Classifying companies in France using machine learning

Thomas Faria & Tom Seimandi (Insee)

1 Introduction

The French company registry, SIRENE, lists all companies in France and assigns them a unique identifier, the Siren number, for use by public institutions. As part of the registration process, companies must provide a description of their economic activity, which is then classified into an industry using the French classification of activities (NACE Rev. 2). Sirene underwent an overhaul in 2018 (Sirene 4) with the goal of improving its management, simplifying exchanges between stakeholders, and reducing waiting times. As part of this transition, a *one-stop shop* for business formalities was established. However, this change brings with it administrative constraints that pose a challenge to the automatic coding of company activities, which is crucial for managing the registry.

Currently, automatic coding is performed by a deterministic rule engine called Sicore, which uses a reference file as an example for encryption. This method is limited, with a 66% automatic coding rate in Sirene 3, and the transition to Sirene 4 is expected to further decrease this rate to 30%. Since all the activities that are not coded automatically are done so manually by Insee agents, this would result in an unmanageable increase in workload. Furthermore, maintaining and updating Sicore is an extremely complicated task. It seems opportune, therefore, to test the performance of statistical learning models to predict NACE codes from activity descriptions in order to replace or supplement Sicore.

In addition to the performance of the chosen model, particular attention has been paid to operational constraints so that these methods can be integrated into Insee's information system in the event of implementation. Given Insee's experience with this type of problem (see Leroy, Malherbe, and Seimandi (2022)) and the operational constraints, we chose to use the classifier from the *fastText* library, a neural network with one hidden layer, to predict NACE codes from activity descriptions.

To supervisedly train our classification model, we used a dataset made available by the Sirene division, containing over 10 million observations from Sirene 3 covering the period 2014-2022. The NACE codes, serving as ground truths, were labeled either by Sicore or manually by a

person. Performance evaluation was performed on both a test subset from Sirene 3 and all data from the *one-stop shop* received since January 1, 2022. We managed to achieve an accuracy of 80% on data from the *one-stop shop* without any manual coding, which is a significant improvement over Sicore's results. By using a confidence indicator, we were able to further increase accuracy to 90% with a manual coding rate of about 15%. The manual coding process can even be facilitated by suggesting the model's top predictions to the annotators. Indeed, in nearly 94% of cases, the correct NACE code is found in one of the model's top 5 predictions.

2 Data

In order to perform our classification task, we utilized a database containing over 10 million company descriptions spanning the period 2014-2022. This sample was labeled either by the Sicore engine or by an Insee agent when Sicore was unable to classify a description. Additionally, we collected all descriptions from the *one-stop shop* (15 696 descriptions) that were recorded since January 1, 2022. These descriptions were not used during the model training phase but only served to evaluate performance on future descriptions from Sirene 4.

An observation consists of free-form text describing the primary activity of the company, as well as 4 categorical variables:

- the nature of the company's activity (23 categories);
- the type of the description (15 categories);
- the type of event associated with the description (24 categories);
- the surface area of the company in square meters (4 categories).

Each observation is labeled with a unique NACE code. There are 732 NACE codes, but only 728 are present at least once in our sample. NACE codes consist of 5 nested levels, and an example of the decomposition of the code "1721B" is illustrated in Table 1.

Table 1: Example of the nested structure of the NACE nomenclature

Level	Code	Description	# **modalities**
Section	C	Industrie manufacturière	21
Division	17	Industrie du papier et du carton	88
Group	172	Fabrication d'articles en papier ou en carton	272
Class	1721	Fabrication de papier et carton ondulés et d'emballages en papier ou en carton	615
Subclass	1721B	Fabrication de cartonnages	732

3 Text classification based on fastText

3.1 Data preprocessing

Text preprocessing is a critical step in natural language processing (NLP) tasks. In our study, we followed traditional preprocessing steps, including removal of punctuations, stop-word removal, lower-case conversion, and stemming. In addition, we concatenated categorical variables at the end of the text, as we found that they could provide useful information for classification and the *fastText* classifier does not support the inclusion of categorical variables out of the box.

3.2 Training fastText model

fastText is a machine learning algorithm for text classification, developed by Facebook AI Research (see Joulin et al. (2016) and Bojanowski et al. (2016)). Texts are first mapped to numeric vectors (embeddings) and a simple one-layer perceptron is then used for classification. *fastText* goes beyond word-level analysis by also considering n-grams when computing the text embeddings, representing each word as a combination of character n-grams. By doing so, *fastText* can handle out-of-vocabulary words, misspellings, and variations of words that are not present in the training data. This is particularly useful in our case because we are dealing with short text that contains a lot of misspellings and abbreviations. In addition to this, *fastText* also takes into account n-grams of words, which gives importance to the word order when capturing the meaning of a sentence. We applied oversampling on the training data to improve the model's performance on minority classes. The model hyperparameters and training parameters that we used are shown in Table 2.

Table 2: Summary table of parameters

Parameter description	Value
Number of epochs	50
Learning rate	0.2
Dimension of the embeddings space	180
Number of different tokens	2 000 000
Max. length of word n-grams	3
Min. length of character n-grams	3
Max. length of character n-grams	4
Min. number of word occurrences (to get an embedding)	3
Min. number of observations per class (for oversampling)	5 000

4 Results

4.1 Performance

To analyze the overall performance of a classification model, it is common practice to use the accuracy rate, which corresponds to the percentage of correct predictions. Figure 1 shows the accuracy rates of the model on the test sample from Sirene 3 and on the *one-stop shop* sample for the different levels of the NACE nomenclature. These correspond to the results of a single model trained to predict the finest level of the nomenclature.

As shown in Table 1, the 5 levels of the NACE nomenclature are nested, so it is possible to deduce the more aggregated levels from the finest level. Although the objective is to optimize the model's performance on the finest level without taking into account the higher levels, analyzing the results on these levels is interesting. From Insee's point of view, two incorrect predictions are not quite equivalent. Predicting code 0126Z ("Cultivation of oleaginous fruits") as 0125Z ("Cultivation of other tree or shrub fruits and nuts") or 4942Z ("Moving services") is not the same in terms of degree of error and therefore external communication.

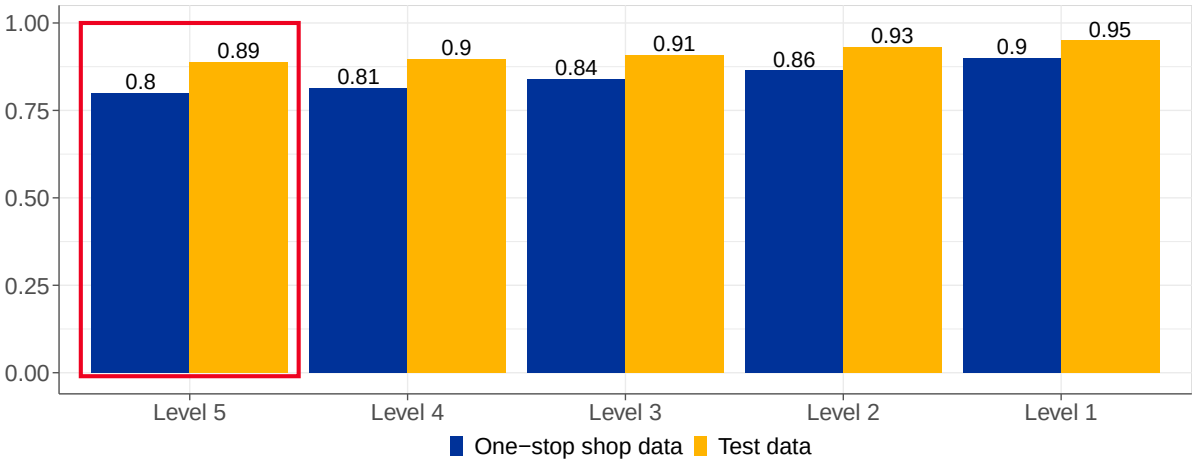

Figure 1: Accuracy for various level of the NACE nomenclature.

The accuracy rate on the Sirene 3 test data is 89% for level 5 and reaches 95% for level 1. The model is thus particularly effective on data similar to those on which it was trained and when the model makes a mistake at level 5, it is often to predict a code close to the ground truth in the nomenclature. As for the performance on data from the *one-stop shop*, it is significantly degraded: 80% of the text descriptions are well classified at level 5. This large difference was predictable and should not undervalue the very good rate of 80%, especially compared to what is currently achieved in Sirene 3. Similarly to the test data, the model does not seem to produce absurd predictions often since the accuracy rate still reaches 90% at level 1.

The *fastText* model has a significant advantage beyond its high overall performance compared to Sicore. Like all classification models using statistical learning methods, the *fastText* model provides a probability for each class. This makes it possible to retrieve the k most probable codes for each prediction made by the model. Analyzing the model's top predictions is another way to ensure that the results are not nonsensical. Using these predictions, we can define the top-k accuracy, which is a generalization of the previously specified accuracy rate, but uses a scoring function equal to 1 when the true value is among the model's top k predictions and 0 otherwise.

Figure 2 shows the top-k accuracy for $k \in \{1, 2, 3, 4, 5\}$ on both test and *one-stop shop* data. Regardless of the dataset used, the true classification is very often among the model's top five predictions. Furthermore, the top-2 accuracy has the highest marginal gain, with a performance gain of 8pp compared to the top-1 accuracy. This information is useful for the manual coding process. The model can be used to provide a number of highly probable classes to the annotators, accelerating the process.

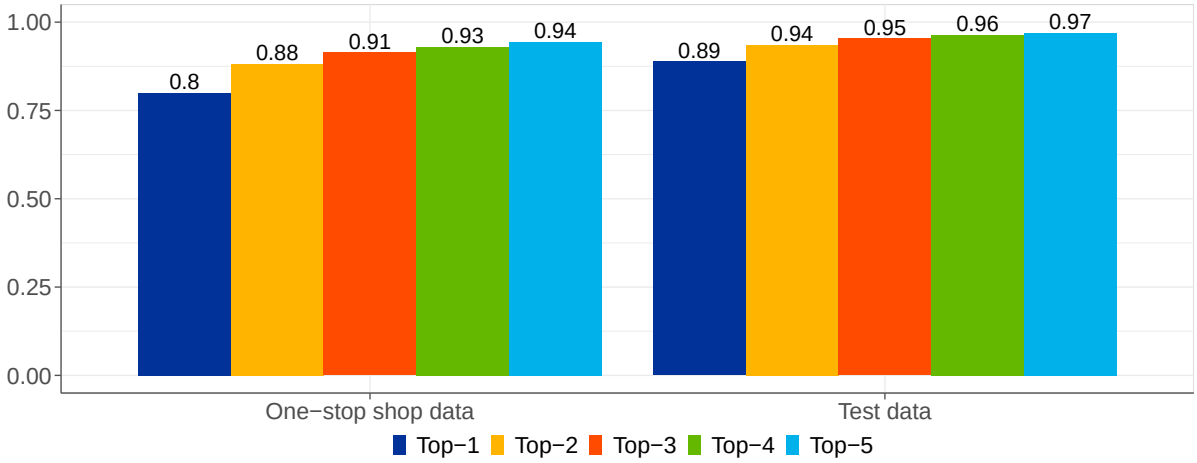

Figure 2: Top-k accuracy per sample.

4.2 Manuel review

Reducing manual intervention is a major objective of our problem. However, this reduction should not come at the expense of the classification quality. Ideally, we should be able to send to manual intervention only the labels on which the model has made mistakes. The challenge is to determine the labels for which the probability that the model made a mistake is high, using a well-defined confidence indicator. The optimal confidence indicator would theoretically be able to perfectly separate good predictions from bad ones. Various approaches can be employed to construct a confidence indicator; however, we have opted to utilize the gap between the two most probable prediction probabilities. This particular indicator has exhibited the most discriminative power amongst all indicators tested, making it the ideal choice. A higher gap

signifies greater confidence in a given prediction. In order to ensure the reliability of the predictions, a review process must be implemented, wherein a threshold is established to determine which descriptions can be automatically coded and which require further review.

The confidence score threshold is defined either (i) by setting a minimum overall precision rate, or (ii) by setting a maximum manual intervention rate. Obviously, it is not possible to simultaneously increase the prediction rate and reduce the manual intervention rate, and a trade-off must be made. The results are presented in Figure 3. An average manual intervention rate of around 25% yields a precision rate of 94%, a gain of 14 percentage points. Even a lower manual intervention rate of around 15% would still significantly increase the precision rate to 90%.

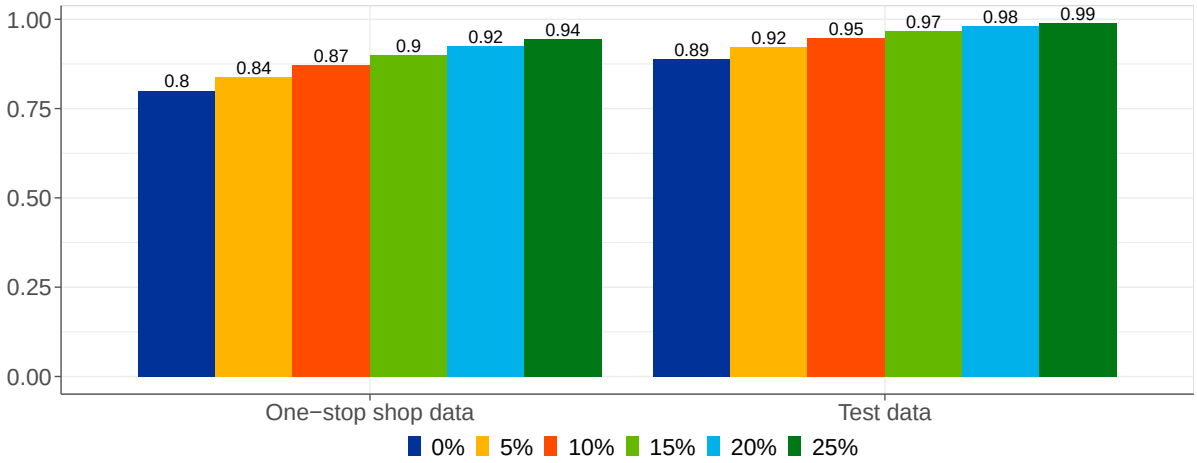

Figure 3: Accuracy as a function of the rate of manual coding performed.

5 Conclusion

The results of this experiment showed that machine learning classifiers are suitable candidates to replace Sicore, significantly reducing the need for manual classification and offering easier maintenance through regular re-training. As a result of these promising results, we have deployed a machine learning model in production for SIRENE, which is a testament to its practical value.

However, it is important to note that having a machine learning model in production also brings new challenges, including the need for constant monitoring, retraining, and management of the model's lifecycle, a list of practices commonly refered to as MLOps. It is critical to implement these practices to ensure the long-term success of machine learning projects and maximize their impact in the production of official statistics.

References

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. "Enriching Word Vectors with Subword Information." arXiv. https://doi.org/10.48550/ARXIV.1607.04606.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. "Bag of Tricks for Efficient Text Classification." arXiv. https://doi.org/10.48550/ARXIV.1607.01759.

Leroy, Théo, Lucas Malherbe, and Tom Seimandi. 2022. "Application de techniques de machine learning pour coder les professions en PCS 2020."