·····

····ESM2·····

····· ESM ·· scripts/extract.py
·· ESM2 ·····
····· 1022 aa ·····

·····

····· FASTA ID····· 64 ·····
····· SHA256·····

```
from pathlib import Path
from collections import defaultdict
import hashlib, re

rows, name, parts = [], None, []
for line in Path('proteins.fa').read_text().splitlines():
    line = line.strip()
    if not line:
        continue
    if line.startswith('>'):
        if name is not None:
            rows.append((name, ''.join(parts)))
        name, parts = line[1:].split()[0], []
    else:
        if name is None:
            raise ValueError('·····FASTA··')
        parts.append(line)
if name is not None:
    rows.append((name, ''.join(parts)))
if not rows or len({n for n, _ in rows}) != len(rows):
    raise ValueError('FASTA···ID··')
allowed = set('ACDEFGHIKLMNPQRSTVWYX')
for n, seq in rows:
    if not re.fullmatch(r'[A-Za-z0-9][A-Za-z0-9_.-]*', n) or not seq or set(seq) - allowed:
        raise ValueError(f'ID·····{n}')
if any(len(seq) > 1022 for _, seq in rows):
    raise ValueError('····1022 aa·····')
groups = defaultdict(list)
for row in rows:
    groups[(len(row[1]) - 1) // 128].append(row)
out = Path('embedding-batches')
out.mkdir(exist_ok=False)
(out / 'source.sha256').write_text(hashlib.sha256(Path('proteins.fa').read_bytes()).hexdigest() + '\n')
with (out / 'plan.tsv').open('x') as plan:
    index = 0
    for _, members in sorted(groups.items()):
        members.sort(key=lambda x: (len(x[1]), x[0]))
        for start in range(0, len(members), 64):
            batch = members[start:start + 64]
            job = f'part{index:05d}'
            (out / f'{job}.fa').write_text(''.join(f'>{n}\n{s}\n' for n, s in batch))
            plan.write(f'{job}\t{len(batch)}\n')
            index += 1
```

·····

·····
···· esm2_t6_8M_UR50D ·· 6 ·· --include mean·····
····· ESM·PyTorch ·····

```
set -euo pipefail
: "${ESM_REPO:?·····/···ESM····}"
mkdir embedding-run-001
python -c 'import torch; print(torch.__version__, torch.cuda.is_available())'   > embedding-run-001/environment.txt
while IFS=$'\t' read -r job count; do
  out="embedding-run-001/$job"
  mkdir "$out"
  python "$ESM_REPO/scripts/extract.py" esm2_t6_8M_UR50D     "embedding-batches/$job.fa" "$out/pt"     --include mean --repr_layers 6 --toks_per_batch 4096     -
-truncation_seq_length 1022 > "$out/run.log" 2>&1
  printf 'expected_records=%s\n' "$count" > "$out/expected.txt"
  python verify_embeddings.py "$out/pt" "embedding-batches/$job.fa" 6
done < embedding-batches/plan.tsv
```

--toks_per_batch ··· token ·····
····· ESMFold · token ····· ESM2 · extract.py·

·····

···· ID ··· .pt ····· --truncation_seq_length ·
····· NaN/Inf··· ID·····
·····

```
from pathlib import Path
import sys, torch, hashlib, json

folder, fasta = Path(sys.argv[1]), Path(sys.argv[2])
layer = int(sys.argv[3])
ids = [line[1:].split()[0] for line in fasta.read_text().splitlines()
       if line.startswith('>')]
seqs = {}
name = None
for line in fasta.read_text().splitlines():
    if line.startswith('>'):
        name = line[1:].split()[0]
        seqs[name] = ''
    elif line.strip():
        seqs[name] += line.strip()
files = {p.stem for p in folder.glob('*.pt') if p.stat().st_size > 0}
if len(ids) != len(set(ids)) or files != set(ids):
    raise ValueError(f'····ID·····={sorted(set(ids)-files)}')
manifest = []
for name in ids:
    obj = torch.load(folder / f'{name}.pt', map_location='cpu', weights_only=True)
    if obj.get('label') != name or layer not in obj.get('mean_representations', {}):
        raise ValueError(f'{name}: ·····')
    vector = obj['mean_representations'][layer]
    if vector.ndim != 1 or not torch.isfinite(vector).all():
        raise ValueError(f'{name}: ·····')
    manifest.append({'id':name, 'sequence_sha256':hashlib.sha256(seqs[name].encode()).hexdigest(),
                     'vector_dimension':vector.numel(),
                     'pt_sha256':hashlib.sha256((folder/f'{name}.pt').read_bytes()).hexdigest()})
(folder / 'checks.json').write_text(json.dumps(manifest, indent=2))
print(f'···{len(ids)}·····{vector.numel()}')
```

···· | ···· | ·····

mean | ····· | ·····

per_tok | ····· | ·····

bos | ····· | ·····

····· ID ·····
·····
·····

·····
·····

····· ID ·····
·····ESM ····· ESM2 ·
·····

····

ESM··extract.py····· https://github.com/facebookresearch/esm/blob/main/scripts/extract.py

ESM····· https://github.com/facebookresearch/esm

ESM·····FASTA···· https://github.com/facebookresearch/esm/blob/main/esm/data.py