ALTeGraD 2023 Data Challenge

Molecule Retrieval with Natural Language Queries

L. Hunout S. Hocine L. Haubert

Master MVA
École Normale Supérieure Paris-Saclay

February 14, 2024

Table of Contents

An example of molecule retrieval

Natural language query → Structure of the related molecule

Cholesteryl linoleate is the (9Z,12Z)-stereoisomer of cholesteryl octadeca-9,12-dienoate. It has a role as a human metabolite and a mouse metabolite. It derives from a linoleic acid.

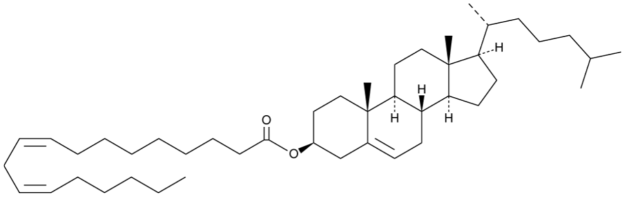

Figure: The idea of molecule retrieval with natural language queries

About the mission

- **Context**:
  - In sciences, medicine, biology, it may be needed to exploit text-molecule relations, e.g for retrieval tasks
  - Retrieval involves obtaining relevant data from a user query
- **Definition**:
  - Extract molecules based on natural language descriptions
  - Queries are textual data and molecules are graph data
- **Key considerations**:
  - Integrating natural language and molecules is complex, due to their different information encoding
  - The challenge consists in building a relevant comparison tool to settle relevant matching scores

Solving and evaluation paradigm

- **Contrastive Learning (CL)**:
  - Text and graph encoders are jointly trained with CL
  - CL maps the similar text-graph pairs close together in the representation space and push other ones apart
  - Contrastive loss is defined with the cross-entropy
- **Label Ranking Average Precision score (LRAP)**:
  - Evaluation metric to quantify the relevance of the retrieval
  - Only one molecule to retrieve $\implies$ LRAP $\sim$ MRR
  - MRR: Average, over the queries, of the inverted retrieval ranks
- **About what follows**:
  - Same paradigm for the baseline and the contributions
  - What will change: models, trainings, adaptations...

Baseline model

- **What was given as a source code**:
  - A pipeline was provided to give a simple baseline
  - LRAP score on half of the test data: about 30%
- **Details about the baseline**:
  - Text encoder: `DistilBERT`
  - Graph encoder: `GCN` (Graph Convolutional Network)
  - Training: 5 epochs, size of batch 32, learning rate 2e-5
- **Our mission**:
  - Develop a new retrieval approach to obtain better LRAP scores
  - Analyze and use, if needed, other text and graph encoders
  - Leverage optimization techniques to improve the pipeline

Contrastive Learning Loss

- Objective: Minimize distance between similar embeddings and maximize for dissimilar pairs
- Cross Entropy (CE) loss function is used for this purpose
- For each graph-text pair, the loss aims to:
  - Maximize probability of correct matching
  - Minimize probability of incorrect matching
- The formula for the contrastive loss using CE is:

$$\mathcal{L}_{CL} = -\frac{1}{N}\sum_{i=1}^{N}\left[\log\left(\frac{\exp(s(z_{\text{graph}_i}, z_{\text{text}_i}))}{\sum_{j=1}^{N}\exp(s(z_{\text{graph}_i}, z_{\text{text}_j}))}\right) + \log\left(\frac{\exp(s(z_{\text{text}_i}, z_{\text{graph}_i}))}{\sum_{j=1}^{N}\exp(s(z_{\text{text}_i}, z_{\text{graph}_j}))}\right)\right]$$

InfoNCE Loss Exploration

- Initially explored InfoNCE loss due to its prevalence in contrastive learning literature
- InfoNCE loss formula:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(s(z_i, z_j)/\tau)}{\sum_{k=1}^{N} \exp(s(z_i, z_k)/\tau)}$$

- Despite it's widely used in contrastive learning, it led to poor convergence in our experiments

Text Encoder

- DistilBERT **model**:
  - Model provided in the baseline, initially adopted
  - Efficient and light due to the distillation process from BERT
  - Train a smaller model to replicate a larger one (here BERT)
- SciBERT **model**:
  - Other tested model, to refine our approach towards efficiency
  - BERT-based architecture, pre-trained on scientific texts
- **Comparison of the two models**:
  - SciBERT: Better convergence rate (which was expected)
  - Both models achieved similar scores/losses upon convergence
  - DistilBERT: Light architecture $\implies$ larger batch size and lower complexity (training) $\implies$ practical advantage
  - We keep DistilBERT as the text encoder

Graph Encoders

- **Common goal**: Encode graph info into numerical representations
- **Shared characteristics**:
  - Designed for graph data
  - Input: Node features, connections. Output: Dense vector
  - Convolutional layers consider graph structure
  - Regularization: Batch Normalization, Dropout
- **Differences**:
  - Types of convolutional layers used
  - Operation on graph data

Graph Encoders: Differences

1. **GCNEncoder:**
   - Uses GCNConv layers to convolve node features, capturing local relationships
   - Aggregates neighbor features without differentiation
2. **GATEncoder:**
   - Utilizes GATv2Conv layers with attention mechanisms, focusing on specific parts of the graph
   - More effective in contexts where relationships between nodes are heterogeneous and the relative importance of neighboring nodes varies
3. **GINEncoder:**
   - Distinguish non isomorphic graphs, relying on the Weisfeiler-Lehman isomorphism test
   - Flexible in representation, doesn't presume structure
   - Enable to get a fine understanding of graph topology

Ensemble Approach

- Correct molecule frequently found among top-ranked molecules
- Matches found jointly by all base models, while some found by certain models only due to differing feature extraction by graph encoders
- Objective: Utilize capabilities of different models via ensemble approach
- Approach: Average results obtained for each model (n-to-n cosine similarity matrix)
- Noticed: Normalizing matrices on rows had little impact on operation result

Optimizations for Training Process (1/2)

- Technical limitations encountered:
  - Training on a single Tesla P100 PCIe 16 GB
  - Time restriction of maximum 30 hours per week
- Realization of the value of time for training models efficiently
- Implemented optimizations to overcome challenges:
  - Optimized DataLoader using Torch library for efficient data loading and reduced wait times
  - Incorporated autocast mixed-precision for exploiting 16-bit floating point calculations, providing significant speedup without compromising results quality
  - Adopted Hugging Face accelerator with Hugging Face library to fully leverage modern hardware capabilities for accelerated training

Optimizations for Training Process (2/2)

- Significance of optimizations:
  - Crucial in addressing CUDA memory errors when processing large volumes of data
  - Enabled increased batch sizes, optimized code, and achieved faster training times
- Contribution to better overall model efficiency

Data and Evaluation

- Data Challenge on Kaggle provides:
  - Host environment and text-graph data
  - Files: train.tsv, val.tsv, token_embedding_dict.npy, cid.graph
  - Test files: test_text.txt, test_cids.txt
- Datasets split into train(80%)/val(10%)/test(10%)
- Shuffled data during training
- Models trained on train data, evaluated by searching all molecules

Results: Baseline Models

- Training details:
  - Adam optimizer with two learning rates:
    - DistilBERT and SciBERT: 3e-5
    - Others: 1e-4
  - StepLR scheduler reduces LR by 50% every 5 epochs
  - Trained for 40 epochs with batch size 64
  - First 256 text tokens used for text encoder
  - Batch size upgraded to 128 with DistilBERT

Results: Baseline Models

- Baseline models:
  - GCN, GAT, Stacked-GAT, and GIN show similar performance
  - Slight edge for Stacked-GAT and GIN with more weight
  - Similar performance attributed to description bottleneck

Model	**Training LRAP**	**Validation LRAP**
`DistilBERT + GCN`	54.26%	75.10%
`DistilBERT + GAT`	54.53%	74.67%
`DistilBERT + Stacked-GAT`	57,68%	77.67%
`DistilBERT + GIN`	57.49%	77.02%
All-Ensemble	NaN	85.68%

Table 1: Results

Results: Ensemble

- Observations:
  - Better LRAPs in validation than training due to dataset size difference
  - LRAP score highly dependent on dataset size
  - Freezing first attention modules in DistilBERT did not improve results
- Ensemble method significantly improves performance
- Increases validation LRAP by 9.5% compared to baseline
- Models with same hyperparameters learn complementary classification methods
- Ensemble incorporates diverse encoder architectures, enhancing efficiency

Conclusion and Future Work

- Study concluded with adoption of DistilBERT as text encoder for its practical advantages
- Ensemble method combining GCN, GAT, and GIN models resulted in significant LRAP score improvements, up to 85.68%
- Future work:
  - Explore data augmentation techniques for more efficient training
  - Consider larger batch sizes for enhanced learning
  - Extend training duration to uncover nuanced patterns
  - Adopt cosine scheduler for dynamic learning rate adjustment
  - Reshuffle and re-split combined datasets to minimize bias
- Intersection of textual and molecular data via machine learning offers new avenues for exploration, highlighting continuous learning in AI

Graph Attention Network v2 (GATv2)

- GATv2 enhances GAT by allowing direct interaction of the attention mechanism with transformed features
- Attention formula:
  $\alpha_{ij} = \text{softmax}_j \left(\text{LeakyReLU} \left(\mathbf{a}^T [W h_i^{(l)} \| W h_j^{(l)}] \right) \right)$
- Update rule: $h_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \cdot W h_j^{(l)} \right)$
  - $\mathcal{N}(i)$ - set of neighbors of node i
  - α_{ij} - attention coefficients
  - W - weight matrix
  - σ - nonlinear activation function
- Increases the expressiveness of the attention mechanism

Graph Isomorphism Networks (GIN)

- GINs aim to match the power of the Weisfeiler-Lehman graph isomorphism test
- Update rule: $h_i^{(l+1)} = \text{MLP}^{(l)}\left((1+\epsilon^{(l)}) \cdot h_i^{(l)} + \sum_{j\in\mathcal{N}(i)} h_j^{(l)}\right)$
    - $\mathcal{N}(i)$ - set of neighbors
    - $\epsilon^{(l)}$ - learnable parameter or fixed scalar
    - $\text{MLP}^{(l)}$ - multi-layer perceptron at layer l
- Effectively captures the graph structure in node representations

Data Augmentation

- Contrastive learning benefits from large datasets
- Graph-text contrastive learning has less data compared to image-text
- Data augmentation can significantly improve model performance such as in DeCLIP

Data Augmentation

- **Text augmentation using EDA**:
  - Synonym substitution, random word shuffling, and word elimination
  - Generates noisy versions of text descriptions to enhance training data
- **Graph augmentation using GraphCL methods**:
  - Effective methods for biological molecule graphs: node dropping and subgraph generation
  - These techniques create variations of the original graph, enriching the dataset

Augmentation Process and Challenges

- For each pair, generate 2 augmented texts and 2 augmented graphs
- Compute embeddings and calculate loss for all augmented pairs and average them
- **Implementation issue**: Increased GPU memory cost hindered successful implementation