lstm_generative_model_flow

Objective

Implement fine-tuned LSTM with reasonable treatment of molecules for designing focused library.

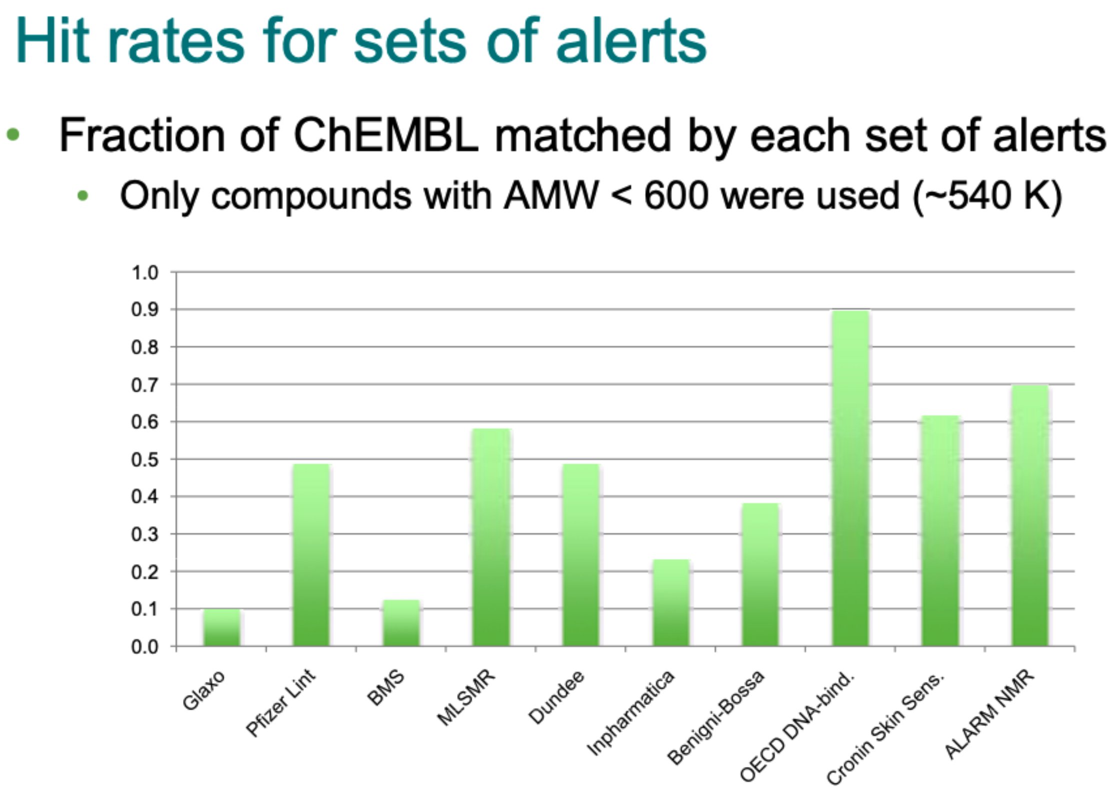
1. Token generation (selection)
2. Data set curation
3. Pre-trained model construction
    1. Statistic values are calculated, inside the program.
    2. Based on a reasonable hyperparameter of the model
4. Fine-tuned model construction
    1. Save sampled molecules in each epoch

1. 2. Token generation and data set curation

ChEMBL & Structural Alerts

Compound selection. Number of heavy atoms was used in addition to three filters.
0. Heavy atom count: 95 percentile (less than.) No restriction for the smaller number.

1. GSK filter: https://pubs.acs.org/doi/10.1021/ci990423o
2. Dundee filter: https://chemistry-europe.onlinelibrary.wiley.com/doi/10.1002/cmdc.200700139
3. PAINS (pan assay interference compounds) https://www.soci.org/~/media/Files/Conference%20Downloads/2011/Designing%20Safer%20Medicines%20in%20Discvery%20Mar%202011/Francis_Atkinson_Presentation.ashx



Hit rates for sets of alerts

- Fraction of ChEMBL matched by each set of alerts
  - Only compounds with AMW < 600 were used (~540 K)

Thus, Glaxo and PAINS filters were utilized.

Loaded molecules: 2,193,110
SMILES conversion to RDKit smiles: no missing
When using 99 percentile in heavy atom counts:

```
passed mols: 2193110,
maximum heavy atom number is 681, minimum heavy atom number is 1

the nha percentile 0.99

the nha threshold 100.0

selected molecules: 2170970
```

Molecules with around 100 heavy atoms might be hard to utilize. Thus eliminated at this moment.

```
assed mols: 2193110

maximum heavy atom number is 681

minimum heavy atom number is 1

the nha percentile 0.95

the nha threshold 48.0
```

Interesting. 95 percent of ChEMBL compounds have less than 48 heavy atoms.

Token frequencies

The threshold: more than (>) 50 counts for eliminating rare tokens due to discrepancy. The number of eligible tokens are 48. The number of tokens used including padding, start and end is 51.
The number of eligible SMILES so far is 1880711.

What is the best data sets for pre-training, this is a different research topic.

3. Pretrain Model (restart 25.03.2024)

Pre-trained model will be built.
0. Dataset class: SmilesDataset is created

1. Model
2. Trainer
   will be created.

Hyper parameter search should be conducted (how).

1. Learning rate: 0.001, 0.0001
2. num layers: 2, 3
3. hidden vector size: 256, 512
4. embedding size: 64
5. using layernorm or not (after lstm layer)

In my opinion. A smaller embeding size would be okay due to 51 tokens (words) in this case.
GPU machine is quite fast.
Let's see.

Parameters and performance comaprison (geenrative models) (per 10000) cpds newly generated.

1. Learning rate (1e-3 or 5e-4)
2. Batch size (bigger is worse)
3. Embedding dimensions and layers and hidden dimensions (network architecture) are investigated under the "best" learning rate and batch size (1, 2)
4. Learning rate
   comparing lr:0.001 and 0.0005 at embed128_layer2_hdim256_batch128.
   lr0.001
   best epoch: 14 trloss: 0.593413233757019 val loss: 0.545327365398407
   lr0.0005
   best epoch 14 trloss: 0.5990570783615112 val loss: 0.5566878914833069
   almost the same, so 0.0005 was chosen for increasing network size
5. batch size
   Testing: 128 ,256, 512, 1024 at embed_128_layer2_hdim256_lr0.0005. Clearly 128 or smaller size batch is better.
   Batch size: 1024
   best epoch 19 trloss: 0.6321582794189453 val loss: 0.609783649444580
   Batch size: 512
   best epoch 26 0.6014785766601562 0.5870824456214905
   and so on.

6. Overall performance:
   Batch size: 128, Use layernorm, top 6 parameter combinations.

Ranking (Validity)	embed_dim	dropout_ratio	nlayers	hidden_dim	training_avg_loss	validation_avg_loss	validity10000	uniqueness10000	novelty10000
1	512	0.20	5	512	0.51	0.48	0.97	1.00	0.94
2	256	0.20	5	512	0.52	0.48	0.97	1.00	0.95
3	256	0.20	4	512	0.51	0.48	0.97	1.00	0.94
4	128	0.20	5	512	0.51	0.47	0.97	1.00	0.95
5	256	0.20	3	512	0.51	0.47	0.97	1.00	0.95
6	512	0.20	4	512	0.54	0.50	0.96	1.00	0.96

Ranking 3 conditions looks good based on the complication.
So, embed_dim=256, dropout_raito=0.2, nlayers=4, hidden_dim=512, layernorm=True, batch size=128, learning rate=0.0005 would be preferable.