AAPVL Tool Technical Documentation: Introduction and installation

0. Acknowledgment

The project was supported by funds of the Federal Ministry of Food and Agriculture (BMEL) 2819102112 based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support programme. The project wass carried out in cooperation with the German Federal Office of Consumer protection and Food Safety (BVL).

1. Preface

The offered software prototype is one of the working results out of the research project called *AAPVL (automated computer-aided identification and evaluation of potentially non-compliant food products traded via electronic commerce)*. It was developed from February 2013 to September 2018 as part of the mentioned project. It is technically a software consisting of various modules, written in different programming languages and has to be hosted as a web application on a server like Apache. The state of the software is declared as prototype because there was no intention to build a full functionally commercial software and because there is no warranty of flawlessly working components. The components are more like templates and start points to build on to develop a fully operational holistic software.

2. Software Architecture

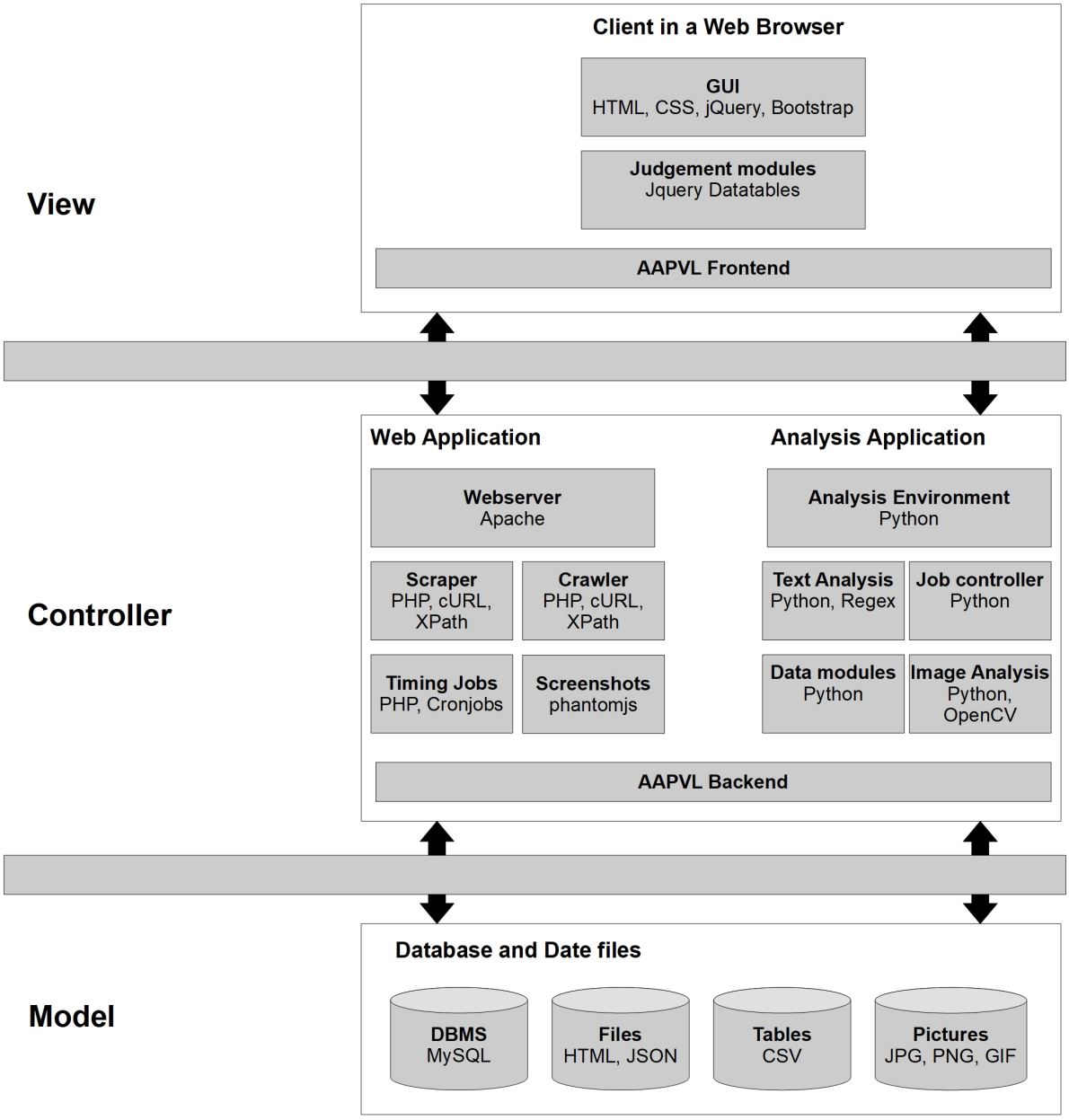

Figure 1: MVC pattern of AAPVL Tool

The software prototype is a modular server application built on several tools developed in PHP and Python. While the PHP parts provide the front-end application with a graphical user interface, the classification tasks are implemented with the components in Python (see fig. 1). The concept behind the software is the MVC pattern to make the software stable and extendable.

3. Software Installation

As mentioned before, the software is a hybrid tool built on different modules and technologies using a MySQL-Database as their main interface to exchange data. There are some requirements to install all the modules and packages to run the prototype. Unfortunately, the software is not offered as a whole package, but its components can be installed manually.

3.1. Requirements and dependencies to build up the software environment

The software prototype was developed and tested on a Debian Linux System, but it can be installed on other distributions too if they meet the requirements. It is also possible to run the software tools on a Microsoft Windows System as well (for example by installing the XAMPP Package consisting of Apache, PHP and MySQL (https://www.apachefriends.org/download.html).

3.1.1. Versions of the used programming languages

The software prototype has been developed on PHP 5.6 and Python 2.7. So, there is no guarantee that all components will work on higher versions of the used programming languages.

3.1.2. Apache Web Server

The software prototype is optimized to run on an Apache Web Server. So, a first step is to install the server environment on the operating system you prefer.

The server environment can be installed on a Debian system in that way for example

apt-get install apache2

3.1.3. PHP

Next step is to install PHP and the related apache modules on the system.

apt-get install php5 libapache2-mod-php5

3.1.4. MySQL

MySQL is the interface to exchange data in between all tools. To install it, type:

apt-get install mysql-server mysql-client

apt install python-mysqldb

3.1.5. Python

You need to follow these steps to install Python on the Debian system:

apt-get install build-essential checkinstall

apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev

cd /usr/src/Python-2.7.13

./configure

make

make altinstall

wget https://bootstrap.pypa.io/get-pip.py

python2.7 get-pip.py

Python can be also installed on a Windows System by downloading the installer at https://www.python.org/downloads/windows/

3.2. Installation of the Web Application

The web application needs no particular packages to run it in the software environment. All packages and libs (like jQuery, Bootstrap, phantomjs) are included but will be outdated at some time. Updates must be conducted manually.

3.2.1. Setup MySQL

The files come with a mysql script "/mysql/aapvl_install_mysql.sql ". Just install that file to the MySQL Database.

3.2.2. Copy of the application files

Just copy all application files with all sub folders to the desired directory on the Web-Server. The web application consists of the following directories:

directory	explanation
application	Main folder for all classes, controller, models and view.
helper	Self-developed libs with functions for recurring tasks to simplify some operations
inc	Folder for all additional files to run the web app like all JavaScript, CSS and libraries of third party vendors.
resources	Folder to save the local texts and screenshots of web sites.
results	Generated reports of classifications and analysis.
server	Interface and template files to connect the functionality of jQuery Datatables and PHP via Ajax
tmp	Just a temporary folder to save temporary cookie information.
user	Holds all JSON-Files to configure the modules of the application and to create and edit language files for the GUI.
./	Root with error.log, index.php and also a phantomjs.exe to run the software on a Windows system. Here is also a .htaccess which needs to be adjust to run the software on the Webserver.

3.2.3. Setup the configuration for the web application[1]

Open the config file of the application in “/application/config.php”. Change the following parameters to meet your requirements:

// Standard Configuration for the app in the local environment, needs to be adjusted to run the application on a Web Server

define('URL', 'http://localhost/aapvl/'); // // Main URL of the app

define('DB_TYPE', 'mysql'); // DBMS

define('DB_HOST', 'localhost'); // DB Host

define('DB_NAME', 'aapvl'); // DB Name

define('DB_USER', 'root'); // DB User

define('DB_PASS', ''); // DB Password

define('WEB_PATH', 'resources/'); // Internal Direcotry to save screenshots and websites for the cases

define('RESOURCE_URL', 'http://localhost/aapvl/resources/'); // External URL to access the resources

define('EXPORT_PATH', 'http://localhost/aapvl/reports/'); // AEternal URL for the reports

The next step is to open the “/.htaccess” to change the Rewritebase there to meet your requirements there. Change the value to the path on the server where you want to install the software.s

RewriteBase

Rewritebase /aapvl/

3.3. Installation of Analysis modules written in Python

The installation of the analysis modules is more complex because of some dependencies and non standard libs.

3.3.1. Python packages

These packages can be installed through the package manager of your distro or through the python package manager pip.

• numpy

• scipy

[1] Further information to configure the software can be found in the document “Documentation_AAPVL_Web”

• sklearn (version 0.18)

• opencv and python-opencv (you should use the official packages here: aptitude install opencv python-opencv)

• nltk

• sklearn-crfsuite

• bs4

• dateutil

• sphinx (to build the documentation only. If you use virtual environments, sphinx has to be installed in the same virtual enviroment to be able to build the api reference.)

3.3.2. non-standard libraries and programs

For installation of the non-standard libraries and programs please follow the installation guides for that program.
• libpostal with python bindings (package name postal): https://github.com/openvenues/libpostal
• ParZu: https://github.com/rsennrich/ParZu
• Zmorge model for ParZu: http://kitt.ifi.uzh.ch/kitt/zmorge

3.3.2.1. libpostal

If you install libpostal into a user specific path and you want to install the python package postal afterwards, you have to specify the path to libpostall for the installation. With pip you can use:

*pip install --global-option=build_ext --global-option="-L/home/foo/libs/libpostal_*
↪build/lib" --global-option="-I/home/foo/libs/libpostal_build/include" postal

3.3.2.2. ParZu

It is sufficient to follow the installation instructions for ParZu until step 3 first part. You don't have to use the script install.sh. All the components installed with this script are not used from this program.

3.3.3. Setup the configuration for analysis applications

The program has some parameters (e.g. user name for the database connection etc.) that should be configured. Therefore a default configuration file (short: config file) exists and can be customized. The file is located in the root directory of the program. The different parameters and their meaning is explained in this chapter and the default config file is shown as example. Note: the keywords are case sensitive, so make sure you spell them correctly.

- **host** IP adress to which the database is reachable. You can leave the default value, when you have your database local

on your machine. default: 127.0.0.1

- **user** Username for the database user. default: foo
- **passwd** Password for the database user. default: bar
- **db** Name of the used database. default: aapvl
- **max_tries** Maximum number of failed tries to obtain jobs from the database before ending the program. If this
number is negative, the program tries four times per day to obtain jobs and then sleeps in the background for the
rest of the time. default: 3
- **delay** Time in seconds to sleep before trying to get jobs from the database. The delay time is only used, after a try to
obtain jobs was not succesful and is not used when max_tries is negative (see also **max_tries**). default: 3600
- **delay_module** Time in seconds after which each module has to finished. If a module needs more than delay_module
seconds, it is terminated with all its subprocesses and None is returned as a result from this module. default:
3600
- **day** Number of times per day the program should wakeup and try to obtain jobs from the database. default: 4
- **update_rate** Number of days after which the database should be checked for altered classification results for online
learning. default: 7
- **food_vocab** Path to a file containing food relevant vocabulary for the food classifier (see also chapter *The food vocabulary (module 3):*). default: data/food_vocab.txt
- **map_file** Path to a file containing an assignment between postal codes and german regions. For more details also information on the assumed format see chapter *Postal codes for address extraction (module 1):*. default: data/plz.csv
- **legal_numbers** Path to a file containing all approved boards of control for ecological traders. For more information on the file and format see also chapter *Legal numbers for ecological control posts (module 6):*. default: data/legal_oeko_numbers.txt
- **health_claim_substances** Path to a file with common substances used in health claims. For more information on the
file, the format and how to update it see also chapter *List with substances*

(module 7): and *List with substances:*.
default: data/health_claim_substances.t