维护 MTEB：面向嵌入基准的长期可用性和可重复性

Isaac Chung [*1] Imene Kerboua [*2 3] Márton Kardos [4] Roman Solomatin [5] Kenneth Enevoldsen [4]

Abstract

大规模文本嵌入基准测试（MTEB）已成为文本嵌入模型的标准评估平台。虽然先前的研究已经建立了核心基准方法论，但本文重点关注确保 MTEB 持续可重复性和扩展性的工程方面。我们介绍了保持健壮的持续集成管道的方法，这些管道验证数据集的完整性、自动化测试执行并评估基准结果的泛化能力。我们详细说明了集体增强可重复性和可用性的设计选择。此外，我们讨论了处理社区贡献以及通过新任务和数据集扩展基准的战略。这些工程实践对于将 MTEB 扩展为更全面的平台至关重要，同时保持质量，并最终与领域相关性。我们的经验为面临确保机器学习评估框架中可重复性和可用性类似挑战的基准维护者提供了宝贵的见解。

MTEB 存储库可在以下位置获取：https://github.com/embeddings-benchmark/mteb

1. 介绍

基准测试对于指导机器学习的进步至关重要，特别是在快速发展的嵌入和表示学习领域。MTEB(Muennighoff et al., 2022) 已经成为跨多种任务和语言的嵌入模型综合评估框架。虽然最初设计为独立基准测试，但 MTEB 已演变为一个多功能评估生态系统。这种自然演变是由研究社区的持续兴趣和参与推动的，集成了众多扩展和专业变体：特定语言的基准如 C-MTEB(Xiao et al., 2024)、MTEB-French(Ciancone et al., 2024) 以及 German Text Embedding Clustering Benchmark(Wehrli et al., 2024)；地区性基准例如 SEB(Enevoldsen et al., 2024)；通过 MMTEB(Enevoldsen et al., 2025) 涵盖超过 250 种语言的多语扩展；特定领域的适应如 ChemTEB(Kasmaee et al., 2024)；以及跨模态评估如 MIEB(Xiao et al., 2025)。

然而，随着基准测试范围的扩大和采用率的提高，保持其可持续性和有效性面临着显著的工程挑战，在文献中这些挑战受到了有限的关注。这些挑战涵盖多个方面：**可重复性**— 确保给定模型的情况下结果可以一致地复现；**可用性**— 有效地评估和展示比较结果；**可扩展性**— 在不破坏现有功能的前提下容纳新的任务、语言和模式；**完整性**— 确保结果反映真实的泛化能力，不受训练数据污染的影响；**相关性**— 随着领域的发展解决当前的研究挑战。文献中强调了基准测试的相关性，但往往忽视了为保持其长期效用所需的实用基础设施。

本文探讨了支撑 MTEB 可持续性和可扩展性的技术设计选择。我们关注为实现**确保数据集质量**、2) **评估基准零样本水平**和 3) **促进社区参与和扩展**而实施的软件工程实践。通过具体的案例研究，我们展示了我们的方法如何解决基准测试维护中的实际挑战，从多语言扩展到污染检测以及社区贡献的扩展。

通过分享我们在维护 MTEB 方面的经验，我们旨在强调在基准开发中经常被忽视的、对长期使用至关

[*]Equal contribution [1]Zendesk [2]Esker [3]INSA Lyon, LIRIS [4]Aarhus University [5]ITMO University. Correspondence to: Isaac Chung <chungisaac1217@gmail.com>, Imene Kerboua <imene.kerboua@insa-lyon.fr>.

Proceedings of the ICML 2025 Workshop on Championing Open-source Development in Machine Learning (CODEML '25).

重要的方面。这里描述的工程方法解决了基准维护中的常见挑战，并可以为机器学习社区内的类似工作提供参考，最终有助于建立更加可重复和值得信赖的评估标准。

2. MTEB 框架

维护像 MTEB 这样的大规模基准库需要稳健的基础设施，以平衡可重复性和灵活性，从而适应多样化的新型模型和评估场景。我们强调的是设计原则，这些原则使 MTEB 能够实现可持续增长和创新潜力，而不仅仅是关注实现细节。

2.1. 系统架构

MTEB 包含一组代码仓库和一个排行榜。Figure 1 在高层次上显示了模型、任务和结果聚合之间的关系：**标准化模型接口**用于生成嵌入，允许在最小适应的情况下评估从句子转换器到指令调整的 LLM 的各种架构；**任务定义**封装不同范式（分类、聚类、检索等）的评估逻辑，同时对特定数据集或模型保持无关性；**数据集处理器**管理数据加载、预处理和验证，同时在语言和领域之间保持一致的接口；**结果处理器**标准化输出格式并在任务间统一计算指标。

为了自动化验证和发布，我们在 MTEB 存储库中采用了标准的 CI/CD 实践，这在 Appendix A中有详细说明。

2.2. 排行榜

除了评估包之外，MTEB 特色一个开放排行榜，显示模型在基准测试中的性能表现，使企业和用户能够选择适合其需求的模型，并支持研究人员推动该领域的进步。排行榜的可视化，请参见 Figure 3中的 Appendix B。

结果提交到排行榜通常是由更广泛的社区完成的。这些可以通过向结果仓库提交拉取请求来实现。

3. 确保基准可重复性

虽然 MTEB 的模块化架构为其功能提供了基础，但特定的可重复性措施对于确保基准测试随时间的有效性至关重要。

我们识别的主要挑战围绕评估工作流程中的歧义：使用的是数据集或模型的哪个版本，执行评估的是代码的哪个版本，以及保存的结果是否可以可靠地加载和信任。

3.1. 多级版本控制系统

在基准测试维护中的一个关键挑战是在管理各种组件的更改的同时保持结果可比性。我们的版本控制系统通过以下方式解决这一问题：**任务版本控制**：当评估协议发生变化时，任务会进行更新，例如使用分层抽样。每个版本都保留自己的评估协议。**数据集版本控制**：每个数据集引用其来源（通常是 Hugging Face）的特定修订版，确保在不同评估运行中数据相同。当发现如错误标记的例子或重复项等问题时，版本会被更新。**模型版本控制**：模型引用具体的检查点或 API 版本以保持一致性，因为模型行为在不同发布之间可能会有显著变化。**代码版本控制**：MTEB 包本身遵循语义化版本控制，具有明确的兼容性边界。

3.2. 可重复的评估环境

某些评估方法引入了必须控制的潜在变异性。例如，使用 K-均值进行聚类任务和使用线性探测进行分类任务需要超参数搜索过程，这些过程可以根据初始化产生不同的结果。为了解决这个问题，我们为评估任务提供了确定性执行的一致种子。此外，考虑到基准测试所需的重大计算资源，我们在 CO_2 排放日志记录中添加了codecarbon 库 [1]。这有助于用户量化并报告其评估运行的环境影响，促进基准测试计算成本的透明度。

[1] https://codecarbon.io/

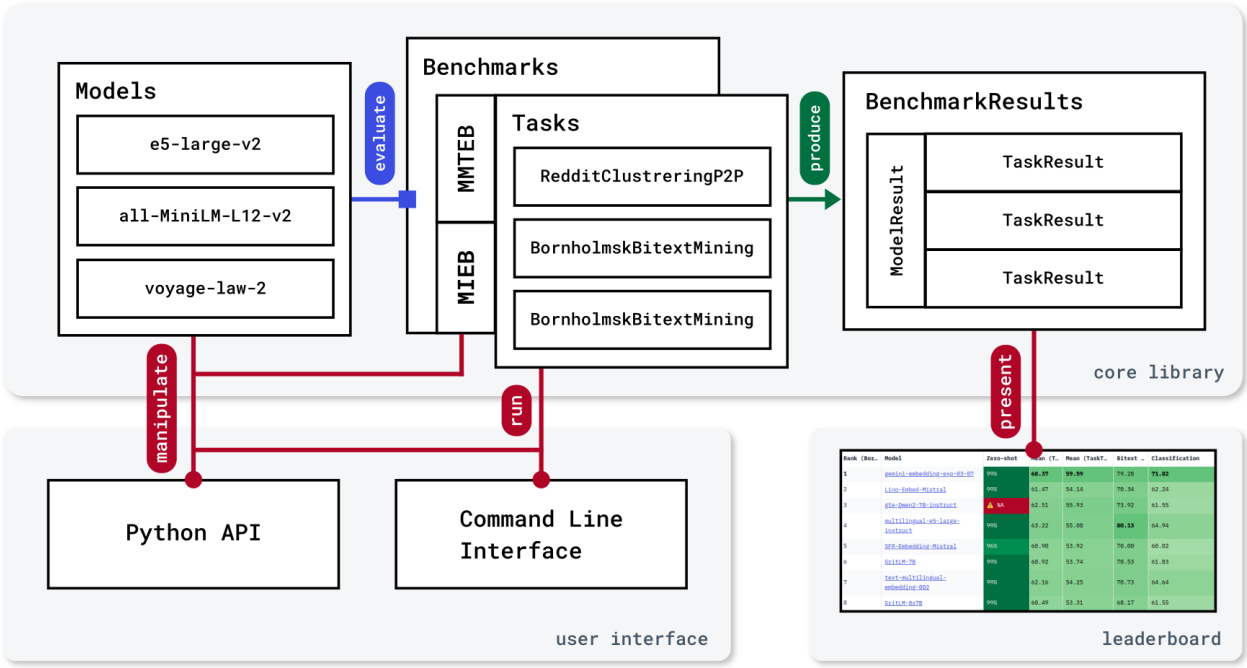

Figure 1. MTEB 采用模块化架构设计，将关注点分离到不同的组件中。用户可以通过 Python API 或命令行界面选择要运行的模型、任务或基准测试。生成的结果可以提交到结果仓库，该仓库由排行榜读取。

3.3. 可验证结果

MTEB 排行榜的设计目的是最大限度地提高透明度和可验证性。提交要求：**参考实现**：每个模型必须有详细的参考实现文档。对于开源模型，这包括直接链接到代码库和模型权重。对于专有模型，实现可以依赖 API 调用，尽管这样引入了一个信任假设，即 API 一致地提供相同版本的模型。**同行评审过程**：提交的结果需要通过向结果存储库提出拉取请求接受社区审查，在那里维护者和社区成员可以验证方法并质疑不寻常的模式。**版本特定跟踪**：结果明确记录了使用的是哪个版本的 MTEB、数据集和模型，从而能够精确再现评估条件。

这种方法确保了任何人都可以通过使用特定的 MTEB 版本将指定的数据集版本与指定的模型版本进行对比，从而在不同的机器和环境中重现基准测试结果并获得相同的结果。

这些可重复性措施反映了从过去的基准测试失败中吸取的经验教训（见 subsection 5.2），并有助于 MTEB 作为公平和一致评估嵌入模型的稳健基础。

4. 设计可扩展性

MTEB 是一个评估框架，首先应当可重复且因此稳定。为了实现这一点，MTEB 采用了模块化设计，旨在允许贡献者扩展任何组件而不修改其他部分，从而在保持向后兼容性和结果可比性的同时促进增长。

构建 MTEB 以具有可扩展性也鼓励创新。任何新添加的自定义模型、任务和基准都可以使用相同的底层框架进行复现。

与许多现有基准测试强制执行一个往往限制性的接口以评估模型 (Nielsen et al., 2024; Li et al., 2024) 不同，MTEB 向模型提供了丰富的上下文信息，这些信息可能在推理时可用，包括任务类型、目标语言或特定于任务的提示，使研究人员能够探索多样化的模型架构和提示策略。这种灵活性允许更现实和创造性地评估模型，促进表示学习的进步。通过

整合最近的模型、任务和基准测试，MTEB 有机地保持了其在该领域的相关性。

5. 案例研究

基准的可持续性不仅取决于基础设施和可重复性，还取决于社区参与和展示出的实用性。社区参与不仅加强了基准的可靠性，还有助于确保其持续维护、相关性和代表性，特别是对于资源匮乏的社区 (Singh et al., 2024)。总体而言，超过 170 名贡献者参与了该项目的发展。他们的共同努力对将 MTEB 塑造成一个多语言、多领域、多模式、由社区驱动的基准至关重要。本节介绍了我们在维护 MTEB 过程中遇到的关键挑战以及我们实施的解决方案，为其他基准维护者提供了实用见解。

5.1. 案例研究 1：评估零样本水平。

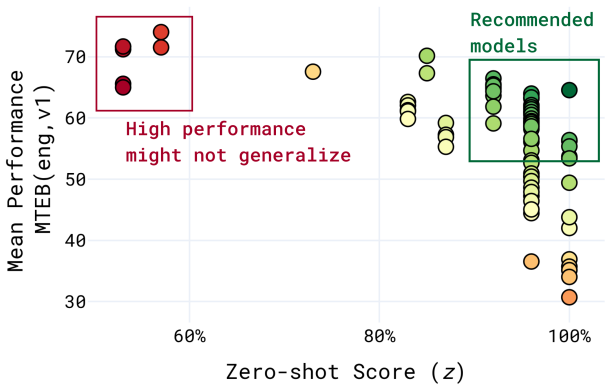

Figure 2. 模型在传统英语 MTEB 上的零样本分数的平均表现。排名最高的模型通过训练基准任务来获得其分数，即使得分较低的模型可能更好地泛化到分布外环境。

在嵌入基准测试中，一个重要的挑战是确定模型是否真正展示了分布外的泛化能力。与传统数据泄露（其中测试样本出现在训练数据中）不同 (Magar & Schwartz, 2022; Choi et al., 2025)，MTEB 面临着更加微妙的挑战：模型被训练于具有类似于基准任务分布的数据集上，特别是在使用来自与基准评估任务相同来源的训练拆分时。为了解决这个问题，我们实施了一种透明度方法，该方法：1) 鼓励模型贡献者披露训练数据集，2) 计算一个零样本分数量化分布重叠程度：

$$z = 1 - \frac{n_{\text{train}}}{n_{\text{total}}}$$

其中，z 是零样本得分，n_{train} 是模型训练所用的基准数据集数量，而 n_{total} 是总的基准数据集数量。

该指标提供了模型相对于基准测试是“在领域内”还是“在领域外”的粗略估计。例如，e5-mistral-7b-instruct 模型 (Wang et al., 2024a) 在 MTEB（英语，v2）上的零样本得分为 95%，表明它仅在基准测试的训练分割的 ~5% 上进行了训练，即 5% 的泄漏。尽管如此，该模型在各方面表现都很出色，这表明其对未见过的任务具有强大的泛化能力。

我们的初始实现过滤了排行榜，仅显示零样本得分为 100%或训练数据未知的模型。这种方法被证明过于严格，因为社区成员指出它通过隐藏那些诚实地披露其训练数据的强大模型而损害了透明度。根据这一反馈，我们开发了一种更精细的方法，在保持透明度的同时为用户提供解释基准分数的关键背景信息。本案例研究展示了如何在方法论严谨性和实际可用性之间平衡，并且说明了社区反馈如何推动基准设计的改进。关于此主题的详细关键交换可以在 Appendix C中找到。

5.2. 案例研究 2：重现报道的结果。

MTEB 排行榜最近从依赖 Hugging Face 模型卡片中的自报结果转向了一个集中式的验证结果存储库。这一过渡揭示了再现报告模型性能时的几个挑战：1) 嵌入模型可以使用前缀（E5 使用 (Wang et al., 2024b) 的查询：和段落：作为检索中的查询和段落提示），2) 提示可以根据任务类型不同（Nomic 模型），3) 在检索过程中可以将提示添加到查询和段落中（E5-mistral(Wang et al., 2023)）或仅添加到查询中（NV Embed），4) 某些模型不归一化嵌入（Nomic ModernBERT），5) 编码期间模型可以有自定义参数（jina-v3 的 (Sturua et al., 2024) 任务类型 参数在推理时加载 LoRAs），6) 模型可能具有附加阶段（CDE(Morris & Rush, 2024) 在存储嵌入的

编码过程中有不同的阶段）。

为了确保在这一多样化的嵌入模型架构中进行准确评估，我们实现了额外的功能以满足这些需求。例如，实现前缀支持使我们能够复现 BGE 模型报告的性能，该模型在之前的评估中曾表现出显著的性能下降。改进的可重复性增强了社区对排行榜排名的信心。相关拉取请求的详细信息请参见 Appendix D。

6. 限制

该框架目前尚未解决偏差问题 (Rakivnenko et al., 2024; Cao, 2025)。在艺术、文化、健康等领域 (Cao, 2024) 的覆盖面也有限。

MTEB 也因对语义版本控制和向后兼容性的松散遵守而受到了模型制作者的批评。这些问题主要源于为多语言扩展所需进行的大规模重构工作。虽然这些重构确实给社区和整个软件包带来了益处，但在次要/补丁发布中有时会引入破坏性更改，这使得该软件包的频繁使用者感到困惑和沮丧。自那以后，我们更加重视保持与旧版本库的向后兼容性以避免进一步的不便。

自 MTEB 首次发布以来，该包一直保持简单的文档结构，最初是在 README 中，随着内容扩展到多个相互链接的 Markdown 文件中。

虽然这在一开始很容易维护，但代码库已变得更大更复杂，并且现在更容易出现代码和文档不一致的情况。我们计划在未来版本（2.0.0 及以上）中将文档纳入 docstrings 并自动生成文档。

7. 结论

MTEB 从一个独立的基准转变为可扩展的评估生态系统，依赖于稳健的设计选择。我们介绍了维护 MTEB 的方法，以实现更好的可重复性、可维护性和新任务及模态的无缝集成。这些基础设施的选择使广泛的社会参与和持续增长得以实现，同时不会牺牲质量。我们的经验为设计和维护基准提供了实用指导，有助于实现更具可重复性、可靠性和全球代表性的机器学习研究。

影响声明

MTEB 通过实现对高资源和低资源语言的严格、可扩展评估，促进了语言技术的公平进步。

致谢

我们感谢审稿人的深刻反馈以及所有对 MTEB 库做出贡献的人。

参考文献

Cao, H. Recent advances in text embedding: A comprehensive review of top-performing methods on the mteb benchmark, 2024. URL https://arxiv.org/abs/2406.01607.

Cao, H. Writing style matters: An examination of bias and fairness in information retrieval systems. In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM ' 25, pp. 336 – 344. ACM, March 2025. doi: 10.1145/3701551.3703514. URL http://dx.doi.org/10.1145/3701551.3703514.

Choi, H. K., Khanov, M., Wei, H., and Li, Y. How contaminated is your benchmark? quantifying dataset leakage in large language models with kernel divergence, 2025. URL https://arxiv.org/abs/2502.00678.

Ciancone, M., Kerboua, I., Schaeffer, M., and Siblini, W. Mteb-french: Resources for french sentence embedding evaluation and analysis, 2024. URL https://arxiv.org/abs/2405.20468.

Enevoldsen, K., Kardos, M., Muennighoff, N., and Nielbo, K. The scandinavian embedding benchmarks: Comprehensive assessment of multilingual and monolingual text embedding. In Advances in Neural Information Processing Systems, 2024. URL https://nips.cc/virtual/2024/poster/97869.

Enevoldsen, K., Chung, I., Kerboua, I., Kardos, M., Mathur, A., Stap, D., Gala, J., Siblini, W., Krzemiński, D., Winata, G. I., Sturua, S., Utpala, S., Ciancone, M., Schaeffer, M., Misra, D., Dhakal, S., Rystrøm, J., Solomatin, R., Çağatan, Ö. V., Kundu, A., Bernstorff, M., Xiao, S., Sukhlecha, A., Pahwa, B., Poświata, R., GV, K. K., Ashraf, S., Auras, D., Plüster, B., Harries, J. P., Magne, L., Mohr, I., Zhu, D., Gisserot-Boukhlef, H., Aarsen, T., Kostkan, J., Wojtasik, K., Lee, T., Suppa, M., Zhang, C., Rocca, R., Hamdy, M., Michail, A., Yang, J., Faysse, M., Vatolin, A., Thakur, N., Dey, M., Vasani, D., Chitale, P. A., Tedeschi, S., Tai, N., Snegirev, A., Hendriksen, M., Günther, M., Xia, M., Shi, W., Lù, X. H., Clive, J., K, G., Anna, M., Wehrli, S., Tikhonova, M., Panchal, H. S., Abramov, A., Ostendorff, M., Liu, Z., Clematide, S., Miranda, L. J. V., Fenogenova, A., Song, G., Safi, R. B., Li, W.-D., Borghini, A., Cassano, F., Hansen, L., Hooker, S., Xiao, C., Adlakha, V., Weller, O., Reddy, S., and Muennighoff, N. MMTEB: Massive multilingual text embedding benchmark. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=zl3pfz4VCV.

Kasmaee, A. S., Khodadad, M., Saloot, M. A., Sherck, N., Dokas, S., Mahyar, H., and Samiee, S. Chemteb: Chemical text embedding benchmark, an overview of embedding models performance & efficiency on a specific domain. In Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop, PMLR 262:512-531, 2024. URL https://doi.org/10.48550/arXiv.2412.00532.

LAION-AI. CLIP_benchmark: CLIP-like model evaluation. https://github.com/LAION-AI/CLIP_benchmark, 2025. Accessed: 2025-05-03.

Li, X., Dong, K., Lee, Y. Q., Xia, W., Yin, Y., Zhang, H., Liu, Y., Wang, Y., and Tang, R. Coir: A comprehensive benchmark for code information retrieval models, 2024. URL https://arxiv.org/abs/2407.02883.

Magar, I. and Schwartz, R. Data contamination: From memorization to exploitation. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 157–165, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.18. URL https://aclanthology.org/2022.acl-short.18/.

Morris, J. X. and Rush, A. M. Contextual document embeddings, 2024. URL https://arxiv.org/abs/2410.02525.

Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. Mteb: Massive text embedding benchmark. arXiv preprint arXiv:2210.07316, 2022. doi: 10.48550/ARXIV.2210.07316. URL https://arxiv.org/abs/2210.07316.

Nielsen, D. S., Enevoldsen, K., and Schneider-Kamp, P. Encoder vs decoder: Comparative analysis of encoder and decoder language models on multilingual nlu tasks. arXiv preprint arXiv:2406.13469, 2024.

Rakivnenko, V., Maslej, N., Cervi, J., and Zhukov, V. Bias in text embedding models, 2024. URL https://arxiv.org/abs/2406.12138.

Singh, S., Vargus, F., Dsouza, D., Karlsson, B. F., Mahendiran, A., Ko, W.-Y., Shandilya, H., Patel, J., Mataciunas, D., OMahony, L., Zhang, M., Hettiarachchi, R., Wilson, J., Machado, M., Moura, L. S., Krzemiński, D., Fadaei, H., Ergün, I., Okoh, I., Alaagib, A., Mudannayake, O., Alyafeai, Z., Chien, V. M., Ruder, S., Guthikonda, S., Alghamdi, E. A., Gehrmann, S., Muennighoff, N.,

Bartolo, M., Kreutzer, J., Üstün, A., Fadaee, M., and Hooker, S. Aya dataset: An open-access collection for multilingual instruction tuning, 2024. URL https://arxiv.org/abs/2402.06619.

Sturua, S., Mohr, I., Akram, M. K., Günther, M., Wang, B., Krimmel, M., Wang, F., Mastrapas, G., Koukounas, A., Koukounas, A., Wang, N., and Xiao, H. jina-embeddings-v3: Multilingual embeddings with task lora, 2024. URL https://arxiv.org/abs/2409.10173.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. Improving text embeddings with large language models. arXiv preprint arXiv:2401.00368, 2023.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. Improving text embeddings with large language models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 11897–11916, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. Multilingual e5 text embeddings: A technical report, 2024b. URL https://arxiv.org/abs/2402.05672.

Wehrli, S., Arnrich, B., and Irrgang, C. German text embedding clustering benchmark. arXiv preprint arXiv:2401.02709, 2024. URL https://arxiv.org/abs/2401.02709.

Xiao, C., Chung, I., Kerboua, I., Stirling, J., Zhang, X., Kardos, M., Solomatin, R., Moubayed, N. A., Enevoldsen, K., and Muennighoff, N. Mieb: Massive image embedding benchmark. arXiv preprint arXiv:2504.10471, 2025. doi: 10.48550/ARXIV.2504.10471. URL https://arxiv.org/abs/2504.10471.

Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., and Nie, J.-Y. C-pack: Packed resources for general chinese embeddings. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, pp. 641 – 649, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

A. 技术基础设施的完整描述

A.1. 持续集成

我们的持续集成 (CI) 管道验证了代码和数据集，这一步骤在其他基准测试中经常被忽视 (Thakur et al., 2021; LAION-AI, 2025)。

A.2. 数据集和模型验证

任何添加或修改数据集的拉取请求都会触发自动化检查，包括：1) 格式一致性，2) 元数据完整性，3) 元数据字段验证，4) 以及在 Hugging Face 上的数据集可用性。同样地，任何新的或已修改的模型元数据也会自动检查其完整性和模型加载情况。这两种类型的验证都严重依赖于 Pydantic[2]。这有助于尽早发现格式错误示例和预处理不一致性。

每个数据集和模型都有各自的 Pydantic 模型，用于指定捕获可重复性所需的关键信息的必填字段。具体来说，每个数据集都有任务元数据，而每个模型都有模型元数据。例如，表 1 显示了截至版本 1.38.4 的所有模型元数据参数及其描述。

A.3. 拉取请求检查

对代码库的贡献需经过一个包含三个连续阶段的自动化验证工作流程：1) 代码检查和格式化，2) 执行单元测试和集成测试，3) 验证任何必要的文档更新。

通过将常规检查委托给此管道，维护人员的负担减轻了，并且该项目仍能大规模接受贡献。

A.3.1. 语法检查

所有传入的更改均根据强制编码规范（例如，导入顺序、现代化 Python 语法）和检测常见错误的共享配置使用 ruff[3] 进行分析。

A.3.2. 测试

为了确保可重复性，测试套件在 Linux 和 Windows 平台上执行，其中 Linux 任务涵盖了 Python 版本 3.9 到 3.12。这种跨平台策略在开发过程中对于发现特定环境缺陷起到了关键作用。

模拟基础设施我们使用模拟任务——每个由两到三个示例定义——和合成模型来发出随机嵌入。这些模拟通过锻炼任务评估者和模型交互代码，能够在不承担完整模型实例化开销的情况下提供快速反馈。

参数验证测试断言所有必需参数都能正确地从评估器模块传递到模型类，且各种配置下提示例行程序能按预期应用输入。

数据集成某些测试从 Hugging Face Hub 动态检索数据集，以验证端到端的数据加载和预处理。

可靠性增强我们集成了 pytest-重运行失败测试插件插件，自动重试因临时连接错误而失败的测试，从而提高整体套件稳定性。

A.4. 版本控制、发布和自动化

版本控制：我们遵循一种适应基准测试的类似语义化版本策略：1）主要版本表示代码不兼容的更改，2）次要版本添加任务或数据集而不影响现有结果，3）补丁版本修复错误或更新文档。

变更日志和发布自动化：自动生成包含：1) 贡献者名单和 2) 变更描述的变更日志。带有版本标签的发布将触发：1) PyPI 发布，2) 文档部署，以及 3) 发布说明准备。这确保了可重现且可追溯的版本。

自动化制品：为了标准化结果报告，我们支持自动生成：1）带有描述和引用的基准和任务的 Markdown 表格，2）数据集的描述性统计信息，3）整个存储库的语言覆盖范围。这些工具提高了结果的清晰度并减少了手动报告错误。LaTeX 表格可以随意生成用于学术出版物，并且不是自动化的。

排行榜：MTEB 排行榜每天都会刷新和重建。之前的流程涉及抓取 huggingface 模型卡片上的自报分数。当前的流程需要主动提交结果到结果仓库。

[2] https://github.com/pydantic/pydantic
[3] https://github.com/astral-sh/ruff

参数	类型	描述
加载器	Callable or None	Function to load the model; defaults to SentenceTransformer if None.
名称	str	Name of the model, ideally in the format “organization/model_name”.
参数数量	int or None	Number of parameters in the model, e.g., 7_000_000.
内存使用量_mb	float or None	Memory usage of the model in MB.
最大标记数	int or None	Maximum tokens the model can handle.
嵌入维度	int	Dimension of embeddings produced.
修订	str or None	Model revision identifier.
发布日期	str	Date the revision was released.
许可	str	License under which the model is released.
开放权重	bool	Whether model weights are open source.
公共训练代码	str or None	URL to training code if publicly available.
公共训练数据	str or None	URL to training data if publicly available.
相似性_函数_名称	str	Distance metric used by the model.
框架	list of str	List of frameworks used, e.g., [” 句子变换器”, ”PyTorch”].
参考	str	URL to the model’s page (e.g., HuggingFace).
语言	list of str	Languages supported, e.g., [” 英-拉丁文”].
使用说明	bool	Whether the model uses instruction-based formatting.
训练数据集	dict	Datasets used in training, e.g., {” 论证分析”: [” 测试”]}.
改编自	str or None	Base model this one was adapted from.
被取代的	str or None	Model that supersedes this one.
是交叉编码器	bool	Whether the model functions as a cross-encoder.
模态	list of str	Modalities supported, default is [” 文本”].

Table 1. 模型元数据类的参数

B. 排行榜

Figure 3显示了浅色模式下的 MTEB 排行榜。

C. 模型零样本得分

关于零样本分数的关键讨论在此部分中：

- 讨论零样本定义的问题：https://github.com/embeddings-benchmark/mteb/discussions/2351
- 社区对初始版本的反馈以及排行榜上零样本模型的过滤：https://github.com/embeddings-benchmark/mteb/discussions/2119

D. 重现报道的结果

我们在这一时期的每个要点上都提供了关于基准可重复性的示例拉取请求：

1. 嵌入模型可以使用前缀（e5、bge 模型）:https://github.com/embeddings-benchmark/mteb/

issues/1912

2. 指令可以因任务类型而异（Nomic 模型）:https://github.com/embeddings-benchmark/mteb/pull/1685

3. 检索过程中可以在查询和文档中添加提示（E5-mistral(Wang et al., 2023)）或者仅在查询中添加（Nvembed）：https://github.com/embeddings-benchmark/mteb/pull/1436

4. 某些模型不规范化嵌入（Nomic ModernBERT）：https://github.com/embeddings-benchmark/mteb/pull/1684

5. 模型在编码过程中可以具有自定义参数（jina-v3(Sturua et al., 2024) 任务类型参数在推理过程中加载 LoRAs）：https://github.com/embeddings-benchmark/mteb/pull/1319

6. 模型可以具有额外的阶段（CDE(Morris & Rush, 2024) 具有不同的编码阶段以存储嵌入）：https://github.com/embeddings-benchmark/mteb/pull/2076

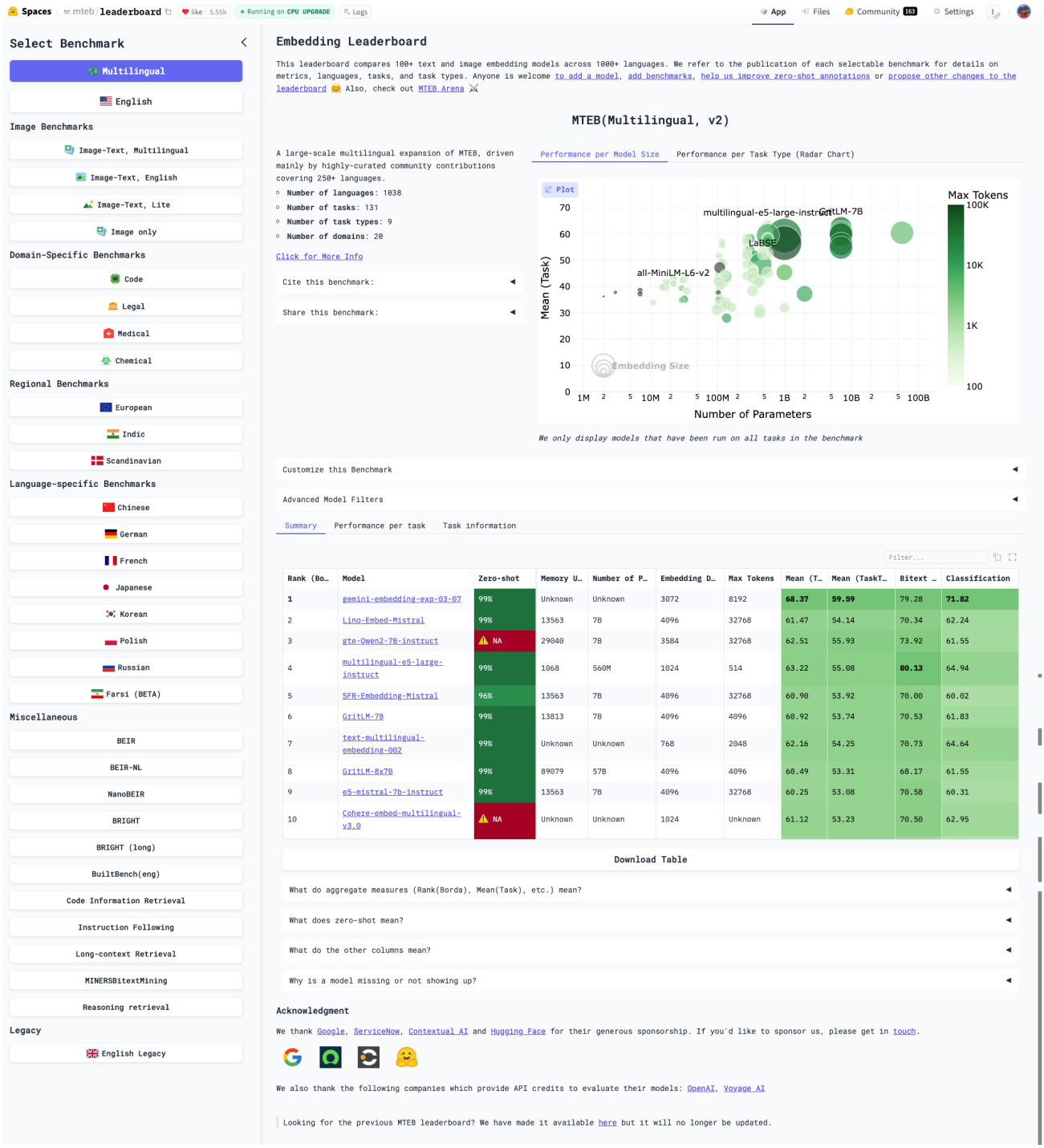

Figure 3. MTEB 排行榜提供了一个可扩展的嵌入基准集合，其默认设置为多语言（MMTEB，(Enevoldsen et al., 2025)）。对于每个基准，都会显示按模型大小划分的模型性能图，并且可以选择在单独的选项卡中显示按任务类型划分的模型性能。模型默认按照博达计数进行排名。MTEB 排行榜还提供了通过筛选任务类型、语言和领域等字段来定制基准的选项。左侧边栏包含按组高亮显示的基准，而所有基准在代码中均可获得。