Protein Embedding ANN Search

Αναζήτηση Πρωτεϊνών με Embeddings & ANN (Python):

Περιγραφή: Η εργασία υλοποιεί αναζήτηση ομοιότητας πρωτεϊνών στον χώρο των embeddings (ESM2) με προσεγγιστικές μεθόδους ANN (Euclidean LSH, Hypercube, IVF-Flat, IVF-PQ, Neural LSH). Για ένα σύνολο queries (targets.fasta) και βάση (swissprot.fasta / embeddings) δημιουργούνται embeddings, εκτελείται αναζήτηση γειτόνων στο embedding space (L2) και παράλληλα BLAST (local alignment) στην ίδια βάση, ώστε να υπολογιστούν Recall@N έναντι BLAST Top-N και QPS (Queries per Second). Επιπλέον γίνεται βιολογική αξιολόγηση πιθανών remote homologs (περιπτώσεις όπου τα embeddings βρίσκουν κοντινά hits που δεν εμφανίζονται ψηλά στο BLAST).

Κατάλογος αρχείων & ρόλοι (source tree)

Παρακάτω περιγράφονται τα βασικά Python αρχεία της 3ης εργασίας:

Αρχείο	Περιγραφή
protein_embed.py	Δημιουργία embeddings (ESM2) από FASTA. Γράφει το matrix vectors (np.save) και συνοδευτικά αρχεία <output>.ids.txt (ID ανά γραμμή) και <output>.meta.tsv (id, length, description).
esm_embedder.py	Wrapper για το pretrained ESM2 (fair-esm). Υποστηρίζει batching με περιορισμό tokens και εξαγωγή αναπαραστάσεων από transformer layer (mean pooling).
protein_fasta.py	Ελαφρύς FASTA parser (iter_fasta) που παράγει (id, sequence, description). Χρησιμοποιείται για embeddings & BLAST queries.
protein_search.py	Κύριο πρόγραμμα αναζήτησης και αξιολόγησης. Φορτώνει βάση embeddings και queries, χτίζει / φορτώνει indices, εκτελεί ANN αναζητήσεις και γράφει report με Recall@N vs BLAST Top-N και QPS. Προαιρετικά τρέχει BLAST (makeblastdb / blastp).
blast_utils.py	Parsing / κανονικοποίηση BLAST outfmt 6. Παρέχει top-N hits ανά query και %identity lookup (με default φίλτρο E-value <= 1e-3).
utils.py	Βοηθητικές συναρτήσεις (π.χ. απλές ρουτίνες για path / formatting).
ann/_init_.py	Ορισμοί / exports για τις μεθόδους ANN και mapping ονομάτων (lsh, hypercube, ivfflat, ivfpq, neural).
ann/lsh.py, hypercube.py	Υλοποιήσεις Euclidean LSH και Hypercube

	projection για L2 αναζήτηση.
ann/ivf_flat.py, ivf_pq.py, kmeans.py	Υλοποιήσεις IVFFlat και IVFPQ με k-means clustering (training) και αναζήτηση με nprobe. Το IVFPQ υποστηρίζει optional refine reranking.
ann/neural_lsh.py	Υλοποίηση Neural LSH: εκπαίδευση μοντέλου διαμέρισης (m bins) στο train subset και multi-probe αναζήτηση με Top-T bins.

Προαπαιτούμενα

Λειτουργικό: Linux (όπως ζητείται στην εκφώνηση).

Python: 3.10+.

BLAST+: NCBI BLAST+ εγκατεστημένο (εντολές makeblastdb, blastp στο PATH).

Βιβλιοθήκες Python:

- numpy
- torch
- tqdm
- fair-esm (esm)
- scikit-learn (προαιρετικό, αν χρησιμοποιηθεί σε πειράματα / plots)

Εγκατάσταση βιβλιοθηκών;

Δημιουργία virtual environment (προαιρετικό αλλά προτεινόμενο)

```
python3 -m venv venv
source venv/bin/activate
```

Εγκατάσταση Python βιβλιοθηκών:

```
pip install --upgrade pip
pip install -r requirements.txt
```

Εγκατάσταση BLAST+ (ενδεικτικά σε Ubuntu / Debian):

```
sudo apt-get update && sudo apt-get install -y ncbi-blast+
```

Μορφή datasets

Η εργασία χρησιμοποιεί τα παρακάτω αρχεία:

• swissprot.fasta: FASTA βάση πρωτεϊνών (database).

• targets.fasta: FASTA με queries (στόχους) για αξιολόγηση.

• protein_vectors.dat: NumPy matrix embeddings [N, d] αποθηκευμένο με np.save (η κατάληξη μπορεί να είναι .dat).

Συνοδευτικά αρχεία (ίδιο basename):

- protein_vectors.dat.ids.txt: ένα protein ID ανά γραμμή (row-aligned με το matrix).

- protein_vectors.dat.meta.tsv (προαιρετικό): TSV (id, length, description).

• blast_results.tsv: BLAST outfmt 6 tabular output (qseqid sseqid pident ...).

Οδηγίες χρήσης (CLI)

Η υλοποίηση περιλαμβάνει δύο βασικά σενάρια:

- Παραγωγή embeddings από FASTA (protein_embed.py)
- Αναζήτηση & αξιολόγηση ενός run (protein_search.py)

Παραγωγή embeddings - protein_embed.py:

Βασικό usage:

```
python protein_embed.py \
  -i <INPUT_FASTA> \
  -o <OUTPUT_VECTORS> \
  [--model esm2_t6_8M_UR50D] [--device {auto|cpu|cuda}] [--max_aa 1022]
```

Αναζήτηση & αξιολόγηση - protein_search.py:

Βασικό usage:

```
python protein_search.py \
  -d <VECTORS_FILE> \
  -q <QUERIES_FASTA> \
  -o <REPORT_TXT> \
  --method {lsh|hypercube|ivfflat|ivfpq|neural|all} \
  [--blast_tsv <BLAST_TSV>] [--run_blast --db_fasta <DB_FASTA>] \
  [--N_eval 50] [--N_print 10] [--cache_dir <DIR>] [--seed 1] \
  [method-specific params ...]
```

Σημείωση: Αν δοθεί --run_blast, το πρόγραμμα δημιουργεί BLAST DB από το db_fasta και τρέχει blastp για τα queries.

Παραδείγματα εκτέλεσης

1) Δημιουργία embeddings για τη βάση (swissprot.fasta):

```
python protein_embed.py \
  -i data/swissprot.fasta \
  -o data/protein_vectors.dat \
  --model esm2_t6_8M_UR50D --device auto
```

2) Σύγκριση όλων των μεθόδων σε ένα run:

```
python protein_search.py \
  -d data/protein_vectors.dat \
  -q data/targets.fasta \
  -o out/report_all.txt \
  --method all \
  --run_blast --db_fasta data/swissprot.fasta --blast_tsv
data/blast_results.tsv \
  --N_eval 50 --N_print 10 --seed 1
```

Μορφή εξόδου & μετρικές

Το report που παράγει το protein_search.py (-o) περιλαμβάνει ανά query:

- Summary πίνακα ανά μέθοδο με Time / query, QPS και Recall@N έναντι BLAST Top-N.
- Αναλυτικό πίνακα Top neighbors ανά μέθοδο (Rank, Neighbor ID, L2 distance, BLAST %identity, αν είναι στο BLAST Top-N, bio comment).
- Συνολικά (στο τέλος του αρχείου) Average QPS και Average Recall@N ανά μέθοδο, averaged σε όλα τα queries.

Ενδεικτική μορφή (απόσπασμα):

```
================================================================================
Query Protein: A0A009HL96
N = 50 (Top-N used for Recall@N evaluation)
[1] Summary comparison
--------------------------------------------------------------------------
Method        | Time/query (s) | QPS     | Recall@N vs BLAST Top-N
--------------------------------------------------------------------------
IVF-Flat      |         0.0004 | 2323.95 | 0.70
BLAST (Ref)   |         0.0523 |   19.12 | 1.00
--------------------------------------------------------------------------
```

```
[2] Top-N neighbors per method (printed N = 10)

Method: IVF-Flat

Rank | Neighbor ID | L2 Dist | BLAST Identity | In BLAST Top-N? | Bio
comment

-----------------------------------------------------------------------

   1 | Q5HI09      |  0.6392 | 26.13%         | Yes             | In
twilight zone - possible remote homolog

   2 | P33112      |  0.6421 | 30.84%         | Yes             | --

   3 | Q99VW2      |  0.6598 | 26.13%         | Yes             | In
twilight zone - possible remote homolog

   4 | Q8CQ37      |  0.6748 | 27.93%         | Yes             | In
twilight zone - possible remote homolog

   5 | Q99RR6      |  0.7503 | 31.28%         | Yes             | --

   6 | Q9AE24      |  0.7792 | 27.63%         | Yes             | In
twilight zone - possible remote homolog

   7 | P28257      |  0.7835 | 32.75%         | Yes             | --

   8 | P0CAW8      |  0.7894 | 31.44%         | Yes             | --

   9 | A6WZ81      |  0.8532 | 33.19%         | Yes             | --

  10 | P45189      |  0.8847 | 34.19%         | Yes             | --

=======================================================================
```

Recall@N: κλάσμα των BLAST Top-N hits που ανακτώνται μέσα στα Top-N της ANN μεθόδου (ίδιο N).

QPS: queries per second, υπολογισμένο από τον χρόνο της προσεγγιστικής αναζήτησης.

Περιγραφή υλοποίησης (σύνοψη)

- Embeddings: Από τα πρωτεϊνικά sequences (FASTA) παράγονται διανυσματικές αναπαραστάσεις με ESM2 και mean pooling ανά θέση.
- ANN indices: Για τη βάση embeddings χτίζονται indices για κάθε μέθοδο (LSH / Hypercube / IVFFlat / IVFPQ / Neural). Για τα ακριβότερα indices υπάρχει cache ώστε να αποφεύγεται rebuild σε επαναλήψεις.
- Evaluation vs BLAST: Για κάθε query, το BLAST (outfmt 6) παρέχει τα Top-N hits (με φίλτρο E-value <= 1e-3 από τον parser). Η Recall@N μετρά πόσα από αυτά ανακτώνται από την ANN στα Top-N.
- Remote homologs: Αναζητούνται περιπτώσεις με χαμηλή ομοιότητα ακολουθίας (π.χ. %identity < 30%) αλλά υψηλή εγγύτητα στο embedding space και τεκμηριώνονται με annotations (UniProt / Pfam / InterPro / GO / EC) στο report.