ALTeGraD 2023 Data Challenge

Molecule Retrieval with Natural Language Queries

Ahmad Rammal
ahmad.rammal@polytechnique.edu

Elyas Benyamina
elyas.benyamina@ens-paris-saclay.fr

Rim El Mallah
elmallahrim@gmail.com

1 Introduction

The objective of the data challenge is to, given a text query and list of molecules (represented as graphs), retrieve the molecule corresponding to the query. This presents some difficulties as textual data and molecular graphs possess fundamentally different information representations. To address this, we used the proposed approach of a co-training framework that simultaneously trains a text encoder and a molecule encoder using contrastive learning. Through this approach, the goal is to create a model that learns to map similar text-molecule pairs closer together while pushing dissimilar pairs apart, ultimately enabling accurate molecule retrieval from text-based queries.

2 Model choice, Tuning and Training

2.1 Model

We keep the same architecture with a graph encoder and a text encoder that we use to do contrastive learning.

2.1.1 Text Encoder

The baseline method uses DistilBERT as the text tokenizer, a lightweight version of the BERT model known for its efficiency and relatively lower computational resource requirement. DistilBERT is effective for general-purpose language understanding tasks, making it a suitable choice for initial implementations.

However, for our approach, we chose to use SciBERT, a variation of the BERT model specifically trained on a large corpus of scientific texts, introduced by Beltagy et al. (2019). The reason behind this choice is that SciBERT's is specific to the scientific domain, and could be particularly beneficial for accurately interpreting and processing the technical language found in chemical descriptions of molecules. This is in contrast to more general-purpose language models, which may not exhibit the same level of proficiency with scientific content.

We also evaluated SBERT (Sentence-BERT), a modification of the original BERT model designed to produce more meaningful sentence embeddings. SBERT adapts the BERT architecture to generate embeddings that are specifically tuned for sentence-level representations. While SBERT offers considerable advantages in generating high-quality embeddings for text, including efficiency and effectiveness in capturing sentence semantics, we ultimately favored SciBERT for our specific application.

To improve the training process and enhance model efficiency, we integrated an additional linear layer into our architecture. This layer serves the pivotal role of transforming the original 768-dimensional

embeddings produced by BERT into a more manageable 350-dimensional space. This dimensional reduction was chosen to balance between retaining significant semantic information and ensuring computational efficiency. Furthermore, we added a layer normalization step, which significantly improves the performance of cosine similarity calculations by ensuring consistent scales across the embeddings.

2.1.2 Graph Encoder

The input process for the graph encoder, which we have not modified, involves using pre-established molecular graph representations and substructure token embeddings. The molecular graphs, representing atoms as nodes and bonds as edges, are combined with a given dictionary that maps each node's substructure with its embedding.

For the graph encoder, we tested various architectures and modifications to try to enhance the performance of the baseline model. Our experiments included the following:

- Of course, we first tested the baseline GCN model. This model utilizes three GCN convolution layers, which are effective in capturing the local graph structure and features of nodes.
- Our first idea was to replace the GCN layers with Graph Attention Network (GAT) layers. GATs introduce an attention mechanism, allowing the model to weigh the importance of nodes in a neighborhood differently, potentially capturing more nuanced relationships in the graph data. This made us get better results than with the GCNs.
- As we got good results with the GATs, we tried to expand the GAT model by increasing the number of GAT layers from three to five. This was done to investigate if a deeper attention-based network would yield better representations of the molecular graphs. We ended up with some overfitting and didn't get better result than the model with only 3 layers.
- We also experimented a hybrid approach combining GCN, GAT, and Graph Isomorphism Network (GIN) layers. This combination aimed to leverage the strengths of each graph neural network type to capture different aspects of the graph's structure and features. After passing the input through each of this convolution independently, we combined them by getting the mean along the axis=1 or trying to sum them up. We didn't get better results.
- In another variation, we explored the impact of removing the fully connected layers that followed the convolutional layers. This was done to assess whether a purely convolution-based approach would be more effective in encoding the graph information. And it wasn't the case so we didn't keep this idea.
- Similarly to the text encoder, we added a normalization layer at the end.

The results obtained with the different approaches can be seen in Table 1.

2.2 Loss

In our approach, we explored four different loss functions for optimizing the model: the traditional contrastive loss introduced by Radford et al. (2021), a modified version inspired by Edwards et al. (2021), the CosineEmbeddingLoss from PyTorch, and the Triplet Loss.

2.2.1 Contrastive Loss

The baseline model uses the following standard contrastive loss function:

$$\text{CL}(v_1, v_2) = \text{CE}(\text{logits}, \text{labels}) + \text{CE}(\text{logits}^T, \text{labels})$$

where $\text{logits} = v_1 \times v_2^T$, n is the number of rows of logits and $\text{labels} = \text{diag}(\text{range}(0, \text{logits.shape}[0])) = \begin{bmatrix} 0 & & \\ & \ddots & \\ & & n-1 \end{bmatrix}$ and CE is the cross-entropy loss.

So this loss computes the logits by taking the dot product of two sets of embeddings (in our case text and graph embeddings) and applies cross-entropy loss using the identity matrix as labels to match the

corresponding graph and texts and all the other non-matching pairs in the batch serve as negatives (correponding to the zeros in the identity matrix).

To improve how our model tells apart embeddings that are similar from those that are not, we added a temperature parameter to the contrastive loss function. This parameter helps to improve the model's decisions by adjusting the scale of logits, which smooths out the distribution of scores and makes it easier to differentiate between correct matches and incorrect ones. The modified loss function would look like this:

$$\text{CL}_\tau(v_1, v_2) = \text{CE}\left(\frac{\text{logits}}{\tau}, \text{labels}\right) + \text{CE}\left(\frac{\text{logits}^T}{\tau}, \text{labels}\right)$$

Instead of setting the temperature value once and for all, we made it adjustable during training, allowing the model to learn the best setting from the data it's trained on.

2.2.2 Negative Sampling Contrastive Loss

Inspired by the Text2Mol model introduced by Edwards et al. (2021), we tried a modified version of the contrastive loss with negative sampling defined as:

$$\text{NegativeSamplingCL}(v_1, v_2, \text{labels}) = \text{BCEL}\left(\text{logits}, \text{eye}\right) + \text{BCEL}\left(\text{logits}^T, \text{eye}\right)$$

where $\text{logits} = v_1 \times v_2^T$ and $\text{eye} = \text{diag\_embed}(\text{labels})$ and BCEL is the Binary Cross-Entropy with Logits Loss.

This loss function introduces the concept of negative sampling by modifying the identity matrix with zeros for mismatched pairs, thereby creating a binary classification task.

To adapt to our specific requirements, we modified the data loader to output a triplet for each instance: a graph, a text description, and a label indicating their correspondence. The label is binary, with 1 signifying that the graph and text accurately correspond to each other and 0 indicating a non-corresponding pair.

Specifically, for each text description, with 0.5 probability we take the corresponding graph (molecule) so with label 1 and with 0.5 probability we take a non corresponding graph and with label 0. This allows us to use a loss that takes as input the text embeddings, graph embeddings and also the labels.

Advantages of Negative Sampling The introduction of negative sampling in the loss function could offer several benefits. Firstly, it allows for a more targeted approach to learning distinctions between matching and non-matching pairs. This is particularly useful in scenarios where the model needs to differentiate between closely related but distinct items. Secondly, by focusing on specific negative samples, the model is trained more effectively, avoiding potential issues of information leakage between encoders. This ensures that the model considers both the molecule and its textual description, leading to a more robust cross-modal understanding.

Similarly to the standard Contrastive Loss, we add a trainable temperature scaling parameter to smooth the distribution of scores:

$$\text{NegativeSamplingCL}_\tau(v_1, v_2, \text{labels}) = \text{BCEL}(\frac{\text{logits}}{\tau}, \text{eye}) + \text{BCEL}(\frac{\text{logits}^T}{\tau}, \text{eye})$$

2.2.3 Cosine Embedding Loss

The Cosine Embedding Loss is defined as follows:

$$\text{CEL}(v_1, v_2, \text{label}, \text{margin}) = \begin{cases} 1 - \cos(v_1, v_2) & \text{if } label = 1 \\ \max(0, \cos(v_1, v_2) - \text{margin}) & \text{if } label = -1 \end{cases}$$

Similarly to the Negative Sampling Contrastive Loss, we created a custom dataloader where for each text description, with 0.5 probability we take the corresponding graph (molecule) so with label 1 and with 0.5 probability we take a non corresponding graph and with label -1.

2.2.4 Triplet Loss

The triplet loss is defined as follows:

$$\text{TripletL}(v, v_p, v_n, \text{margin}) = max(d(v, v_p) - d(v, v_n) + \text{margin}, 0)$$

In this loss we consider three embeddigs:

1. Embedding v associated to the graph of a certain molecule.
2. Embedding v_p (called positive) linked to a text describing the previous molecule.
3. Embedding v_n (called negative) linked to a text describing another molecule.

Intuitively, we want that the distance (here the cosine similarity) between v and v_p to be much smaller than the distance between v and v_n.
We add a margin in order that the model learns to distance negative embeddings from positives and to facilitate the pairing of texts and graphs.
We created custom dataloaders that are adapted to the following loss. In the training phase, we used two random datasets for each iteration, each containing texts and graphs corresponding to the same molecules. The first dataset contains $(x_{\phi(i)}, y_{\phi(i)})$ and the second $(x_{\psi(i)}, y_{\psi(i)})$, where ϕ and ψ are two bijective functions on the indices of the training set. Then, for each index i, we use the triplet loss four times (we add up the values),

1. First use: $x_{\phi(i)}$ plays the role of embedding v, $y_{\phi(i)}$ plays the role of positive embedding v_p and $y_{\psi(i)}$ plays the role of negative embedding v_n.
2. Second use: $y_{\phi(i)}$ plays the role of embedding v, $x_{\phi(i)}$ plays the role of positive embedding v_p and $x_{\psi(i)}$ plays the role of negative embedding v_n.
3. Third use: $x_{\psi(i)}$ plays the role of embedding v, $y_{\psi(i)}$ plays the role of positive embedding v_p and $y_{\phi(i)}$ plays the role of negative embedding v_n.
4. Fourth use: $y_{\psi(i)}$ plays the role of embedding v, $x_{\psi(i)}$ plays the role of positive embedding v_p and $x_{\phi(i)}$ plays the role of negative embedding v_n.

As showcased in Table 1 we obtained poor performance with this loss while using different margins of 0.1, 0.5, and 1.

2.3 Training

In our training process, we experimented with both freezing and unfreezing layers of the pre-trained text encoder. The outcomes showed that not freezing any layers led to better performance. We explored various model architectures and monitored validation metrics closely to prevent overfitting. Our training sessions ran for 80 to over 100 epochs for the best model, starting with a learning rate of 2×10^{-5} and gradually reducing it to 1×10^{-7}. Due to the limitations of the Google Colab GPU, we set our batch size to 32 and couldn't increase it any further. The total training duration ranged from 6 to 30 hours, varying by the number of epochs, the architecture and the loss function used.

3 Results

The performance of the models is evaluated using the Label Ranking Average Precision (LRAP) metric after applying the cosine similarity between the text embeddings and graph embeddings. For a given sample i, the LRAP score is calculated using the formula:

$$\text{LRAP}(y, y') = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{1}{\|y_i\|_0} \sum_{j:y_{ij}>0} \frac{|L_{ij}|}{\text{rank}_{ij}}$$

where $L_{ij} = \{k : y_{ik} = 1, y'_{ik} \geq y'_{ij}\}$ and $\text{rank}_{ij} = |k : y'_{ik} \geq y'_{ij}|$. Here, $\|y_i\|_0$ is the count of true labels for sample i, L_{ij} is the set of labels that are true and have a score at least as high as the threshold, and rank_{ij} is the rank of the correct label based on the score from the cosine similarity between embeddings.

Model Architecture				Loss				Train Setting		Result
Graph Enc.	Text Enc.	Out. Dim.	Temp. Scaling	Ctr. Loss	Neg. Sam.	Triplet Loss	Cos. Emb.	Epochs	Freeze Layers	Score
3 GAT	SciBERT	350	✓	✓	✓	X	X	120	X	0.87
3 GCN	SciBERT	350	✓	✓	✓	X	X	120	X	0.82
3 GAT	SciBERT	768	✓	X	✓	X	X	100	X	0.81
3 GAT	SciBERT	768	X	X	✓	X	X	60	X	0.79
5 GAT	SciBERT	768	X	X	✓	X	X	60	X	0.76
GCN+GAT+GIN	SciBERT	768	X	X	✓	X	X	60	X	0.74
3 GAT + 2 GCN	SciBERT	768	X	X	✓	X	X	50	X	0.71
3 GCN	SciBERT	768	X	✓	X	X	X	10	X	0.52
3 GAT	SciBERT	768	X	X	X	✓	X	50	X	0.46
3 GAT	SciBERT	768	X	X	X	X	✓	60	X	0.45
3 GCN	DistilBERT	768	X	✓	X	X	X	10	X	0.43
3 GCN	DistilBERT	768	X	✓	X	X	X	10	✓	0.4
3 GCN	SBERT	768	X	✓	X	X	X	20	X	0.38
3 GCN	SBERT	768	X	✓	X	X	X	20	✓	0.35

Blue = Baseline Model, Green = Best Model

Table 1: Scores obtained for multiple models and training strategies tested

The following table summarises the scores obtained for the multiple models and training strategies that we considered for this data challenge:

Our top-performing model utilized three GAT layers for graph encoding and SciBERT for text encoding, with an output dimension of 350. We chose not to freeze any layers based on our prior experiences. Initially, we trained the model using standard contrastive loss for 80 epochs, during which the validation loss plateaued, resulting in a validation score of 0.77. Subsequently, we saved this model and extended the training with Negative Sampling Contrastive Loss for an additional 40 epochs, achieving an improved validation score of 0.87. This model emerged as our most effective configuration.

4 Conclusion

In conclusion, we have demonstrated the efficacy of a co-training framework to address the challenge of molecule retrieval from textual queries. By integrating SciBERT with a graph attention network, we leveraged the strengths of NLP and graph theory to handle the complex domain of chemical compounds. Through our experiments, we noticed that the choice of the loss function was key to obtain good results. We obtained a satisfactory score of 0.878, but lacked the resources and time to test certain approaches. Here are some of them:

- Exploring ensemble methods that combine the predictions from multiple model architectures
- Test other loss functions.
- Finetune some parameters. For example parameters of the scheduler or the margin in Triplet Loss.
- Use Negative Sampling Contrastive Loss at first and then continue the training with other loss functions such as Triplet Loss.

References

Beltagy, I., Lo, K., and Cohan, A. (2019). Scibert: A pretrained language model for scientific text.

Edwards, C., Zhai, C., and Ji, H. (2021). Text2Mol: Cross-modal molecule retrieval with natural language queries. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 595–607, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision.