Instructions For Training, Testing and Visualizing Results with PSALM

Requirements

1. Requirements for visualization are listed in `requirements.yml` (the complete list for training/test is listed in `full_requirements.yml`)
2. HMMER can be downloaded from hmmer.org, which also contains a link to the documentation/user manual.
3. ESM-2 can be installed from the ESM GitHub repository, which contains the documentation and installation guide.
4. UniProt release March 2021 can be downloaded from the UniProt ftp site
5. The Pfam-35.0 HMM database can be downloaded from the Pfam ftp site
6. The models (each one larger than the supplemental file limit) and rebuilt profile HMM database can be accessed at this anonymized link. These PyTorch state dictionary files should be downloaded into the `models` folder.

Dataset Creation (from scratch)

Training/Test Sequences

1. IDs of every sequence used in training are stored in `train_ids.txt` file located in the datasets folder and IDs of every sequence in each test set (max pid in 0-20, 20-40, 40-60, 60-80, and 80-100) are stored in `test_ids_<bin>.txt`

2. Use the `esl-sfetch` tool from esl mini apps in the HMMER tool kit to obtain the full length sequences for training/test sequence IDs. UniProtKB release March 2021 should be used as the `<database>` .

```
esl-sfetch -o <output_file> -f <database> <seq_ids>
```

3. Split the training/test (subset) sequences into the desired number of shards (default is 50 shards) using `split_fasta_file()` from `hmmscan_utils.py`

`split_fasta_file(<whole_fasta_file>, <out_dir>, <num_shards>)` will split `<whole_fasta_file>` into `<num_shards>` non-overlapping fasta files in `<out_dir>`

Training/Test Ground Truth Annotations

Ground truth annotations are determined via the `hmmscan` tool from HMMER. The fasta files post sharding will be named `split_{1-num_shards}_{train/test}_ids_full.fasta`, shortened as `<shard_fasta>` in the following:

```
hmmscan --acc -o <scan_data_dir>/<shard_fasta>_scan.txt <hmm_db>
<fasta_dir>/<shard_fasta>
```

Fine-tune/Evaluation Datasets

Create the sequence data required for fine-tuning/evaluation (shuffled unannotated regions) using `create_ft_sequences()` from `hmmscan_utils.py`

`create_ft_sequences(<input_fasta_file_dir>,<output_fasta_file_dir>,` will create the shuffled sequences for all fasta shards in `<input_fasta_file_dir>` in `<output_fasta_file_dir>`. `<maps.pkl>` can be found in the `info_files` folder.

Formatting/Saving Notes

1. Train sequence shards should be saved in the folder titled `train_fasta`. Test sequence shards should be saved in folders titled `test_fasta_<bin>`. Fine-tuning sequences should be saved in the folder titled `train_fasta_shuffled` and the test sequence shards suitable for evaluation should be saved in fodlers titled `test_fasta_<bin>_shuffled`.
2. Train hmmscan shards should be saved in the folder titled `train_scan`. Test hmmscan shards should be saved in folders titled `test_scan_<bin>`

Training

The training script can be run with the following command: `python psalm_train.py -o <name_of_results_directory>`

For additional options, run: `python psalm_train.py -h`

Testing

The training script can be run with the following command: `python psalm_test.py`

For additional options, run: `python psalm_test.py -h`

Visualization

Visualization of a single sequence from a custom fasta file can be run using the following command: `python psalm_viz.py -i <path_to_fasta_file>`

For additional options, run: `python psalm_viz.py -h`

Recreating Figure 4 (ROC plots)

1. All ROC data is saved in the `roc_data` folder.
2. Running the first two cells in `ROC_final.ipynb` will generate the exact figure 4 in the paper