ALTEGRAD 2025-26 Data Challenge

Molecular Graph Captioning

DaSciM Team, LIX, École Polytechnique

Welcome to the ALTEGRAD Data Challenge! This project is designed to provide you with a hands-on experience in building and evaluating sophisticated machine learning/deep learning models on a novel and multi-modal task.

1 Challenge Description

The goal of this project is to study and apply machine learning/artificial intelligence techniques to bridge the gap between structured graph data and unstructured natural language. One of the most challenging and rapidly evolving tasks in this multi-modal domain is that of **graph-to-text generation**. This task has attracted significant attention recently, as it enables the "translation" of complex relational data into human-readable text [2, 3].

For instance, in chemo-informatics, this specific task is known as **Molecular Graph Captioning**. This task is crucial for translating complex 2D structures into summaries for scientific literature, augmenting chemical database entries, and assisting in AI-driven drug discovery pipelines [1].

In this challenge, your primary objective is to develop a model that takes a molecule's atomic structure, represented as a graph, and predicts a coherent, human-readable text caption. This caption should accurately describe the key structural and functional aspects of the molecule.

This task lies at the challenging intersection of Graph Representation Learning and Natural Language Processing (NLP). It requires your model to bridge the gap between two fundamentally different data modalities: the symbolic, structured-graph representation of a molecule and the sequential, semantic structure of natural language.

Success in this task has significant implications for scientific research, with applications in automated report generation, enhancing the accessibility of chemical databases, and assisting in drug discovery by summarizing molecular properties.

For detailed instructions about how to participate, please refer to Section 6.

2 Dataset Overview

The full dataset we provide contains approximately 33,000 molecule-text pairs. We have split this dataset as follows:

- Training and Validation Set: About 32,000 molecules are provided with both their full graph structure and their corresponding ground-truth text description. You will use them for training and local model evaluation.
- The rest 1,000 molecules are provided with their graph structure only. The text descriptions for this set are withheld and are used for the leaderboard evaluation.
  - Public Leaderboard: When you submit your predictions, half of these test molecules will be evaluated immediately to update the public Kaggle leaderboard. This allows you to track your performance and compare it with others during the competition.
  - Private Leaderboard: The remaining half will be used to compute the private leaderboard, which will be revealed after the competition deadline.

Note on Evaluation and Scoring:

You will not know which molecules belong to which leaderboard half, as they are all mixed together in the test set. Your final rank on this private leaderboard will be used as a component of your final evaluation for the course.

2.1 Overall Data Structure

The preprocessed data is distributed in *.pkl* files (*train_graphs.pkl*, *validation_graphs.pkl* and *test_graphs.pkl*). These files are standard Python "pickles" containing a list of *torch_geometric* data objects.

We provide a custom *PreprocessedGraphDataset* class in *data_utils.py* to easily load these files for use in your PyTorch models. Each molecule in the dataset is represented as a single *torch_geometric.data.Data* object. This object bundles all information for one graph. The key attributes you will use are:

- *data.x*: The node (atom) feature tensor.
- *data.edge_index*: The graph connectivity (bond) tensor.
- *data.edge_attr*: The edge (bond) feature tensor.
- *data.id*: A unique identifier string for each molecule. This is used to align the graphs with their text captions and for making your final submission.
- *data.description*: The ground-truth human-readable text. (This will be absent for the test set).

2.2 Graph Node Features

The node features are a tensor where each row represents one atom (node) and contains 9 categorical indices. Please note that these are not one-hot encodings, but rather integer indices. In the provided baseline in Section 3, they are fed into *torch.nn.Embedding* layers of the Graph Convolutional Network. The 9 features for each atom are, in order:

1. *atomic_num*: Index of the atom's atomic number (covering atomic numbers 0 to 118).
2. *chirality*: Index of the atom's chirality tag (you may find the complete list in Appendix A.1).

3. *degree*: Index of the atom's total degree (representing number of bonded neighbors 0 to 10).
4. *formal_charge*: Index of the atom's formal charge (representing charges -5 to +6).
5. *num_hs*: Index of the total number of hydrogens attached (representing 0-8 hydrogens).
6. *num_radical_electrons*: Index of radical electrons (representing 0-4 radical electrons).
7. *hybridization*: Index of the atom's hybridization state (Unspecified, S, SP, SP2, SP3, SP3D, SP3D2, Other).
8. *is_aromatic*: Boolean index for aromaticity.
9. *is_in_ring*: Boolean index for ring membership.

The specific integer-to-value mappings are defined in the *x_map* dictionary in *data_utils.py*.

2.3 Graph Edge Features

Edge information is stored in two tensors:

- *data.edge_index*: This tensor defines the graph connectivity in COO (coordinate) format. For each undirected bond between atoms i and j, two directed edges are included in this list: one from i to j and one from j to i.
- *data.edge_attr*: This tensor provides features for each directed edge listed in *edge_index*. Like the node features, these are 3 categorical indices per edge, corresponding to:
  1. *bond_type*: Index of the bond type (you may find the complete list in Appendix A.2).
  2. *stereo*: Index of the bond's stereochemistry (Stereonone, Stereoany, Stereoz, Stereoe, Stereocis, Stereotrans).
  3. *is_conjugated*: Boolean index for conjugation.

The mappings for these indices are defined in the *e_map* dictionary in *data_utils.py*. The edges are deterministically sorted to ensure consistent ordering.

3 Baseline Implementation

To ensure a smooth start, we provide a complete, functioning baseline model. This baseline frames the problem not as a generative task, but as an embedding-similarity-based retrieval task.

Baseline Architecture:

- Graph Encoder: A simple Graph Convolutional Network (GCN) with a simple linear projection layer is used to process the molecular graph and produce a d-dimensional vector embedding, v_{graph}.
- Text Encoder: A pre-trained BERT model is used to encode the corresponding text caption into a d-dimensional embedding, v_{text}.

Training Objective: The model is trained using mean square error (MSE) as a similarity-based loss function. The objective is to train the GCN such that for every molecule i, the

embedding of its graph (v^i_{graph}) is "pulled" as close as possible to the embedding of its corresponding text caption (v^i_{text}) in the shared latent space.

Prediction Mechanism: During inference on the test set, this baseline does not generate new text. Instead: the GCN computes the graph embedding v^{test}_{graph} for a test molecule. It then searches the entire training set to find the text caption whose embedding v^{train}_{text} is the nearest neighbor (i.e., has the highest cosine similarity) to the query v^{test}_{graph}. This retrieved training set caption is used as the final prediction for the test molecule.

4 Evaluation and Leaderboard

All submissions will be evaluated as a text generation task, regardless of the method used. Your model's predicted caption for each test molecule will be compared against its hidden ground-truth caption using a standard suite of NLP metrics.

- BLEU-4 F1 Score: A precision-oriented metric that measures the overlap of n-grams (word sequences) between the generated and reference texts.
- BERTScore with *RoBERTa-base*: A modern, semantic-focused metric that computes the cosine similarity between the embeddings of tokens in the generated and reference texts, using a pre-trained BERT-style model.

This suite ensures a holistic evaluation, balancing lexical/syntactic overlap (BLEU) with semantic accuracy (BERTScore). Your final rank on the private leaderboard will be determined by a composite score derived from these two metrics.

5 Suggestions

The provided baseline is a preliminary approach, and there is substantial room for innovation. We propose two potential paths for improvement below, but we strongly encourage you to be creative and explore your own novel methods.

- **Enhancing the retrieval system:** The baseline's effectiveness is entirely dependent on the quality of the shared embedding space.
- **Building a generative model:** Instead of retrieving an existing caption, you can train a model to generate a new one from scratch. This is a more complex but also more powerful and flexible approach.
- **Or something else?** We will leave you to figure it out.

Note on Computational Constraints:

If your approach involves a Large Language Model (LLM) as the decoder, we strongly advise you to start with smaller pre-trained models (e.g., GPT-2 Medium at 345M parameters) considering the significant computational complexity.

In case your implementation is limited by computational resources, please do not be discouraged. We would still like to hear about your complete, ambitious idea in your report and during your final presentation.

Please remember that the scores on the leaderboard represent only a portion of our final evaluation. We are more interested in the novelty of your approach, your engineering abilities, and the motivations behind your design choices.

6 Participation and Submission

To participate in the challenge, please first fill in the following form before 15/12/2025:

https://forms.gle/jfUKPKdWGHjSMWpZ8

The challenge is hosted on Kaggle, a platform for predictive modeling on which companies, organisations, and researchers post their data, and statisticians and data miners from all over the world compete to produce the best models. You may join the Kaggle competition through the link below:

https://www.kaggle.com/t/9c777e8a055d41e7ab4fc7a6a8e3d481

Rules. The following rules apply to this challenge: (i) one account is allowed per participant; (ii) there is a limit on the size of each team (at most 3 members); (iii) privately sharing code outside of teams is not permitted; (iv) there is a limit in the number of submissions per day (at most 15 entries per day); (v) any use of external data is now allowed, and any usage of external code should be declared (including word embeddings and pre-trained models). (vi) **your code must be reproducible**: please provide all the necessary in the submission so that we can run your code and achieve the same results as you report.

Evaluation and Submission. Your final evaluation for the project will be based on (1) the presentation you will give (50%), (2) your total approach to the problem and the quality of the report (30%), and (3) your position on the private leader-board and the score that will be achieved (20%).

Deliverable. As part of the project, you have to submit the following:

- At least one submission on Kaggle in the format of a *csv* file with "ID" and "description" as two columns, storing your prediction of the entire test split. Please note that the evaluation of every submission on Kaggle may take around 10 minutes.
- A 4-5 pages report (excluding references), in which you should describe the approach and the methods that you used in the project. Since this is a real generative task, we are interested to know how you dealt with each part of the pipeline, e.g., how you created your representation, which features did you use, which contrastive learning techniques did you use and why, the performance of your methods (loss, accuracy and training time), approaches that finally didn't work but are interesting, and in general, whatever you think that is interesting to report.
- A directory with the code of your implementation (not the data, just the code).
- Create a *.zip* file containing the code and the report and submit it via the link below, make sure that the name of the file is as follows: **team_name.zip.**
  https://forms.gle/avw9KR2rBYW4iARK9

- **Deadline (for both competition and submitting code and report): 15/01/2026 23:59.**

Presentation: As mentioned above, you will be asked to present the approach you followed. Therefore, you will need to prepare some slides (using *ppt* or any other tool you like).

Date of presentation: We will publish the presentation date after the team submission deadline.

References

[1] Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between Molecules and Natural Language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5723–5736. Association for Computational Linguistics, 2022.

[2] Sixing Ji, Yumo Zhang, Yuan Zhao, Yutong Wan, Ziang Guo, Ming Zhong, Meng Niu, Philip S Yu, and Dejing Dou. A survey on graph-to-text generation: Tasks, methods, and challenges. *arXiv preprint arXiv:2203.04803*, 2022. URL `https://arxiv.org/abs/2203.04803`.

[3] Shuzhi Yuan and Michael Faerber. Evaluating generative models for graph-to-text generation. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 1413–1422. INCOMA Ltd., 2023.

A Appendix

A.1 Chirality Types

In chemistry, chirality is a critical stereochemical concept. A molecule is considered chiral if it is non-superimposable on its mirror image. In our dataset, the chirality tag for an atom (node) describes the stereochemical configuration of the atom. Here is a detailed explanation of each indexed value:

1. *CHI_UNSPECIFIED*: This is used to mark an atom that is not chiral (i.e., it is "achiral") or one whose chirality is unknown, undefined, or irrelevant to the task.
2. *CHI_TETRAHEDRAL_CW*: Tetrahedral Chirality - Clockwise (CW). This is the most common type of chiral center, typically a carbon atom bonded to four different groups. This tag specifies that its neighboring groups (when ordered by RDKit's internal rules) are arranged spatially in a clockwise direction.
3. *CHI_TETRAHEDRAL_CCW*: Tetrahedral Chirality - Counter-Clockwise (CCW). The opposite of CW, this tag specifies that the neighboring groups are arranged in a counter-clockwise direction.
4. *CHI_TETRAHEDRAL*: Tetrahedral Chirality (General). This tag indicates that the atom is a tetrahedral chiral center, either clockwise or counter-clockwise.
5. *CHI_ALLENE*: Allene-type Chirality (Axial Chirality). Allenes are compounds with cumulative double bonds (C=C=C). When the substituents at the ends of the allene are different, they create a special type of chirality called Axial Chirality, as the two end groups lie in perpendicular planes. This tag is used to mark this type of chiral center (typically the central carbon of the allene).
6. *CHI_SQUAREPLANAR*: Square-Planar Chirality. This typically occurs in transition metal complexes. A central atom (like Platinum) bonded to four different ligands in a square-planar geometry can be chiral (a form of planar chirality).
7. *CHI_TRIGONALBIPYRAMIDAL*: Trigonal-Bipyramidal Chirality. This is another geometry found in (often 5-coordinate) complexes or specific organophosphorus compounds. The five ligands are arranged around the central atom in a trigonal bipyramid, and specific arrangements can lead to chirality.
8. *CHI_OCTAHEDRAL*: Octahedral Chirality. Common in 6-coordinate transition metal complexes. When ligands are arranged in an octahedral geometry around a central metal atom in specific ways (e.g., three bidentate ligands), the entire complex can be chiral.
9. *CHI_OTHER*: This is a fallback tag for other (often rare) forms of chirality that do not fall into any of the specific categories listed above.

A.2 Bond Types

The `bond_type` feature describes the nature of the chemical bond between two atoms. These categories cover a wide range of interactions, from standard covalent bonds to more complex chemical concepts. Here is a detailed explanation of each indexed value:

1. *UNSPECIFIED*: The bond type is unknown, undefined, or irrelevant to the task.

2. *SINGLE*: A standard single covalent bond, where two atoms share one pair of electrons (e.g., a C-C bond).
3. *DOUBLE*: A double covalent bond, where two atoms share two pairs of electrons (e.g., a C=C bond).
4. *TRIPLE*: A triple covalent bond, where two atoms share three pairs of electrons (e.g., a C≡C bond).
5. *QUADRUPLE*: A quadruple covalent bond, a rare bond involving four pairs of electrons, typically found between transition metal atoms.
6. *QUINTUPLE*: A quintuple covalent bond, an extremely rare bond involving five pairs of electrons, also found between metal atoms.
7. *HEXTUPLE*: A hextuple covalent bond, the highest order bond known, found in specific dimetal complexes.
8. *ONEANDAHALF*: A bond with a formal order of 1.5, often used to represent delocalized bonds in resonance structures (e.g., in a carboxylate anion).
9. *TWOANDAHALF*: A bond with a formal order of 2.5, representing further delocalization.
10. *THREEANDAHALF*: A bond with a formal order of 3.5.
11. *FOURANDAHALF*: A bond with a formal order of 4.5.
12. *FIVEANDAHALF*: A bond with a formal order of 5.5.
13. *AROMATIC*: A delocalized bond found within an aromatic ring (e.g., in benzene). It is distinct from single or double bonds and typically has a bond order of ∼1.5.
14. *IONIC*: Represents an ionic bond, the electrostatic attraction between two oppositely charged ions (e.g., Na^+ and Cl^-).
15. *HYDROGEN*: A weak electrostatic attraction between a hydrogen atom (covalently bonded to a highly electronegative atom like O or N) and another nearby electronegative atom.
16. *THREECENTER*: A "three-center, two-electron" bond, where three atoms share two electrons (e.g., the B-H-B bonds in diborane).
17. *DATIVEONE*: A single-electron dative bond, where one atom contributes one electron to a bond with another atom that contributes none.
18. *DATIVE*: A coordinate (or dative) covalent bond, where one atom contributes *both* shared electrons to the bond.
19. *DATIVEL*: Used internally by RDKit to specify the direction of a dative bond (from the Left atom) in a SMILES string.
20. *DATIVER*: Used internally by RDKit to specify the direction of a dative bond (from the Right atom) in a SMILES string.
21. *OTHER*: A fallback category for any bond type that does not fit into the other classifications.

22. *ZERO*: A special type used to represent non-bonded interactions or to manage atoms in the context of chemical reactions.