BRAUDE
College of Engineering, Karmiel

Software Engineering Department

User Guide

Analysis of Medieval Arabic Creations

25-1-R-19

https://github.com/YosraDaso/CapstoneProject.git

- **Purpose**

This user guide provides instructions for running the authorship attribution analysis code developed in this project. It outlines the necessary files, environment, and steps required to execute the research workflow successfully.

- **System Overview**

This system performs authorship verification of medieval Arabic texts using a signal-based deep learning framework. The notebook walks through all stages of the pipeline: Arabic text preprocessing, embedding generation with AraBERT, similarity computation via a Siamese network, signal processing, and clustering analysis.

- **Requirements**
  - Google Colab (recommended: Pro+ with an A100 GPU and High RAM).
  - Python (via Google Colab).
  - Internet access and an active Google Drive account.
  - GitHub access to the repository: https://github.com/YosraDaso/CapstoneProject.git

- **Repository Location**

/Phase B/Run Environment/

- **Files Provided**
  - Analysis of Medieval Arabic Creations.ipynb: The main notebook containing the full experimental pipeline.
  - impostors_group.zip: Compressed folder containing impostor texts.
  - test_group.zip: Compressed folder containing test texts.
  - test_group_index.csv: Index mapping for processed test files.
  - test_group_index_original.csv: Original index of full texts.

- **Setup Instructions**

1. Open Google Colab (preferably Pro+ with an A100 GPU and High RAM).
2. Unzip impostors_group.zip and test_group.zip
3. Upload the required files to your Google Drive
4. Open Analysis of Medieval Arabic Creations.ipynb in Google Colab.
5. Mount Google Drive when prompted by the notebook.

6. Update the file paths in the notebook if needed (according to your folder structure).
7. Run all cells in order to execute the full pipeline.

Note: Training the Siamese network for one impostor pair takes approximately 30 minutes on Colab Pro+ (A100 GPU). Since the experiment includes 300 pairs, we recommend running this phase on a more powerful setup if available. We used Lambda Cloud with an NVIDIA GH200 GPU, where training time per pair was reduced to around 10 minutes. The resulting signals are saved in the /impostor_results/signals directory.

- **Google Drive Folder Structure**

We begin the process with the essential files listed in the Files Provided section, which include the main notebook, text datasets, and index files. Once the entire pipeline is executed successfully, a structured set of output directories and files is generated automatically in the user's Google Drive. These outputs, illustrated in Figure 1, reflect the different stages of the pipeline and support further analysis.

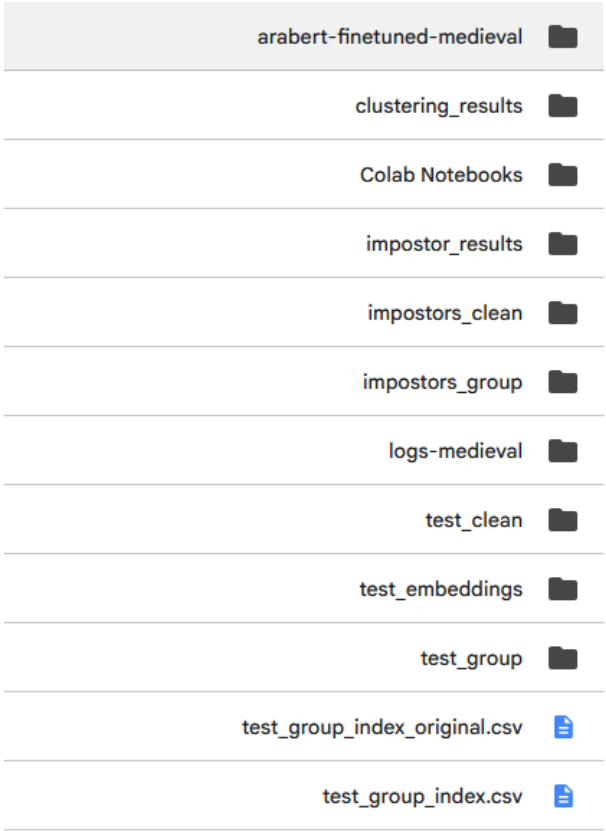

Figure 1. Google Drive folder structure after full pipeline execution

After running the full pipeline, the following directories are created automatically:

- test_clean and impostors_clean: generated after the preprocessing stage and contain the cleaned text files.
- arabert-finetuned-medieval: stores the fine-tuned AraBERT model checkpoint.
- test_embeddings: includes the output embeddings for all segments in the test group.
- impostor_results: created to store results from the training and classification stages. It contains four subdirectories:
    - signals: includes the signal vectors produced by the Siamese network for each book.
    - dtw_matrices: contains the distance matrices generated by Dynamic Time Warping for every run.
    - isolation_forest_output: includes the raw anomaly scores for each book in every run.
    - isolation_forest_output_scores: includes the normalized anomaly score matrix used for clustering.
- clustering_results: stores the outputs of the clustering stage, including cluster labels, silhouette score evaluation, and final visualizations.

Most of the output files are saved in .npy or .png formats. .csv is used for index files, anomaly score matrices, and clustering outputs when tabular structure is needed.