ALTEGRAD Data Challenge - Molecular Graph Captioning

Antoine Gilson
antoine.gilson@ensae.fr

Paul Lemoine
paul.lemoine@ensae.fr

Alexandre Zenou
alexandre.zenou@polytechnique.edu

1 Introduction

First, we would like to state that the challenge was particularly intuitive to approach, thanks to the clear baseline pipeline provided and the guidance offered in the accompanying PDF, which suggested several meaningful directions to explore.

The first crucial choice we made was not to pursue a generative approach for the output, as we considered it would be unnecessarily complex compared to the retrieval-based approach and the scope of the task. Instead, we decided to focus on strengthening the retrieval pipeline by targeting three key components explicitly highlighted in the assignment through the python files: the text embeddings, the graph encoder, and the retrieval mechanism.

2 Improvement of the embeddings

First of all, our analysis identified text embeddings as a critical component of the retrieval-based molecular captioning pipeline, as they play a major role in the training of the model later on. Indeed, we will train the model to align with the embeddings, so if the embeddings are poor, even the best architecture won't be relevant. In the baseline system, textual descriptions are encoded using a single general-purpose language model (`bert-base-uncased`), with a fixed truncation length of 128 tokens and a pooling strategy based solely on the `[CLS]` token. While computationally efficient, this setup presents several limitations.

To address these issues, we first stated that the pipeline should support multiple backbone models, including scientific models (e.g., ChemBERTa, SciBERT., as well as sentence-level embedding models explicitly optimized for nearest neighbor search (mpnet, GTE, and BGE models). We thus imported several pre-trained models. This choice was made to ensure strong baseline performance while maintaining a simple, reproducible pipeline, avoiding the substantial computational cost of training large language models from scratch. By evaluating multiple and different text embedding models, we explored different embedding strategies to identify promising configurations and gain initial insights into the structure of the problem.

On a more technical note, we increased the maximum input length to 512 tokens, substantially reducing information loss for longer descriptions. In addition, we moved beyond a single `[CLS]`-based pooling strategy by adopting mean pooling over non-padding tokens. Moreover, all embeddings are explicitly L2-normalized, ensuring stable cosine similarity computations.

Besides, we further enhanced robustness and reproducibility by systematically handling empty or malformed descriptions, encapsulating inference in safe execution blocks.

As a result, we explored several embedding strategies, including ensembling multiple models by combining a chemistry-oriented encoder with a general semantic model. Empirically, we did

Preprint. Under review.

not observe large performance differences across embedding families, whether they were used individually or in an ensemble configuration. Nevertheless, across all settings, improving the overall quality and stability of the embedding space consistently led to better downstream performance, all else being equal. In particular, replacing BioBERT or SciBERT with retrieval-optimized models such as GTE resulted in clear and consistent performance gains, highlighting the importance of embeddings explicitly designed for similarity-based retrieval (according to our results).

Overall, we believe our modifications strengthened the expressiveness and robustness of the text embedding space, which should directly benefit the alignment between graph and text representations..

3 Improvement of the encoder and its training

To provide a comprehensive view of our retrieval-based framework, Figure 1 illustrates the overall architecture designed for this challenge. The model operates as a dual-branch system: the molecular branch extracts structural features from the graph, while the text branch encodes semantic descriptions. Both representations are projected into a shared latent space and aligned using a contrastive objective, ensuring that chemically similar molecules lie close to their corresponding textual descriptions.

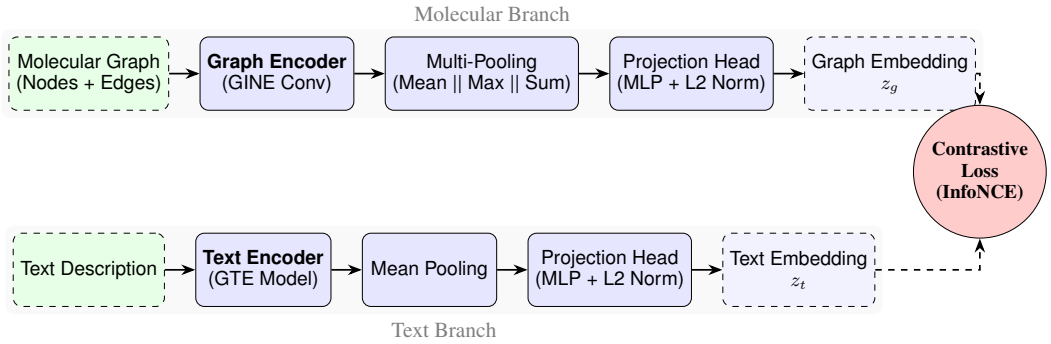

Figure 1: **Overview of the proposed architecture.** The model consists of two parallel branches projecting inputs into a shared embedding space. Left-to-right flow illustrates the transformation from raw data to L2-normalized embeddings, optimized via Contrastive Loss.

In the original baseline, molecular graphs are initialized with identical node representations, meaning that all atoms start from the same shared embedding and no explicit chemical information is provided to the model. As a result, the graph encoder can only learn anonymous connectivity patterns and is effectively limited to graph isomorphism–style reasoning, which discards most of the chemically relevant signal. To address this limitation, we enrich the graph representation by explicitly incorporating both node- and edge-level features provided in the dataset. These categorical features are embedded using learnable embedding layers and combined either through summation or concatenation followed by a projection MLP to form a chemically meaningful initial node representation. This enables the model to distinguish between different chemical interactions and to use all the features provided in the dataset.

This naturally motivates the use of message-passing layers that explicitly consume the edge features we just added. In our final configuration, we adopt a GINE-based architecture, which extends GIN by conditioning messages on embedded bond attributes and is particularly well suited for molecular graphs. Compared to the feature-free baseline, this enriched encoding enables the graph encoder to reason over chemically grounded structures rather than purely topological patterns, yielding a substantially more informative representation. Moreover, each message-passing block applies LayerNorm and dropout, and Jumping Knowledge is optionally used to select the final-layer representation or aggregate intermediate layers, providing flexibility in how multi-scale information is retained.

At the graph level, node representations are aggregated using a multi-pooling readout that concatenates global mean, max, and sum pooling. This design captures complementary statistics of the node distribution and proved more robust than a single pooling operator. The resulting graph representation is then mapped to the text embedding space through a deeper projection head composed of linear

layers, LayerNorm, dropout, and a lightweight residual MLP. Finally, the graph encoder outputs L2-normalized embeddings, ensuring a well-behaved embedding space consistent with cosine-similarity-based retrieval at both training and inference time.

A key part of the project was to redesign the training objective by replacing a pure embedding regression loss with a contrastive loss inspired by CLIP-style formulations. While the baseline can be framed as an embedding retrieval problem, using a mean squared error objective does not fully exploit negative examples and encourages absolute alignment rather than relative similarity. In contrast, the proposed loss explicitly normalizes both graph and text embeddings, computes similarity scores using cosine similarity or dot products, and leverages all other elements in the batch as negatives through an InfoNCE (NT-Xent) formulation with a temperature parameter. This objective is better aligned with the retrieval-based nature of the task and with the final evaluation protocol, which depends on nearest-neighbor similarity rather than exact embedding reconstruction. The loss is implemented symmetrically, optimizing both molecule-to-text and text-to-molecule alignment within each batch.

Concerning model training, we adopt a standard and robust set of hyperparameters and deliberately refrain from extensive fine-tuning in order to limit computational cost. Indeed, in preliminary experiments, we observed that moderate variations in architectural details or hyperparameter values had a relatively limited impact on performance compared to changes in the text embedding space or more substantial architectural modifications, such as switching from GNN-based message passing to transformer-style convolutions. To further stabilize optimization, gradients are clipped to a fixed norm during training, which proved beneficial when optimizing the contrastive objective.

4 Improvement of the retrieval system

In the baseline system, retrieval is performed by selecting the single nearest neighbor in the text embedding space based on cosine similarity between the predicted graph embedding and all training text embeddings. While simple and efficient, this hard nearest-neighbor strategy is highly sensitive to noise in the embedding space and to small geometric distortions introduced during graph embedding regression.

Our improved retrieval module replaces this brittle decision rule with a topk_weighted strategy, which retrieves the top-k most similar candidate descriptions and combines them using similarity-based weights rather than relying on a single best match. Concretely, cosine similarities are first computed and normalized, then transformed into a smooth weighting distribution (e.g. via softmax or normalized positive scores), which is used to aggregate information from multiple close neighbors. This approach is motivated by the observation that, in high-dimensional semantic spaces, several neighboring descriptions often capture complementary aspects of the target molecule, and that the exact ranking among the closest candidates is not always reliable. The retrieval process becomes more robust to small embedding errors and reduces variance across predictions. Importantly, this soft aggregation better aligns with the downstream evaluation metrics: BERTScore rewards semantic similarity rather than exact lexical matches, and BLEU can benefit when the retrieved description reflects more general phrasing shared across several close neighbors. As a result, the topk_weighted strategy is a low-cost yet effective refinement of the overall retrieval-based pipeline.

5 Performance of our methods

The problem could have been approached in many different ways, and we explored several design choices across the pipeline, including text embedding models (e.g., ChemBERTa, GTE...), graph encoder architectures (model type, pooling strategy, and jumping knowledge mode), and retrieval mechanisms (such as top-k aggregation or ensembling). One possible direction we chose not to pursue was an exhaustive or brute-force search over all combinations of hyperparameters, including training hyperparameters. Instead, since the objective was not to maximize leaderboard performance at all costs nor to overfit the test set through external data, we focused on configurations that were well motivated from a methodological standpoint. Our main results are showcased in the Table 1. While alternative combinations might yield additional performance gains, our choices prioritize interpretability, robustness, and alignment with the underlying assumptions of the task.

While BLEU and BERTScore are used for final evaluation, we relied on ranking-based metrics during local validation. Indeed, BLEU or BERTScore are less informative during local development, as

our model does not generate novel text but retrieves existing descriptions, making ranking-based metrics more directly aligned with the training objective and inference procedure. We relied on Mean Reciprocal Rank (MRR), Hit@1, and Hit@5. as they directly assess the quality of the ranking induced by cosine similarity in the shared embedding space.

Let $\mathcal{Q}$ denote the set of evaluation queries, and let $rank_q$ be the rank position of the correct (ground-truth) item for query q, according to the similarity-based ranking.

The Mean Reciprocal Rank is defined as:

$$\text{MRR} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{rank_q}$$

Hit@K measures whether the correct item appears among the top-K ranked results:

$$\text{Hit@K} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{I}(rank_q \leq K)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

In our experiments, we reported MRR, Hit@1 and Hit@5. Indeed, for each molecular graph in the validation set, the model retrieves a ranked list of candidate textual descriptions based on embedding similarity. Hit@K measures whether the correct description appears among the top K retrieved candidates, thus providing an intuitive notion of retrieval accuracy at different operating points. In particular, Hit@1 reflects strict top-1 accuracy, while Hit@5 captures the model's ability to place the correct description within a small set of plausible candidates. Besides, Mean Reciprocal Rank (MRR) provides a more fine-grained evaluation of ranking quality by considering the exact position of the correct description in the retrieved list. It is defined as the average of the reciprocal ranks of the correct items across all queries, thereby rewarding systems that consistently rank the correct description higher. Compared to Hit@K metrics, MRR is more sensitive to improvements in the top of the ranking and better reflects overall retrieval quality.

About hyperparameters, as explained earlier, although we briefly explored a limited grid search over a small subset of hyperparameters, we did not observe significant performance gains, suggesting that further hyperparameter tuning was not the main bottleneck of the system.

Eventually, about the form of the repository, we chose to preserve the overall structure of the existing pipeline and repository, introducing only minimal modifications. The original design proved to be simple, stable, and sufficient for our purposes, and this decision allowed us to focus on understanding the core challenges of the task and improving the files rather than engineering a substantially different system.

At the end of the day, our results looked like the following table :

Table 1: Progression of our important model performances milestones

Pipeline Stage	Configuration / Method	Key Feature	Score
1. Baseline	Basic Retrieval	BERT Base, [CLS] Pooling	0.487
2. Graph Encoder	Graph Transformer	Transformer Conv	0.578
	GINE + Multipool	**Explicit Edge Feat., Mean/Max/Sum**	**0.595**
3. Text Embeddings	Final Single Model	GTE Base (Optimized)	0.638
	Ensemble (Best)	**ChemBERTa + GTE Base**	**0.644**

6 Conclusion

To conclude, with our last architecture, our model achieved a score of 0.64 on the public leaderboard. The final system combines GTE text embeddings with a GINE-based graph encoder using explicit node and edge features, a multi-pooling readout (mean, max, sum), and Jumping Knowledge in `last` mode. Predictions are produced using a weighted top-3 nearest-neighbor retrieval strategy in the shared embedding space.

Embedding generation requires approximately 20 minutes on the Onyxia SSP cloud machines. Model training with fixed hyperparameters takes a similar amount of time, while the final retrieval step is comparatively lightweight and completes in roughly one minute.

Overall, our results do not place the method at the very top of the leaderboard, but rather within the average range for a comparable number of submissions. We consider this outcome satisfactory, as it reflects a solid understanding of the task and its underlying challenges. Importantly, we emphasize the robustness of our approach: we did not rely on aggressive hyperparameter tuning, and multiple architectural and hyperparameter configurations consistently led to performance within the same order of magnitude. This suggests that the proposed method generalizes reasonably well and that its performance is driven by principled design choices rather than finely tuned parameters. Moreover, we didn't rely on any external data at any point.

To conclude, potential improvements include further strengthening the text embedding space, for instance through more advanced or domain-adapted pretrained models, as well as exploring generative approaches for molecular description beyond pure retrieval. Additional gains could come from improved contrastive training strategies, more expressive graph encoders, or hybrid retrieval–generation frameworks.