Data Challenge ALTeGraD Report
Molecule Retrieval with Natural Language Queries
Team Numpy

Adib Habbou
MVA - ENS Paris-Saclay
adib.habbou@ens-paris-saclay.fr

Steven Zheng
MVA - ENS Paris-Saclay
steven.zheng@ens-paris-saclay.fr

Teddy Alexandre
MVA - ENS Paris-Saclay
teddy.alexandre@ens-paris-saclay.fr

1 INTRODUCTION

This project delves into the intricate task of retrieving molecules, represented as graphs, using natural language queries. The challenge lies in integrating information encoded in text descriptions and molecular graphs. By developing a contrastive training pipeline employing distinct encoders specialized in handling textual data and graphical data, we are able to capture both nuances.

2 FEATURES REPRESENTATION

To represent the features of molecules in our dataset, we employ a combination of graph-based and text-based representations. The graph-based representation involves encoding the molecular structure as a graph, where nodes represent atoms and edges represent chemical bonds. For this purpose, we utilize the *GraphDataset* class provided in the baseline code, that processes raw graph data, extracting edge indices and node features.

Moreover, we integrate textual information by employing a tokenizer to encode natural language descriptions associated with each molecule. Utilizing the Hugging Face Transformers library through the custom *TextDataset* provided in the baseline code, we convert these text descriptions into numerical representations. The encoding process includes the application of attention masks, ensuring that the model attends to relevant portions of the input sequence while disregarding padding tokens.

The resulting text encodings, enriched with attention masks, are merged with the graph-based features. The obtained feature representation for each molecule comprises graph information, including edge indices and node features, along with tokenized and encoded natural language descriptions. This combined representation allows our model to learn relationships between molecular structures and associated text descriptions.

3 PIPELINE SETUP

3.1 Parallelization

In order to harness the computational power of two GPU T4 devices on Kaggle, we strategically parallelized our model by assigning distinct encoders to separate devices. The graph encoder, responsible for processing graph-based features, is allocated to *cuda:0*, while the text encoder, handling tokenized and encoded natural language descriptions, resides on *cuda:1*. This parallelization enables concurrent and efficient utilization of both GPUs during training, thereby accelerating the overall computation and enhancing the model's capability to handle large-scale molecular datasets.

3.2 Contrastive Loss

The loss used is computed based on the dot product of the two text and graph embeddings. The dot product results in a matrix of logits, capturing the pairwise similarities between the vectors. The Cross Entropy Loss function is then applied twice: first, with the logits matrix and labels representing the indices along the diagonal, and second, with the transposed logits matrix.

3.3 Learning Rate Scheduler

The Cosine Annealing Learning Rate Scheduler is designed to improve the training process by gradually reducing the learning rate in a smooth and cyclical manner. The scheduler follows a cosine curve, starting with a higher learning rate and smoothly annealing it towards a minimum value *eta_min*. This cyclical adjustment helps the model escape from local minima and explore the loss landscape more effectively, potentially leading to better convergence and improved generalization. The *T_max* parameter specifies the number of iterations in one cosine cycle, and as training progresses, the learning rate oscillates between its maximum and minimum values.

3.4 Optimizer

AdamW optimizer, a variant of the Adam optimization algorithm, which is commonly used for training neural networks. The optimizer is responsible for updating the model's parameters during the training process to minimize the loss.

The optimizer incorporates *weight decay* (L2 regularization) with a coefficient of 0.01 to prevent overfitting by penalizing large weights. The *betas* parameter sets the exponential decay rates for the moving averages of past gradients and squared gradients, influencing the momentum-like behavior of the optimizer.

3.5 Training

In each epoch of the training loop, the model's two components, *model_part1* and *model_part2*, are set to training mode. For each batch in the training data, the graph and text encoders process the respective inputs, and the contrastive loss is calculated between the encoded graph and text representations. The model parameters are updated using back-propagation, and a learning rate scheduler is applied, to dynamically adjust the learning rate.

After training, the model components are switched to evaluation mode, and the validation loss is computed. The best validation loss is tracked, and if an improvement is observed, a checkpoint containing the model states, optimizer, scheduler, and relevant metrics is saved.

4 GRAPH ENCODER

4.1 Graph Attention Networks

The authors of [1] propose Graph Attention Networks (GATs), a novel neural network architecture for processing graph-structured data. GATs leverage masked self-attentional layers to overcome limitations of previous methods, such as graph convolutions. By allowing nodes to attend over their neighborhood's features, the model can specify different weights to different nodes without costly matrix operations.

The models achieve state-of-the-art results across various graph benchmarks, demonstrating their effectiveness. The attention mechanism, inspired by recent work on sequence-based tasks, efficiently handles variable-sized inputs and is applied to node classification in graph-structured data. GATs offer computational efficiency, interpretability, and inductive learning capabilities, distinguishing them from prior approaches in the field.

The following equation shows how are computed the importance of the features of the neighbor j to the node i in a GAT model using node representations:

$$e(h_i, h_j) = \text{LeakyReLU}\left(a^T \cdot \left[\mathbf{W}h_i || \mathbf{W}h_j\right]\right) \tag{1}$$

After applying a softmax, we obtain the normalized attention scores denoted $\alpha_{ij} = \text{softmax}(e(h_i, h_j))$.

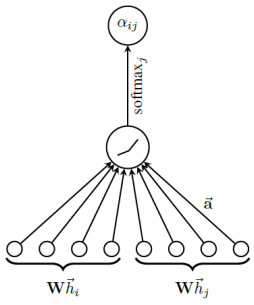

Figure 1: Attention mechanism utilized by GAT

GAT is usually used with several heads. Below we can see a case of multi head attention with nheads = 3, where each arrow style and color refers to a separate attention computations. The features are finally aggregated or averaged from each head.

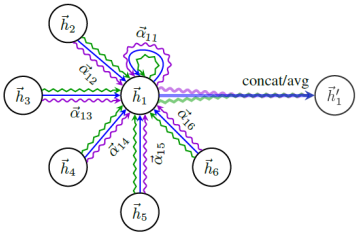

Figure 2: Multi-head Attention mechanism utilized by GAT

4.2 Graph Attention Networks V2

The paper [2] addresses limitations in Graph Attention Networks, by revealing that GATs compute a restricted form of attention called static attention, hindering their expressiveness. The authors propose GATv2, a dynamic graph attention variant, just by modifying the order of operations.

GATv2 is demonstrated to be strictly more expressive than GAT, outperforming it across various benchmarks. The paper introduces formal definitions for analyzing graph attention mechanisms expressive power and shows that GAT cannot compute dynamic attention. The proposed fix involves a simple modification to the attention function, providing a more accurate representation of attention dynamics in graph structures.

The following equation shows how are computed the importance of the features of the neighbor j to the node i in GATv2 model using node representations:

$$e(h_i, h_j) = a^T \text{LeakyReLU}\left(\mathbf{W} \cdot \left[h_i || h_j\right]\right) \tag{2}$$

4.3 Super Graph Attention Networks

In the paper [3], the authors introduce a self-supervised graph attention network (SuperGAT) designed to enhance graph attention models for noisy graphs. The study explores two attention mechanisms, GAT's original single-layer neural network (**GO**) and dot-product (**DP**), and use also a scaled dot-product (**SD**) and a mixed version of GO and DP (**MX**) as building blocks for SuperGAT.

The proposed model leverages self-supervised tasks to predict edges, using the presence and absence of edges to inform the importance of relationships between nodes. The authors conduct experiments to analyze how graph characteristics such as homophily and average degree impact the effectiveness of attention mechanisms. The results lead to a proposed recipe for designing graph attention with edge self-supervision tailored to specific graph characteristics.

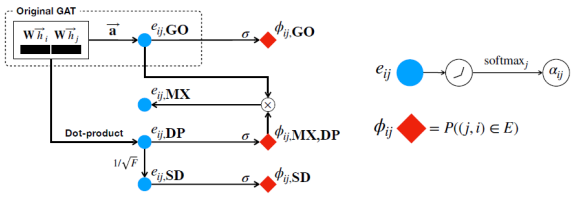

Figure 3: Attention mechanism utilized by SuperGAT

The following equations shows how are computed the importance of the features of the neighbor j to the node i in SuperGAT model using node representations:

$$\begin{aligned}
e_{GO}(h_i, h_j) &= \text{LeakyReLU}(a^T \cdot \left[\mathbf{W}h_i || \mathbf{W}h_j\right]) \\
e_{DP}(h_i, h_j) &= \text{LeakyReLU}((\mathbf{W}h_i)^T \cdot \mathbf{W}h_j) \\
e_{SD}(h_i, h_j) &= e_{DP}(h_i, h_j)/\sqrt{F} \quad \text{(F: number of features)} \\
e_{MX}(h_i, h_j) &= e_{GO}(h_i, h_j) \times \sigma(e_{DP}(h_i, h_j)) \quad (\sigma\text{: sigmoid})
\end{aligned} \tag{3}$$

5 TEXT ENCODER

5.1 BERT

Bidirectional Encoder Representations from Transformers, serves as the pioneering architecture for contextualized language representation. Developed by Google [4], BERT revolutionized natural language processing by introducing a bidirectional context-aware approach to pre-training language models. Unlike traditional models, BERT considers both left and right context in a sentence, capturing intricate semantic relationships.

5.2 DistilBERT

In the paper [5], the authors propose a method for pre-training a smaller general-purpose language representation model, named DistilBERT, through knowledge distillation. They show that it is possible to reduce the size of a BERT model by 40% while retaining 97% of its language understanding capabilities and achieving a 60% faster inference time. The proposed triple loss combines language modeling, distillation, and cosine-distance losses during pre-training. DistilBERT's performance is comparable to larger models on downstream tasks, making it a more efficient alternative.

The use of DistilBERT as in the baseline code provided, contributes to the overall feature representation of molecules by encoding natural language descriptions associated with each molecule, while keeping the training time reasonable.

5.3 SciBERT

SciBERT, a pretrained language model based on BERT, is introduced in [6] to address the lack of labeled scientific data for NLP tasks. Leveraging unsupervised pretraining on a large corpus of scientific publications, SciBERT demonstrates significant improvements over BERT on various downstream scientific NLP tasks. The model is evaluated on tasks such as sequence tagging, sentence classification, and dependency parsing, achieving new state-of-the-art results. SciBERT is trained on a corpus of 1.14M scientific papers and uses a new scientific vocabulary.

By integrating SciBERT as our text encoder, we aim to improve the model's performance on downstream tasks related to molecular retrieval. Surprisingly the model didn't achieved as good performance as DistilBERT, potentially because the training time was double, so we weren't able to run multiple epochs.

5.4 BioBERT

BioBERT, a domain-specific language representation model, is introduced for biomedical text mining in [7]. Pretrained on large-scale biomedical corpora, BioBERT outperforms BERT and previous state-of-the-art models in various biomedical tasks, including named entity recognition, relation extraction, and question answering. The model's pre-trained weights are made freely available, showcasing its effectiveness in understanding complex biomedical texts. The biomedical domain's distinct word distribution is considered, highlighting the importance of training language models on domain-specific corpus.

BioBERT's contributed to the text encoding information by using biomedical texts associated with molecules. This domain-specific knowledge didn't enhance the model's capability, again mainly because we didn't train it on several epochs because of the huge training time.

5.5 BioMegatron

The study [8] investigates factors influencing the performance of domain-specific language models in biomedical text mining, exploring sub-word vocabulary, labeling methods, model size, and domain transfer. The BioMegatron model, trained on a larger biomedical corpus, consistently improves performance on biomedical benchmarks, surpassing the previous state-of-the-art in named entity recognition, relation extraction, and question answering.

By incorporating BioMegatron as our text encoder, we maybe could be able to reach a new step on our learning process, and achieve a better score at the end. But we couldn't even run the model, since the GPU was out of memory.

5.6 Summary

The following tables gives us the number of parameters and the number of words used as training data for each of the cited models that could be used as text encoders:

Model	Parameters	Training Words
DistilBERT	66 M	3.3 B
SciBERT	110 M	3.17 B
BioBERT	110 M	4.5 B
BioMegatron	345 M	6.1 B

Table 1: Number of parameters and training words

To have a better idea of the difference between does models, we choosed to make a visual representation by plotting the above table:

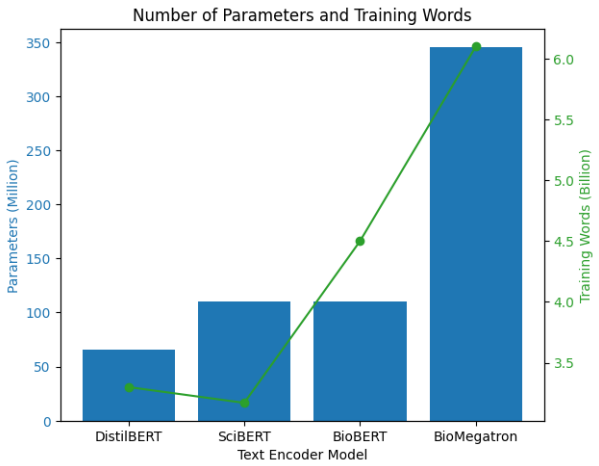

Figure 4: Number of parameters and training words

6 TRAINING PARAMETERS

We developed three distinct Graph Attention Encoders, each with 3 layers of GAT, GATv2, and SuperGAT, incorporating 25 heads per layer. After each layer, we did a ReLU activation. At the end of the conv layers we did a scatter max pooling (by taking the maximum values along dimension 0). Finally, we applied two linear layers.

The models were trained for different number of epochs and batch size, resulting in different training times. This unified architecture across GAT, GATv2, and SuperGAT allowed us to leverage the unique attention mechanisms of each variant while maintaining consistency in the overall model structure and training process.

The text encoder used each time was DistilBERT, as it's the one who achieves the best results regarding the validation loss and the Label Ranking Average Precision score.

We also kept, as in the baseline, the cosine similarity for computing the similarity between text and graph embedding for the generation of the final submission file. As it outperforms the other similarities we tested (euclidean, minkowski, jaccard and cityblock).

Model	Epochs	Batch Size	Training Time
GAT	110	16	12 hours
GATv2	120	16	11 hours
SuperGAT	160	16	11 hours

Table 2: Training information for different models

7 ENSEMBLE APPROACH

To optimize predictive performance, we implemented an aggregation strategy by harnessing the strengths of three distinct Graph Attention Network models: GAT, GATv2, and SuperGAT. Each model was independently trained to capture diverse aspects of graph structures and relationships within the molecular data. This collaborative approach was motivated by the belief that the collective intelligence of multiple models could yield superior predictive results compared to any individual model in isolation, as proved in [9].

The individual predictions generated by these models were aggregated using a simple averaging technique. This approach aimed to capitalize on the unique learning capabilities of each model, creating a well-rounded and more reliable prediction. By combining the outputs of GAT, GATv2, and SuperGAT, our aggregation strategy aims to mitigate the limitations of individual models and provide a more comprehensive solution.

One notable concern revolves around the assumption that each model contributes equally valuable insights. In reality, the models may have varying degrees of expertise or specialization, leading to potential biases in the aggregated predictions. Furthermore, the simple averaging technique employed may not fully capture the intricate relationships between different models, potentially limiting the exploitation of synergies.

8 RESULTS

8.1 Loss Evolution

We clearly see that the 3 models tends to achieve validation loss above 0.08 after training each one of them separately.

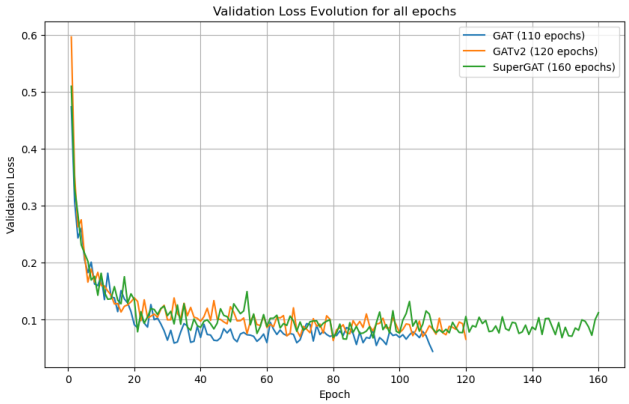

Figure 5: Evolution of the validation loss for all the epochs

In order to have a better insights we plot the evolution of the loss starting from the 20th epoch:

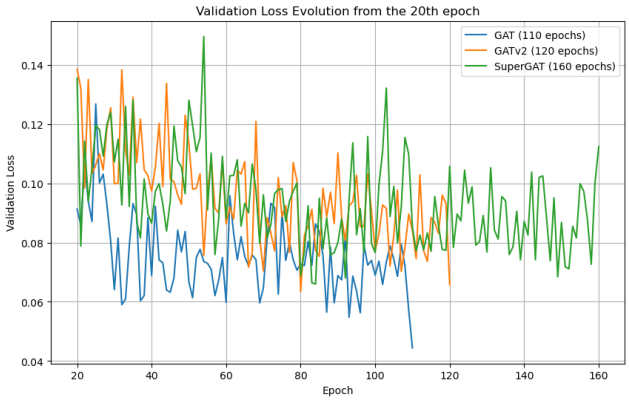

Figure 6: Evolution of the validation loss starting from 20th

GAT exhibits a rapid and consistent reduction in validation loss after epoch 20, outperforming GATv2 and SuperGAT, which display varying patterns and higher losses throughout the training.

The best validation losses achieved by each model are:

Model	Best Validation Loss
GAT	0.0414
GATv2	0.0614
SuperGAT	0.0629

Table 3: Best validation loss for each model

8.2 Kaggle Scores

Finally, we trained our models on both the train dataset and the validation dataset. The obtained LRAP scores when submitting the predictions of each model to the Kaggle Challenge are:

Model	Score
GAT	0.8538
GATv2	0.8356
SuperGAT	0.8371
GAT + GATv2	0.9009
GATv2 + SuperGAT	0.8988
GAT + SuperGAT	0.8977
GAT + GATv2 + SuperGAT	0.9387

Table 4: Predictions scores for different models

9 CHALLENGES

9.1 GPU Availability and Memory

Since we used Kaggle GPU, we only had access to GPU P100 or GPU T4 limited session of 12 hours and a maximum of 30 hours per week. We also couldn't use some specific text encoders such as BioMegatron, or higher batch size, because we didn't have enough GPU memory since the GPU P100 offered 16 GB and each one of the two GPU T4 offered 15 GB of memory.

9.2 Accelerating Baseline

To enhance the computational efficiency of a dual-encoder model with a text and a graph encoder, we utilized the Accelerate library from Hugging Face for comprehensive GPU distribution, coupled with the implementation of FP16 precision, rather than relying on Automatic Mixed Precision (AMP). This approach facilitates the parallel execution of both encoders across multiple GPUs, fully harnessing the parallel processing capabilities of GPUs to significantly diminish training and inference times. By conducting operations simultaneously, we optimize computational resources and eliminate delays stemming from processors waiting on each other.

The shift to FP16 precision directly contributes to increased computational speed and reduced memory consumption during both training and inference phases, without compromising result quality. This decision allows for more efficient computations and, when combined with gradient scaling, addresses the vanishing gradient problem by adjusting gradients to sustain steady training progression.

By leveraging the Accelerate library for full GPU distribution and adopting FP16 precision, we efficiently parallelized our dual-encoder model's workload across GPUs, drastically reducing training and inference times. This optimization, coupled with gradient scaling, allowed us to increase our batch size significantly to 256. This streamlined approach not only accelerates the model's operations but also enhances scalability and efficiency, enabling the use of larger batch sizes and encoders for faster and more effective model development.

9.3 Masking Strategy

We have particularly focused on varying the mask strategy employed by the BERT architectures. We tried to incorporate random masking. This involves randomly selecting words or tokens within a sentence and replacing them with a mask token, thereby compelling the model to infer meaning and context without relying on a fixed sentence structure or specific word placement.

We motivate our attempt of randomly masking tokens by trying to make the overall model more robust, preventing it from over-relying on certain words or phrases that may frequently appear at specific positions in the training data. However, we were not able to make use of random masking, because of the difficulty to really custom the parameters of the BERT architectures, more specifically on the attention mask attributes.

9.4 Graph Data Transformation

We also explored several graph-related features to enhance molecule representation. Apart from node degree, we considered centrality measures like betweenness and closeness centrality, which could reveal the importance of individual nodes. Graph density and clustering coefficients could provide insights into the overall connectivity and local groupings within the graph. Additionally, the average path length and graph diameter were considered for understanding the overall structure and size of the networks.

But, integrating these graph-theoretic features into our existing model posed significant challenges, mainly the manipulation of the raw data, including a new process for data preparation.

9.5 Unsuccessful Experiments

In our quest to improve the model's performance, we experimented various architectural elements and techniques beyond the standard Graph Attention Networks. This included exploring alternative layers from the PyTorch Geometric library (GCN, GraphSAGE, GIN, AttentiveFP...), yet these modifications did not yield significant improvements. Additionally, we experimented with dropout, batch normalization, and layer normalization techniques, in order to avoid overfitting. We also tested different activation functions (LeakyReLU, ELU, SeLU...), in place of the standard ReLU activation. However, these attempts did not lead to notable enhancements.

10 CONCLUSION

In conclusion, this project addresses the really intricate challenge of retrieving molecules represented as graphs through natural language queries. The developed contrastive training pipeline, incorporating specialized encoders for textual and graphical data, proves effective in capturing nuanced information from both modalities.

Leveraging Graph Attention Networks (GAT, GATv2 and SuperGAT) by using ensemble method, we showcase the adaptability of attention mechanisms in handling diverse graph structures. The proposed aggregation strategy, combining predictions from these models, enhances overall predictive performance. Despite all the above challenges, our approach demonstrates good final results by achieving a fair LRAP score at the end.

REFERENCES

[1] Arantxa Casanova Adriana Romero Pietro Liò Yoshua Bengio Petar Veličković, Guillem Cucurull. "graph attention networks". In *arXiv:1710.10903, Published as a conference paper at ICLR*, 2018.

[2] Eran Yahav Shaked Brody, Uri Alon. "how attentive are graph attention networks?". In *arXiv:2105.14491, Published as a conference paper at ICLR*, 2022.

[3] Alice Oh Dongkwan Kim. "how to find your friendly neighborhood: Graph attention design with self-supervision". In *arXiv:2204.04879, Published as a conference paper at ICLR*, 2021.

[4] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. "bert: Pre-training of deep bidirectional transformers for language understanding". In *"arXiv:1810.04805"*, 2019.

[5] Julien CHAUMOND Thomas WOLF Victor SANH, Lysandre DEBUT. "distilbert, a distilled version of bert: smaller, faster, cheaper and lighter". In *"arXiv:1910.01108"*, 2020.

[6] Arman Cohan Iz Beltagy, Kyle Lo. "scibert: A pretrained language model for scientific text". In *"arXiv:1903.10676"*, 2019.

[7] Sungdong Kim Donghyeon Kim Sunkyu Kim Chan Ho So Jinhyuk Lee, Wonjin Yoon and Jaewoo Kang. "biobert: a pre-trained biomedical language representation model for biomedical text mining". In *"Bioinformatics, Volume 36, Issue 4, Pages 1234–1240"*, 2019.

[8] Evelina Bakhturina Raul Puri Mostofa Patwary Mohammad Shoeybi Raghav Mani Hoo-Chang Shin, Yang Zhang. "biomegatron: Larger biomedical domain language model". In *"arXiv:2010.06060"*, 2020.

[9] Heng Ji Carl Edwards, ChengXiang Zhai. "text2mol: Cross-modal molecule retrieval with natural language queries". In *"2021 Conference on Empirical Methods in Natural Language Processing, pages 595–607"*, 2021.