, each case mapping one of the sets of requirements, as shown in Table 2, which also lists the ground truth ("real") similarity of the corresponding problem spaces.

Case	Requirement set	Description of the problem spaces comparison	"Real" similarity
1	R3. The system shall have a power consumption lower than 100 W. R4. The system shall consume less than 100 W.	Different formulations that capture the same problem space.	1 (Total)
2	R1. The system shall have a mass lower than 100 kg.	Slightly different formulations (only units of measurement) that capture the same	1 (Total)

	R2. The system shall have a mass lower than 220.46 lb.	problem space.	
3	R5. The system shall have a mass lower than 220.46 lb. R6. The system shall have a mass lower than 220,46 lb.	Slightly different formulations (only notation formalism) that capture the same problem space.	1 (Total)
4	R1. The system shall have a mass lower than 100 kg. R7. The system shall have a mass lower than 100 kilograms.	Slightly different formulations (only changes using an abbreviated vs the complete unit of measurement) that capture the same problem space.	1 (Total)
5	R1. The system shall have a mass lower than 100 kg. R2. The system shall have a mass higher than 100 kg.	Slightly different formulations that capture disjoint problem spaces.	-1 (None)
6	R1. The system shall have a mass lower than 100 kg. R3. The system shall have a power consumption lower than 100 W.	Different formulations representing different problem spaces. (While the problem spaces are different, they may lead to non-disjoint solution spaces.)	Low
7	R1. The system shall have a mass lower than 100 kg. R8. The system shall have a mass lower than 10 kg	Slightly different formulations, where one of the problem spaces encompasses the other. Here, fulfilling R8 implies fulfilling R1, but not vice versa. The problem space in R8 is somewhat resembled in R1, but not vice versa.	R1 to R8: Higher R8 to R1: Lower

Table 2: Cases under study.

3.2. Sentence transformer

We employed the *all-MiniLM-L6-v2* sentence transformer [33], based on the BERT model, and trained on generic corpora of Internet text ranging from Wikipedia to Stack Exchange pages. We have taken this choice based on the popularity of the model, highly downloaded in Hugging Face [33], and prior use in the literature [24].

3.3. Pair-wise similarity metric

We employed cosine similarity to assess semantic similarity because of its high popularity the literature [21] [22] [23] [24]. The pair-wise cosine similarity between two embeddings takes a value between -1 (complete dissimilarity) and 1 (maximum similarity). Geometrically, it captures the Euclidean distance between a pair of vectors. Its mathematical formulation is presented in Equation 2.

$$\text{cosine_similarity}(\overrightarrow{E_{R1}}, \overrightarrow{E_{R2}}) = \cos\left(\theta_{\overrightarrow{E_{R1}},\overrightarrow{E_{R2}}}\right) = \frac{\overrightarrow{E_{R1}} \cdot \overrightarrow{E_{R2}}}{\|\overrightarrow{E_{R1}}\| \|\overrightarrow{E_{R2}}\|} = \frac{\sum_{i=0}^{n} E_{R1,i} E_{R2,i}}{\sqrt{\sum_{i=0}^{n} {E_{R1,i}}^2} \sqrt{\sum_{i=0}^{n} {E_{R2,i}}^2}} \quad (2)$$

If two requirements capture the same problem space, the maximum similarity score should be obtained: 1 in this case. Conversely, if two requirements capture disjoint problem spaces, the minimum similarity score should be obtained: -1 in this case.

We have employed cosine similarity's implementation in the popular *scikit-learn* library for Python [34].

4. Results and discussion

The results are shown in Table 3. An initial look reveals two major issues semantic similarity metric. First, it is unable to identify identical problem spaces (i.e., none of the results are 1). Second, and most importantly, it is unable to identify strong differences in problem spaces (i.e., all results indicate high similarity).

Case	Requirement set	"Real" similarity	Cosine similarity
1	R3. The system shall have a power consumption lower	1 (Total)	0.8613

	than 100 W. R4. The system shall consume less than 100 W.		
2	R1. The system shall have a mass lower than 100 kg. R2. The system shall have a mass lower than 220.46 lb.	1 (Total)	0.8045
3	R5. The system shall have a mass lower than 220.46 lb. R6. The system shall have a mass lower than 220,46 lb.	1 (Total)	0.9811
4	R1. The system shall have a mass lower than 100 kg. R7. The system shall have a mass lower than 100 kilograms.	1 (Total)	0.9801
5	R1. The system shall have a mass lower than 100 kg. R2. The system shall have a mass higher than 100 kg.	-1 (None)	0.9543
6	R1. The system shall have a mass lower than 100 kg. R3. The system shall have a power consumption lower than 100 W.	Low	0.6121
7	R1. The system shall have a mass lower than 100 kg. R8. The system shall have a mass lower than 10 kg.	R1 to R8: Higher R8 to R1: Lower	R1 to R8: 0.9056 R8 to R1: 0.9056

Table 3: Summary of the results.

The results seem to indicate that the similarity assessment is heavily influenced by generic elements, such as expletive particles or frequently repeated words like “system” and “shall”, unable to semantically reason about the underlying problem space. This is particularly visible with Case 5, which obtains one of the highest results (both requirements differ in only one word) albeit representing a set of completely dissimilar problem spaces. The only requirements sets with a higher cosine similarity are Case 3, which only differ in one character (a comma vs a point) and Case 4, which only differ in using ‘kg’ or ‘kilogram’ as the unit of measurement at the end of the requirement.

Case 7 hints at the same issue, as metric is unable to capture the directionality in the requirements, claiming that each requirement is equally similar to the other when such is not the case.

Cases 1, 2, and 6 further support this pattern, as the cosine similarity result seems to be inversely correlated with the number of differing words.

Under these assumptions, it is curious, however, that Case 7 reports a cosine similarity below that of Cases 4 or 5, given that Case 7’s requirements only differ in one number (10 vs 100). We do not have an explanation for this result yet.

6. Conclusion and limitations

In this paper, we have built a simple experiment to demonstrate that pair-wise similarity metrics with generic text embeddings do not properly capture requirements similarity. Using seven sets of requirements expressed in natural language, we have shown that using pair-wise cosine similarity with a well-known generic embedding model fails at trivial cases of requirements expressing identical or disjoint problem spaces.

Given the poor performance on the selected trivial cases, we argue that these metrics are ineffective for proper requirements comparison. This should caution against using this type of similarity metric to evaluate LLM-generated system requirements, or, potentially, SE artifacts at large.

This work is the result of an initial exploration, and as such, it comes with a series of limitations. When defining the experimental cases, we have not aimed at being exhaustive. Additional cases and, particularly, larger sets of requirements, could enrich the data and evaluation.

Additionally, we have used a generic text embedding model to create the vector embeddings from the requirement texts. We are aware that there has been work regarding the creation of specific sentence embedding models for software engineering to assess requirements similarity [35]. These embedding

models have not been assessed due to not being publicly available. However, domain-specific embedding models alone are unlikely to solve the fundamental issue behind the inability of effectively capturing requirements similarity when understood from the viewpoint of problem space similarity. Indeed, research depicts an accuracy increase of 10% compared to generic embedding models [35], which, when put into the context of our study's results, hardly change the overall picture.

In this paper, we have only used pair-wise cosine similarity. While this is a popular choice, other similarity metrics with word embeddings exist. Testing a range of these metrics could yield insights on potential behavioural differences among them, as well as hinting at overperformances or underperformances of certain semantic similarity metrics. Yet, we also feel that the overall result will remain unchanged.

Given the poor performance of the similarity metric shown in this work, there are least hints that such a metric alone cannot be effective. We suggest that a novel methodology is necessary to adequately determine requirements similarity.

References

[1] M. A. Zadenoori, J. Dąbrowski, W. Alhoshan, L. Zhao, and A. Ferrari, "Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review," Sept. 14, 2025, *arXiv*: arXiv:2509.11446. doi: 10.48550/arXiv.2509.11446.

[2] P. Vasudevan and S. Reddivari, "The Role of Generative AI Models in Requirements Engineering: A Systematic Literature Review," in *Proceedings of the 2025 ACM Southeast Conference*, Southeast Missouri State University Cape Girardeau MO USA: ACM, Apr. 2025, pp. 188–194. doi: 10.1145/3696673.3723053.

[3] H. Cheng *et al.*, "Generative AI for Requirements Engineering: A Systematic Literature Review," Jan. 23, 2025, *arXiv*: arXiv:2409.06741. doi: 10.48550/arXiv.2409.06741.

[4] "AIxValispace – Valispace." Accessed: Aug. 30, 2025. [Online]. Available: https://www.valispace.com/ai/

[5] L. Kröll, "Generative AI in Requirements Engineering," microTOOL. Accessed: Oct. 08, 2025. [Online]. Available: https://www.microtool.de/en/using-objectif-rpm/generative-ki-in-requirements-engineering/

[6] A. Ferrari, S. Abualhaija, and C. Arora, "Model Generation with LLMs: From Requirements to UML Sequence Diagrams," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, June 2024, pp. 291–300. doi: 10.1109/REW61692.2024.00044.

[7] S. Lubos *et al.*, "Leveraging LLMs for the Quality Assurance of Software Requirements," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, June 2024, pp. 389–397. doi: 10.1109/RE59067.2024.00046.

[8] J. O. Couder, D. Gomez, and O. Ochoa, "Requirements Verification Through the Analysis of Source Code by Large Language Models," in *SoutheastCon 2024*, Mar. 2024, pp. 75–80. doi: 10.1109/SoutheastCon52093.2024.10500073.

[9] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and Applications of Large Language Models," July 19, 2023, *arXiv*: arXiv:2307.10169. doi: 10.48550/arXiv.2307.10169.

[10] D. Chandrasekaran and V. Mago, "Evolution of Semantic Similarity—A Survey," *ACM Comput. Surv.*, vol. 54, no. 2, p. 41:1-41:37, Feb. 2021, doi: 10.1145/3440755.

[11] A. Salado, R. Nilchiani, and D. Verma, "A contribution to the scientific foundations of systems engineering: Solution spaces and requirements," *J. Syst. Sci. Syst. Eng.*, vol. 26, no. 5, pp. 549–589, Oct. 2017, doi: 10.1007/s11518-016-5315-3.

[12] P. Wach and A. Salado, "The Need for Semantic Extension of SysML to Model the Problem Space," in *Recent Trends and Advances in Model Based Systems Engineering*, A. M. Madni, B. Boehm, D. Erwin, M. Moghaddam, M. Sievers, and M. Wheaton, Eds., Cham: Springer International Publishing, 2022, pp. 279–289. doi: 10.1007/978-3-030-82083-1_24.

[13] J. Peer, Y. Mordecai, and Y. Reich, "Self-Generative Requirements Engineering," Oct. 2025.

[14] K. Ronanki, C. Berger, and J. Horkoff, "Investigating ChatGPT's Potential to Assist in Requirements Elicitation Processes," July 14, 2023, *arXiv*: arXiv:2307.07381. doi: 10.48550/arXiv.2307.07381.

[15] M. Binder and V. Mezhuyev, "A framework for creating an IoT system specification with ChatGPT," *Internet of Things*, vol. 27, p. 101218, Oct. 2024, doi: 10.1016/j.iot.2024.101218.

[16] A. Nouri, B. Cabrero-Daniel, F. Törner, H. Sivencrona, and C. Berger, "Engineering Safety Requirements for Autonomous Driving with Large Language Models," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, June 2024, pp. 218–228. doi: 10.1109/RE59067.2024.00029.

[17] M. Krishna, B. Gaur, A. Verma, and P. Jalote, "Using LLMs in Software Requirements Specifications: An Empirical Evaluation," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, June 2024, pp. 475–483. doi: 10.1109/RE59067.2024.00056.

[18] D. Jin, Z. Jin, X. Chen, and C. Wang, "MARE: Multi-Agents Collaboration Framework for Requirements Engineering," May 06, 2024, *arXiv*: arXiv:2405.03256. doi: 10.48550/arXiv.2405.03256.

[19] X. Lian, J. Ma, H. Lv, and L. Zhang, "ReqCompletion: Domain-Enhanced Automatic Completion for Software Requirements," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, June 2024, pp. 142–154. doi: 10.1109/RE59067.2024.00023.

[20] C. Jain, P. R. Anish, A. Singh, and S. Ghaisas, "A Transformer-based Approach for Abstractive Summarization of Requirements from Obligations in Software Engineering Contracts," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*, Sept. 2023, pp. 169–179. doi: 10.1109/RE57278.2023.00025.

[21] R. M. García Alarcia, P. Russo, A. Renga, and A. Golkar, "Bringing Systems Engineering Models to Large Language Models: An Integration of OPM with an LLM for Design Assistants:," in *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering*, Rome, Italy: SCITEPRESS - Science and Technology Publications, 2024, pp. 334–345. doi: 10.5220/0012621900003645.

[22] G. Malik, M. Cevik, D. Parikh, and A. Basar, "Identifying the requirement conflicts in SRS documents using transformer-based

sentence embeddings," June 28, 2022, *arXiv*: arXiv:2206.13690. doi: 10.48550/arXiv.2206.13690.
[23] N. Klievtsova, J. Mangler, T. Kampik, and S. Rinderle-Ma, "Utilizing Process Models in the Requirements Engineering Process Through Model2Text Transformation," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, June 2024, pp. 205–217. doi: 10.1109/RE59067.2024.00028.
[24] J. R. Beilstein, "A Framework for Semantic Textual Similarity Integration with Requirements and System Models," Thesis, Massachusetts Institute of Technology, 2024. Accessed: Oct. 13, 2025. [Online]. Available: https://dspace.mit.edu/handle/1721.1/153990
[25] R. M. García Alarcia and A. Golkar, "Generative AI for the Design of Complex Systems: Roadmap and Space Mission Case Study," *Procedia Computer Science*, vol. 268, pp. 348–356, Jan. 2025, doi: 10.1016/j.procs.2025.08.213.
[26] B. Cratsley, S. Regmi, P. Wach, and A. Salado, "Interpretation Discrepancies of SysML State Machine: An Initial Investigation," in *Recent Trends and Advances in Model Based Systems Engineering*, A. M. Madni, B. Boehm, D. Erwin, M. Moghaddam, M. Sievers, and M. Wheaton, Eds., Cham: Springer International Publishing, 2022, pp. 361–370. doi: 10.1007/978-3-030-82083-1_31.
[27] A. Salado and N. Shadab, "A comparative experiment between textual requirements and model-based requirements on proxies for contractual safety," *Systems Engineering*, vol. 27, no. 3, pp. 556–569, 2024, doi: 10.1002/sys.21738.
[28] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 311. doi: 10.3115/1073083.1073135.
[29] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries".
[30] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3980–3990. doi: 10.18653/v1/D19-1410.
[31] P. Sitikhu, K. Pahi, P. Thapa, and S. Shakya, "A Comparison of Semantic Similarity Methods for Maximum Human Interpretability," in *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, Nov. 2019, pp. 1–4. doi: 10.1109/AITB48515.2019.8947433.
[32] Gij&#243, M. n Agudo, A. Vilalta Arias, and D. Garcia-Gasulla, "Analyzing Distances in Word Embeddings and Their Relation with Seme Analysis," in *Artificial Intelligence Research and Development*, IOS Press, 2019, pp. 407–416. doi: 10.3233/FAIA190153.
[33] "sentence-transformers/all-MiniLM-L6-v2 · Hugging Face." Accessed: Oct. 13, 2025. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[34] "cosine_similarity," scikit-learn. Accessed: Oct. 13, 2025. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
[35] S. Das, N. Deb, A. Cortesi, and N. Chaki, "Sentence Embedding Models for Similarity Detection of Software Requirements," *SN COMPUT. SCI.*, vol. 2, no. 2, p. 69, Feb. 2021, doi: 10.1007/s42979-020-00427-1.