Robotoff Category Classifier Performance

ykulizhskaya@

OBJECTIVES

Recently the Robotoff Category Classifier has been re-trained to:

- Upgrade it to TF v2.6 and allow the model to be served via the TensorFlow Serving node, rather than bundling it with the Docker image.
- Increase the set of categories the model can predict from (the category set is derived from a snapshot of the category taxonomy).
- Train on a bigger dataset (the training dataset is a dump of the Product Opener products).

The training data used can be found here and the training code here.

As part of this effort, we would also like to evaluate the category classifier's performance and to evaluate whether there is room for automatic application of some of the classifier's output.

MULTI-LABEL CLASSIFICATION THRESHOLD

Predicting product categories is a multi-label classification problem - each product in the Open Food Facts database can belong to several categories (or labels).

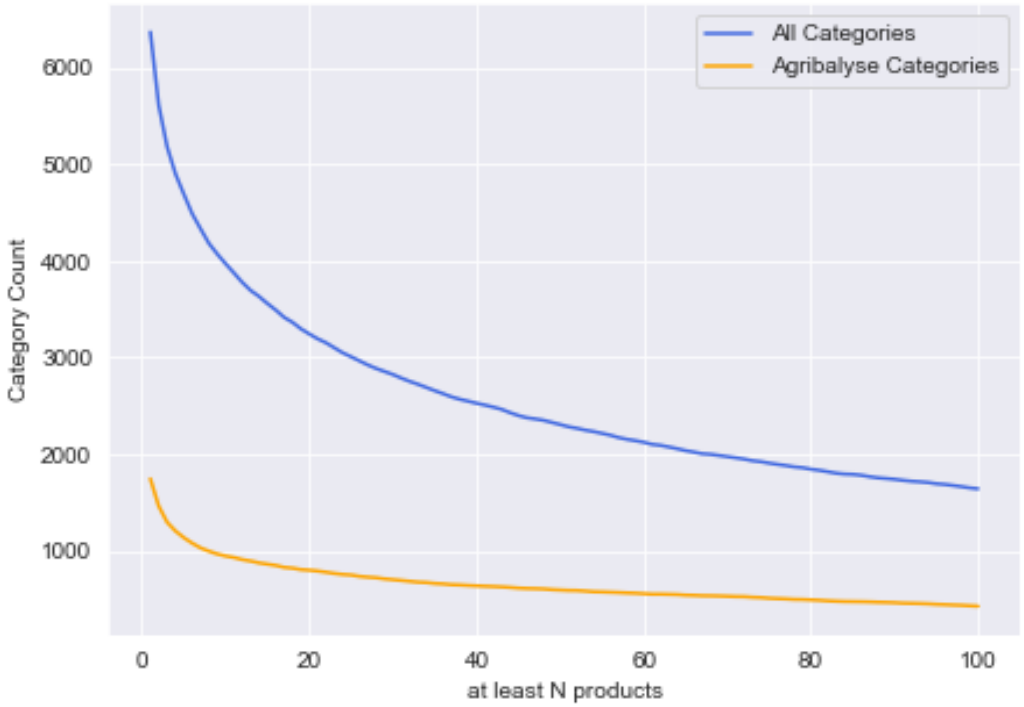
Since all of the categories are extracted from the category taxonomy, we have a relatively large label set: **6,529 unique categories**. In addition, some of these categories are assigned to a relatively small number of products - 1-2 samples per category.

Ideally we would want a balance between being able to predict from a wide range of categories, provide high quality data to the training model (e.g. enough data samples per category) and the ability to accurately evaluate the model's performance. The way to achieve this is to drop the labels with the smallest representation from the training dataset, while maintaining enough label coverage for every product.

This section attempts to provide some evaluation for how the set of labels the model is predicting from has been chosen.

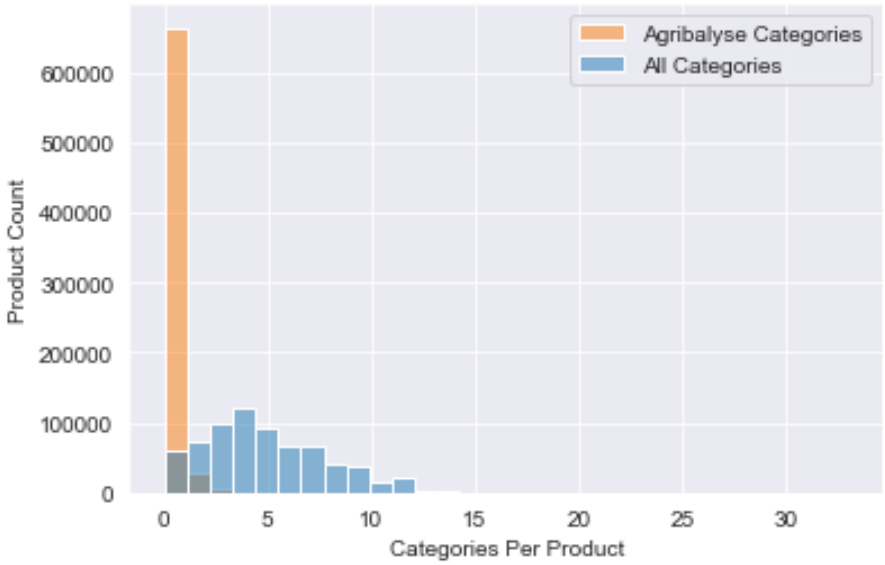
First, the entire Open Food Facts products dataset was split into the *training*, *validation* and *test* subsets in the *80:10:10* ratio. Since the validation and the test sets are only used for model verification, they are omitted from the data presented below.

We use a filter to drop all of the categories where the sample set is insufficiently small. The graph below shows the possible filter threshold vs. the number of categories we will still be able to predict from.



The sharply falling plotline indicates that the dataset contains a large number of very small categories. Therefore, a good filter value to choose could be just after the line begins to flatten - maximising the number of categories we're able to predict while removing enough noise from the input.

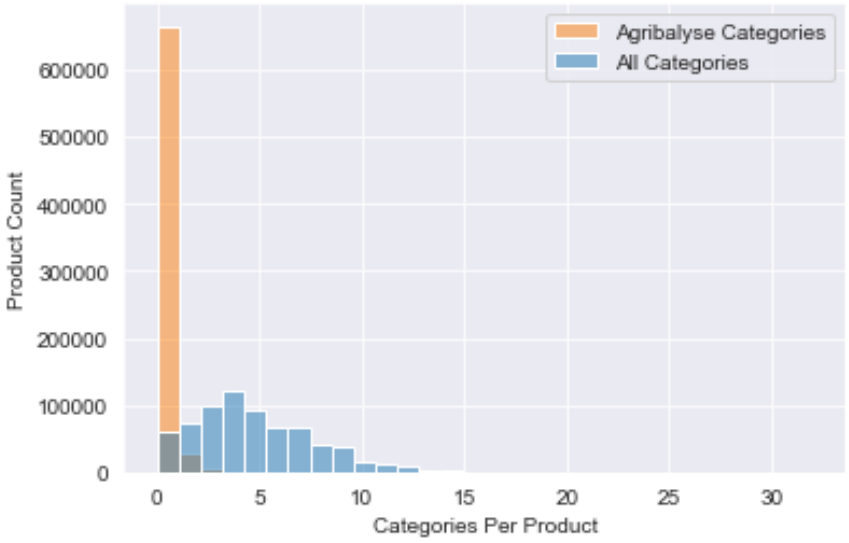
Looking at the number of categories a product has assigned to it gives us the following distribution:



Ideally, the label set we chose to predict from would give us a similar distribution of labels, such that we have similar category coverage over the OFF dataset.

Choosing the **category filter of at least 10 products**, gives us a label set of **3,969 categories** (62% of the total number of categories), out of which **949 are Agribalyse categories**.

Looking at the category per product distribution, gives us a similar distribution to the graph below (with the exception of the 'over 10 categories' buckets). This shows that the pruned categories were likely taxonomy leaves and that by using this category set, we're still allowing the ML model to provide good coverage over the products.



CATEGORY CLASSIFIER PERFORMANCE

The evaluation metrics for the re-trained category classifier are given below:

	precision	recall	f1
micro-	0.89	0.77	0.82
macro-	0.44	0.34	0.37
weighted-	0.86	0.77	0.8
samples	0.84	0.78	0.79

A brief description of the difference between micro-, macro-, weighted- and samples can be found here.

For the problem of product category classification, we're particularly interested in the 'micro-' metrics, as they put less emphasis on how balanced the training dataset is, don't assume each class has equal importance and better capture the performance of the smaller labels.

For precision vs. recall, we are more interested in precision - predicting (and applying) correct labels for products is more important than failing to predict all of the labels for a given product.

Therefore, the micro-precision of 0.89 and the sample precision of 0.84, along with recall of ~0.77 implies that the model performs well, both per-category and per-product.

Manual Evaluation of The Classifier's Performance

As well as calculating the metrics above, I also manually inspected 50 products that either had wrong predictions assigned to them (poor precision) or where the classifier had failed to assign all of the correct labels (poor recall).

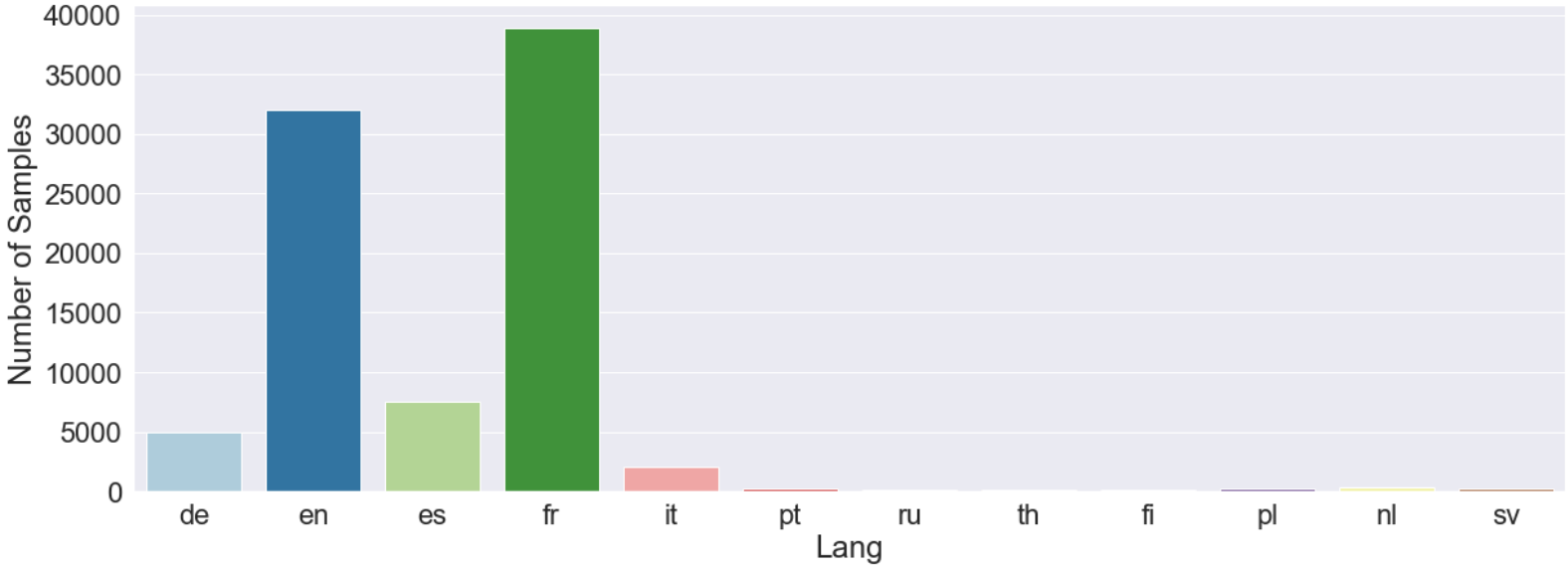
This was done to get a better sense of the classifier's performance and to see whether there were any patterns in the poorly performing products. My findings are summarised below, with the following patterns emerging:

- Many of the predictions classified as 'wrong' are actually good predictions that *should be* applied to the product (22 out of 35, or **63%**):
    - This likely means that the actual precision of the model is higher than the numbers given above.

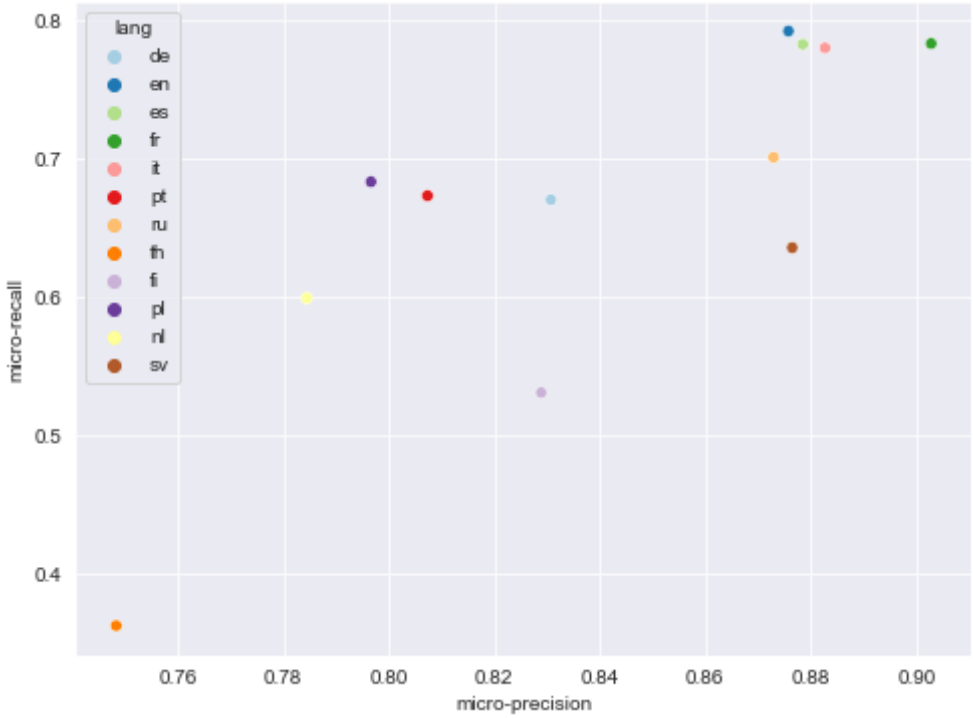
- Missed predictions tend to be at the end of a taxonomy chain, i.e. if a product has categories A->B->C->D, the most likely missing prediction would be ‘D’.
    - This indicates that we are still able to provide reasonable coverage over a product and only miss the smaller categories.
- If there are several wrong predictions or missed labels on a product, they tend to form a taxonomy chain that connects to the ‘true’ category graph. Example, the classifier might try to assign ‘fresh vegetables’ -> ‘mushrooms’ -> ‘white mushrooms’ to a jar of pickled white mushrooms.
- Number of products where the classifier was not able to predict anything useful - 2 out of 50.

Category Classifier Performance by Language

The test dataset contains products for **51 languages** - the breakdown of languages with at least 100 products in the dataset is given below:



Plotting micro-precision vs micro-recall of these languages gives us the following plot:

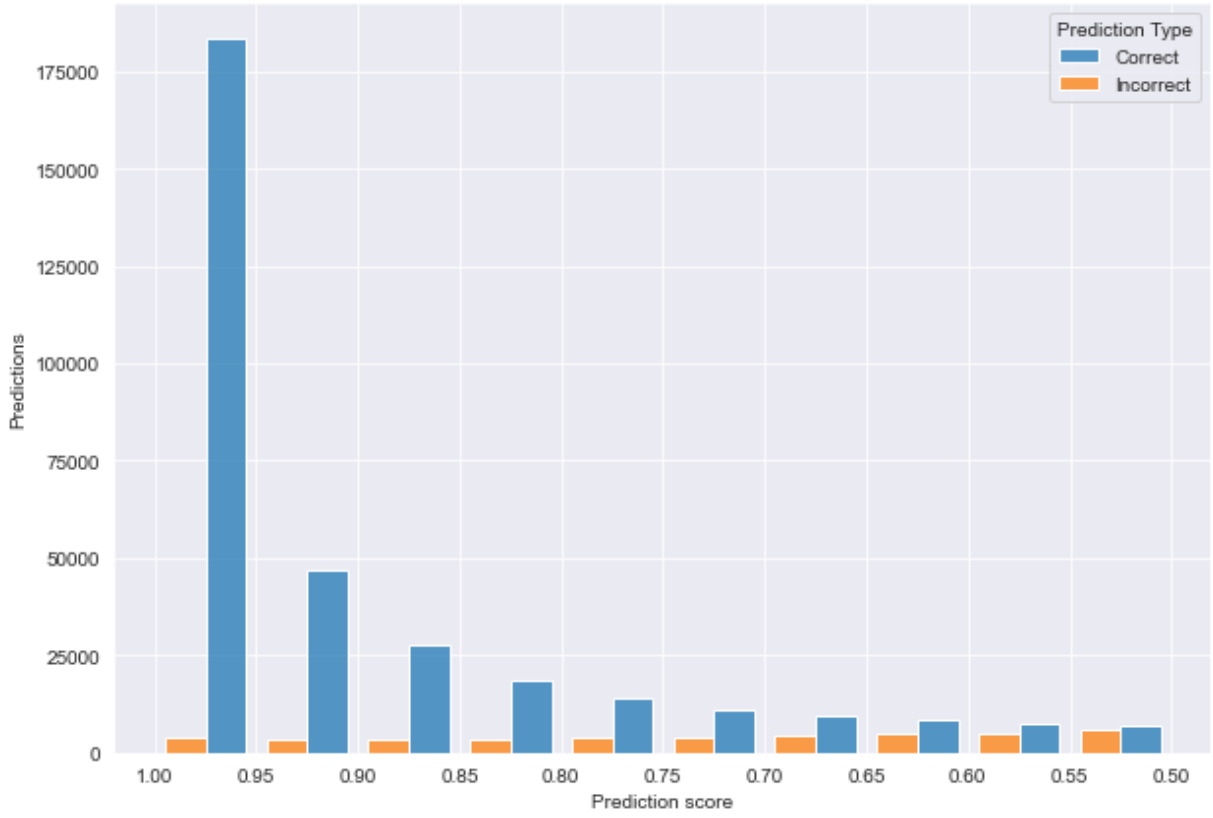


Automatic Application of The Classifier's Predictions

Given the high precision of the model, it is possible to automatically apply some of these predictions.

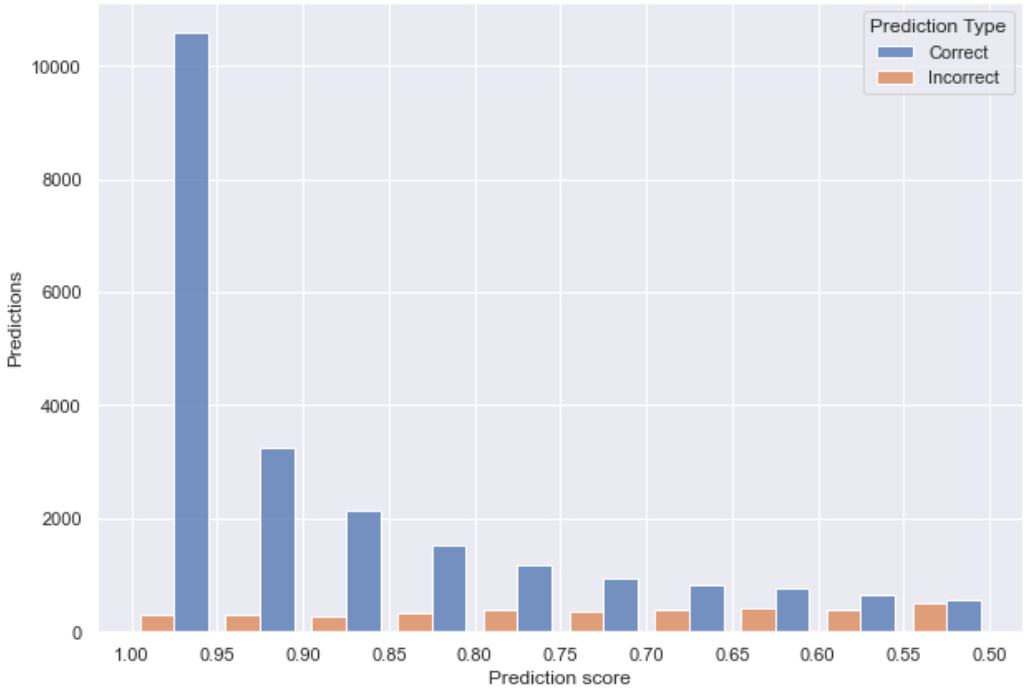
The automatic application can be done either based on the prediction score the model produces for a label, or based on where this label appears in the taxonomy graph (closer to a source node == the bigger the category == more likely it is correct).

Given the complexity of estimating a good heuristic for graph node location-based automatic application, I have instead chosen to look at the prediction score thresholds instead. Below is the breakdown of 'correct'/'incorrect' predictions vs the model's prediction score (keep in mind that for the manual inspection 60% of 'incorrect' predictions happened to be actually correct):



From the plot above, we can see that the vast majority of predictions have a prediction score of >0.9. Additionally, this is where the biggest difference in the number of correct and incorrect predictions is, i.e. it is much more likely that a prediction with a score of 0.9 and above is a good one.

The plot below shows the same prediction score breakdown for only Agribalyse categories:



Both of these graphs imply that **0.9 is a good threshold for automatic application of categories.**