Technical report of the first machine learning class competition submission

Introduction

The assignment consists in predicting the scores of user reviews originating from TripAdvisor with a linear model. To do that, we receive a set of 194000 reviews that we will divide in a **training** and a **validation** with a ratio of 80% - 20%. The training reviews are grouped by hotel and contains the following fields f_{tr} :

- Author: text
- Content: text
- Date: date
- Hotel ID: text
- Rating: a natural in [0, 5], -1 otherwise
- Subratings: a vector of 5 natural in [0, 5], -1 otherwise

The predictions for the competition are runned on a set of approximately 6000 reviews with the following fields:

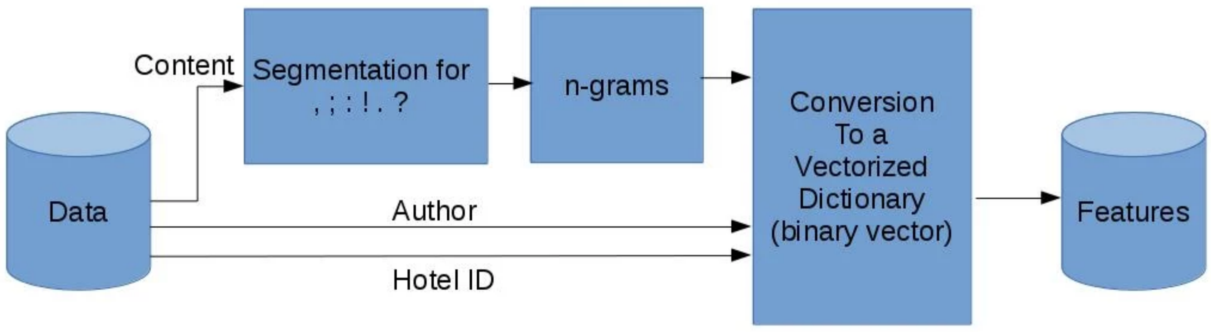
- Author: text
- Content: text
- Date: date
- Location: text
- Price: a positive real

The goal is to predict the *Rating* field. For this purpose, we will divide our work into three parts: feature extraction (*features.py*), training (*train.py*) and validation (*validate.py*).

Methods

Features extraction

In this part part, we try to summarize the reviews to a sparse binary matrix according to the following schema:


Content
Segmentation for , ; : ! . ?
n-grams
Conversion To a Vectorized Dictionary (binary vector)
Data
Author
Hotel ID
Features

To segment the sentences in a natural way, we splitted them according to the punctuation marks. Then, we agglomerated consecutive words together into n-grams (or shingles) to keep a representation of the temporality present in natural language [1]. In the current version, we use only 2-grams as it seems to give the best results. A possible explanation is that the database size is too small when we process shingles composed of too many words. That can be assessed by measuring the number of occurence of each shingle in the database: starting from n=3, the number of n-grams that appear only one time in the database excess 70%, which mean that a new review has poor chance to be processed properly. A possible solution would then be to filter the meaningless shingles instead of keeping them (certainly with adapted python libraries).

An other important remark is that the features matrix is very sparse. In future work, we'll try to reduce dimensionnality with an algorithm like MCA [2].

Finally, authors and hotel ID are just added as binary inputs in the vector list but we should try to come up with a representation more homogeneous with the n-grams and also including the date of the review. The price could also be integrated in the test phase as it certainly bring new information to the prediction.

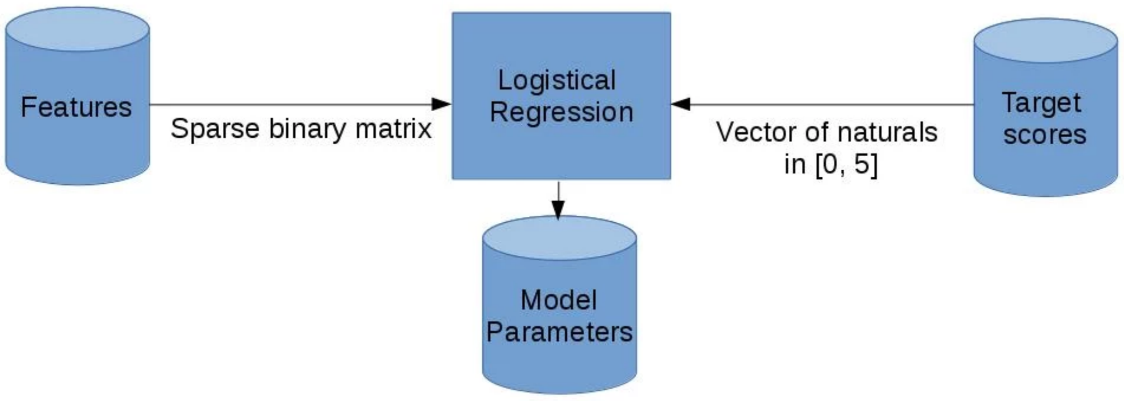
To start feature extraction, type:

```
python2 features.py
```

The features pickle file and the feature processing model pickle file are saved in the *Features* folder.

Training

Training consists in a linear supervized learning method: logistic regression.



It has been applied by using the module **linear_model.LogisticRegression** in the python sklearn library with all the default parameters. The main disadvantage is that we are not currently using any technique to determine when the algorithm starts overfitting. This will be the first priority in future work.

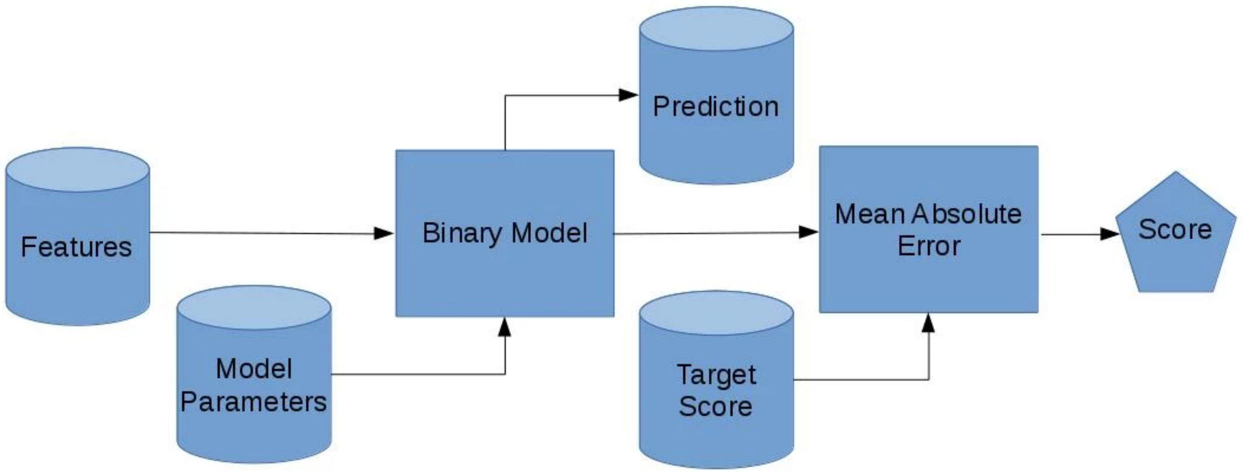
The training algorithm can be started with:

```
python2 train.py <features_file.pkl>
```

The model file (and inherently its parameters) and the a training score file are saved in te folder *Models* .

Validation

Finally, though we are not monitoring the regression algorithm with a validation set during its execution, it is interesting to validate the results manually afterwards by computing the score on a validation set and adapt certain parameters.



The best score we obtained so far in a very short period of tryouts (half a day) is 0.593 when taking 80% of the dataset for training and 20% for validation. The results in the competition should probably be a bit better, as the dataset hasn't been shuffled and the reviews are still ranked by hotel ID (therefore some information in the validation dataset is not present in the training set).

To run the validation script:

```
python2 validate.py <feature_model_file.pkl> <regression_model_file.pkl>
```

It produces a validation score file in *Models* in order to compare it with the one produced during trianing and discuss overfitting.

Prediction

To create a prediction numpy file then the CSV file for kaggle, just start:

```
python2 predict.py <feature_model_file.pkl> <regression_model_file.pkl>
python2 create_submission.py <prediction_numpy_file.npy> [<prediction_csv_fil
```

Results

In the best models we got so far (using 3-grams, 2-grams and words together), we have the following scores:
- **Training** : 0.117
- **Validation** : 0.584
- **Test** : 0.538

Future work

As explained previously, the first next steps for future work will be:

- Reducing overfitting
- Extract better features for natural language processing, maybe also using some external knowledge about language and homogenize features for all review fields
- Test non-linear regression methods and tune their parameters better

References

[1] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets, Cambridge University Press.
[2] Abdi, H., & Valentin, D. (2007). Multiple correspondence analysis. Encyclopedia of measurement and statistics, 651-657.