Exploring Problem Statements and Source Code Submissions in Online Judges via 2D Embedding Projections

Vinicius R. P. Borges[1,3], Maria Eduarda M. de Holanda[1], Vladimir Molchanov[2,3], Maria Cristina F. Oliveira[4], Lars Linsen[3]

[1] University of Brasília, Brazil
viniciusrpb@unb.br,eduarda.holanda@aluno.unb.br
[2] Leibniz FH, University of Applied Sciences, Hanover, Germany
vladimir.molchanov@leibniz-fh.de
[3] University of Münster, Germany
linsen@uni-muenster.de
[4] University of São Paulo, Brazil
cristina@icmc.usp.br

Abstract. Online Judges (OJs) are widely used as supporting tools in computer programming courses. These platforms store large collections of problem statements and student code submissions, offering rich opportunities for analyzing behaviors and performance patterns. However, their built-in analytics capabilities are limited, making systematic data exploration difficult. In this paper, we investigate how embeddings of problem statements and student source code, combined with multidimensional projection techniques, can support exploratory analysis of student activity. We evaluate multiple embedding models for Portuguese problem statements, as well as for source code, using metrics such as Neighborhood Preservation, Trustworthiness, Spearman correlation, and Stress. Our results indicate that ModernBERT-PT and GraphCodeBERT are the most suitable embeddings for Portuguese statements and source code, respectively. Regarding projection methods, t-SNE outperformed UMAP and TriMap in preserving local structure. Two case studies illustrate the practical value of the approach, showing how projections reveal groups of students with similar topic preferences and highlight potentially suspicious submissions.

CCS Concepts: • **Computing methodologies** → **Natural language processing**.

Keywords: visual data mining, embeddings, multidimensional projection, online judges, educational data mining

1. INTRODUCTION

Online Judges (OJs) are widely used in programming courses to support the teaching and assessment of problem-solving and coding skills. They store large collections of programming problems and automatically evaluate student submissions, generating rich data about learning behaviors. Although valuable, these platforms offer limited analytical capabilities, making it difficult for instructors to systematically explore student performance and problem-solving patterns.

Educational Data Mining (EDM) research has proposed several approaches for OJ data, often relying on supervised learning or Natural Language Processing (NLP) to classify problems by topic or difficulty, or on clustering methods for unlabeled datasets. However, instructors frequently seek to understand how students approach problems—identifying strategies, similarities, and anomalies—rather than obtaining predictive models alone. This requires analyzing heterogeneous data, including problem statements, source code, difficulty levels, topic tags, and verdict outcomes.

Exploratory visualization with multidimensional projections is a promising alternative, as it en-

ables instructors to investigate patterns and relationships in the data without relying on predefined labels [Keim et al. 2008]. Multidimensional projection techniques support this process by mapping high-dimensional representations into low-dimensional visual spaces while preserving neighborhood relationships, to an extent [Nonato and Aupetit 2018]. In programming education, such projections can reveal groups of students with similar programming styles, clusters of related problems, and potentially suspicious submissions.

The effectiveness of projections in preserving structures from high-dimensional manifolds depends on the underlying data representation. For text data, earlier approaches based on TF–IDF, Bag-of-Words, or Abstract Syntax Trees capture mostly surface-level features and often fail to model the semantic meaning of problem statements or the functional intent of source code. Neural embedding models address these limitations by producing dense vector representations that encode richer semantic and contextual information.

Despite the increasing use of embeddings in EDM and NLP, little is known about how the choice of embedding affects the quality of multidimensional projections used for visual exploration of data from OJ environments. This gap is particularly relevant for course instructors and competition coaches, who could rely on visual analytics to monitor progress, identify difficulties, compare strategies, and detect anomalous behavior.

In this work, we investigate the suitability of modern sentence embedding models for generating multidimensional projections of programming problems and source-code submissions. We evaluate embeddings for English and Portuguese problem statements, as well as specialized code embeddings, combined with projection techniques such as t-SNE, UMAP, and TriMap. Since preserving the local structure of the data in the 2D embedding is relevant for tasks such as identifying similar student programming styles and detecting suspiciously similar code submissions, we use multiple neighborhood preservation metrics to assess projection quality and illustrate the practical utility of the approach through two case studies.

Our main contribution is a systematic evaluation of text and source code embeddings for analyzing data stored in OJ environments based on multidimensional projections, bridging representation learning and visual analytics for programming education. We address the following research questions:

—To what extent do different embedding models preserve meaningful relationships among programming problems and source code submissions?
—How do embedding choices impact clustering quality and visual organization in dimensionality-reduced spaces?

The remainder of this paper is organized as follows. Section 2 reviews related work on EDM and embedding-based representations for programming problem statements and source code. Section 3 describes the proposed methodology for comparing embedding approaches and multidimensional projections. Section 4 presents the experimental setup, discusses the results, and presents two illustrative case studies. Finally, Section 5 concludes the paper and outlines directions for future work.

2. RELATED WORK

Analysis of programming problem statements and source code is a long-standing topic in Educational Data Mining (EDM). Recent advances in Large Language Models (LLMs) have enabled new approaches for text and code analysis, though most methods still operate on a single modality rather than jointly modeling both.

A growing line of work explores multimodal or joint representations. Wang et al. [Wang et al. 2024] proposed a multimodal framework for estimating problem difficulty by combining pre-trained encoders for textual descriptions and example code solutions. Their results showed that C-BERT outperformed

unimodal BERT and CodeBERT, achieving an AUC of 87.25 on a Codeforces dataset. Yoshida et al. [Yoshida et al. 2024] fine-tuned GPT-3.5 Turbo using problem descriptions and example solutions to predict difficulty ratings; interestingly, fine-tuning on problem descriptions alone yielded the best performance, suggesting that combining modalities does not always improve results. Kim et al. [Kim et al. 2024] introduced a transformer-based feature extractor with dual classification heads to jointly predict algorithm tags and difficulty levels, also using Codeforces data.

Beyond machine-learning-based prediction tasks, visual exploration approaches have leveraged structured or embedding-based representations to support knowledge discovery in programming data. Early work used Abstract Syntax Trees (ASTs) to visualize code evolution over time [Feist et al. 2016]. More recent efforts rely on neural embeddings, e.g., Carissimi et al. [Carissimi et al. 2025] applied BERTopic to summaries of Python functions generated by Gemma2 2B-it and visualized topic clusters using UMAP. Mosquera and Bojorque [Mosquera and Bojorque 2026] compared embedding models for detecting code smells such as Long Method, God Class, and Feature Envy, projecting high-dimensional embeddings with UMAP, PCA, and t-SNE. Their findings indicate that CodeBERT produces more structured and separable latent spaces than OpenAI Embeddings.

To the best of our knowledge, no prior work has systematically compared dense representations of both programming problem statements and source code to generate multidimensional projections aimed at visual exploration. This gap is particularly relevant for understanding students' programming styles, topic preferences, and submission behaviors. In this work, we evaluate modern embedding models for textual problem descriptions and source code, assessing their suitability for visual analytics through projection quality metrics that measure neighborhood preservation and projection fidelity.

3. METHODOLOGY

3.1 Corpora Selection

We selected two corpora of problem statements and source codes, all from OJs. The first is a publicly available corpus of problem statements in Portuguese from the Beecrowd OJ curated by [Silvestre et al. 2024]. It includes 3,654 problems with difficulty scores in the range [0, 10] and originally labeled by problems' authors according to 9 topic categories: *Beginner*, *Data Structures*, *Graphs*, *Paradigms*, *Computational Geometry*, *Strings*, *Ad-Hoc*, *Math*, and *No-Category*.

For source code, we randomly subsampled the Codeforces Submissions dataset[1], restricting the sample to five programming languages (Python, C, C++, JavaScript, and Java) and constraining the number of submissions per problem to 100 to avoid over-representation. This resulted in a corpus of 10,000 code submissions, with categorical labels for the final verdict (*accepted* (ok), *wrong answer*, *time limit exceeded*, *memory limit exceeded*, and *runtime error*) and programming language.

3.2 Sentence Embeddings Setting

For problem statements, we evaluated models specifically pre-trained or fine-tuned on Portuguese text, namely `neuralmind/bert-base-portuguese-cased` (BERTimbau) [Souza et al. 2020], a BERT model pre-trained on a large Portuguese corpus comprising Common Crawl and Wikipedia; and `juridics/bertimbau-base-portuguese-sts-scale`, which extends BERTimbau with fine-tuning on Portuguese semantic textual similarity pairs, making it more suitable for sentence-level comparisons. We also included `intfloat/multilingual-e5-large` [Wang et al. 2022], a multilingual contrastive model covering over 100 languages, which serves as a cross-lingual baseline.

The following sentence embeddings were considered for source code based on their prior use in the literature (Section 2). CodeBERT [Feng et al. 2020] relies more heavily on textual information,

[1] `https://huggingface.co/datasets/open-r1/codeforces-submissions`

including identifiers and comments; GraphCodeBERT [Guo et al. 2020] explicitly incorporates data-flow relationships to capture structural dependencies within the program; CodeT5+ [Wang et al. 2023] learns semantic representations through a variety of generative code-related tasks, enabling it to model higher-level program semantics; and Jina v2 Code [Günther et al. 2023] is directly optimized for semantic similarity and code retrieval tasks.

3.3 Selection of Multidimensional Projections

We consider three dimensionality-reduction techniques designed to preserve local structures in high-dimensional data: Uniform Manifold Approximation and Projection (UMAP) [McInnes et al. 2018], t-Distributed Stochastic Neighbor Embedding (t-SNE) [Van der Maaten and Hinton 2008], and TriMap [Amid and Warmuth 2019]. These methods are well suited to our tasks because their underlying strategy is to map semantically or structurally related instances to nearby points in the projected space, allowing the observation of clusters and other local patterns. All projections consider the cosine distance to compare embeddings.

4. EXPERIMENTS AND RESULTS

We conducted experiments to evaluate how different embedding models represent programming problem statements and source code submissions, in order to identify the most suitable configurations for visual exploration through multidimensional projections. All experiments were conducted using Python 3.10.12 on a Ubuntu 22.04 operating system, running on an RTX 3060 GPU (12GB) machine with CUDA version 12.0. All embedding models are applied with no input pre-processing or fine-tuning. The source code and the corpora used are available in a public repository[2].

We assess projection quality using metrics that quantify how well local neighborhood structures are preserved in the 2D embedding, directly reflecting tasks such as identifying similar problems and detecting shared coding patterns [Nonato and Aupetit 2018]. *Neighborhood Preservation* (NP) and *Trustworthiness* assess how well a projection retains local relationships; both metrics were computed across a range of neighborhood sizes $k \in \{1, 30\}$, and we report the resulting average and standard deviation. *Spearman correlation* measures the preservation of distance rankings, while *Stress* quantifies the global geometric distortion between the original and projected spaces.

We performed initial experiments for each corpus described in Subsection 3.1 to optimize the hyperparameters of each multidimensional projection technique via an exhaustive grid search, executed independently for each embedding/projection combination. For UMAP, we varied the number of neighbors and the minimum distance; for t-SNE, the perplexity and learning rate; for TriMap, the number of inliers, outliers, and learning rate. For each combination, we selected the configuration that yielded the highest Neighborhood Preservation for the comparisons. For each embedding/projection combination, the optimal hyperparameters obtained after grid search, the ranges of tested values, and the generated layouts are all available in our public repository.

Table I summarizes the results obtained with embeddings from source code submissions, in which bold indicates the best values per embedding and underline indicates the best combination for each metric. Regarding the neighborhood preservation metrics, t-SNE with GraphCodeBERT yielded the best combination across all metrics, except for Trustworthiness. The difference between the code sentence embeddings is minimal when analyzing each embedding's performance across individual projections for neighborhood preservation, whilst some variation is observed for Spearman and Stress values. This indicates that, although the embeddings are roughly equivalent at preserving local neighborhoods of similar code submissions, they diverge in how faithfully they preserve the global structure of distances between different solution strategies.

[2] `https://gitlab.com/gvic-unb/kdmile2026-progprobs-ojs-visualization`

Table I. Metrics for each embedding/visualization combination for source code submissions from Codeforces.

Embedding	Vis.	NP ↑	Trust ↑	Spearman ↑	Stress ↓
CodeBERT	t-SNE	**0.4649 ± 0.0289**	**0.9838 ± 0.0077**	**0.5688**	**0.3851**
	UMAP	0.3517 ± 0.0465	0.9677 ± 0.0107	0.3347	0.6075
	TriMap	0.2781 ± 0.0652	0.9618 ± 0.0065	0.6826	0.8938
GraphCodeBERT	t-SNE	**0.4688 ± 0.0277**	**0.9804 ± 0.0089**	**0.6963**	**0.3403**
	UMAP	0.3520 ± 0.0467	0.9591 ± 0.0149	0.4707	0.5109
	TriMap	0.2683 ± 0.0565	0.9484 ± 0.0078	0.6534	0.9042
CodeT5+ 110M	t-SNE	**0.4541 ± 0.0470**	**0.9618 ± 0.0174**	0.4504	**0.3966**
	UMAP	0.3469 ± 0.0409	0.9385 ± 0.0162	0.3492	0.5947
	TriMap	0.2571 ± 0.0520	0.9156 ± 0.0090	**0.5058**	0.8645
Jina v2 Code	t-SNE	**0.4578 ± 0.0409**	**0.9523 ± 0.0185**	0.4289	**0.3866**
	UMAP	0.3606 ± 0.0442	0.9368 ± 0.0179	0.4482	0.5360
	TriMap	0.2270 ± 0.0409	0.8647 ± 0.0143	**0.5270**	0.8856

Table II. Metrics for each embedding/visualization combination for problem statements from Beecrowd.

Embedding	Vis.	NP ↑	Trust ↑	Spearman ↑	Stress ↓
BERTimbau	t-SNE	**0.2793 ± 0.0549**	**0.9126 ± 0.0256**	0.6930	**0.3671**
	UMAP	0.1963 ± 0.0387	0.8834 ± 0.0104	0.7407	0.5430
	TriMap	0.0791 ± 0.0235	0.8788 ± 0.0025	**0.7862**	0.6157
BERTimbau-STS	t-SNE	**0.3634 ± 0.0610**	**0.9070 ± 0.0213**	0.6684	**0.3843**
	UMAP	0.2872 ± 0.0464	0.8804 ± 0.0128	**0.7173**	0.6325
	TriMap	0.2322 ± 0.0409	0.8865 ± 0.0071	0.6778	0.8207
ModernBERT-PT	t-SNE	**0.3077 ± 0.0538**	**0.9162 ± 0.0244**	0.7419	0.3675
	UMAP	0.2301 ± 0.0273	0.8955 ± 0.0127	**0.7437**	**0.3507**
	TriMap	0.0871 ± 0.0272	0.8887 ± 0.0022	0.7401	0.5159
Multilingual-E5-Large	t-SNE	**0.2646 ± 0.0788**	**0.8628 ± 0.0410**	0.5374	**0.4159**
	UMAP	0.1907 ± 0.0355	0.8287 ± 0.0160	0.6637	0.7113
	TriMap	0.1507 ± 0.0286	0.8348 ± 0.0098	**0.6837**	0.8528

Table II presents the best visualization configurations for problem statements in Portuguese. Looking at local neighborhood preservation, BERTimbau-STS combined with t-SNE achieved the best NP (0.3634), while ModernBERT-PT combined with t-SNE achieved the best Trustworthiness (0.9162). In general, t-SNE yielded higher local metric values compared with UMAP and TriMap across all four embeddings. However, when considering the Spearman correlation, BERTimbau combined with TriMap achieved the highest value overall (0.7862), with each embedding reaching its best Spearman score under a different projection method rather than consistently under t-SNE. These results also show that no single projection method is best across both local and global criteria: t-SNE is the most reliable choice for preserving neighborhoods, but TriMap and UMAP can better preserve the overall distance structure depending on the embedding. Despite this advantage in Spearman correlation, TriMap yielded the worst Stress values across all embeddings.

An interesting finding is that the average word embeddings produced by BERTimbau and ModernBERT-PT outperformed specialized sentence embedding models in most metrics, except for NP. A plausible explanation is that BERTimbau-STS and Multilingual-E5-Large were fine-tuned on short, general-domain sentences for semantic similarity tasks, making them less effective for longer, technical problem statements. Results for English problem statements from Codeforces, evaluated using the same methodology, are reported in the supplementary material available in our repository.

We visualize the best-performing t-SNE configuration for each embedding, comparing their layouts in Figure 1 using the same hyperparameters as those presented in Table I. Point colors indicate the programming language of each source code submission (C, C++, Java, Python), as this categorical variable is present in the corpus. Because C++ is the most frequent language in the corpus, its large cluster dominates the layouts, while the remaining languages form smaller, distinguishable clusters. Gaining deeper insights into these smaller clusters requires interaction mechanisms such as lasso

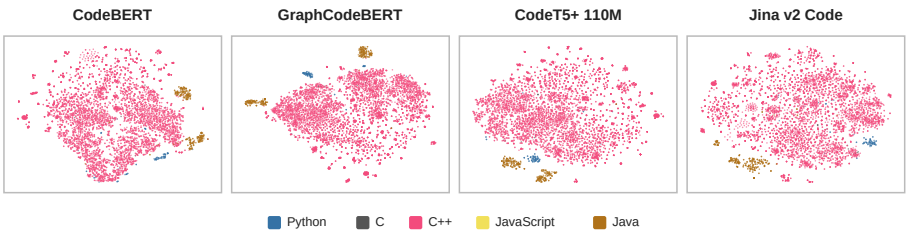


Fig. 1. t-SNE projections obtained using different embeddings for source code.

selection. A demonstration is available in our repository, showing that most of these clusters consist of submissions solving the same problem.

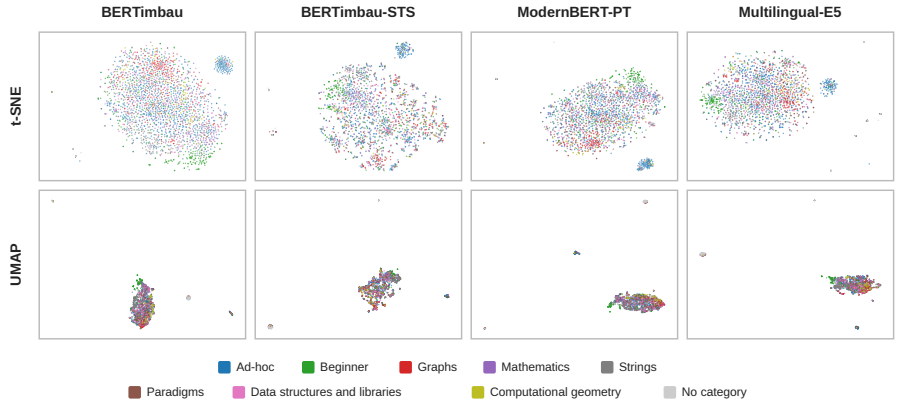


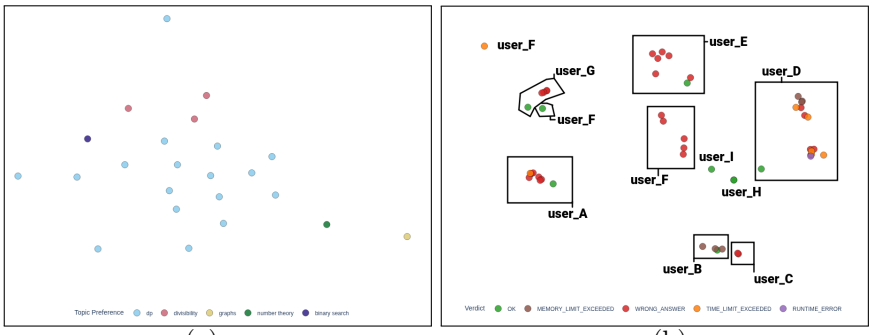
Fig. 2. t-SNE projections of multiple embeddings.

Figure 2 shows the layouts for the Portuguese sentence-embedding models applied to problem statements. Across all t-SNE layouts, we observe a dense cluster alongside a larger and more diffuse one, within which ad-hoc problems appear scattered throughout the space. This pattern can be attributed to the fact that the ad hoc categories cover both broad implementation tasks and more specific topics. Beginner-level problems tend to be more similar to one another because their statements are typically simpler, with lower mathematical complexity, a trend particularly evident in the BERTimbau-STS and ModernBERT-PT layouts. UMAP projections appear more compact and exhibit greater clutter than t-SNE, with problem categories remaining mixed within the layout as well.

We present two case studies illustrating possible applications in a real-world OJ scenario. For this purpose, we used anonymized data collected from an undergraduate Competitive Programming course, with 23 students and customized problem statements written in Portuguese. The following examples were generated using t-SNE as the projection method, with GraphCodeBERT for source code and BERTimbau-STS for problem statements. Usage of this educational data for research purposes was reviewed and approved by the Institutional Review Board (IRB) of the University of Brasília[3].

A first case study aimed to identify students with similar coding styles and topic preferences over an academic semester. Students completed five assignments and four exams, each containing approximately the same number of programming problems. For each student, following prior work [Reimers

[3]Project ID 93678925.5.0000.5540, Ruling 8.304.447



(a) (b)

Fig. 3. t-SNE visualizations: (a) Case Study I shows programming styles based on most preferred student topics; (b) Case Study II depicts submissions to a complex dynamic programming problem.

and Gurevych 2019], we computed a representative embedding by averaging the embeddings of all their submissions. Each submission embedding was obtained by concatenating the embedding of the problem statement with the embedding of the corresponding source-code solution. These combined embeddings were then projected into a two-dimensional space using t-SNE with the optimal hyperparameters identified in our experiments. Point colors indicate each student's most frequently solved problem topic throughout the semester.

The final layout is shown in Figure 3(a). We observe that most students solved problems in dynamic programming (dp), which may be related to the fact that this topic is consistently emphasized in programming contests and constitutes a major focus of the training. Students preferring math-related problems and other topics such as graphs, number theory and binary search form smaller, distinguishable sparse groups, consistent with their proximity in the layout. Since the projection preserves neighborhood relationships, nearby students share similar programming styles and topic preferences, providing useful information for coaches composing balanced and diverse competition teams.

Figure 3(b) presents a second case study, which uses only the source-code embeddings from submissions by nine students who attempted to solve a hard-level dynamic programming problem, with point colors indicating the final verdict. As expected, most points are non-green (failed submissions). The rectangles and polygons were manually added to the image to highlight the result of hovering over the corresponding points, which reveals the user identities. However, two isolated green points in the central region, corresponding to accepted submissions from user_I and user_H, drew our attention. Moreover, an isolated green submission from user_F also appears very close to a green submission from user_G. This unusual proximity between green points from different users prompted the instructor to inspect the corresponding source codes further, confirming that they were highly similar.

5. CONCLUSION

This paper evaluated sentence-transformer models for encoding problem statements and source code, combined with different multidimensional projections. The objective was to evaluate how sentence embeddings can be leveraged for tasks requiring multidimensional projections, which demands assessing how well the embeddings' local structures are preserved in the low-dimensional space.

The results showed that t-SNE consistently achieved the highest Neighborhood Preservation and Trustworthiness values for the two corpora considered. Among the evaluated embeddings, ModernBERT-PT and GraphCodeBERT produced the most suitable representations for Portuguese statements and source code, respectively. The case studies further demonstrated that the proposed approach can support the identification of students with similar programming styles and assist instructors in detecting potentially suspicious submissions that warrant closer inspection.

As future work, we plan to investigate fusion strategies between embeddings with statistical features and metadata from OJs. We also intend to incorporate temporal information to analyze student evolution throughout a course and to develop more advanced interactive visualizations integrating clustering, source code inspection, and explainability techniques.

6. ACKNOWLEDGEMENTS

The authors would like to thank *Fundação de Apoio à Pesquisa do Distrito Federal (FAPDF)* for funding this research (Grant #00193-00001398/2024-21) and the National Council for Scientific and Technological Development (CNPq) (Grant 302652/2025-6).

REFERENCES

Amid, E. and Warmuth, M. K. TriMap: Large-scale Dimensionality Reduction Using Triplets. *arXiv preprint arXiv:1910.00204*, 2019.

Carissimi, M., Saletta, M., and Ferretti, C. Towards leveraging large language model summaries for topic modeling in source code. In *Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering*. EASE '25. Association for Computing Machinery, New York, NY, USA, pp. 776–781, 2025.

Feist, M. D., Santos, E. A., Watts, I., and Hindle, A. Visualizing project evolution through abstract syntax tree analysis. In *2016 IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE, pp. 11–20, 2016.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., and Zhou, M. Codebert: A pre-trained model for programming and natural languages, 2020.

Günther, M., Ong, J., Mohr, I., Abdessalem, A., Abel, T., Akram, M. K., Guzman, S., Mastrapas, G., Sturua, S., Wang, B., et al. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *arXiv preprint arXiv:2310.19923*, 2023.

Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Svyatkovskiy, A., Fu, S., et al. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*, 2020.

Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. Visual analytics: Definition, process, and challenges. In *Information Visualization: Human-centered Issues and Perspectives*. Springer, pp. 154–175, 2008.

Kim, J., Cho, E., and Na, D. Problem-solving guide (psg): Predicting the algorithm tags and difficulty for competitive programming problems. In *AI for Education Workshop*. PMLR, pp. 48–56, 2024.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Mosquera, M. and Bojorque, R. Evaluating embedding representations for multiclass code smell detection: A comparative study of codebert and general-purpose embeddings. *Applied Sciences* 16 (8): 3622, 2026.

Nonato, L. G. and Aupetit, M. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Transactions on Visualization and Computer Graphics* 25 (8): 2650–2673, 2018.

Reimers, N. and Gurevych, I. 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)*. pp. 3982–3992, 2019.

Silvestre, A. S. S., De Souza, B. V., Lisboa, V. H. F., and Borges, V. R. P. A multi-label classification approach for categorizing beginner programming problems from online judges. In *2024 IEEE Frontiers in Education Conference (FIE)*. pp. 1–8, 2024.

Souza, F., Nogueira, R., and Lotufo, R. Bertimbau: Pretrained bert models for brazilian portuguese. In *Intelligent Systems*, R. Cerri and R. C. Prati (Eds.). Springer International Publishing, Cham, pp. 403–417, 2020.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research* 9 (11): 2579–2605, 2008.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Wang, Y., Le, H., Gotmare, A., Bui, N., Li, J., and Hoi, S. Codet5+: Open code large language models for code understanding and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. pp. 1069–1088, 2023.

Wang, Z., Zhang, W., and Wang, J. Estimating difficulty levels of programming problems with pre-trained model. *arXiv preprint arXiv:2406.08828*, 2024.

Yoshida, C., Matsushita, M., and Higo, Y. Estimating the Difficulty of Programming Problems Using Fine-tuned LLM . In *IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA)*. pp. 28–34, 2024.