# AC-PLT (Automatic Codification of PLT)

This project presents a novel algorithm that uses natural language processing (NLP) techniques to semi-automatically code feature listing data. The corresponding code is experimental and was applied to two Spanish datasets. We also provide an English option that has not been tested.

The code performs 3 main steps, data cleaning, word embedding, and classification.

To run the algorithm with the provided example, download all codes and required files (embedding), install the corresponding libraries, and run the code without any change. Otherwise, to apply the model to your own data and language, you have to just change the paths of files for training and of the data to be codified, following the specifications of the CSV files (see below for more details).

## Requirements

This project runs under python==3.10.9. You can install all the libraries using pip install -r src\requirements.txt. The original code uses the vectors of the [Spanish-word-embedding](https://github.com/dccuchile/spanish-word-embeddings#word2vec-embeddings-from-sbwc) (specifically the Binary format .bin.gz). After you download the file, add it to the data folder. In the case of an English dataset, you can use the [word2vec-google-news-300](https://huggingface.co/fse/word2vec-google-news-300/tree/main). Remember that the English version has not been tested.

Uses

Suggestion

The jupyter notebook file AC\_PLT.ipynb has the main code and the corresponding instructions. To use this model the dataset needs to be in a csv file with the three following columns: concept (the name of the concept, e.g. DOG), feature (the raw property listed by a subject, e.g. “has four legs”, and the code manually assigned by a human coder to the raw property (e.g., “quadruped”) (please keep the order of the columns, but the labels of the head columns are not important).

The *Important Variables* section of the code contains the main variables for the model, which may be changed according to your needs:

- pathTrainData: The path and name of the file that contains the training dataset.

- pathData: The path and name of the file that contains the data (raw properties) to be coded.

- pathEmbedding: The path and name of the embedding file that must be previously downloaded.

- outputFile: The path and name of the output file, where the results must be saved.

- numberCluster: The number of clusters that the tool will generate. This corresponds to the maximum number of codes that can be learned and assigned by the algorithm.

- numberCodes: The number of unique human-generated codes that will be suggested by the algorithm.

- language: language of the training and dataset files. It can be set to "spanish" or "english".

The section *Setting the Language* can manage Spanish and English. For other languages, please modify the code at your own convenience, but remember that the code has been tested only in Spanish.

The code in the *Important functions* section implements the function for the *Data Cleaning* process.

The *Word Embedding* section applies the embedding process based on the selected embeddings. The algorithm transforms the training data and the data to be coded (raw properties) into numeric vectors of 300 dimensions, using the embedding defined in pathEmbedding.

Section *Model* has the main algorithm: AC\_PLT(). This class needs a single hyperparameter (n\_clusters), which corresponds to the important variable numberCluster defined before. Remember that this is the number of clusters for the *k*-Means algorithm. The important functions of this class are:

* fit(): receives the training dataset and trains the *k*-means algorithm with n\_clusters clusters.
* suggestions(): receives the data to be coded, and returns a matrix with the numberCodes codes assigned by the model to the dataset.

The *Codification Suggested* section trains the model with the number of clusters using the training data, and applies the learned model to the data to be codified. Finally, a CSV file is generated with all the codifications suggested by the AC-PLT model, which is saved in the defined path and name by the outputFile variable.

Replication of results (using the provided files)

The file Replication.ipynb has the code for replicating the results corresponding to the CPN27 and CNP120 datasets. This code does not generate a codification in a CSV file. Instead, it calculates the top-*p* (numberCodes) accuracies applying *k*-fold cross validation to the training data. Two new sections were added to this code: *Experiments* and *Search of hyperparameter k*. If run properly, you can replicate the results of the paper using the original datasets CPN27 and CPN120.