Journées de méthodologie statistique de l'Insee 2025

fastText, tout juste en production et déjà obsolète? Présentation du package torchTextClassifiers pour la modernisation de la codification automatique via l'exemple de l'APE

Meilame Tayebjee (*), Cédric Couralet (**), Nathan Randriamanana (***), Julien Pramil (****)

(*) Insee, DMCSI, Unité SSP Lab
(**) Insee, DMCSI, Unité SSP Lab
(***) Insee, DSE
(****) Insee, DMCSI, Unité SSP Lab

meilame.tayebjee@insee.fr *cedric.couralet@insee.fr* *nathan.randriamanana@insee.fr*
julien.pramil@insee.fr

Mots-clés : Classification de texte, codification automatique, fastText, torchTextClassifiers, deep learning, PyTorch

Domaines : Données administratives et qualité, Données et traitements statistiques

Résumé

La codification automatique de l'Activité Principale Exercée (APE) à partir de libellés textuels représente un enjeu opérationnel majeur pour l'Insee. Depuis 2020, un modèle fastText est utilisé en production pour cette tâche. À la fois méthodologie et librairie logicielle développée par Facebook AI Research, fastText repose sur des classificateurs linéaires efficaces, combinés à des représentations vectorielles de mots et de sous-mots. Il offre une solution simple, rapide et précise pour la classification de texte à large échelle, particulièrement adaptée aux nomenclatures métiers.

Mais quatre ans après sa mise en oeuvre, ce modèle soulève une double préoccupation. D'une part, la pérennité technique est fragilisée par l'usage d'un binaire compilé, difficile à maintenir dans le temps, susceptible d'engendrer des conflits de version, et éloigné des standards actuels de développement collaboratif. D'autre part, le besoin de modernisation se fait sentir - les évolutions rapides des outils de deep learning ouvrent de nouvelles perspectives que l'approche fastText ne permet pas d'exploiter.

Pour répondre à ces défis, l'Insee a initié le développement de torchFastText, un nouveau package Python basé sur PyTorch, conçu pour reproduire l'approche de fastText dans un cadre plus moderne, transparent et modulaire. Ce package évolue désormais vers torchTextClassifiers, qui distribuera toutes les architectures classiques de classification de texte, à destination de tous usagers souhaitant entraîner rapidement et efficacement un modèle d'apprentissage profond pour la classification de texte.

Ce développement, réalisé en open source, répond à un besoin partagé entre plusieurs instituts statistiques européens, dans une logique - espérée - d'interopérabilité et de mutualisation. L'objectif - garantir la frugalité des modèles, tout en préservant les performances du modèle initial en termes de rapidité et de précision. torchFastText

a été validé à l'issue d'une série de tests rigoureux sur les jeux de données de production. Il est aujourd'hui en production.

Outre l'aspect technique, ce projet incarne une démarche d'appropriation collective des outils d'IA, en s'appuyant sur des langages et frameworks largement adoptés dans la communauté scientifique, favorisant leur diffusion et leur maintenance dans le temps.

En plus de la simple réimplémentation, le passage à PyTorch ouvre des perspectives de développement ambitieuses :

1. Amélioration du modèle - grâce à la flexibilité du framework, il devient possible d'explorer de nouvelles architectures, d'optimiser les fonctions de perte ou encore d'incorporer la hiérarchie propre aux nomenclatures dans l'apprentissage. C'est l'objectif de la nouvelle mouture, torchTextClassifiers.
2. Explicabilité - les outils d'interprétabilité compatibles avec PyTorch (comme Captum) permettent d'apporter des éclairages plus fins sur les prédictions du modèle, favorisant leur auditabilité.
3. Intégration à l'ecosystème deep learning - l'intégration de tokenizers entraînés sur des corpus spécifiques, ou encore l'utilisation de modèles préentraînés issus de bibliothèques comme Hugging Face (de type BERT ou CamemBERT3), autorise une personnalisation plus poussée, éloignant progressivement l'approche du modèle fastText originel.
4. Meilleure expérience pour les data scientists - l'environnement PyTorch facilite le suivi de l'apprentissage (logs, visualisation des courbes de perte, early stopping. . .), accélérant les cycles de développement et de validation.

Au-delà de ce cas d'usage spécifique, l'objectif est aussi de diffuser la culture PyTorch à l'Insee, pour en faire un socle commun pour les futurs projets de deep learning en statistique publique. Des formations internes sont en cours pour accompagner cette montée en compétence collective.

Abstract

The automatic coding of the Activité Principale Exercée (APE) from textual labels is a major operational challenge for Insee. Since 2020, a fastText-based model has been used in production to perform this task efficiently. However, after four years, both technical sustainability and methodological modernization have become pressing issues. To address them, Insee has developed torchFastText, a new open-source Python package built on PyTorch that reproduces the fastText approach within a modern, modular, and maintainable framework. Now evolving into torchTextClassifiers, the package aims to provide a suite of lightweight, high-performance text classification models for broad institutional and public use. Beyond its technical benefits, this initiative reflects a collective effort to strengthen sovereignty over AI tools while fostering interoperability and shared development across European statistical institutes. It also paves the way for future advances in model interpretability, hierarchical learning, and deep learning adoption within public statistics.

1. Introduction et contexte

1.1. Historique de la codification automatique de l'APE : de Sicore à fastText

Tâche commune au sein des instituts de statistique publique, la codification automatique de libellés textuels dans une nomenclature donnée peut s'avérer complexe à grande échelle, complexité d'autant plus relevée que la nomenclature est de grande taille.

Dans le cas de l'Activité Principale Exercée (APE), l'enjeu est particulièrement sensible : chaque entreprise du répertoire Sirene doit être classée dans l'une des sous-classes de la nomenclature d'activités française (NAF rév. 2, 732 postes), à partir d'un libellé textuel décrivant son activité. La qualité

et la rapidité de cette codification conditionnent directement l'efficacité du système d'information statistique.

Historiquement, le système Sicore a été développé dans les années 90 (E. Meyer et P. Rivière [1]) pour les grands usages de codification à travers l'Institut. Basé sur un principe d'« apprentissage », l'algorithme permettait d'identifier de façon déterministe un code unique, ou bien de choisir de ne rien proposer, menant à une reprise manuelle par un gestionnaire. C'était justement Sicore qui était utilisé dans le cadre de la codification automatique de l'Activité Principale de l'Entreprise dans le répertoire Sirene des entreprises.

L'arrivée du guichet unique dans le cadre de la loi PACTE (n° 2019-486 du 22 mai 2019) a offert aux chefs d'entreprises plus de flexibilité dans la description de leurs activités principales, la rendant mécaniquement plus verbeuse que précédemment. L'efficacité de Sicore s'en est trouvée directement impactée : le taux de codage automatique a chuté à seulement **30%**, entraînant une augmentation importante du taux de reprises manuelles et surchargeant considérablement les équipes de gestion du répertoire Sirene.

L'arrivée des modèles de traitement automatique du langage (NLP) a ouvert la voie à une nouvelle génération de systèmes de codification automatique. Parmi eux, fastText, développé initialement par Facebook AI Research, a été retenu pour son excellent compromis entre rapidité et performance. Son architecture simple, fondée sur des représentations vectorielles de mots et de n-grammes, permet une adaptation rapide à la structure particulière des libellés d'activités.

Depuis le 1er janvier 2023, un modèle *fastText* a donc remplacé aves succès l'algorithme Sicore en production, permettant une automatisation à 80% de la codification, avec un temps de réponse unitaire de moins de 100ms et une qualité de prédiction satisfaisante (évaluée sur des données de test avant le déploiement et également en continu par des gestionnaires).

1.2. L'intégration d'un modèle *fastText* dans le système d'information de l'Insee

fastText se base sur une représentation vectorielle de *n-grams* (association de sous-éléments) de mots et de caractères. Cette représentation est apprise, comme dans tout modèle d'apprentissage profond, à partir d'un jeu de données d'entraînement. L'architecture du modèle est relativement simple, l'efficacité venant pour une grande partie de la transformation des mots en tokens, permettant de se concentrer sur la sémantique des mots (leur racine). De plus, la simplicité du modèle le rend très performant en terme de temps de réponses lors de son utilisation. Nous invitons le lecteur à lire le papier original (A. Joulin, E. Grave, P. Bojanowski, et T. Mikolov [2]) pour comprendre les détails de cette méthodologie et l'architecture du modèle.

Le déploiement d'un modèle de *machine learning* en production a nécessité de nombreuses adaptations, dans une philosophie ***MLOps***. L'idée fondamentale est d'intégrer tout le cycle de vie d'un projet dans un continuum automatisé, de la gestion des données à la surveillance du modèle déployé en passant par l'entraînement et le versionnage des modèles.

Cette approche MLOps s'inscrit fondamentalement dans une perspective d'utilisation de technologies *cloud-native*, en particulier Onyxia. La description des travaux et la philosophie générale du déploiement d'un modèle *fastText* est décrite dans le document de travail *L'apport des technologies cloud pour industrialiser le processus d'innovation statistique* (R. Avouac, T. Faria, et F. Comte [3]).

L'architecture mise en place repose sur plusieurs piliers :

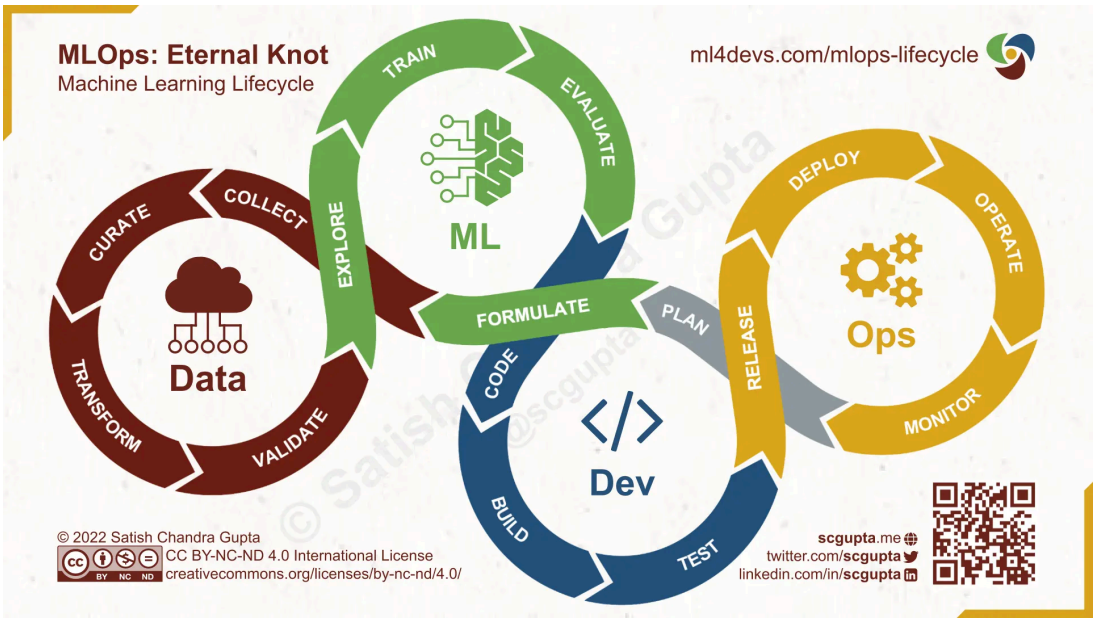

Figure 1. – Le pipeline MLOps idéal

- **Séparation stricte des environnements** : les environnements de calcul et de stockage des données sont découplés, avec utilisation de l'espace de stockage S3 du SSP Cloud pour la gestion des données
- **Automatisation des pipelines** : ArgoWorkflows orchestre l'entraînement avec un minimum d'intervention humaine
- **Gestion du cycle de vie des modèles** : MLFlow assure le suivi de l'entraînement, le stockage et le versionnage des modèles, ainsi que leur encapsulation pour le service (*wrapper* contenant les logiques de prétraitement du texte)
- **Service via API REST** : exposition du modèle pour intégration dans les systèmes de production
- **Collaboration métier-innovation** : travail conjoint entre les équipes Sirene et le SSP Lab
- **Surveillance continue** : dashboards de monitoring basés sur Quarto pour suivre les performances en production

2. PyTorch : enjeux et défis de la mise en production

La librairie *fastText* développée par Meta a été archivée en juin 2024, posant différentes questions quant à la maintenabilité et la viabilité d'un modèle *fastText* en production:

Dépendance à un binaire compilé : fastText s'appuie sur du code C++ compilé, ce qui entraîne :

- Difficultés de maintenance en cas de bug ou de besoins d'évolutions
- Conflits de version potentiels avec d'autres dépendances
- Éloignement du cadre de cohérence technique du système d'information de l'Insee

Absence de perspectives d'évolution : sans support actif, la librairie ne bénéficiera plus de correctifs de sécurité ni d'améliorations méthodologiques, alors que le domaine du NLP évolue rapidement (utilisation de GPU, absence d'explicabilité native, aucune prise en compte de variables additionnelles…)

Le passage à l'écosystème **PyTorch**, devenu standard *de facto* pour le développement de réseaux de neurones, a donc été naturellement envisagé pour les possibilités que cela permettait:

- maintenance et développement en interne du modèle de codification automatique (par exemple, avec une prise en compte mieux gérée des variables additionnelles et la possibilité d'améliorer les performances avec des architectures plus sophistiquées)

- connexion à un écosystème dynamique et donc à de nombreuses librairies permettant des ajouts utiles : utilisation de modèles et/ou de tokenizers pré-entraînés depuis *Hugging Face*, explicabilité avec *Captum*, calibration des modèles avec *torch-uncertainty* etc.

La question principale qui restait en suspens était celle de la performance et du temps d'inférence en utilisant uniquement des CPU. PyTorch est une librairie généraliste, destinée à pouvoir implémenter n'importe quelle architecture de réseaux de neurones, contrairement à la librairie fastText (qui par définition n'implémente que la méthodologie fastText). PyTorch est par ailleurs particulièrement destinée à une utilisation sur GPU : si les GPUs du SSP Cloud permettent un entraînement rapide, la question de l'inférence en production à l'Insee (dont les serveurs ne disposent pas encore de GPU) était cruciale.

Après des tests conclusifs, un modèle PyTorch d'environ 1 million de paramètres a été mis en production, avec des temps d'inférence satisfaisants de moins de 300ms - bien que supérieurs à ceux de fastText. Mais cette légère dégradation s'accompagne d'une meilleure précision, d'une meilleure calibration, d'un meilleur contrôle sur l'entraînement et des fonctionnalités supplémentaires (voir ci-après).

Tableau 1 : Comparaison fastText vs torchTextClassifiers

Critère	fastText (2021)	torchTextClassifiers (2024)	Évolution
Taux d'automatisation	80%	80%	=
Temps d'inférence moyen	<100ms	<300ms	+200ms
Calibration	Limitée	Excellente	↗↗
Explicabilité	Non	Oui (Captum)	✓
Support GPU	Non	Oui	✓
Maintenance	Archivé	Active	✓
Extensibilité	Limitée	Élevée	✓

Les figures ci-dessous illustrent la qualité de la calibration obtenue avec le modèle PyTorch.

3. torchTextClassifiers: architecture et perspectives

Le SSP Lab a centralisé sous la forme d'un package Python ce modèle PyTorch (disponible sur ce lien), et ce pour plusieurs raisons :

- **Séparation des responsabilités** : le modèle a une existence propre, indépendante de son usage spécifique pour la codification APE
- **Portabilité et versionnage** : l'architecture package permet une distribution via PyPI, facilitant l'installation et les mises à jour
- **Diffusion méthodologique** : la méthodologie dépasse le cadre de l'APE et peut servir de standard à l'Institut pour tout projet de classification de texte

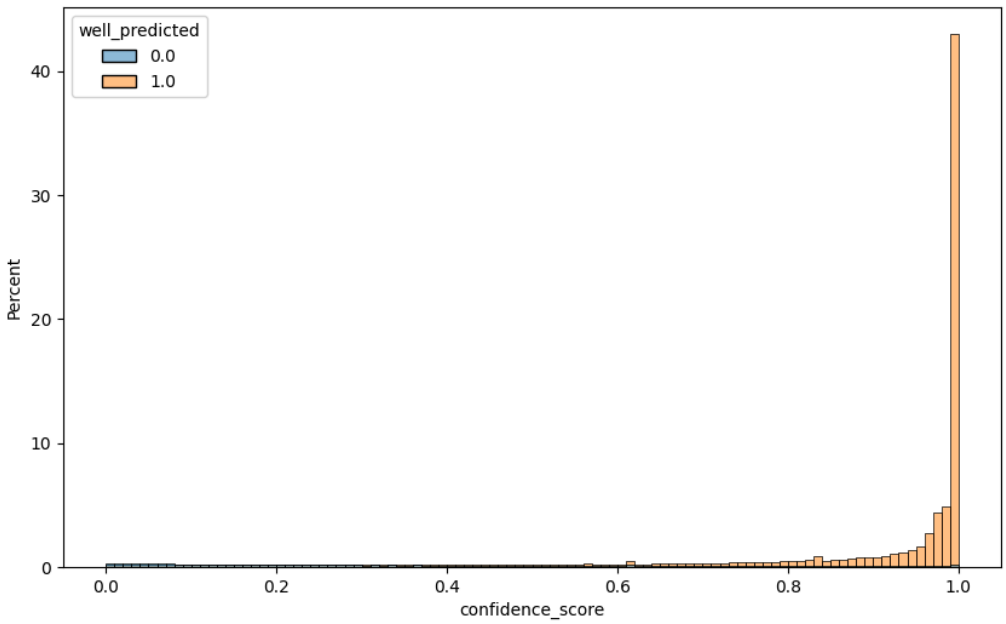

Figure 2. – Sur un échantillon de test de 300 000 libellés, histogramme des niveaux de confiance prédits par le modèle, sur les libellés correctement ou mal classés. Le modèle arrive à bien discriminer, ne se trompant que lorsqu'il a une faible confiance.

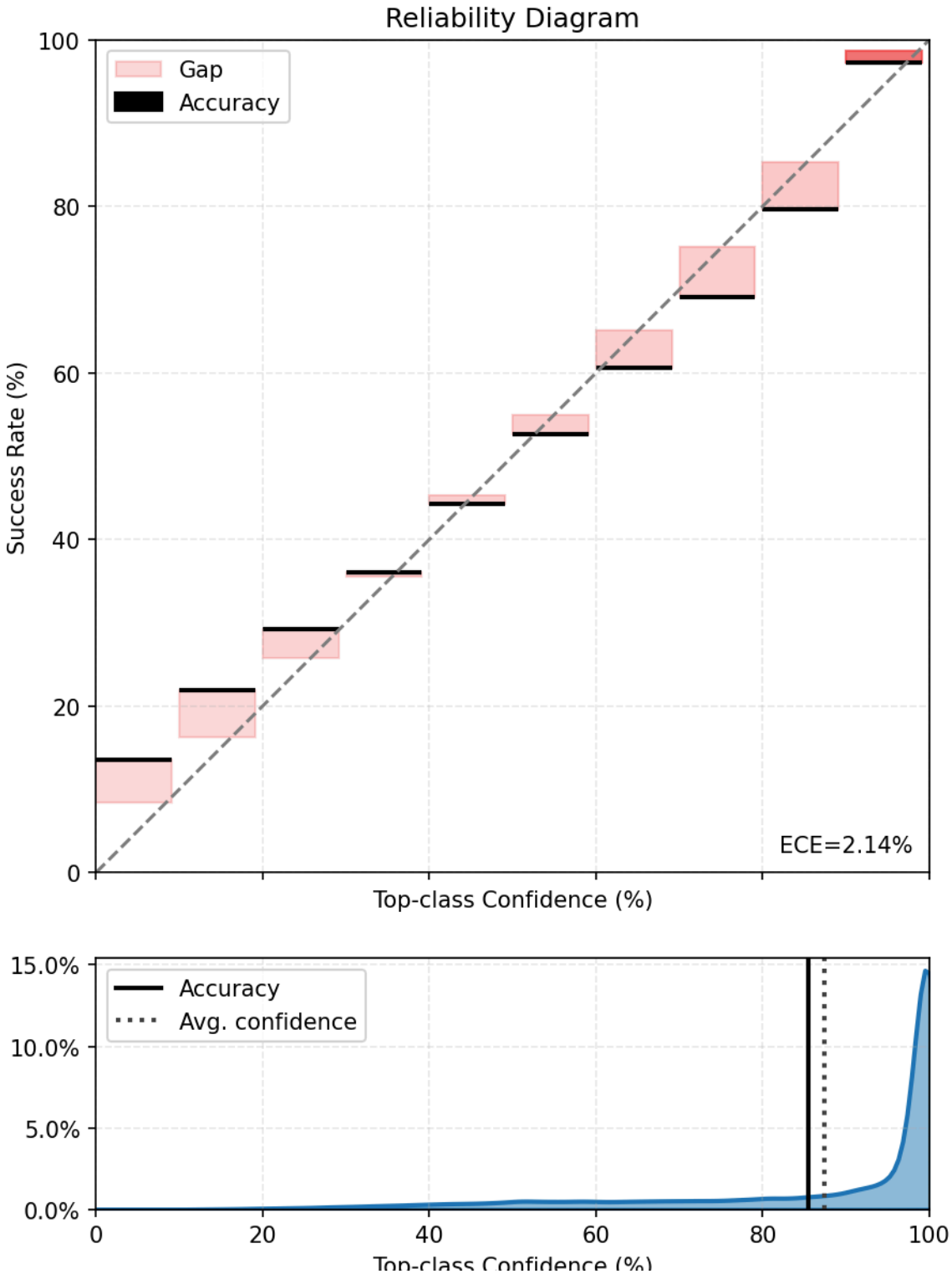

Figure 3. – Sur un échantillon de test de 300 000 libellés, courbe de calibration du modèle. Le modèle est très bien calibré : quand il indique un niveau de confiance de xx%, il est effectivement correct dans xx% des cas.

- **Collaboration institutionnelle** : facilite le partage avec d'autres instituts statistiques européens partageant des besoins similaires

A terme, le but est de disposer (et mettre à disposition) une librairie permettant d'initialiser et entraîner très facilement un modèle de classification de texte utilisant les architectures les plus modernes si besoin. Les utilisateurs visés sont ceux qui ne peuvent pas utiliser de modèles lourds pré-entraînés depuis Hugging Face mais veulent entraîner efficacement leurs propres modèles de taille réduite - pour une inférence CPU par exemple. Les organismes de la statistique publique font donc naturellement partie de cette cible.

Le package repose sur deux piliers :

- **PyTorch natif** pour la définition des modèles et les opérations tensorielles
- **PyTorch Lightning** comme framework d'entraînement, standard actuel pour les réseaux de neurones, qui simplifie la gestion du cycle d'entraînement (boucles, validation, checkpointing, logging)

L'utilisation de dépendances optionnelles (**Captum**, par exemple) permet d'enrichir les possibilités de base. Le package mis à disposition dispose dès aujourd'hui:

- d'une réimplémentation fidèle de l'architecture fastText en PyTorch
- des fonctionnalités d'entraînement permises avec PyTorch Lightning (suivi des métriques, early stopping, checkpointing)
- de la calibration des probabilités de sortie
- de l'explicabilité au niveau des mots et des caractères via Captum

Cette fonctionnalité offre plusieurs bénéfices opérationnels :

- **Auditabilité** : possibilité de justifier les prédictions auprès des gestionnaires et des entreprises
- **Détection d'erreurs** : identification rapide des cas où le modèle s'appuie sur des indices trompeurs
- **Amélioration itérative** : compréhension fine des faiblesses du modèle pour guider les évolutions

A plus long terme, torchTextClassifiers vise à devenir une **boîte à outils complète** pour la classification de texte légère :

1. **Amélioration continue** : grâce à la flexibilité de PyTorch, exploration de nouvelles architectures, optimisation des fonctions de perte, intégration de contraintes métier
2. **Explicabilité renforcée** : développement de méthodes d'interprétation adaptées aux nomenclatures (contribution de chaque niveau hiérarchique, identification des descripteurs discriminants)
3. **Intégration à l'écosystème** : compatibilité avec les modèles pré-entraînés (BERT, CamemBERT) pour des usages hybrides (fine-tuning léger sur des embeddings contextuels)
4. **Expérience développeur** : simplification maximale de l'API pour permettre un prototypage rapide et une mise en production fluide

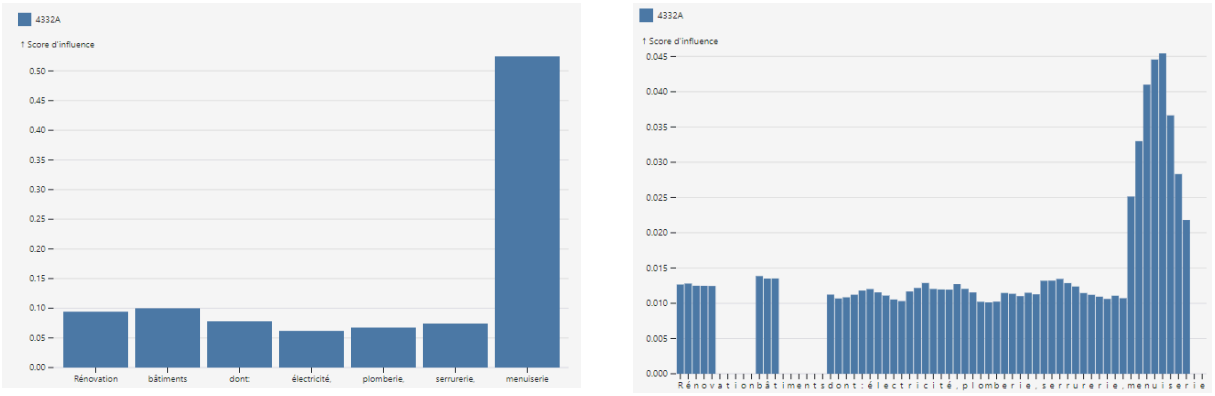

Figure 4. – L'explicabilité est une des fonctionnalités implémentées : quel mot ou caractère a influencé la prédiction ?

Malgré ses avantages, la migration vers PyTorch et l'utilisation dans le système d'information de l'Insee présente quelques défis :

- **Performance** : bien que les temps d'inférence restent acceptables (<300ms) sur le modèle actuellement en production (pour la codification de l'APE), ils représentent une dégradation par rapport à fastText. Des optimisations restent possibles mais n'atteindront probablement jamais les performances du binaire fastText.
- **Complexité organisationnelle** : La mise en place du MLOps suppose la définition de nouvelle organisation entre le métier et les informaticiens, et donc de nouvelles compétences à acquérir des deux côtés
- **Compatibilité** : l'écosystème PyTorch évolue rapidement, ce qui peut entraîner des ruptures de compatibilité entre versions. Une politique de versionnage stricte est indispensable.

4. Conclusion

La migration de fastText vers torchTextClassifiers illustre les tensions inhérentes à la mise en production de modèles d'apprentissage automatique : entre performance et maintenabilité, entre spécialisation et généralité, entre rapidité de déploiement et pérennité.

Le choix de PyTorch, bien qu'impliquant une légère dégradation des temps d'inférence, apporte des garanties de maintenabilité, d'extensibilité, et d'alignement avec l'écosystème open source qui compensent largement cet inconvénient. La calibration améliorée et les fonctionnalités d'explicabilité constituent par ailleurs des atouts majeurs pour l'acceptabilité opérationnelle du système.

Au-delà du cas d'usage de la codification APE, ce projet pose les bases d'une **infrastructure pérenne** pour le développement et le déploiement de modèles d'IA en statistique publique. En s'appuyant sur des standards reconnus (PyTorch, MLflow, API REST), en privilégiant la transparence (open source), et en investissant dans la formation, l'Insee renforce sa capacité d'innovation tout en maîtrisant ses dépendances technologiques.

La prochaine étape consistera à enrichir torchTextClassifiers avec de nouvelles architectures et fonctionnalités, à évaluer son utilisation dans d'autres contextes (autres nomenclatures, autres instituts), et à poursuivre les efforts de recherche pour exploiter pleinement le potentiel de l'apprentissage profond dans le domaine de la classification de texte.

Bibliographie

[1] E. Meyer et P. Rivière, « SICORE, un outil et une méthode pour le chiffrement automatique à l'INSEE ». 1997.

[2] A. Joulin, E. Grave, P. Bojanowski, et T. Mikolov, « Bag of Tricks for Efficient Text Classification », *arXiv preprint arXiv:1607.01759*, 2016.

[3] R. Avouac, T. Faria, et F. Comte, « L'apport des technologies cloud pour industrialiser le processus d'innovation statistique », 2025, [En ligne]. Disponible sur: https://www.insee.fr/fr/statistiques/8614378