**Reference Recommendation for LLM-Generated Text Using Deep Textual Representations**

This repository implements the reference recommendation system described in the paper "Reference Recommendation to Verify Information Produced by Large Language Models". The system helps users verify information from LLM chatbots by recommending relevant references using Siamese and Triplet networks with Sentence-BERT embeddings and submodular optimization.

**1. Requirements**

Install Python 3.8+ and the following packages:

bash

CopyEdit

pip install torch sentence-transformers scikit-learn networkx faiss-cpu tqdm numpy

If faiss-cpu is not available for your platform, the code will automatically fall back to sklearn nearest neighbors.

**2. Configuration**

All important settings are in the CONFIG dictionary at the top of reference\_recommendation.py.

| **Parameter** | **Default** | **Description** |
| --- | --- | --- |
| sentbert\_model | "all-MiniLM-L6-v2" | Base Sentence-BERT model |
| learning\_rate | 2e-5 | Fine-tuning learning rate |
| batch\_size | 16 | Training batch size |
| siamese\_alpha | 0.4 | Scaling factor for Siamese similarity targets |
| triplet\_margin | 1.0 | Margin for Triplet loss |
| siamese\_epochs\_per\_distance | {1:30, 2:20, 3:10, 4:5} | Epochs per citation distance (Siamese) |
| triplet\_epochs\_per\_distance | {1:25, 2:15, 3:7, 4:4} | Epochs per citation distance (Triplet) |
| max\_distance\_positive | 4 | Max citation path length for positives |
| subref\_clusters | 20 | Number of clusters if authors are not used |
| use\_faiss | Auto | Use Faiss if available |

You can edit CONFIG to suit your dataset and hardware.

**3. Running the Pipeline**

Example (with training disabled for quick testing):

bash

CopyEdit

python reference\_recommendation.py

Example (training enabled, using your own files):

python

CopyEdit

from reference\_recommendation import example\_pipeline

import json

# Load data

with open("corpus.json") as f:

corpus = json.load(f)

with open("queries.json") as f:

queries = json.load(f)

# Run pipeline

example\_pipeline(corpus, queries, train=True)

**4. Expected Output**

During execution, you will see:

1. **Citation graph stats**

cpp

CopyEdit

Citation graph nodes: 100 edges: 350

Computed pair distances up to 4

1. **Embedding progress**

shell

CopyEdit

100%|██████████| 100/100 [00:05<00:00, 18.50it/s]

1. **Training logs** (if train=True)

python-repl

CopyEdit

Siamese epoch 1/3: loss=0.042

...

Triplet epoch 1/3: loss=0.317

1. **Final evaluation metrics**

graphql

CopyEdit

SentBERT results (mean): F1@1: 0.452 F1@3: 0.532 F1@5: 0.601 MRR: 0.665