Ensemble Learning

Master in Data Sciences & Business Analytics - Text Classification with Rakuten France Product Data -

March 26, 2020

Instructors: Myriam TAMI - Parantapa GOSWAMI

Michaël ALLOUCHE - Raphaël ATTALI - Thomas DORVEAUX

---

michael.allouche@student-cs.fr - raphael.attali@student-cs.fr - thomas.dorveaux@student-cs.fr

Introduction

The cataloging of product listings through title and image categorization is a fundamental problem for any e-commerce marketplace, with applications ranging from personalized search and recommendations to query understanding. Manual and rule-based approaches to categorization are not scalable since commercial products are organized in many classes.

Deploying multi-modal approaches would be a useful technique for e-commerce companies as they have trouble categorizing products given images and labels from merchants and avoid duplication, especially when selling both new and used products from professional and non-professional merchants, like Rakuten does. Advances in this area of research have been limited due to the lack of real data from actual commercial catalogs.

Here is the link to the Gitthub project : https://github.com/thomasdor/EnsembleLearning_RTM.git. It contains :

- The final code :"template_script_V2.py"
- The inital datasets (X_train, Y_train, X_test) from Rakuten
- The jupyter file of our original code
- The jupyter file we used to process the image and train a ResNet50 on it
- Our final report

1 Problem definition

The project focuses on the topic of large-scale product type code text classification where the goal is to predict each product's type code as defined in the catalog of Rakuten France. This project is derived from a data challenge proposed by Rakuten Institute of Technology, Paris.

This data challenge focuses on multimodal product type code classification using text and image data. For this project we will work with both text and image data.

Goal of the project: large-scale product data classification into 27 product type codes

2 Dataset description

2.1 Dataset presentation

For this project, we have the opportunity to work with real data, provided by Rakuten France. In this real-life situation project, the dataset provided contains 99K product listings. These data, in CSV format, include some information regarding each product:

- Integer ID: this ID allows one to associate the product with its corresponding product type code
- Product designations: the product title and a brief summary of the product
- Product descriptions: this description relies on a more detailed description of the products. Since the marchants don't necessarily fill this specific fiels, this columns may contain "NaN" values
- Product ID: unique ID for each product
- Product images: unique image ID
- Product type code: product code used for classification purposes

2.2 Data exploration

Since the aim of this project is to classify products into product types according the available Rakuten data consists of training and test sets. The training and test set contain respectively 84,916 and 13,812 product listings. For each set, the explanatory variables (Integer ID, Product designations, Product descriptions, Product ID, Product images) and the explained variable (product type code, the necessary classes for the classification task).

Among the listed products, we noticed that the language designation may vary. In fact, as shown on *fig. 1*, French and English are the 2 major languages used to describe the products. The number of product described in French reaches 74.6 % of the entire dataset, while the English ones represents 13.2 % of the dataset provided by Rakuten France.

Since the aim of this project is to be able to classify products per type ID, using specific features such as designation, understanding the products distribution according to the type ID was also a key analysis to perform. As underlined by *fig. 2*, the more represented products belong to the type: 2583.

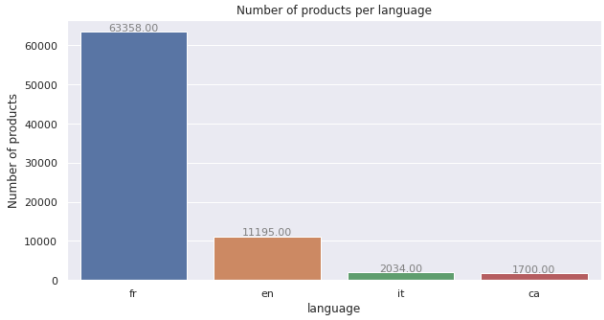

Fig. 1: Number of products per language

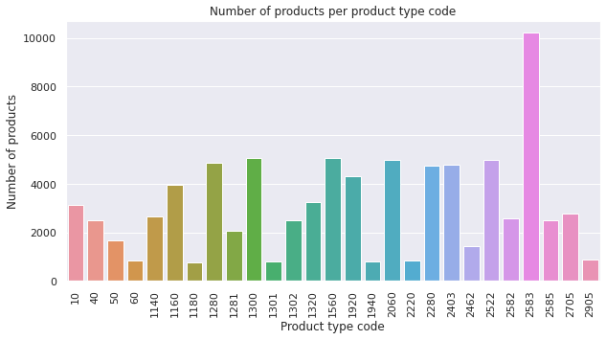

Fig. 2: Number of products per type ID

3 Models used

For this work we naturally chose the "state of the art" models that we have notably studied in the course, which are essentially decisions tree models. Namely, we used the following classifier from the scikit-learn package in python: Decision Tree (one with parameters set by default and one with parameters optimized with Grid Search), Random Forest, AdaBoost and XGboost (all optimized with Grid Search).

3.1 Decision Tree

The Decision Tree classifier from scikit-learn creates a classic decision tree model to classify our data. It splits at each node using a Gini or an entropy function. One of the major drawbacks of decisions trees is that they often overfit. Indeed, their architecture is entirely structured around the training dataset. While they can perform decently on the trainig dataset, decision tree algorithm may generate poor results on the test dataset. As it is the most simple and basic decision tree model, we do not expect this model to perform well compared to the others.

3.2 Random Forest

Random Forest is a model of bagging, meaning that it combines several trees and aggregate their results to improve its performance. It is one of the most famous bagging model and as well as one of the most powerful ones. It aggregates of a collection of trees. It uses bootstrapping to train the trees and randomly chooses the features to split each node during the tree-growing process.

Compared to a classic decision tree classifier, the Random Forest significantly reduces the variances as it uses different samples for training, specifies random feature subsets and builds and combining small (shallow) trees.

3.3 AdaBoost

Contrary to the Random Forest classifier, Adaboost puts a weight on every trees it is using. The decision made to classify an element of the dataset uses all the weak learners to build a strong learner. To do so, trees are ponderated according to their capability to correctly classify. As it is a boosting algorithm and not a bagging algorithm, the trees are not independently generated : each new tree is built in function of the tree created just before.

3.4 XGboost

XGboost has been widely used in the data science community for a couple of years. In particular, this algorithm can be easily optimized and is scalable for high dimensional datasets or "big data". As our tf-idf matrix is huge (84916 rows and 55520 lines), using a scalable algorithm makes sense.

3.5 Image Classification

For the image classification, we trained a ResNet50 as a Convolutional Neural Network (CNN) to train our model.

4 Experimental strategy

4.1 Text pre-processing

Once we analyzed the data, we first pre-processed the text data using common Natural Language Processing Techniques.

Standardizing characters: the first pre-processig step we implemented consists in standardizing all the text-related characters. In other words, we created functions allowing us to switch to lower case characters. Then, we decided to remove all the non-alphabetic characters. In fact, we are convinced that these non-alphabetic characters don't add specific information.

Detecting languages: since we detected 2 main languages, we made the choice to use different spacy models. In fact, the next steps (lemmatization and stop words) depend on the spacy model chosen. It seemed therefore the most appropriate approach for that.

Tokenizing data: then, we transformed strings into tokens. This step remains important to find lemmas and stop words.

Lemmatizing data: lemmas allow us to standardize the written style, by transforming each word to the root of the word.

Removing word: some words don't provide any insightful information to the meaning of the sentences. This is the case of the stop-words which can be removed using the model which corresponds to the sentences languages. We also felt that very small words, don't necessarily have a positive impact on the meaning of the products designations. Thus we removed the words which have a lengths lower or equal than 2.

Removing accents: finally, to completely standardize our dataset, we removed all the accent which occur when dealing with french designations.

Recombining tokens into strings: many applications require strings as inputs, not a list of tokens. So, we merged tokens of the same designation into a single string. This generates clean strings which will be used to implement the ensemble models.

4.2 Vector-space model

once we pre-processed the data, we transformed our set of documents into a set of features, which can be directly used in Machine Learning tasks. To do so, we build the TF-IDF matrix which is more efficient than a simple term count or term frequency scheme by combining two term statistics components: local and global components.

4.3 Parameter Tuning

Cross validation: in order to train validate our models we used a K-fold cross validation method. First, the dataset is randomly shuffled, then it is splited into k-groups. For each unique group, one group is hold out as test data and the remaining groups are set as training data set. At the end we get a validation score for each group. When choosing our parameter K we chose a K large enough because it reduces bias towards overestimating the true expected error. However a very large K increases running time. As our training time was very long with a K=10, we attempted to reduce it. We finally chose K=5, which we found was a good trade-off. In order to reproduce all our results, we used the parameter random state=50 for each of our models.

Grid Search: in order to tune each of our models, we used a Grid-Search method. Grid-searching is the process of scanning the data to configure optimal parameters for a given model. With grid-search for every set of parameters, we get the best parameters with the cross validation scores resulting for these parameters.

Decision Tree: for our decision tree model we chose to use grid-search on the parameters criterion, max_depth, min_samples_leaf.
criterion : The function to measure the quality of a split.
max depth :The maximum depth of the tree, if too small , this increases bias if too large it may cause overfitting and increase running time.
min sample leaf: The minimum number of samples required to be at a leaf node.
For all of our ensemble models, we used the parameter n_jobs=-1 to run our models using all the avalaible CPU's to reduce the training time.

Random Forest: for our Random Forest model we chose to use grid-search on the parameters n_estimators,max_depth. The parameters n_estimators controls the number of trees in the forest. It has to be set large enough to reduce variance but not too large to avoid too long running time. The parameter max_depth is similar to the decision tree.

Adaboost: for Adaboost, we performed a Grid Search on the parameter n_estimators which is the maximum number of estimators at which boosting is terminated.

XGBoost: for XGBoost, we performed a Grid Search on the parameters n_estimators and max_depth. These are similar to random Forest.

5 Comparison and analysis of the results

	Accuracy	Weighted F1 Score	Hyper-parameter(s)
Image (ResNet50)	0.37 imagesize = 32:32	0.51 imagesize = 100:100	
Decision Tree	0.5515	0.5615	criterion = Gini, max_depth = None
Random Forest	0.7493	0.7492	n_estimators=140,random_state = 50, max_depth = None
Adaboost	0.267	0.1938	random_state =50, n_estimators = 60
XGboost	0.713	0.7241	max_depth = 3,n_estimators = 50

As expected, the classic decision tree performs badly compared to bagging (Random Forest) and boosting models (AdaBoost & XGboost).

The tuning of the hyper-parameters does not improve significantly the score of the classic decision tree, since there is only a slight difference of accuracy between the optimized decision tree and the optimized one.

Not surprisingly, XGboost performs better than AdaBoost. It is due to the inherent quality of XgBoost which seems to be the most robust and efficient boosting model as of today.

AdaBoost performs very badly compared to XGboost. It is actually quite difficult to explain such poor performance from AdaBoost.

Finally, the best classifier seems to be the Random Forest classifier.

6 Image classification

We also performed the multi-classification task with the image data from Rakuten. In order to achieve the multi-classification, we used a Resnet-50 model with already pre-trained weights. On this model , we applied a transfer learning method: we re-trained the last layer (the classifier) such that the model is more specific to our specific problem.

We used a batch size of 100, with and trained on compressed images of size (100,100) with a validation set of 30%.

After 2 epochs, and using the cross-entropy loss, we got an accuracy of 0.5193. Compared to the accuracy that certain deep learning level can obtain on the ImageNet dataset, it may seem disappointing. However, it is important to notice that the class or categories of product we are considering are probably far less homogeneous than those of the ImageNet dataset.

Once we trained our CNN model using transfer learning, we created a feature extractor, which allows us to extract feature maps from images. The extracted features are saved in a CSV file. This could be interesting for a further step which could combine both text features and feature maps to perform a more powerful final classifier.

7 Conclusion

7.1 Contribution of each member

	Preprocess	Model experimentation	Report
Michaël Allouche	30%	30%	40%
Raphaël Attali	30%	40%	30%
Thomas Dorveaux	30%	30%	40%

7.2 Conclusion

The scores obtained for the multiclassification are significantly better when using the product description, than using the images. The poor performance for the image method might come from the heterogeneity of the different categories of the Rakuten dataset. Hence it is certainly difficult for the CNN to generalize well on a given class.

It is interesting to observe that in our present context, Natural Language Processing and ensemble methods are more efficient than artificial neural network. It shows that the best algorithms are those who exploit the best the data at their disposal.

We can also notice that among all the tree based models, the Random Forest has the best F1 score and the best accuracy and therefore performs the best.