README for PAL

Maria Skeppstedt

April 3, 2017

Introduction

The PAL package trains a classifier to select suitable data to annotate from a pool of unlabelled data and to carry out pre-annotation of this data. PAL is meant to be used for annotations of named entities or other types of shorter chunks of text. The pre-annotated data can be loaded into the BRAT tool[1], to be revised by the human annotator. The PAL package is written in Python 3. It is developed to be used in a Unix environment, and it has been tested on Ubuntu 12.04.5.

Active learning and pre-annotation

The PAL package is structured around three different data folders:

1. The *labelled* folder, which contains the data that has been manually labelled.

2. The *unlabelled* folder, which contains the pool of unlabelled data.

3. The *tolabel* folder, to which the pre-annotated, actively selected data is written.

When a project starts, it must contain the data and the folders that are shown in the example folder *data/example_project*. In the *labelled* folder, there must be at least one file containing the seed set of annotated data, which is required to start the active learning process. In the example project, the seed set is split into three different files of annotated data (but its okay to have one or more than three files as well). The file that contains the pool of unlabelled data must be called *unlabelled.csv* and must be positioned in the *unlabelled* folder.

When the process of active learning and pre-annotation is run, all .csv-files located in the *labelled* folder are used to train a machine learning model. This model is then employed to perform the active selection of training samples and the pre-annotation. Data samples to be labelled are selected from the file *unlabelled.csv*, pre-annotated and written to the three files that are created in the *tolabel* folder. These three new files receive a name containing their creation timestamp. The two files named with the file extensions *.txt* and *.ann* contain the pre-annotated data in BRAT format. That is, those two files are the ones that can be directly imported into BRAT for manual annotation. The file *unlabelled.csv* is also updated and does not any longer contain the data samples that have been selected for annotation. That is, the selected data samples have been removed from the pool of unlabelled data. A copy of the original version of the unlabelled data is also created in the *unlabelled* folder.

The data in the .csv-files must be available in tab-separated format in which the data has been tokenised with each token on a separate line. The data must also be segmented into sentences with an empty line signalling a sentence

[1]http://brat.nlplab.org/index.html

break. The tokens of labelled data are expected to be labelled according to the BIO-format, i.e., a token could be the *B*eginning of, *I*nside or *O*utside of a named entity (or of another type of text chunk). The data samples in the active selection process consist of the sentences. This means that text units in the form of a number of tokens that are separated by an empty line are the units which are selected in the active sampling process. See the file *unlabelled.csv* for an example of data format for unlabelled data, and see any of the files in the *labelled* folder in *data/example_project* for an example of the data format for the labelled files.

Configuring PAL

To run the package, a directory for the project needs to be created. This directory needs to contain the directories with *labelled* and *unlabelled* data (as described above), as well as a *settings.py* file with configuration information. The most important configuration parameters are:.

```
# Classes to include with their prefix (B or I).
# Classes, apart from "O" the outside-class, not in this list, will be ignored
minority_classes = ["B-speculation", "I-speculation", "B-contrast", "I-contrast"]

# Number of sentences to be actively selected and pre-annotated in each round
# (referred to as k above)
nr_of_samples = 20

# Type of model to use
model_type = NonStructuredLogisticRegression

# The context around the current word to
# include when training the classifiers
number_of_previous_words = 2
number_of_following_words = 1

# A cut-off for the number of occurrences of a token in
# the data for it to be included as a feature (current and context-token)
min_df_current = 2
min_df_context = 2
```

There are, however, also a number of other parameters that can be given to PAL. For a description of those, see the *settings.py* file or the *settings.html* file in the *documentation* folder.

Running PAL

The functionality for selecting and pre-annotating samples can then be run with the following command, in which the location of the data and the settings file is specified:

```
python active_learning_preannotation.py --project=data.example_project
```

The data files that are created by this command (the .txt and the .ann files) can then be moved to the BRAT *data* folder to make it available for the annotator to select for annotation. When the annotator has finished the annotation/correction of the pre-annotated files, they can be transformed back to a tab separated format and positioned in the folder with annotated data with the following command (on one line, without line breaks):

```
python transform_from_brat_format.py
 --project=data.example_project
 --annotated=my_brat_path/data/example_project/brat_tolabel_20160928_174414
```

The name *brat_tolabel_20160928_174414* is the path to the .txt-file and to the human-annotated version of the .ann-file, but without the suffices. The two files must be positioned in the same folder.

Installing the external libraries on which PAL is dependent

PAL makes use of three external libraries, which all are freely available, PyStruct, Scikit-learn and Gensim. These libraries are in turn dependent on cvxopt, numpy and scipy.

Given the existence of a Miniconda installation[2], the following needs to be carried out to install these libraries:

```
conda install numpy
conda install scipy
conda install scikit-learn
conda install gensim
conda install cvxopt
pip install --user --upgrade pystruct
pip install joblib
```

[2]Install script for Miniconda for different OS:s and descriptions of how to create conda environments can be downloaded from:
http://conda.pydata.org/miniconda.html
For instance, to install for Mac write:
bash Miniconda3-latest-MacOSX-x86_64.sh
To create a conda environment, write:
conda create -n py3k python=3
and to activate that environment write:
source activate py3k