**Code Execution and Results :**

1. Create Mapping between labels and their ids present in sparse matrices

1. Move to data/EUR-Lex/label\_num\_mapping

Folder

2. Move to **train** folder (ensuring folder contain “train\_labels.npy” file) and execute readTrainLabelsnpy.py it will create onlyTrainLabelsnpy.txt

3. Execute readTraintxt.py (ensuring folder contain “train.txt” file) it will create “onlyTrainLabels.txt”

4. Execute “readBothFiles.py” , it will take above files as input and create “list\_mappings.txt”. It will give indexes as number to the labels.

5. Execute “create\_mapping.py” it will take “list\_mappings.txt” as input and it will produce “mappings.json” file.

6. Execute “create\_mapping\_rev.py” to get reverse mapping of file as an output. It will Create “mapping\_rev.json” file.

7. “Mappings.json” and “mappings\_rev.json” are created which will be used for processing for train dataset.

8. Move to **test** folder (ensuring folder contain “test\_labels.npy” file) and execute readTestLabelsnpy.py it will create onlyTestLabelsnpy.txt

9. Execute readTesttxt.py(ensuring folder contain “test.txt” file) it will create “onlyTestLabels.txt”

10. Execute “readBothFiles.py” , it will take above files as input and create “list\_mappings.txt”. It will give indexes as number to the labels.

11. Execute “create\_mapping.py” it will take “list\_mappings.txt” as input and it will produce “mappings\_test.json” file.

12. Execute “create\_mapping\_rev.py” to get reverse mapping of file as an output. It will Create “mappings\_rev\_test.json” file.

13. “Mappings\_test.json” and “mappings\_rev\_test.json” are created which will be used for processing for train dataset.

14. Go to parent folder (./data/EUR-LEX) and execute merge.py with instructions provided below:

* Ensure point 7 outputs are copied to parent folder (i.e. ./data/EUR-LEX).
* Ensure point 13 outputs are copied to parent folder (i.e. ./data/EUR-LEX).

With corresponding inputs (mappings.json and mappings\_test.json), program will execute and save output in file named “merged\_mappings.json”

2. Run PLT from AttentionXML for the given dataset and generate the clusters

2.1 Command: bash scripts/run\_eurlex\_fast.sh

2.2 Clusters will be stored in models folder :

FastAttentionXML-EUR-Lex-Tree-0-cluster-Level-0.npy

FastAttentionXML-EUR-Lex-Tree-0-cluster-Level-1.npy

FastAttentionXML-EUR-Lex-Tree-0-cluster-Level-2.npy

3. Preparing data for HAXML Net G :

3.1 Copy original test\_labels.txt and train\_labels.txt in data/EUR-Lex/original directory.

3.2 Create cluster and label mapping using PLT and mapping created in step 1

3.2.1 Command: python3 create\_cluster\_label.py

3.2.2 Two json file are created in data/EUR-Lex Folder :

**cluster\_label.json**

For each cluster and its respective label there is a corresponding mapping.

**label\_cluster.json**

For each label and its respective cluster there is a corresponding mapping.

3.3 Modify test\_labels.txt and train\_labels.txt replacing labels with their corresponding cluster id

3.3.1 Command: python3 modify\_labels\_G.py

3.3.2 The respective text file is modified with its corresponding clusters for each datapoint.

4. Run AttentionXML for the modified dataset

4.0 Prerequisite : Input for AttentionXML is categorised datapoint labelled by clusters ( as shared in point 3.3 )

4.1 Command: bash scripts/run\_eurlex.sh

4.2 Post executions following folders are updates models/**HAXML\_NET\_G\_EUR-LEX** :

Model : corresponding model is saved as :

AttentionXML-EUR-Lex-Tree-0

AttentionXML-EUR-Lex-Tree-1

AttentionXML-EUR-Lex-Tree-2

Results : Ensemble scores will be saved here for further usage.

AttentionXML-EUR-Lex-Ensemble-scores.npy

5. Preparing data for HAXMLNet L:

5.1 Copy current test\_labels.txt and train\_labels.txt files to **data/EUR-Lex/HAXML\_NET\_G** directory

5.2 Generate candidate labels for each data points in test and train dataset

5.3 Below Command will modify test\_labels.txt and train\_labels.txt replacing current labels with candidate labels

5.3.1 Command: python3 modify\_labels\_L.py

5.3.2 We will obtain candidate labels for each datapoint.

6. Run AttentionXML for the modified dataset

6.1 Command: bash scripts/run\_eurlex.sh

6.2 Post executions following folders are updates models/**HAXML\_NET\_L\_EUR-LEX** :

Model : corresponding model is saved as :

AttentionXML-EUR-Lex-Tree-0

AttentionXML-EUR-Lex-Tree-1

AttentionXML-EUR-Lex-Tree-2

Results : Ensemble scores will be saved here for further usage.

AttentionXML-EUR-Lex-Ensemble-scores.npy

7. Combine scores from HAXML Net G and HAXML Net L models to get final predictions on test dataset

7.0 Prerequisite :

In folders models/HAXML\_NET\_L\_EUR-LEX and models/HAXML\_NET\_G\_EUR-LEX corresponding models and scores should be present.

Cluster labels should be present for the dataset. As here Eur-lex is in processing , hence folder “data/EUR-LEX” should have cluster\_label.json.

7.1 Command: python3 compute\_scores\_HAXML.py

Fetch the details from Ensemble score of HAXMLNet G and L then top K labels are considered for each datapoint. ( shown as point 4 in HAXMLNet paper ).

7.2 It will make an output file which is predicted by the model for each datapoint. Locations of file will be : “results/EUR-Lex\_labels\_1024”

8. Get P@k and DCG@k scores for the prediction

8.0 Prerequisite :

Model should produce the corresponding results in the “result” folder.

Actual labels should also be present in the corresponding dataset folder. For Eur-LEX it is “dataset/test\_labels.txt”.

8.1 Command: python3 evaluation\_metrics\_HAXMLNet.py

8.2 For varying K it will produce the respective score for following evaluation metric:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DataSet | Prec@1 | Prec@3 | Prec@5 | nDCG@1 | nDCG@3 | nDCG@5 |
| Eur-LEX | 87.03 | 74.38 | 61.9 | 87.03 | 77.37 | 67.98 |
| Wiki 10-30K | 84.7 | 70.5 | 60.2 | 84.7 | 73.8 | 65.02 |

Control Flow :

1. Cluster will be saved in “results/plt\_model” folder , we are reading “FastAttentionXML-Amazon-670K-Tree-0-cluster-Level-1.npy” for creating the clusters.
2. Execute the python file “modify\_labels\_G.py”