Fairness, Privacy and Ethics in AI   Date: 26/08/2025

Instructor: *Dr. Sujit Gujar*   Scribe: Sravan Kotta, Ravi Teja Dendukuri

Project Report

1 CODE FLOW OVERVIEW

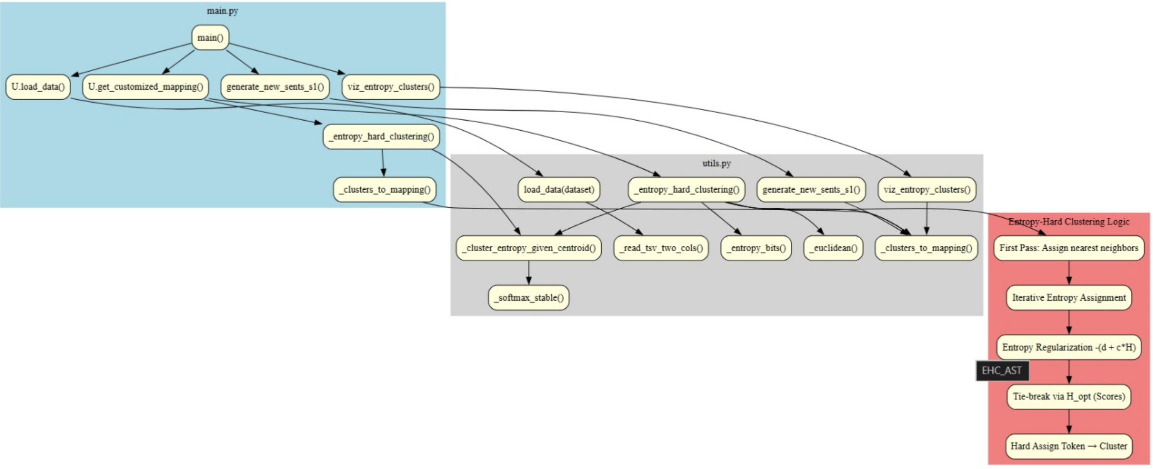
main.py
main()
U.load_data()
U.get_customized_mapping()
generate_new_sents_s1()
viz_entropy_clusters()
_entropy_hard_clustering()
_clusters_to_mapping()
utils.py
load_data(dataset)
_entropy_hard_clustering()
generate_new_sents_s1()
viz_entropy_clusters()
_cluster_entropy_given_centroid()
_read_tsv_two_cols()
_entropy_bits()
_euclidean()
_clusters_to_mapping()
_softmax_stable()
Entropy-Hard Clustering Logic
First Pass: Assign nearest neighbors
Iterative Entropy Assignment
Entropy Regularization -(d + c*H)
EHC_AST
Tie-break via H_opt (Scores)
Hard Assign Token → Cluster

Figure 1: AST Function flow

Run instruction:
1] Original.py for baseline on a sentiment analysis task using SVM (text without change).
2] run main.py to obtain replaced text.
3] Run private.ipynb to check results with new text.

2 Algorithm Overview

2.1 Data Loading

The train and dev splits must contain `sentence` and `label`, whereas the test split may omit labels. A resilient TSV loader:

- tries multiple encodings,
- accepts both headered and headerless formats,
- coerces labels to integers,
- strips and sanitizes text fields.

2.2 Embedding Subset Extraction

Only tokens appearing in the dataset vocabulary are streamed from `embeddings/<embedding_type>.txt`. All vectors are L2-normalized to stabilize Euclidean distances for clustering.

2.3 Entropy-Guided Hard Clustering

Seeding

Select K dataset tokens as fixed centroids (`--num_centroids` determines K).

Assignment Rule

For every unassigned token t, compute the fixed-centroid entropy:

$$H_{\text{fixed}} = H(\text{cluster} \cup \{t\} \mid \text{centroid fixed})$$

Assign t to the cluster that maximizes H_{fixed}.

Tie-breaking

If multiple clusters are within an ε margin of the best entropy, compute the virtual re-centering entropy:

$$H_{\text{opt}} = \max_{\text{member } c} H(\text{cluster} \cup \{t\} \mid c \text{ as centroid})$$

Blend the two using:

$$\text{score} = \alpha \cdot H_{\text{fixed}} + (1 - \alpha) \cdot (\text{reg\_hopt} \cdot H_{\text{opt}})$$

and choose the highest-scoring cluster. This prevents unstable or degenerate assignments.

Penalizing High Entropy

To ensure entropy does not take priority over distance (utility) we introduce c:

$$\text{base\_score} = -\left(d + c \cdot H_{\text{fixed}}\right)$$

where d is the centroid distance and c is a hyperparameter. This promotes tighter and more coherent clusters.

Outputs

Two mapping artifacts are written to disk:

- `sim_word_dict/.../<tag>.txt`: cluster membership and per-token candidate lists.
- `p_dict/.../<tag>.txt`: per-token replacement probabilities.

2.4 Token-to-Candidate Mapping

For token t in cluster C, the candidate set is $C \setminus \{t\}$. Probabilities are computed via a temperature-controlled softmax over negative distances to the centroid:

$$p_i = \text{softmax}\left(-\frac{d_i}{T}\right)$$

where T controls distribution sharpness. Degenerate distributions are renormalized defensively.

2.5 Text Privatization Strategy

For each sentence:

- preserve stopwords if `--save_stop_words` is enabled,
- jitter numeric tokens to reduce linkability,

- replace other tokens with sampled cluster candidates.

An optional enforcement step guarantees at least one semantic-preserving replacement per sentence.

Outputs are saved under:

```
privatized_dataset/cf-vectors/conservative/eps_<eps>_<strategy>_save_stop_words_<flag>/
```

3 Results

The hyperparameter refers to the term the entropy is multiplied with when calculating the cost of adding a point to the cluster.

Clusters K	**Hyperparameter** c	**Accuracy**
30	0.5	50.4%
60	0.5	58.1%
60	0.2	55.0%

Table 1: Accuracy Results for 1005 tokens