PROGRAMMER GUIDE

ABAI : Authored By AI

Version 0.0.1 (May 2024)

Contents

1 Introduction

1.1 Purpose of ABAI

In a world where generative AI has been made available to the public (in the shape of various tools), demand for a tool capable of authorship attribution has emerged. Authorship attribution is the task of identifying the author of a given text. ABAI was created specifically for this purpose. It will assist you in identifying the source of a certain piece of text.

1.2 License

The ABAI project is licensed under the MIT License and it's local GUI (based on the PyQt5 python module) is licensed under the GPLv3 License. For more information, please read the license segment in the about section.

2 First steps

2.1 Contribution to ABAI

If you want to contribute to `Authored by AI`, this is the place to start.

2.1.1 Required skills

Authored by AI is majorily written in `Python`. Python knowledge would therefore be a precious asset to you and will make it easier for you to contribute to the source code. Furthermore, `Django` is used as a framework for the front-end development of the web interface.

In addition, it could be beneficial to have affinities with the following packages used throughout the project :

- `Scikit-learn`
- `NLTK`
- Numpy
- PyQt

The `User guide` and `Programmer guide` are written in `LaTeX`, therefore, if you want to contribute to the documentation, it would be good to have some familiarity with it.

2.1.2 Where to start?

If you are still interested in contributing to Authored by AI, you will find guidance on how to do so by reading the following sections of the guide:

- Contributor's reference: Contains useful references for any contributor of ABAI.
- Setup guide: Explains how to set up your development environment.
- Source code documentation: Understanding the module distribution.
- Project structure: Explains how the project is structured.

3 Contributor's reference

3.1 Writing code

This section contains useful information for contributors interested in modifying the code base.

3.1.1 Setting up the application

To contribute to `Authored by AI`, you have to setup a working development environment. The recommended method to check if your installation of the project works as expected is to run it and use features looking for eventual failures related to your setup. To quickly set up `ABAI` check out the setup guide.

3.1.2 Writing a patch

To write a `patch`, make the change that you want in the code base. Please be careful to respect the coding style. There is a topic guide that explains what conventions are in use.

Note

If you have troubles finding in which file to make changes, try reading the topic guide about the project structure

3.1.3 Test the patch

For a modification to be considered and added to future releases, it has to be tested and approved with acceptable stability. It will then be added to the next release. To run the unit tests of the application, please refer to the `testing` section.

4 Setup guide

4.1 Setting up the development environment

In this section you will find a step by step walk-through on how to setup your development environment.

4.2 System requirements

ABAI has the following system requirements :

Operating System

- `Windows / MacOS / Linux`

Disk Space

- At least `100 MB` of free disk space.

4.3 Software requirements

ABAI makes use of the following main software :

Library and Environment

- `Python 3.12`
- `Pip v24.0` (installed with Python)
- `Git`

During installation of the previous software, please pay attention that they correctly define their respective variables in **PATH**.

4.4 Download

The project's source code is available on GitHub.

The recommended way to download the source code is to use the following git command in the folder you want:

```
$ git clone https://github.com/UNamurCSFaculty/2324_INFOB318_ABAI2
```

4.5 Setup

First, open a terminal and navigate to the project folder. The name of the folder should be **2324_INFOB318_ABAI2**

Note

It may be useful to create a virtual environment before installing the application. This is to make sure that Python will not have any dependency conflicts.

Now, run the following command to setup the development environment:

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

5 Source code documentation

This sections contains the documentation information about the source code. The source code is divided in two parts:

- The **local** backend application contains the local python PyQt GUI and the logic of the application (preprocessing, feature extraction, etc.).
- The django **web** application contains the code for the web interface.

5.1 The local backend application

The local backend application is further divided into sub-modules.

5.1.1 The main module

The main module resides at the following location :

```
2324_INFOB318_ABAI2/code/backend/main.py
```

It contains the code related to the main function of the local application, the `PyQt GUI`. It specifically sets up the **main window** with a main menu and **child windows** for every one of it's **options**.

5.1.2 The pipelines module

The pipelines module resides at the following location :

```
2324_INFOB318_ABAI2/code/backend/scripts/pipelines.py
```

It contains the code necessary to **generate** a custom **scikit-learn pipeline** that contains the steps specified by the **user configuration**. This pipeline can then be **saved** and **loaded** at any time.

A pipeline is essentially a list of steps constituting what we will call a model. These steps (also called transformers) can then be sequentially applied to data and end with a final predictor that will be used for predictive modelling.

5.1.3 The preprocessing module

The preprocessing module resides at the following location :

```
2324_INFOB318_ABAI2/code/backend/scripts/preprocessing.py
```

It contains the custom transformers and feature unions that will be integrated to the scikit-learn pipelines. This includes a transformer for pre-processing the data and every transformer that will be applied sequentially before and after pre-processing is applied.

The decision of applying a transformer before or after pre-processing should be taken according to the requirements and compatibility of each method with the pre-processing steps.

5.1.4 The database functions module

The database functions module resides at the following location :

```
2324_INFOB318_ABAI2/code/backend/scripts/database_functions.py
```

It contains all the functions that will be used to manage the project's database. The PyQt window for the database actions are also contained in this module.

5.1.5 The detect origin (local) module

The detect origin (local) module resides at the following location:

```
2324_INFOB318_ABAI2/code/backend/scripts/detect_origin_local.py
```

It contains a function to predict labels for unknown texts. Additionally, it contains the PyQt windows for the authorship attribution task and it's result.

5.1.6 The train, validate and test module

The train, validate and test module resides at the following location:

```
2324_INFOB318_ABAI2/code/backend/scripts/train_validate_test.py
```

It contains multiple functions to call training, grid search, learning curve, validation and test methods from custom scikit-learn pipeline.

Additionally, it contains the PyQt windows responsible for performing actions on models. This includes previously cited methods like training, validation and testing.

5.1.7 The configuration module

The configuration module resides at the following location:

```
2324_INFOB318_ABAI2/code/backend/utils/configurating.py
```

It contains the config parser methods that extract the user's configuration by reading the config.ini file located at:

```
2324_INFOB318_ABAI2/code/backend/config/config.ini
```

Additionally, it contains the PyQt windows that let the user edit and save options to the config.ini file.

5.1.8 The logging module

The logging module resides at the following location:

```
2324_INFOB318_ABAI2/code/backend/utils/log.py
```

It contains a singleton method to initiate a logger object that will be used throughout every backend module.

5.1.9 The slicing module

The slicing module resides at the following location:

```
2324_INFOB318_ABAI2/code/backend/utils/slice_up_essays.py
```

It contains a methods to slice up local `.TXT` files that contain long and continuous essay texts into paragrahps and stores them in labeled `.TXT` files inside the raw data folder located at :

```
2324_INFOB318_ABAI2/code/backend/data/raw
```

Furthermore, a second method slices up texts with their labels to prepare them as sliced lists to later be added into the database.

6 Testing

6.1 How to run unit tests?

Tests are performed using the unittest module of the standard `Python` library.

6.2 Requirements

To run the unit tests, you need to have a working development environment. Please refer to the setup guide if this is not yet the case.

6.3 How to write a unit test?

If you want to write a unit test to test out a feature, please proceed like this :

1. If a test module doesn't yet exist for the feature you need to test, create a new test module named **`test_[module name]`** with the module name being the name of the module that contains the feature to test.

2. Next, write your test method named like this : **`test_[feature name]`**. The feature name is the name of the method you need to test.

6.4 Unit testing

To run the unit tests, run the following command at the root of the project:

```
$ python -m unittest discover -s code/backend/tests -p "test_*.py" -v
```

6.5 Test coverage

Tests coverage analysis is performed using the coverage module. To perform the coverage analysis use the following commands sequentially at the root of the project:

```
$ coverage run -m unittest discover -s code/backend/tests -p "test_*.py"
$ coverage html
```

After this, you should be able to visualize the coverage analysis result by opening the generated **index.html** file from the **htmlcov** folder.

7 Topics

7.1 Coding style

The source code follows the PEP 8 and a variant of the google style docstrings. In addition, the entire code is also formatted with Black to remain consistent. Type hinting can be added and is considered as a most welcome bonus. Especially where it seems the most useful to resolve confusion.

7.2 Project structure

The ABAI project follows the recommended Python project structure. The main project structure for ABAI is described as follows:

```
2324_INFOB318_ABAI2 (root with license and readme files)
├── code (1)
│   ├── backend (2)
│   │   ├── config (3)
│   │   ├── data (4)
│   │   ├── logs (5)
│   │   ├── models (6)
│   │   ├── scripts (7)
│   │   ├── static (8)
│   │   ├── tests (9)
│   │   └── utils (10)
│   ├── django_abai (11)
│   │   ├── abai_website (12)
│   │   │   ├── static (13)
│   │   │   │   ├── css (14)
│   │   │   │   └── images (15)
│   │   │   └── templates (16)
│   │   └── django_abai (17)
│   ├── docker (18)
│   ├── method (19)
│   │   ├── analysis
│   │   ├── design
│   │   └── implementation
│   └── planning (20)
└── doc (21)
    ├── PDFs (22)
    └── sources (23)
```

(1). Contains the **source code** of Authored by AI.

(2). Contains code for the **local application**.

(3). Contains code for **user configuration** (GUI included).

(4). Contains code for `.TXT` **data storage** and **database** (GUI included).

(5). Stores **logs** for the project.

(6). Stores **saved trained models**.

(7). Contains **code scripts** for the local application (GUI included).

(8). Contains **images** for the local application's GUI.

(9). Contains **unittests**.

(10). Contains **utilitary modules** (logging, slicing, configuration (GUI included)).

(11). Contains code for the **web application**.

(12). Django application for the website application.

(13). Contains **CSS styling** and **images** for the website.

(14). Contains **CSS styling** for the web interface.

(15). Contains **images** for the web interface.

(16). Contains **html templates** for the web interface.

(17). **Django** project folder.

(18). Contains **dockerfiles** and **yaml** file.

(19). Contains gathered **artefacts** (interviews, research and preparation) of the project.

(20). Contains the planning of the project (`Gantt` chart).

(21). Contains documents for **documentation**.

(22). Contains `PDFs` for the `user guide` and the `programmer's guide`.

(23). Contains the `LaTeX` source files for the `user guide` and the `programmer's guide`.

7.3 Type checking

If type hinting is used correctly, it can be useful to detect type related errors before execution. We use the MyPy module to perform static type checking. To type check the source code, run the following command at the root of the project:

```
$ mypy ./code
```

Note

If you have troubles with the MyPy module, please refer to the official MyPy documentation.

8 Troubleshooting

8.1 Pip is not recognized as internal/external command

Error Message: This error occurs when `pip` is not in your PATH environment variable. You can try the following instructions to solve this problem:

1. If you are running Linux, try using `pip3` instead of `pip`.
2. Add `pip` to your PATH environment variable by following this tutorial.
3. Reinstall Python and `pip`.

8.2 WSL not supported

The error message indicates that your Windows machine is not able to run WSL. This could be caused by a **disabled Windows feature** called "Virtual Machine Platform". WSL depends on it. And in turn, Docker Desktop depends on WSL.

For more information on WSL and **virtualisation** on windows, please check out the following guides :

1. Windows Subsystem for Linux
2. Windows 11 Virtualisation
3. Windows 10 Hyper-V

9 About

9.1 License

9.1.1 Can I redistribute the application ?

Yes, you can redistribute the application even under another license, but you must be cautious about how you handle the GPL3-licensed Python module. The GPL3 license requires that any derivative work or modifications made to the GPL3-licensed module must also be licensed under the GPL3 or a compatible license. Therefore, if your modifications are to the GPL3-licensed module itself, those modifications must also be released under the GPL3 license.

9.1.2 Can I modify the application?

Yes, you can modify the application, but again, you need to consider the implications of the GPL3 license for the module. If your modifications are limited to the MIT-licensed parts of the application and do not involve modifications to the GPL3-licensed module, then you are free to modify the MIT-licensed portions as you see fit. However, if your modifications extend to the GPL3-licensed module, those modifications must comply with the GPL3 license terms, including the requirement to release the modified code under the GPL3 or a compatible license.

9.2 Developpers

People that took part in this project:

- **Developer** Kevin Schweitzer, Kevin.schweitzer@student.unamur.be.