



Projet Rakuten

Classification de produits selon 2 modalités (texte + image)

oct24_cds_Rakuten

-
- Fahim HERRICHE
 - Frédéric DE MOLLIS
 - Bakari TRAORE
 - Didier BALLOUARD



I. Vue d'ensemble.....	4
A. Contexte du projet.....	4
B. Objectif.....	4
C. L'équipe projet.....	4
II. Méthodologie.....	5
A. Organisation.....	5
1. Approche.....	5
2. Rythme.....	5
3. Outils.....	6
B. Expertises externes.....	6
C. Interaction avec un projet en cours.....	6
III. Exploration & Data visualisation.....	6
A. Description des données.....	6
1. Introduction.....	6
2. Modalité Texte.....	6
3. Modalité Image.....	14
4. Volumétrie.....	14
IV. Preprocessing.....	15
A. Modalité Texte.....	15
1. Préambule : problématique de langues multiples.....	15
2. Traitements.....	17
a) Initialisation.....	17
b) pré-traitements.....	18
(1) Création de variables.....	18
(2) Traitement des doublons.....	18
(3) Traitements du texte.....	19
B. Modalité Image.....	21
1. Préambule.....	21
2. Traitements.....	21
V. Modélisation.....	22
A. Introduction.....	22
1. Nature du projet.....	22
2. Métrique.....	22
3. Méthodologie.....	22
B. Modalité Texte.....	23
1. Préambule.....	23
2. Approches développées.....	23
3. Récapitulatif des meilleurs scores.....	31
4. Approche retenue.....	32



C. Modalité Image.....	37
1. Travail exploratoire.....	37
2. Récapitulatif des meilleurs scores.....	38
3. Architecture retenue.....	39
D. Architecture multi-modale.....	41
1. Préambule.....	41
2. Principes.....	42
3. Résultats et perspectives.....	44
E. Interprétabilité.....	45
1. Modalité Texte.....	45
2. Modalité image.....	47
VI. Conclusion.....	49
VII. Bibliographie.....	53

Remerciements

Nous souhaitons ici remercier les équipes pédagogiques DataScientest pour les moyens mis en œuvre, la qualité des cours, l'animation des Master Classes, le suivi et le support à la formation.

Nous souhaitons ensuite saluer en particulier Aïda, notre Program Manager et Kalomé, notre Chef de Cohorte, pour l'accompagnement prodigué tout au long de notre parcours.

Enfin, nos remerciements vont à notre mentor, Elliott, pour son écoute, sa rigueur dans notre suivi et ses précieux conseils.

I. Vue d'ensemble

A. Contexte du projet

Le projet fait référence à un challenge proposé par le e-commerçant Rakuten qui, pour sa marketplace, est confronté à une problématique de catégorisation des produits régulièrement vendus sur sa plateforme.

La société Rakuten souhaite mettre à profit les technologies de Machine Learning afin d'automatiser cette catégorisation à grande échelle et ainsi faciliter la recherche sur son site ou encore la recommandation de produits.

Rakuten met à disposition les détails et résultats de 2 modèles de référence (ResNet50 / Keras pour les images et un classifieur CNN pour le texte) qui permettra de situer les performances du modèle mis en œuvre dans le cadre de ce projet. Le score F1 pondéré obtenu séparément par les 2 modèles sont de 0.8113 (texte) et 0.5534 (image).

B. Objectif

La problématique de catégorisation, ou classification de données, automatique consiste à prédire le code type de chaque produit, tel que défini dans le catalogue de Rakuten France, sur la base de données d'entrée descriptives selon les 2 modalités : texte + image.

L'objectif du projet vise à modéliser un classifieur pour classer les produits dans son code type de produit correspondant.

La mesure utilisée pour représenter la performance du modèle est le score pondéré F1 (cf implémentation Python scikit-learn avec paramètre average défini sur "weighted")

C. L'équipe projet

Fahim Herriche, cadre expert de la gouvernance des données au sein d'un ministère. Parmi ces missions, développe et conçoit notamment des solutions numériques innovantes au bénéfice de son administration.



Bakkari Traore est actuellement enseignant en mathématiques au sein de l'éducation nationale. Après un Master en probabilités appliquées et quelques années dans la recherche sur les modèles épidémiques SIR, à l'aise dans la programmation avec R, Matlab, Python et passionné par le code et la data, il entame une reconversion vers le métier de Data Scientist.

Frédéric de Miollis, docteur en technologie médicale, possède une grande expérience dans le domaine de la data (collecte, traitement, nettoyage, analyse et visualisation avec R & Python). Il souhaite se former en machine learning pour approfondir ses compétences et évoluer dans sa carrière. Il espère développer des modèles et contribuer à des projets innovants dans le domaine médical.

Didier Ballouard dispose de compétences en business intelligence exercées dans le domaine de la grande distribution et de la logistique. A l'aise avec le data processing, la modélisation (sous SQL) et la data visualisation, il n'a, en revanche, pas d'expérience en Machine Learning.

II. Méthodologie

A. Organisation

1. Approche

L'approche retenue par l'équipe est un travail personnel à chaque étape du projet et une mise en commun pour en synthétiser une construction collective. Chacun peut ainsi appliquer à discréption les enseignements reçus, tester ses intuitions et explorer de nouvelles pistes. La consolidation des travaux individuels permet à chacun de renforcer la compréhension globale du projet, des réalisations, de la trajectoire suivie et contribue à bâtir une solution optimale.

2. Rythme

La mise en place d'un point hebdomadaire participe à maintenir une dynamique soutenue pour l'équipe sur le projet. L'objectif de ces rendez-vous est la mise en commun des travaux réalisés par chacun, d'une semaine à l'autre, de partager ses expériences, les idées, d'assurer un niveau d'investissement constant au projet, de veiller au respect des plannings mais aussi d'affiner notre vision.

3. Outils

- Mise en place d'un environnement Google Drive partagé pour une construction collaborative du document support du projet et pour le partage des comptes-rendus de réunions.
- Mise en œuvre d'un dépôt Github interne pour partager les réalisations de chacun au travers de notebooks individuels et pour synthétiser le code principal de la solution.
- Mise en œuvre d'un canal Slack interne dédié pour un partage permanent.

B. Expertises externes

A date, aucune demande d'expertise externe n'a été sollicitée.

C. Interaction avec un projet en cours

A date, pas d'interaction identifiée avec un projet en cours.

III. Exploration & Data visualisation

A. Description des données

1. Introduction

L'ensemble des données utilisées pour ce projet est issu d'un dépôt sur la plateforme Kaggle, accessible à cet emplacement. Ces données, propriété de la société Rakuten, ne doivent pas être exploitées en dehors de ce contexte d'apprentissage.

Ce projet consistant en l'exploitation de données textuelles et d'images, et non basé sur des données chiffrées, l'exploration et la data visualisation seront nécessairement limitées.

2. Modalité Texte

Les données de texte utilisées seront issues des sets d'entraînement uniquement (train), les seuls à fournir la variable cible.

Le set des données textuelles reprend les éléments suivants :

ID entier du produit / #

Cet ID est utilisé pour associer le produit à son code de type de produit correspondant et a permis le merge initial des sets de train pour constituer le set initial du projet.

C'est un entier, compris entre 0 et 84 916, qui représente l'index technique du dataset. En cette qualité, il ne présente pas de valeur informative pour l'atteinte de l'objectif.

designation

Il s'agit du titre du produit, un court texte résumant le produit. C'est cette unique variable textuelle du dataset qui a été utilisée dans le modèle de référence de Rakuten.

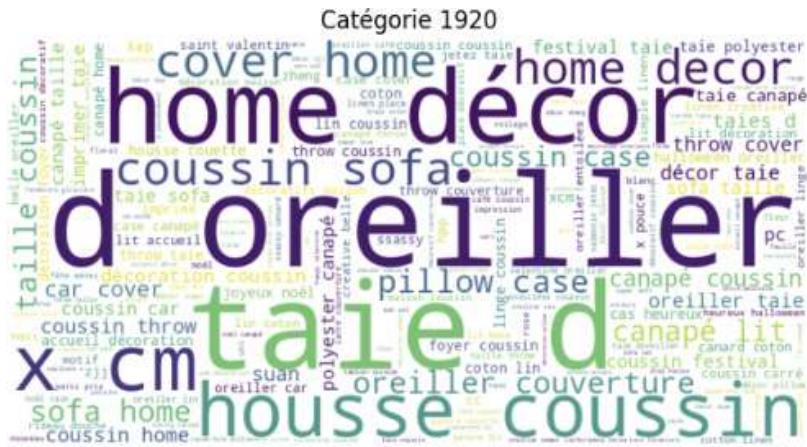
C'est une variable de type objet (string) de taille limitée à 250 caractères. Elle compte environ 3% de doublons et son contenu semble être majoritairement rédigée en Français.

Elle est la principale candidate pour l'approche textuelle de classification des produits (renseignée de manière exhaustive, caractère concis à forte valeur discriminante probable)

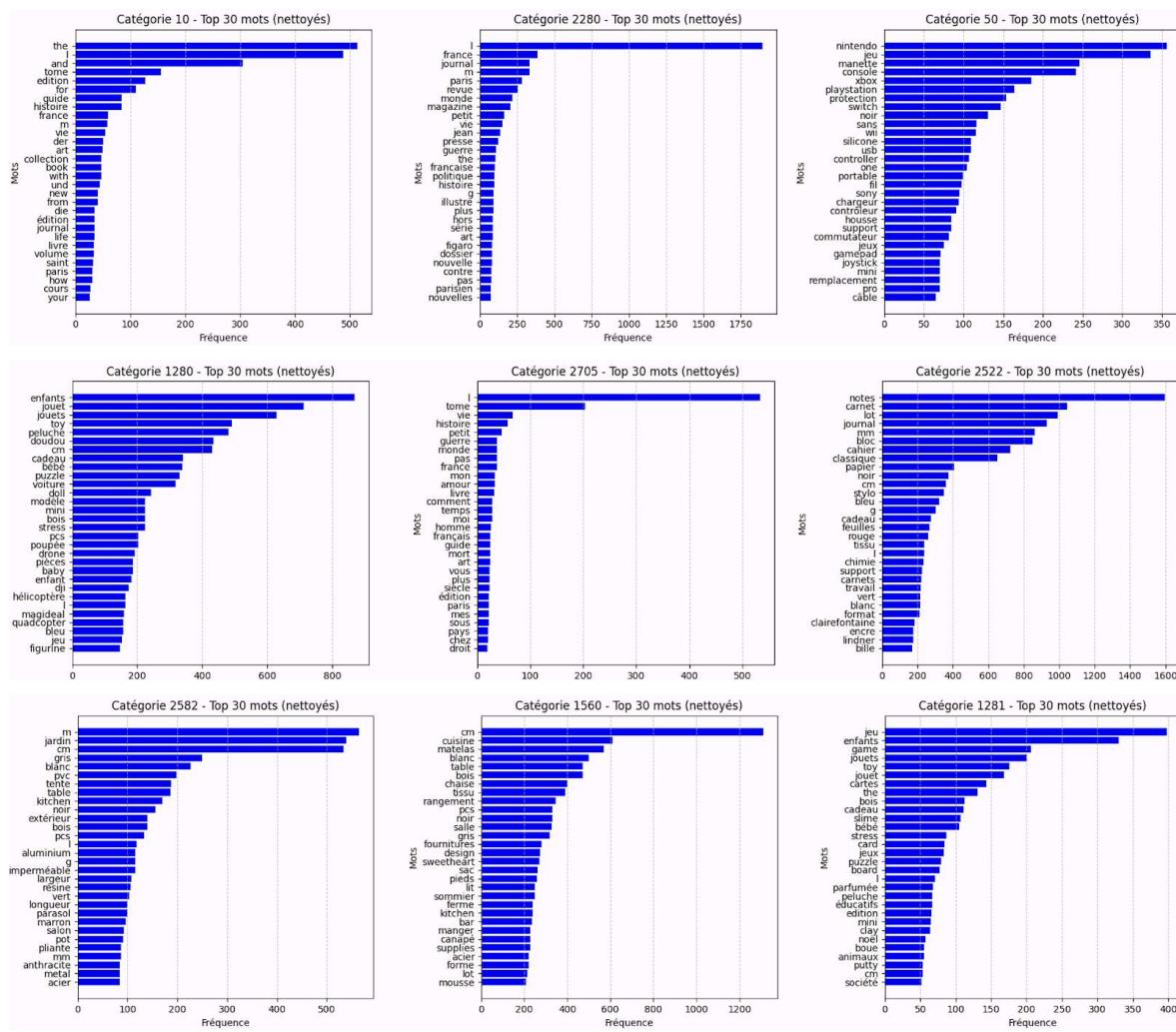
Ci-après, un rendu visuel de type nuage de mots pour quelques catégories de produits permettant de représenter le contenu textuel majoritaire de la variable.

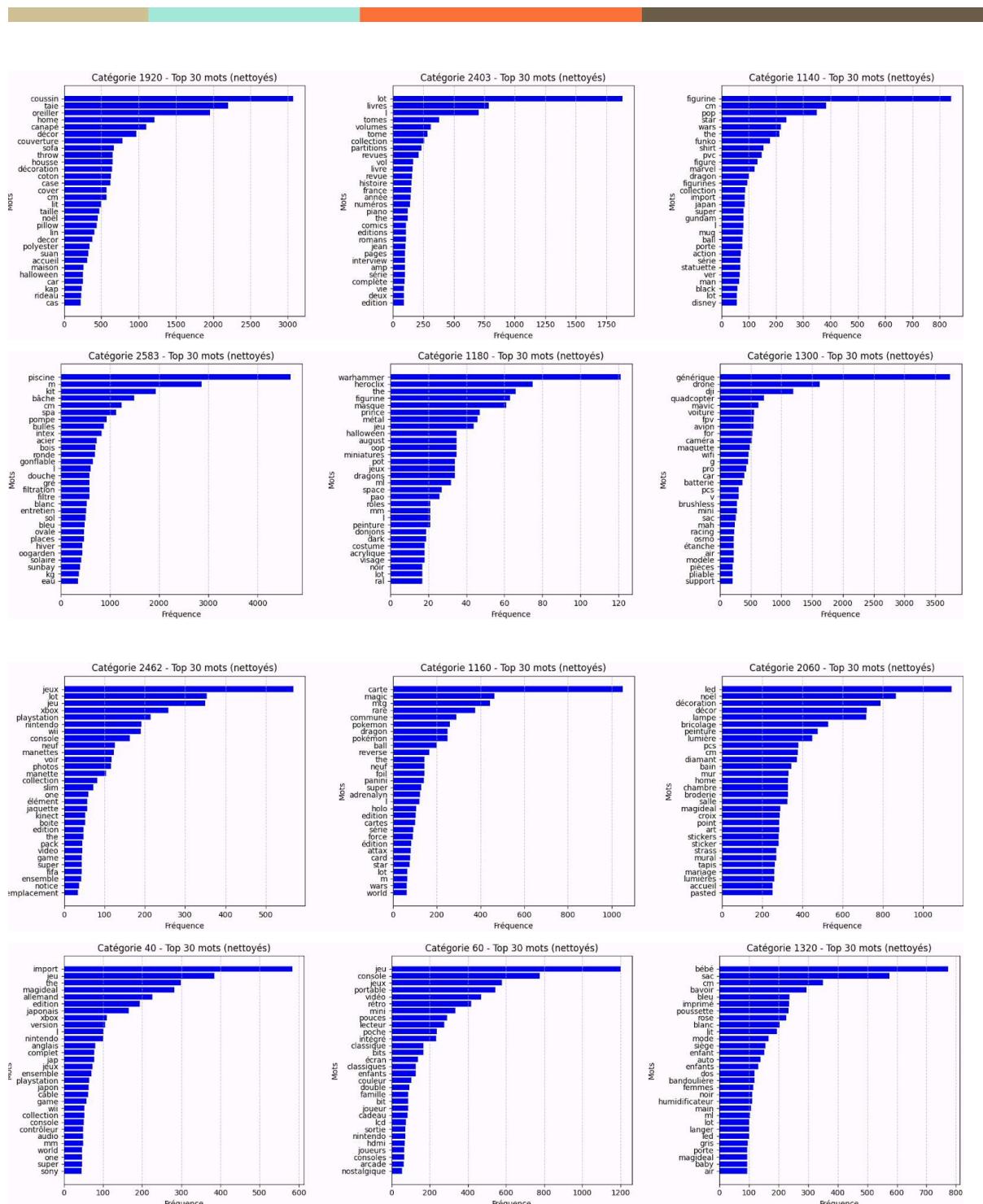
Ceci permet d'avoir rapidement une idée de la thématique adressée par la catégorie et du travail de preprocessing nécessaire afin de nettoyer la variable des mots vides de sens qui polluent l'exercice de synthèse.

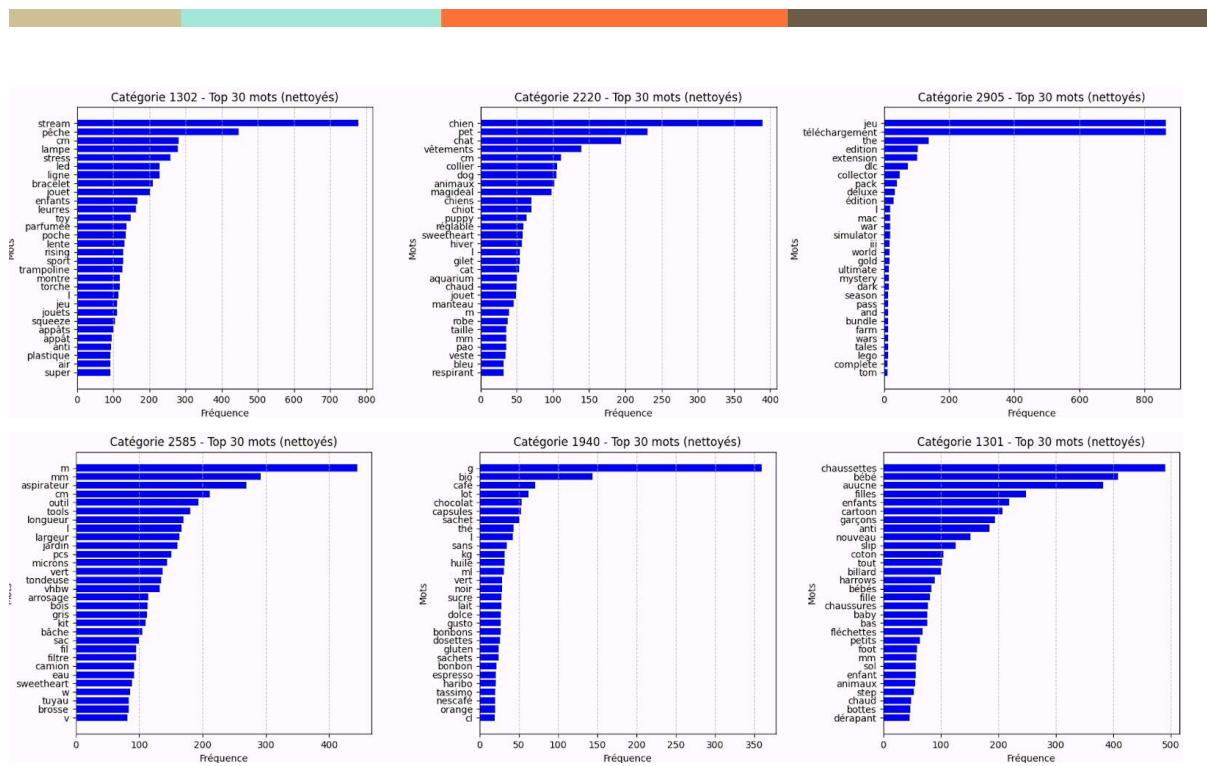




Un autre code a permis, après un pré-nettoyage du champ, d'établir un top 30 des mots les plus représentés par catégorie, renforçant ainsi l'exercice de synthèse.







A l'issue du preprocessing, nous devrions être en capacité d'identifier le contenu "métier" de chaque catégorie, participant ainsi à rendre plus intelligible la problématique introduite par le projet.

description

Il s'agit d'un texte plus détaillé décrivant le produit. Tous les commerçants n'utilisant pas ce champ, ce dernier présente un nombre important de valeurs manquantes (NaN). Environ 35% des entrées sont concernées.

Cette variable de type objet (string) d'une taille allant jusqu'à 12 451 caractères.

Son contenu semble être majoritairement rédigée en Français et l'on note la présence récurrente de balises HTML de mise en forme dont il faudra possiblement tenir compte dans son exploitation.

Par opposition à la variable designation, le contenu de cette variable plus dense peut être plus complexe (non exhaustivité, taille des contenus, bruit probable, opérations de nettoyages préalables probables (balises html))

Quand sont contenu n'est pas très proche, voire identique à la variable designation (1% des cas), il semble de qualité très variée. On rencontre, en outre, 9% de doublons sur cette variable.

Un exercice de représentation visuel analogue à la variable designation, mais cette fois, sur l'ensemble de la variable description, laissent entrevoir la difficulté probable à extraire des informations utiles à l'objectif :



Le preprocessing à appliquer à ce champs sera sans doute plus dense.

productid

Identifiant unique pour le produit, de type int64. Aucune valeur manquante trouvée.

Cette variable ne présente aucune valeur informative pour le modèle de classification textuel.

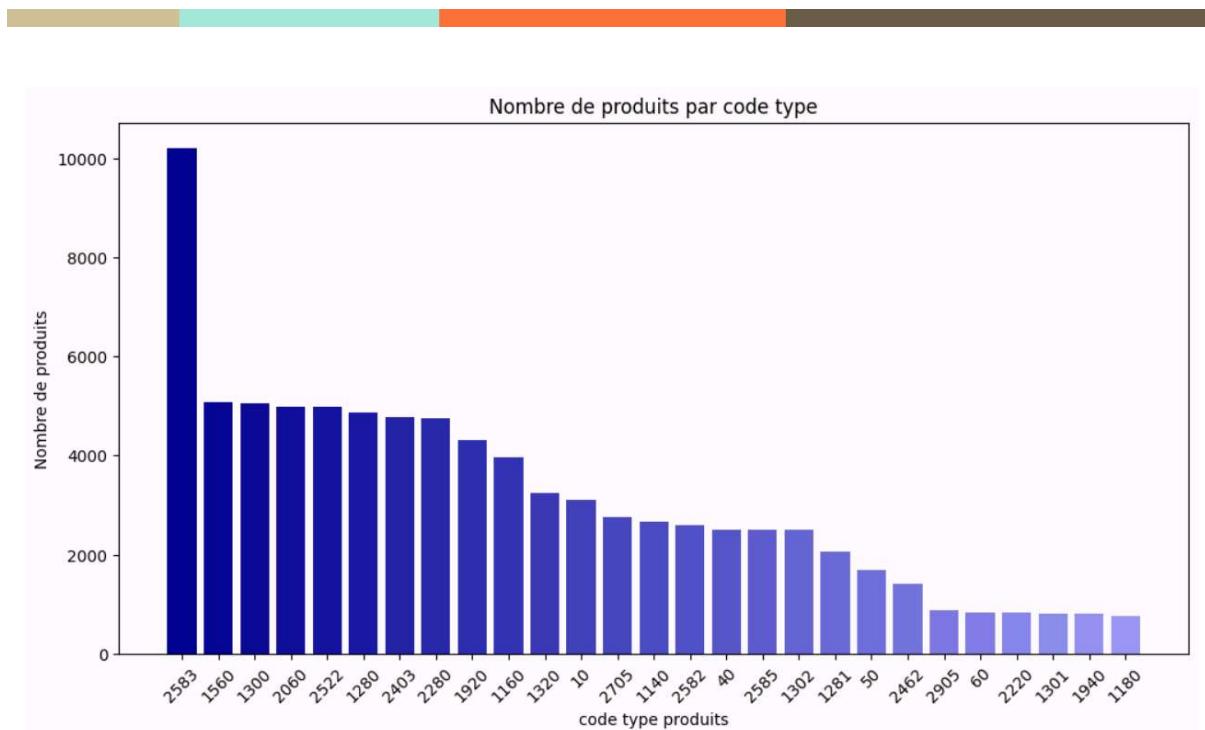
imageid

Identifiant unique pour l'image associée au produit, de type int64. Aucune valeur manquante trouvée.

Cette variable ne présente aucune valeur informative pour le modèle de classification textuel.

prdtypecode

Code type du produit, de type int64. C'est la variable cible à prédire. Ci-après les 27 codes type produits représentés dans le jeu de données. Le jeu de données apparaît déséquilibré.



Le travail de synthèse des éléments textuels conjugué à un rapprochement manuel sur des échantillons d'images et la consultation du site de Rakuten permet de proposer ces labels pour chacune des 27 catégories.

Catégorie	Libellé
10	Livres
40	Jeux Vidéo
50	Accessoires jeux vidéos
60	Jeux vidéo & Consoles
1140	Figurines
1160	Cartes de jeux
1180	Jeux de rôle et jeux de figurines
1280	Jouets & Enfant
1281	Jeux de société
1300	Véhicules radiocommandés et miniatures
1301	Chaussettes bebe
1302	Sports & Loisirs
1320	Puériculture
1560	Maison



1920	Linge de maison
1940	Petit déjeuner
2060	Décoration
2220	Animalerie
2280	Revue
2403	Lots de Livres et de Revues
2462	Lots de consoles et jeux vidéo
2522	Fournitures Papeterie
2582	Mobilier de jardin
2583	Equipement piscine & spa
2585	Outilage de jardin
2705	eBooks
2905	Jeux en téléchargement

3. Modalité Image

Les images correspondant au set d'entraînement uniquement seront utilisées pour la même raison que celle exprimée pour les données d'entrée textuelles.

Chaque produit identifié dans le dataset de texte dispose d'un fichier image qui lui correspond.

Le nom du fichier qui lui correspond est construit ainsi : "image_" + imageid + "_product_" + productid + ".jpg". imageid et productid étant 2 variables disponibles dans le dataset

Les images, en couleur, semblent toutes de format identique (500 x 500 pixels) et pèsent de 3 à 105 ko.

4. Volumétrie

Le set textuelle représente 84 916 entrées pour environ 52 Mo tandis que les images représentent autant de fichiers que d'entrées textuelles pour environ 2.5 Go.



IV. Preprocessing

A. Modalité Texte

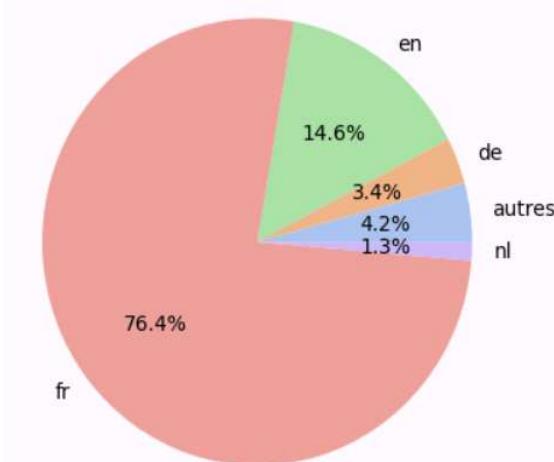
1. Préambule : problématique de langues multiples

Dans l'étape d'exploration des données, nous avons pu remarquer que les données étaient majoritairement rédigées en français mais qu'il y avait un certain nombre d'entrées dans d'autres langues.

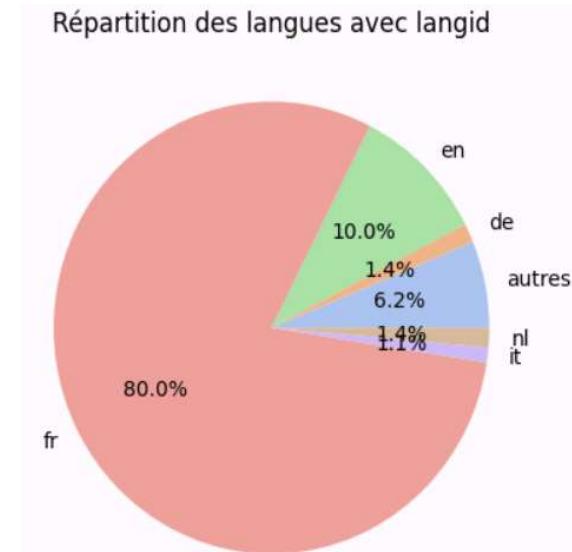
A ce stade, nous pressentons que la prise en compte de la langue peut avoir une influence sur l'étape de preprocessing et sur les performances des modèles qui seront testés. Nous décidons de quantifier la présence de ces autres langues afin de mieux aiguiller nos choix.

En utilisant une détection de langue à l'aide de la bibliothèque langdetect, sur une variable textuelle composée par la fusion des variables designation et description, il apparaît que près d'un quart des contenus n'est pas rédigé en français. Précisons toutefois que la performance de la détection n'est pas optimale et produit un certain nombre de faux positifs. A noter que langdetect trouve 31 langues différentes dans le set. La catégorie "autres" représente le cumul des langues dont la représentation unitaire dans le set de données est inférieure à 1%.

Répartition des langues avec langdetect



En utilisant une bibliothèque alternative, langid, la détection semble un peu meilleure mais il reste que 20% des contenus sont rédigés dans une autre langue que le français. A noter que langid trouve 68 langues dans le set.



Par la suite, nous identifierons et utiliserons la bibliothèque Fasttext pour détecter, de manière plus performante, la langue des observations.

S'agissant des bibliothèques disponibles sous Python pour traiter du NLP et utiles à notre problématique, il apparaît qu'elles ont un support hétérogène des langues multiples. Toutes les langues trouvées dans notre set de données ne sont pas couvertes.

En outre, nos prospections en lien avec l'étape de modélisation nous conduisent à envisager l'utilisation d'algorithmes dérivés de BERT, qui, dans l'univers du NLP, a introduit des avancées majeures en 2018. Ainsi, des modèles tels que camenBERT ou flauBERT semblent, aujourd'hui, faire référence. Cependant, ces modèles ne sont pas multilingues et n'existent, à date, que pour la langue française.

A ce stade, nous pouvons envisager plusieurs approches qui impacteront le contenu de l'étape de Data Engineering :

- Nous choisissons d'implémenter l'un des deux modèles cités ci-avant, associé à une approche de Transfert learning. La gestion de la langue se résumerait alors à 2 options :
 - Ne conserver dans le set, que les entrées en français, partant du principe que c'est la langue majoritaire ($> 75\%$ des entrées) et que les classes y sont largement représentées,
 - Traduire préalablement les entrées qui ne sont pas en français, vers le français.
- ou bien nous choisissons d'utiliser d'autres modèles couvrant les langues majoritaires en y associant une préparation des données avec prise en compte de la langue des entrées.

Finalement, ces différentes approches seront testées par le groupe. Et si la prise en charge des langues au maximum des capacités de chaque bibliothèque tout au long du pipeline (tokenisation, stop words, lemmatisation, ...) associée à l'utilisation de modèles de machine learning se révèlera peut encourageante (voire, avec une dégradation du scoring), la traduction préalable des observations en langue française, associée à des modèles de deep learning cumulent les avantages :

- Toutes les observations sont exploitées, ce qui favorise la performance,
- La gestion des langues au plus tôt dans le pipeline simplifie notamment le code,
- La taille du vocabulaire, un paramètre sensible lors de la vectorisation, n'explose pas puisqu'il se limite au vocabulaire d'une seule langue. Cela permet de garder une taille confortable de vocabulaire tout en limitant l'impact calculatoire.

2. Traitements

Nous présentons ci-après les différentes étapes ordonnées que nous envisageons dans la préparation des données textuelles et qui pourront cependant être ajustées selon le choix que nous effectuerons dans l'étape de modélisation.

a) Initialisation

Pour permettre de prendre connaissance des données et préciser les étapes nécessaires à leur préparation, il a fallu tout d'abord charger les 2 fichiers sources de données initiaux, dont features et target étaient respectivement séparées en X_train et Y_train, et les joindre (merge) dans un même dataframe. Cette étape a

permis de réaliser l'exploration de données, reprise en détail dans le chapitre III du présent document.

A noter que les données d'entrées que nous avons utilisées, et qui ont été fournies par Rakuten pour le challenge, sont extraites du sous-ensemble d'entraînement uniquement, seul jeu de données pour lequel les labels sont fournis.

Une analyse approfondie des langues présentes au niveau des variables textuelles nous incite à initialiser un dictionnaire de mots vides de sens, établi pour chaque langue majoritaire. Ce dernier est enrichi de mots vides additionnels. Les exercices de synthèse, par wordcloud notamment, pour chaque catégorie, ayant permis de les mettre en évidence.

b) pré-traitements

Ce premier examen statistique réalisé conjugué à notre appropriation grandissante du contenu des variables textes nous amènent à pratiquer la préparation suivante :

(1) Création de variables

Variable textuelle par fusion : à ce stade, il nous apparaît nécessaire de créer une unique variable textuelle réunissant les variables designation et description afin de préserver un maximum de valeur informative pour chaque entrée. Bien que disponible pour deux tiers des entrées seulement et ajoutant un bruit notable à designation, la variable description renferme des informations importantes utiles à l'exercice de synthèse. L'exploration des données a montré que, dans certains cas, la variable désignation est équivoque (titre d'une œuvre seul, par exemple). La variable description apporte alors des termes qui vont, eux, être discriminants pour la catégorie. Pour chaque entrée, la fusion est réalisée en remplaçant les valeurs manquantes de description par un texte vide (`fillna("")`) et lorsqu'elle est différente de la variable designation (égalité dans 1% des cas). Cette nouvelle variable recevra les traitements qui suivent,

Chemin d'accès aux images : pour faciliter l'exploitation des images de chaque entrée, une variable contiendra la construction du chemin d'accès à l'image selon les spécifications transmises par Rakuten, via l'utilisation des variables productid et imageid.

(2) Traitement des doublons

L'étape d'exploration des données a pu mettre en évidence la présence de doublons sur les variables textuelles. Bien que moins importants une fois la fusion réalisée, une analyse plus détaillée de ces doublons montre qu'aucun ne partage le même imageid. En regardant de plus près les images de doublons (par échantillonnage), on s'aperçoit qu'il s'agit souvent du même produit mais présenté en différentes

couleurs, ou encore, pour des broderies, puzzles ou peintures, par exemple, il s'agit de motifs différents à réaliser. Nous concluons qu'il ne s'agit pas de véritables doublons et qu'aucun traitement de doublons n'est à appliquer.

(3) Traitements du texte

Le pré-traitement est une étape essentielle dans l'analyse des données textuelles afin d'extraire des informations pertinentes. La succession des étapes suivantes visent à nettoyer et structurer le texte de chaque entrée dans la perspective d'une catégorisation optimale :

- **Conversion en minuscule** : permet d'uniformiser la casse et d'éviter d'inutiles doublons (ex : "Produit" vs "produit")
- **Remplacement des caractères d'encodage HTML** : conversion des entités HTML en leurs équivalents textuels pour garantir une analyse correcte,
- **Traduction des langues** : selon le scénario de prétraitement choisi, la traduction peut être appliquée à toutes les entrées non françaises vers le français ou uniquement aux langues marginales (moins de 1% des entrées) vers l'anglais. L'anglais est souvent la langue native de nombreuses bibliothèques NLP, facilitant ainsi l'analyse,
- **Nettoyage du Texte via les Expressions Régulières (RegEx)** : suppression des balises HTML, caractères spéciaux, nombres et autres éléments indésirables *via* des expressions régulières. Cette étape vise à conserver uniquement le texte pertinent pour l'analyse,
- **Tokenisation** : segmentation du texte en unités significatives (mots, phrases). Cette étape est essentielle en Machine Learning pour préparer le texte à une analyse plus approfondie, en identifiant les mots individuels ou les groupes de mots,
- **Suppression des mots vides de sens** : retrait des stopwords (ex. "le", "la", "et") qui n'apportent pas de valeur sémantique. Si support multilingue pour la solution, un dictionnaire contextuel à la langue de l'entrée est utilisé.
- **Lemmatisation** : réduction des mots à leur forme de base (ex. "courir" au lieu de "courant"). Cette étape aide à normaliser le texte et à améliorer la précision de l'analyse. Il s'agit ici de la synthèse ultime de la valeur informative de l'entrée, selon la langue (si support multilingue),
- **Rééchantillonnage** : équilibrage des classes sous-représentées dans le corpus. L'étape d'exploration ayant permis de caractériser un déséquilibre dans les catégories de produits, il est indispensable de rééquilibrer le set pour maintenir de bonnes performances de classification sur les classes moins représentées,
- **Vectorisation** : Conversion du texte en représentation numérique pour faciliter l'analyse statistique et l'apprentissage automatique. La vectorisation par TF-IDF est appliquée à la variable textuelle, résultante de toutes les opérations



précédentes, et enregistrée au sein d'un nouveau dataframe, spécialement créé pour accueillir cette unique variable qui servira d'entrée au modèle. Ce qui ponctue la préparation de la modalité textuelle,

- **Réduction de dimension** : des tests de réduction de dimensions seront effectués avec truncatedSVD pour synthétiser la matrice creuse générée par la vectorisation TF-IDF. Ce afin de réduire les temps de calcul, tout en conservant la totalité des observations,
- **Encodage des labels** : pour éviter toute interférence lors de l'apprentissage et les prédictions des modèles ML,
- **Séparation du jeu de données et sauvegarde** : séparation en deux ensembles de données (Train et Test) et sauvegarde sous forme de fichiers CSV du texte vectorisé à l'étape précédente et du mapping des labels encodés et des classes originelles.

A ce stade, nous n'excluons pas d'itérer sur l'étape de feature engineering, dans le cadre de la modélisation, afin de mesurer notamment l'impact de certaines tâches de préparation des données (stratégie d'échantillonnage, approche de la gestion des langues, ...) sur les résultats des modèles.

B. Modalité Image

1. Préambule

Pour exploiter les images, il est envisagé d'utiliser la technique du Transfert Learning afin de bénéficier d'un modèle pré-entraîné à la reconnaissance d'image et de fine tuner ce dernier sur notre ensemble d'images. Le modèle envisagé est ResNet. Les étapes de prétraitements des images ci-après s'inscrivent dans cette perspective.

2. Traitements

Le prétraitement des images prévoit :

- La structuration du filesystem. Au sein du dossier image, création de 2 sous-dossiers selon les jeux Train et Test. Dans chacun d'entre eux, création des 27 dossiers représentant les 27 catégories avec, comme nom de dossier, le numéro ou label retenu pour chaque catégorie. Copie des fichiers dans leur emplacement respectif,
- Redimensionnement. Passage de 500x500 pixels à 224x224 pour coller à la préparation du modèle pré entraîné,
- Augmentation de données. Rotation, flip, translation, zoom, filtres ..,
- Normalisation. Selon les statistiques du jeu utilisé dans l'entraînement de ResNet.

V. Modélisation

A. Introduction

1. Nature du projet

Le projet Rakuten, auquel nous sommes confrontés, présente un problème de classification multi-classes faisant intervenir deux modalités : le texte et l'image. Ces 2 modalités vont pouvoir être exploitées pour répondre à l'objectif qui est de classer les produits et annonces, mis en ligne par les vendeurs, en catégorie de produits sur le site de Rakuten. Ces catégories ont pour objet d'organiser les quelques millions de produits vendus sur le site afin de faciliter notamment la recherche pour de potentiels acheteurs.

Cette problématique est largement répandue sur les sites de ce type et plus généralement, dès lors qu'il s'agit de structurer un ensemble conséquent d'objets et de pouvoir les retrouver rapidement et simplement.

2. Métrique

Pour évaluer la performance de nos modèles, guider leur sélection et leur optimisation, la métrique que nous retenons est le F1-score pondéré. En effet, pour ce type de classification multiple, il faut être capable de mesurer la performance de prédiction pour chaque classe, ou, catégorie de produits ici. C'est, de plus, la métrique choisie dans le cadre du challenge Rakuten pour observer les performances des solutions proposées. L'accuracy est également utilisée, notamment en phase de pré-évaluation de modèles de machine Learning pour une première sélection des modèles les plus adaptés.

3. Méthodologie

L'approche globale du groupe a d'abord été de permettre à chacun d'explorer la modélisation sans contraintes. L'objectif était ici, pour chacun, de mettre en application les enseignements des sprints de la formation sur un cas concret et de s'approprier la problématique de classification proprement dite. Lors de nos points de partage hebdomadaires, nous pouvions nous enrichir mutuellement des pistes suivies par chacun et des résultats qu'elles produisaient.

Dans un second temps, nous avons cherché des voies complémentaires au sein du collectif afin de décupler notre capacité d'exploration. Enfin, dans un troisième temps, l'idée était, pour chacun, de pousser nos convictions afin de nous challenger



en interne et rechercher le meilleur score, tant sur la modalité texte que sur l'image. Avec l'intention fine, d'adopter collégialement, pour la suite du projet, la solution la plus performante.

Notons que l'ensemble des expérimentations de modélisation a été mené à l'aide de la plateforme Google Colab. Ce, afin de bénéficier d'un environnement matériel apte à la prise en charge de volumes de calculs conséquents, des nombreuses expérimentations et itérations qui allaient être pratiquées, tout en restant accessible économiquement.

B. Modalité Texte

1. Préambule

Le chapitre suivant vise à présenter les approches différenciantes et complémentaires, menées au sein du collectif, afin de restituer autant que possible leur originalité, au-delà de l'inévitable cheminement commun. Le chapitre suivant reprend les différents modèles testés avec les meilleurs scores obtenus. S'ensuit un chapitre dédié à la présentation de l'architecture de la solution retenue par le groupe, qui a obtenu les meilleurs résultats et constitue le socle de la solution pour la suite du projet.

A noter qu'à ce stade de modélisation, l'exercice a volontairement été mené de manière individuelle par chaque membre de l'équipe sur l'ensemble du pipeline (de la préparation du set à l'entraînement des modèles), ce afin de favoriser la richesse exploratoire et en repoussant la nécessaire étape de convergence à un moment ultérieur. Cette approche a notamment été motivée par le fait que nous souhaitions, chacun, être libre d'ajuster la préparation des données afin d'en comprendre les impacts sur les résultats des prédictions.

2. Approches développées

a) Scénarios testing

(1) Principe

Une des approches développées par le groupe a consisté à créer un programme, relativement générique, qui permettrait de tester de nombreuses combinaisons d'architectures de solutions (scénarios).

Au stade actuel de la formation, il n'est pas encore possible d'appréhender avec précision, l'impact sur la performance de la solution de choix opérés dans l'étape clé de preprocessing. En effet, de nombreuses options sont possibles dans la préparation des données, que ce soit dans l'étape de nettoyage du texte, dans le

choix des différentes bibliothèques et algorithmes à disposition, le support avancé ou non des langues multiples présentes dans le set, etc ...

L'idée a donc été, ici, d'identifier les principaux leviers qui séparent les données d'entrées brutes du set à la performance mesurée de la solution in fine, de manière à rendre chacun d'entre-eux, aisément paramétrable et permettre de pratiquer de nombreuses combinaisons de tests afin de guider nos choix.

(2) Leviers

Les principaux leviers identifiés sont :

- langue : niveau de support des langues (français seul, français et anglais, langues majoritaires, max)
- stopwords : choix de l'algorithme (*stopwords, spaCy, ...*)
- tokenisation : choix de l'algorithme (*split, word_tokenize, ...*)
- synthèse : choix de l'algorithme (racinisation, lemmatisation, ...)
- sampling : choix de la stratégie (random, under, over, *smote, ...*)
- vectorisation : choix de l'algorithme (*countVectorizer, TF-IDF*)
- modèle : choix du modèle (*LR, SVM, KNN, RF, CNN, ...*)
- optimisation : choix de la stratégie (*GridSearch, RandomSearch, ...*)
- langueAlgo : choix de l'algorithme (*langid, langdetect, ...*)

Hormis les leviers vectorisation, modèle et langueAlgo, chacun des leviers peut également être désactivé dans le but d'avoir une mesure de performance "baseline" avec une prise en charge de préparation limitée et permettre ainsi de mesurer l'impact de chaque levier sur la performance globale de la chaîne de traitement.

(3) Définition des scénarios

Une matrice de paramétrage permet de définir chaque levier selon les valeurs prises en charge par le programme afin de construire les différents scénarios de test. Ceux-ci sont ensuite constitués en un dictionnaire, soumis en entrée du programme principal, qui va exécuter chacune des entrées et stocker les résultats dans un fichier de trace.

capture matrice de paramétrage des scénarios

La concaténation du code sur un digit de chaque levier constitue un code de scénario interprété par chaque étape de la chaîne de traitement qui aiguille la prise en charge au sein des programmes.

(4) Bench

La matrice de confusion produite pour chaque scénario constitue le résultat tracé. L'intérêt est de conserver le détail des résultats par classe. Une fonction permet ensuite d'en déduire l'ensemble des métriques dont l'accuracy et le F1-Score pondéré. Un bench des différents scénarios peut ainsi être établi afin de comparer les performances des options choisies et guider l'équipe vers les choix les plus prometteurs.

Extrait du bench de scénarios

(5) Interprétation des résultats

L'objectif des tests successifs visait à mesurer l'impact d'une sophistication progressive de la chaîne de traitement des données sur le score final de la solution.

Afin de constituer une référence, le premier scénario testé s'est voulu des plus basiques. Le texte brut a ainsi été vectorisé via un *CountVectorizer* et soumis à une régression logistique. Le F1-Score pondéré, qui est la métrique retenue pour l'évaluation des solutions, s'établit alors à 77,19%.

Dans le second test, le seul changement apporté est un sampling du set ne conservant que les données d'entrées en langue française (plus des 3/4 du set initial). Cette opération se traduit par un gain de plus de 2 points sur le F1-Score pondéré. La conservation des autres langues en l'état, ici dans le set, se traduit davantage par du bruit qu'une plus value informationnelle utile à la classification. Ce qui peut se comprendre puisque les relations statistiques entre features et targets sont alors traduites de manière bien différente selon le corpus.

A partir du 3ème test, on active les différents leviers identifiés (*stopwords*, lemmatisation, choix d'une vectorisation *TF-IDF*) mais pas de travail sur l'équilibrage des classes. Le score évolue peu à 79,63%. Ce résultat est surprenant car il implique une préparation bien plus importante. On aurait pu s'attendre à un score sensiblement plus élevé.

Le quatrième test visait à mesurer si la prise en charge des langues majoritaires, au travers des différents leviers, influençait favorablement le score. Ici, les différentes bibliothèques utilisées le long du pipeline sont paramétrées dynamiquement selon la langue de chaque entrée pour un traitement optimal. Cependant, là encore, aucun effet positif sur le score. Pis, ce dernier recule de plus d'un point et demi ! Score qui recule encore d'un demi point avec une prise en charge poussée au maximum du support de langues des bibliothèques utilisées au niveau des différents leviers (test 5). On conclut alors que la recherche de prise en charge au plus près des langues à chaque étape de la chaîne n'est pas la bonne approche. Nous comprendrons plus tard qu'il y a de nombreux avantages à régler cette question des langues le plus tôt possible dans le pipeline.

Pour la plupart des tests à partir du 6ème, le support des langues n'ayant pas donné satisfaction, seules les entrées en français sont conservées et les différents leviers de prétraitement restent activés et fixés aux mêmes valeurs, hormis le travail sur l'équilibrage des classes (sampling). L'objectif est, cette fois, de multiplier les tests de modèles distincts.

Sur une régression logistique jusqu'alors, pour ne pas nuire à la mise en évidence des impacts liés à la préparation du set, on introduit désormais, dans le 6ème test,

un nouveau modèle : le *SVM*. Le score s'établit à 81,23%. De tous les modèles testés par la suite, c'est ce modèle qui obtiendra les meilleurs scores et qui semble donc le mieux adapté à la problématique.

Un certain nombre d'autres modèles sont ensuite testés (*KNN*, *Random Forest*, *multinomialNB*) qui, tous, produisent des résultats de plusieurs points inférieurs.

Une reproduction de l'architecture du *CNN* utilisé par Rakuten n'atteint que 54% de F1-score pondéré contre plus de 81% pour le modèle de référence. Il n'a pas été possible de mettre en évidence les raisons de cette contre performance.

Le *SVM* constituant le modèle le plus prometteur, les tests suivants visent l'optimisation d'une solution basée sur ce modèle. Un travail sur le sampling est alors pratiqué (sous-échantillonnage sur classe minoritaire, de la classe majoritaire uniquement, ramenée à 5k observations et *SMOTE*) mais tous auront pour effet de dégrader le score. Il semble, dans le contexte de notre projet, qu'il soit davantage crucial de conserver le maximum d'observations pour un F1-score pondéré optimal.

S'ensuit une dernière campagne de tests visant à optimiser les hyperparamètres du modèle *SVM* avec une première approche grossière sur *GridSearchCV*, suivie d'une approche plus fine avec *BayesSearchCV* qui permettra d'atteindre le F1-Score pondéré de 82,95%. A noter que ce score a été obtenu en ajustant à 60k, la taille du vocabulaire de la vectorisation *TF-IDF*. En effet, plusieurs tests ont été fait en augmentant progressivement la taille du vocabulaire avec un score qui progressera jusqu'à cette taille pour se dégrader au-delà.

b) L'efficacité est dans la bibliothèque !

Une autre initiative, au sein du collectif, s'est attachée, par une exploration et expérimentation aiguës et rigoureuses, à rechercher les bibliothèques et algorithmes *NLP* (Natural Language Processing) les plus efficaces, gages soupçonnés d'une performance globale optimale de la solution de classification.

Cette initiative a évalué, selon différents leviers du pipeline de préparation du texte, l'efficacité comparée de plusieurs bibliothèques pour, in fine, alimenter le collectif sur celles qui devaient être retenues. C'est ainsi qu'ont pu être isolées les bibliothèques *fastText*, pour la détection de la langue d'un texte, à la fois précise et rapide, *spaCy*, pour nombreux d'algorithmes de manipulation (tokenization, stopwords, lemmatisation, ...) ou encore *Beautiful Soup*, plutôt que le classique recours à *regex*, afin d'éliminer efficacement et simplement toutes surcharges *HTML* dans un texte et pour pratiquer la conversion des entités *HTML* en texte intelligible.

Outre l'identification de bibliothèques efficaces, l'évaluation en avance de phase de techniques et divers modèles a permis un gain de temps significatif au collectif. Il s'agit en particulier de :

- l'évaluation des bénéfices d'une réduction de dimension après vectorisation, sur matrice creuse, avec *TruncatedSVD*,
- l'évaluation de la référence *BERT*,
- l'expérimentation de grilles d'hyperparamètres avec *GridSearchCV*, *RandomizedSearchCV* et *BayesSearchCV*,
- le feedback sur une grande diversité de modèles testés : *Logistic Regressor*, *Random forest*, *multinomialNB*, *SVM*, *KNN*, *XGBoost*, *CNN*, donnant des indications précieuses au collectif et orientant ses choix.

c) L'optimisation est dans le résultat !

Au registre de l'optimisation, une idée originale est venue de l'examen et de l'exploitation des matrices de confusion avec une interrogation : comment permettre au modèle de discriminer les classes qu'il a tendance à confondre ? De cette question naît une idée : "forcer le destin" ou, plus prosaïquement, s'agit-il, pour les classes concernées, de lister les mots clés "charnières" afin de les "booster" artificiellement lorsqu'ils sont rencontrés dans le texte. Ils auront alors plus de poids et infléchiront en conséquence, favorablement, les prédictions, pensions nous.

Cette approche hybride, mêlant rigueur linguistique et pragmatisme statistique, a permis, en effet, une amélioration sensible des performances, notamment sur les classes difficiles à prédire.

```
--- Optimisation SVM (fr) k_svd=300 ---
Fitting 3 folds for each of 15 candidates, totalling 45 fits
SVM best params: {'svm_kernel': 'rbf', 'svm_gamma': 'auto', 'svm_C': 5}
SVM best F1 (CV): 0.7950
[SVM - fr] Accuracy: 0.7950 | F1-weighted: 0.8021
```

Un des scores obtenus sur SVM bénéficiant de cette technique de "boosting"

Mais nous devions finalement renoncer à cette technique qui ne pouvait être généralisée et s'apparente, dans les faits, à un surapprentissage car spécifique aux mots du corpus d'entraînement !

d) Benchmarks et reproductibilité

Au registre des enseignements glanés durant l'étape de modélisation, la nécessité de maîtriser le set en entrée de chaque entraînement de solution, quelle qu'elle soit, s'avèrera déterminant pour comparer les performances relatives de manière fiable.

Afin de s'assurer de cette reproductibilité du set, il est vite apparu nécessaire de fixer les sets d'entraînement et de test et ce, pour l'ensemble des pipelines évalués. Et le plus simple et lisible pour le faire était encore de découper les sets selon l'index des enregistrements et de stocker ceux-ci distinctement sous forme de fichiers csv, appellable en entrée de chaque évaluation et itérations.

L'assurance d'utiliser rigoureusement les mêmes jeux pour des comparaisons d'une grande fiabilité.

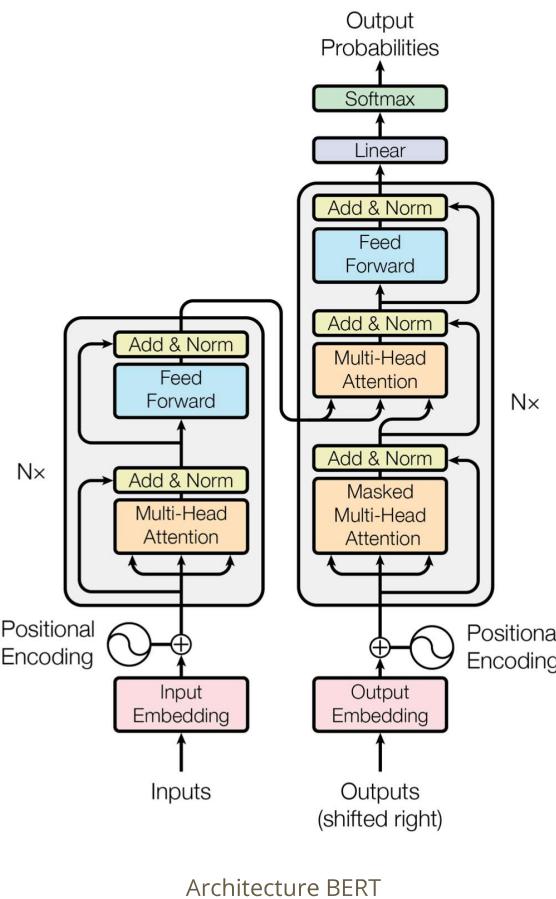
e) To deep or not to deep ?

La dernière approche exposée ici, et qui constituera finalement le choix du collectif, a été d'examiner d'emblée les solutions les plus avancées utilisables dans notre type de problématique. L'évolution constante des technologies d'IA et particulièrement, celles au cœur du deep learning et du *NLP* est jalonnée de progrès décisifs. L'un de ces progrès disruptifs, introduit par Google en 2018, a été la mise au point du modèle *BERT* pour "Bidirectional Encoder Representations from Transformers".

BERT est une architecture spécialisée sur la réalisation efficace de deux tâches de *NLP* :

- MLM (masked language model) : approche *NLP* visant à entraîner le modèle pour qu'il retrouve les mots masqués dans une phrase. Non séquentielle, cette technique traite l'ensemble de la phrase en une fois et non un mot après l'autre, dans un ordre déterminé, comme c'est le cas avec les techniques qui ont cours jusqu'alors,
- NSP : prédiction de la phrase suivante, connexion logique entre 2 phrases.

Pré-entraîné et doué d'une connaissance générale et bidirectionnelle d'une langue, une architecture basée sur *BERT* est capable de saisir le contexte et de déterminer si deux phrases sont consécutives ou ayant une connexion logique. Cette capacité est au cœur de la performance de cette architecture.



Architecture BERT

L'utilisation de modèles de deep learning, suggérant de maximiser le nombre d'observations à présenter lors de la phase d'entraînement, associée aux sirènes de la performance du deep, ont participé à structurer l'approche dont il est question ici et a conduit à deux choix majeurs :

- L'utilisation des modèles *flaubERT* (CNRS) et *camemBERT* (Facebook AI Research et Inria), tous deux dérivés de l'architecture BERT et qui font autorité en langue française,
- La traduction en français des entrées du set, formulées en langues étrangères, préservant ainsi l'intégralité des données pour l'entraînement des modèles !

3. Récapitulatif des meilleurs scores

Ci-après, on retrouve la diversité des modèles testés par le collectif ainsi que les meilleurs F1-score pondérés obtenus pour chacun d'entre eux.

Modèle	%	Meilleur F1-Score pondéré	Commentaire
Regression Logistique		82,34%	Sur les seules observations en langue française, Stopwords, word_tokenize, lemmatisation, TF-IDF, à l'issu d'un RandomSearch
MultinomialNB		78,59%	Sur les seules observations en langue française, Stopwords, word_tokenize, lemmatisation, TF-IDF, à l'issu d'un RandomSearch
KNN		59,73%	
SVM		82,99%	Sur les seules observations en langue française, Stopwords, word_tokenize, lemmatisation, TF-IDF (vocab_size = 60k), à l'issu d'un GridSearchCV puis d'un BayesSearchCV
CNN		80,32%	Sur les seules observations en langue française.
Decision tree		67,87%	
XGBoost		79,91%	
FlaubERT		88,45%	
CamemBERT		88,19%	
Meta modèle LR		90,00%	On entraîne ici une régression logistique sur la concaténation des logits des modèles FlaubERT et camemBERT
Meta modèle LightGBM		90,05%	Idem avec le classifieur LightGBM
Meta modèle XGBoost		90,31%	Idem avec le classifieur XGBoost
Meta modèle fusion final		90,52%	Post optimisation Optuna

Meilleurs F1-Score obtenus - modalité Texte

EPOCH 26 - RÉSULTATS DÉTAILLÉS: F1-weighted: 0.8845 NOUVEAU RECORD! F1-macro: 0.8844 Accuracy: 0.8850 Eval Loss: 0.4650	EPOCH 46 - RÉSULTATS DÉTAILLÉS: F1-weighted: 0.8819 NOUVEAU RECORD! F1-macro: 0.8824 Accuracy: 0.8826 Eval Loss: 0.4733	F1-weighted: 0.9031 Precision-weighted: 0.9030 Logistic Regression F1-weighted: 0.9000 LightGBM F1-weighted: 0.9005 XGBoost F1-weighted: 0.9031
Flaubert	Camembert	Meta modèles

Aperçu des meilleurs scores des modèles retenus pour la solution

Pour mémoire, le score de référence du challenge Rakuten pour la modalité texte est de 81,13%



4. Approche retenue

a) Chronique d'une solution de classification

Les premiers modèles SVM, dont chaque membre de l'équipe confirma le potentiel par ses expérimentations propres, avaient permis de dégager des tendances claires, notamment sur les classes problématiques. Il devenait alors naturel de franchir un nouveau palier en termes de modélisation, en nous tournant vers des approches plus profondes, capables de capturer la structure syntaxique, les nuances sémantiques, et les subtilités propres à chaque catégorie de produits.

Cette approche, entièrement dédiée au deep learning et appliquée aux textes, avait un double objectif : tester la robustesse de modèles pré-entraînés de type Transformers dans ce contexte multiclass, et vérifier si la richesse issue de la traduction pouvait pleinement s'exprimer dans un environnement d'apprentissage profond.

Comme introduit dans les chapitres précédents, *camemBERT* et *flauBERT* se sont rapidement distingués par leur performance, leur stabilité en fine-tuning, et leur capacité à généraliser, même avec peu d'exemples pour certaines classes.

L'approche a été rigoureuse : jeux stratifiés, tokenisation dédiée, encapsulation en dataset PyTorch, early stopping, surveillant en permanence la validation pour éviter tout surapprentissage. Et ça a payé ! Les premiers résultats furent des plus encourageants avec un F1-score pondéré autour des 86% pour chacun des 2 modèles, bien plus que les résultats obtenus avec les modèles de Machine Learning classiques !

Cela validait bien la pertinence de cette approche qui fut alors adoptée par le groupe pour les étapes suivantes du projet, à commencer par l'optimisation des architectures de deep learning utilisées et le développement des techniques de mise en œuvre de méta modèles.

	0	0.247760	0.366659	0.659197	0.653659	0.651617
6	0.213700	0.568919	0.865657	0.863959	0.868552	
7	0.203500	0.688428	0.866342	0.863867	0.868089	

```
CamemBERT: {'eval_loss': 0.49727100133895874, 'eval_accuracy': 0.8588826305373725, 'eval_f1_weighted': 0.8574339762679193, 'eval_f1_macro': 0.8597622224747915, 'eval_runtime': 98.7028, 'flaubert': {'eval_loss': 0.4661736786365509, 'eval_accuracy': 0.8625361546658548, 'eval_f1_weighted': 0.8609118894349957, 'eval_f1_macro': 0.8642558042153223, 'eval_runtime': 95.2243, 'model': 'flaubert'}}
```

Scores individuels des modèles flauBERT et camemBERT

b) Une architecture optimisée

Ces premiers bons résultats unitaires auraient pu suffir. Mais l'examen des résultats de classification de ces deux modèles laissaient apparaître des différences notables.

	precision	recall	f1-score	support
Livres	0.72	0.66	0.69	467
Jeux Vidéo	0.78	0.81	0.80	376
Accessoires jeux vidéos	0.85	0.96	0.90	300
Jeux vidéo & Consoles	0.98	0.98	0.98	300
Figurines	0.83	0.81	0.82	401
Cartes de jeux	0.97	0.95	0.96	593
Jeux de rôle et de figurines	0.80	0.96	0.87	300
Jouets & Enfant	0.82	0.70	0.76	731
Jeux de société	0.66	0.75	0.70	311
Véhicules RC & miniatures	0.98	0.96	0.97	750
Chaussettes bébé	0.99	0.99	0.99	300
Sports & Loisirs	0.82	0.85	0.83	374
Puericulture	0.82	0.81	0.82	486
Maison	0.87	0.87	0.87	750
Linge de maison	0.93	0.94	0.94	646
Petit déjeuner	0.98	0.99	0.99	300
Décoration	0.89	0.82	0.85	749
Animalerie	0.94	0.99	0.96	300
Revue	0.91	0.96	0.93	714
Lots Livres & Revues	0.92	0.80	0.85	716
Lots consoles & jeux	0.91	0.93	0.92	300
Fournitures Papeterie	0.95	0.94	0.94	748
Mobilier de jardin	0.81	0.87	0.84	388
Équipement piscine & spa	0.96	0.98	0.97	750
Outilage de jardin	0.93	0.89	0.91	374
eBooks	0.75	0.88	0.81	414
Jeux en téléchargement	1.00	1.00	1.00	300
accuracy			0.88	13138
macro avg	0.88	0.89	0.88	13138
weighted avg	0.89	0.88	0.88	13138

Exemple de rapport de classification pour le modèle flauBERT

Mais lequel avait raison ? Et si tantôt c'était l'un et tantôt l'autre ? Avec cette interrogation en tête, les recherches, dans le cadre de la modélisation, ont permis d'identifier une pratique avancée dans ce domaine : la conjugaison de modèle !

En effet, si nos deux modèles ont bien appris, ils semblent surtout avoir appris différemment. Et c'est là que réside la clé de la performance atteinte in fine : ne pas choisir mais réunir pour bénéficier de leur complémentarité !

Ainsi, l'architecture de la solution prévoit-elle d'exploiter les logits, c'est-à-dire les sorties brutes de l'extraction des caractéristiques du réseau, avant les couches de classification à proprement parler, qu'il est possible de sauvegarder et combiner à ceux d'autres modèles !

Dès lors, un méta-classifieur, combinant les logits des deux modèles, pouvait-il être créé, lissant les incertitudes de l'un avec la confiance de l'autre. Cette stratégie de fusion tardive a renforcé la stabilité globale, tout en corrigeant les erreurs propres à chaque architecture. Les gains ont été clairs, avec un F1-Score pondéré dépassant les 90% !

c) Un pipeline stabilisé

Nous proposons ci-après, une description détaillée des différentes étapes du pipeline appliqué à la modalité textuelle, en particulier, dans l'architecture retenue pour notre solution.

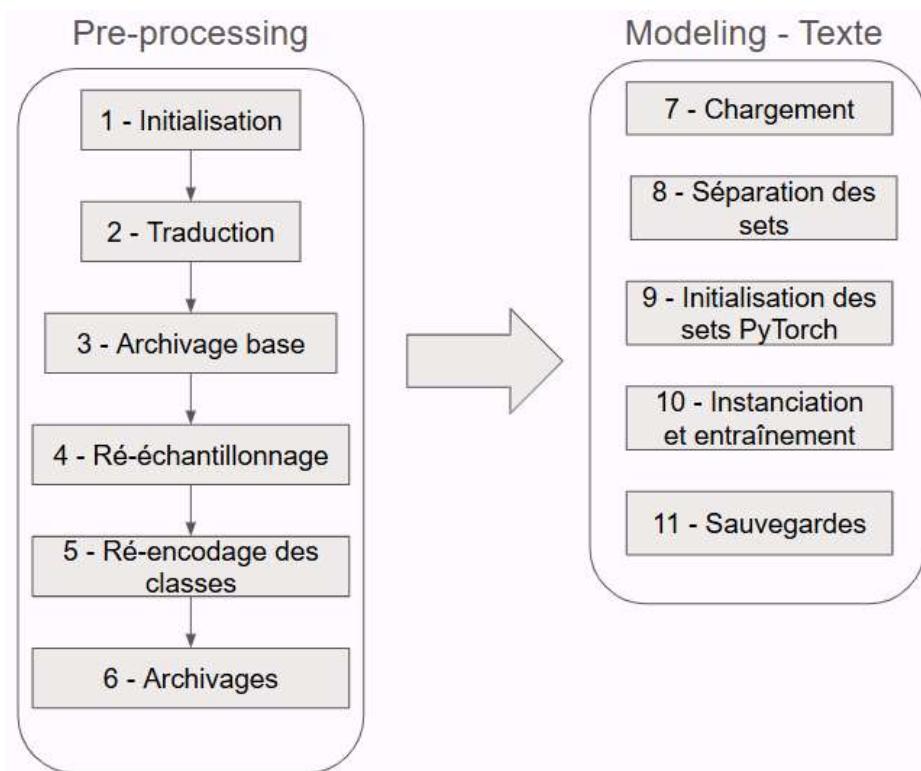


Schéma d'architecture de la solution retenue

1 - Initialisation

L'initialisation prévoit notamment la fusion des champs "designation" et "description" du set initial, qui devient la donnée d'entrée textuelle pour l'étape de modélisation.

Cette donnée est ensuite nettoyée avec parcimonie. En effet, l'approche consistant à exploiter une solution full deep learning, cela impacte sensiblement la préparation des données car, contrairement aux actions classiques pratiquées dans ce domaine dans le cadre de l'utilisation de modèles de machine learning, l'utilisation des modèles *flauBERT* et *camemBERT* ne nécessite que peu de travail sur le set, dans la mesure où l'essentiel des transformations nécessaires est nativement intégré aux modèles eux-mêmes.

Et de manière générale, là où l'utilisation de modèles de machine learning suggère une synthèse ultime du texte pour ne garder qu'un contenu à haute valeur informative et discriminant (suppression des mots vides de sens, conservation des seuls lemmes, ...), les modèles de deep learning utilisés en transfert learning sont capables de saisir le contexte et nécessitent de préserver au maximum le texte originel, tout au plus, nettoyé de caractères indésirables (balises et encodage HTML notamment dans notre cas).

Dans l'initialisation on trouve également la constitution d'un champ additionnel qui contient le chemin d'accès complet à l'image qui accompagne chaque donnée d'entrée textuelle.

2 - Traduction

La traduction de près d'un quart du set initial est confiée à la bibliothèque *Deep Translator* de Google. Cette traduction n'est effectuée que pour les entrées concernées, identifiées grâce à la bibliothèque *fastText*, au sein d'un champs ajouté au jeu de données.

La traduction nécessitant beaucoup de ressources, plusieurs types de mécanismes de reprise sur incidents sont implémentés.

3 - Archivage base

Le dataframe, nettoyé des champs inutiles, est sauvegardé au format csv.

4 - Ré-échantillonnage

Le set présentant un déséquilibre important au sein des classes, les classes minoritaires sont sur échantillonnées à 2k observations, tandis que les classes majoritaires sont ramenées à 5k.

5 - Ré-encodage des classes

Pour éviter tout parasitage lors de la phase d'entraînement, les classes sont ré-encodées avec la bibliothèque LabelEncoder.



6 - Archivages

Sont archivés :

- Le dataframe nettoyé, traduit et ré-échantillonné au format csv,
- Les classes encodées à l'aide de la bibliothèque Joblib,
- Le mapping des classes au format json à des fins de présentations ultérieures

7 - Chargement

Chargement, depuis les archives correspondantes, de la base de données (dataframe) et des classes encodées.

8 - Séparation des sets

La base de données est séparée à l'aide du module train_test_split en set d'entraînement et de validation selon leur index. Le set de validation est constitué à hauteur de 15% du jeu de données total. Les index du set de validation est archivé au format json pour constituer le point de départ du chargement des données à modéliser des sets d'entraînement (par différence) et de validation.

9 - Initialisation des sets PyTorch

Constitution des datasets (dérivés de la classe éponyme de PyTorch) à partir de notre donnée textuelle, des labels encodés et du tokenizer instancié selon le modèle (flauBERT ou camembERT) et des sous sets correspondants (entraînement et validation, selon leur index).

10 - Instanciation et entraînement

Instanciation et entraînement des modèles avec comme principaux paramètres :

- Evaluation et sauvegarde du modèle à chaque epoch
- Learning Rate : 2e-5
- Taille des batchs (train et val) : 16
- Nb d'epoch : 20
- Décroissance du poids : 0.01
- Métrique : f1-pondéré
- Patience Early stopping : 3

Les deux modèles partagent les mêmes données d'entrée et paramètres.

Une fonction pour le calcul des métriques est définie, retournant l'accuracy, le f1-macro et le f1-pondéré via récupération du label bénéficiant de la plus forte probabilité lors de la prédiction.

11 - Sauvegardes

Sont sauvegardés :

- les modèles ayant produit le meilleur score,
- les logits et labels au format binaire sur l'ensemble du dataset PyTorch

C. Modalité Image

1. Travail exploratoire

Chaque produit de notre jeu de données est accompagné d'une image, source d'information visuelle complémentaire au texte descriptif. L'association des deux modalités (texte + image) est essentielle pour améliorer la performance globale de la classification.

L'exploitation de la modalité image s'appuie principalement sur des techniques de **deep learning** avec transfert learning, afin de bénéficier des poids pré-entraînés sur de larges bases d'images (notamment ImageNet). Cette approche est nécessaire à la fois pour limiter les ressources requises et pour maximiser l'efficacité du modèle.

Contrairement à la préparation du texte, qui implique un pipeline complexe (nettoyage, vectorisation, etc.), la préparation des images se concentre essentiellement sur :

- **L'adaptation des images au format attendu par les modèles**, incluant un redimensionnement à 224×224 pixels et une normalisation avec les moyennes et écarts-types utilisés lors du pré-entraînement sur ImageNet. Cette normalisation garantit la cohérence statistique entre nos images et celles utilisées initialement pour entraîner les modèles, condition indispensable à un transfert learning efficace.
- **L'augmentation de données via des transformations aléatoires** (rotation, flou, variations de contraste...) appliquées dynamiquement pendant l'entraînement grâce à la méthode **TrivialAugmentWide**. Cette étape renforce la robustesse du modèle face à la variabilité naturelle des images.

Les expérimentations sur la modalité image ont été menées individuellement par chaque membre, avec des approches assez similaires. En effet, la nature des modèles d'image (réseaux convolutifs et transformers) et les contraintes techniques réduisent la diversité des options possibles, contrairement au texte.

Par ailleurs, ce travail nécessite un accès à des ressources matérielles importantes, notamment des GPU pour entraîner efficacement les modèles sur un dataset de taille conséquente (près de 60 000 images). Nous avons utilisé la plateforme Google Colab, qui offre un accès gratuit mais limité (environ 1h30 d'utilisation GPU par jour), ce qui a conditionné nos choix d'entraînement et de découpage.

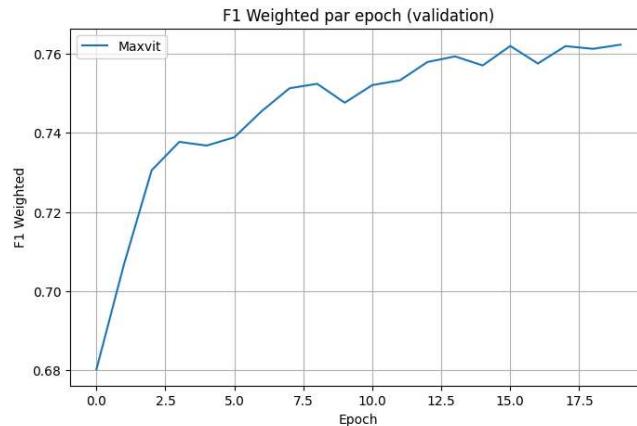
Enfin, il est rapidement apparu que **l'utilisation intégrale des données d'entraînement est indispensable** pour atteindre des performances satisfaisantes : toute réduction significative du dataset (under-sampling) s'est avérée préjudiciable à la qualité des modèles. C'est pourquoi le travail a principalement porté sur la diversité des architectures testées, afin d'identifier les modèles les plus performants et complémentaires pour la fusion tardive.

2. Récapitulatif des meilleurs scores

Voici résumé ci-après, les différents modèles testés sur la modalité image avec leur F1-score respectif :

Modèle	F1-Score pondé	Commentaire
VGG16	47,00%	
ResNet50	60,87%	
ResNet50_Weights	69,97%	
VGG16 + ResNet50	72,29%	Moyenne des logits
EfficientNet	73,97%	
EfficientNetV2	75,25%	
ConvNext	75,17%	
ConvNextv2	77,37%	
Swin	74,04%	
Maxvit	76,20%	
CoatNet2	74,52%	
Gcvit	76,45%	
Meta modèle fusion final	79,84%	Post optimisation Optuna

Meilleurs scores obtenus pour les modèles Image testés



Exemple de courbe d'apprentissage pour l'un des meilleurs modèles Image

Pour mémoire, le score de référence du challenge Rakuten pour la modalité texte est de 55,34%

3. Architecture retenue

Dans un premier temps, nous avons expérimenté des architectures classiques bien établies en vision par ordinateur, notamment **VGG16** et **ResNet50**. Ces modèles servent souvent de baseline solide grâce à leur robustesse et à leur large utilisation. Ils ont permis d'implémenter et de valider notre pipeline complet incluant :

- Le chargement optimisé des images via un champ **image_path** dans le dataset,
- La normalisation des images selon les moyennes et écarts-types ImageNet standards ($\text{mean} = [0.485, 0.456, 0.406]$, $\text{std} = [0.229, 0.224, 0.225]$), garantissant la cohérence avec les poids pré-entraînés,
- L'augmentation de données dynamique avec **TrivialAugmentWide**, augmentant la diversité des images d'entraînement et la robustesse du modèle,
- L'utilisation de couches de Dropout pour limiter le surapprentissage,
- L'extraction des logits, vecteurs de sortie intermédiaire avant la décision finale,
- Un découpage identique des données (train/validation) pour tous les modèles, assurant la comparabilité directe des résultats.

Bien que ces modèles aient donné des résultats corrects (F1-score pondéré autour de 71-72%), notre volonté d'amélioration nous a conduits à explorer des architectures plus récentes et performantes.

Nous avons ainsi intégré plusieurs modèles de nouvelle génération via la bibliothèque **timm**, qui offre des implémentations optimisées et pré-entraînées, respectant toutes les mêmes standards de prétraitement et de split :

- **EfficientNet V2 (efficientv2L)** : célèbre pour son excellent compromis vitesse/performance, grâce à une nouvelle approche de mise à l'échelle et à des couches convolutives repensées,
- **ConvNeXt V2 (convnext2)** : un modèle convolutionnel modernisé, conçu pour rivaliser avec les transformers, combinant simplicité et puissance,
- **MaxViT (maxvit_base_tf_224)** : un hybride novateur mêlant convolutions et mécanismes d'attention multi-dimensionnelle, permettant de capter finement aussi bien les détails locaux que la structure globale des images,
- **CGViT (gcvit)** : un transformer convolutionnel qui associe l'efficacité des convolutions à la capacité des transformers à modéliser les relations globales, apportant un équilibre intéressant entre précision et coût computationnel.
- **CoaNet2(coatnet_rmlp_2_rw_224.sw_in12k_ft_in1k)**: un modèle hybride combinant couches convolutionnelles et blocs transformers, conçu pour capter efficacement à la fois les détails locaux et les dépendances globales dans l'image, offrant ainsi un excellent compromis entre précision et complexité computationnelle.

Chaque modèle a été entraîné avec la même méthodologie rigoureuse, utilisant le même découpage train/validation, la même normalisation ImageNet standard, et un pipeline d'augmentation identique, garantissant ainsi une évaluation rigoureuse et comparable. Ces modèles avancés ont tous fourni des F1-scores pondérés compris entre 74% et 77%, témoignant de leur performance très prometteuse dans ce contexte.

Cette homogénéité dans le prétraitement et la gestion des données est un élément clé de notre démarche. En effet, pour assurer une évaluation rigoureuse et fiable des différents modèles testés, il est essentiel d'appliquer une **standardisation stricte du pipeline de traitement des images** ainsi qu'un **découpage identique du dataset** pour l'entraînement et la validation.

La normalisation des images selon les valeurs moyennes et écarts-types utilisées dans ImageNet ([0.485, 0.456, 0.406] pour la moyenne, [0.229, 0.224, 0.225] pour l'écart-type) est une étape incontournable, car tous les modèles pré-entraînés que nous exploitons ont été entraînés sur cette base. Cette normalisation garantit que la

distribution statistique des images d'entrée correspond à celle des données initiales de pré-entraînement, assurant ainsi que les poids pré-apris soient exploités de manière optimale.

Par ailleurs, utiliser un découpage train/validation identique pour tous les modèles est fondamental afin que les comparaisons soient justes et ne soient pas biaisées par une variation des données d'évaluation. Chaque modèle est donc entraîné et évalué sur les mêmes ensembles d'images, permettant d'attribuer toute différence de performance uniquement aux qualités intrinsèques du modèle et non à un artefact du jeu de données.

De plus, appliquer la même stratégie d'augmentation de données (notamment via **TrivialAugmentWide**) contribue à homogénéiser les conditions d'apprentissage et à favoriser une robustesse comparable entre modèles.

Finalement, la plupart des modèles testés sera retenue, la conjugaison de leurs logits apportant diversité et complémentarité au service de la performance globale de la solution, comme nous le verrons dans le chapitre suivant.

D. Architecture multi-modale

1. Préambule

Le projet Rakuten propose un défi de classification produit où les données disponibles sont hétérogènes : une description textuelle (désignation, description) et une image associée à chaque produit. Ces deux modalités sont complémentaires : le texte apporte un contexte sémantique précis, tandis que l'image fournit une représentation visuelle souvent indispensable pour une classification fine.

L'enjeu principal est donc de construire des modèles capables de tirer parti de ces deux sources d'information, aussi bien indépendamment que conjointement, afin d'optimiser la précision de classification. La stratégie retenue repose sur la construction de trois méta-modèles distincts :

- Un méta-modèle basé uniquement sur les représentations issues du **texte**,
- Un méta-modèle basé uniquement sur les représentations issues de **l'image**,
- Un méta-modèle combiné, ou **fusion multimodale**, exploitant simultanément les deux sources.

Dans notre démarche de classification multimodale, nous avons développé une architecture progressive et hiérarchique qui maximise la complémentarité des informations issues du texte et de l'image.

2. Principes

a) Concaténation des logits issus des meilleurs modèles

Nous exploitons d'abord les meilleurs modèles spécialisés sur chaque modalité :

- Pour le texte, les transformers CamemBERT et FlauBERT,
- Pour l'image, des architectures puissantes de deuxième génération telles que ConvNeXt V2, EfficientNet V2, MaxVit, CGViT et CoatNet2.

Chacun de ces modèles produit des logits représentant des vues différentes et complémentaires des données. Ces logits sont extraits sur l'ensemble du dataset, respectant un découpage train/validation strict et une normalisation homogène.

b) Le MLP de fusion comme booster

Afin de renforcer encore cette représentation, nous avons conçu un modèle de fusion dédié, un **MLP profond** qui prend en entrée la concaténation des logits de CamemBERT, FlauBERT ainsi que des meilleurs modèles images.

Ce MLP agit comme un **booster**, un modèle supplémentaire capable de capturer des interactions complexes entre les caractéristiques texte et image à un niveau intermédiaire, produisant une sortie affinée sous forme de logits ou probabilités.

Le MLP est entraîné avec un découpage rigoureux, utilisant une architecture adaptée (couches linéaires, ReLU, Dropout) et optimisé avec Adam et une fonction de perte CrossEntropy. Ce modèle est sauvegardé et ses sorties (softmax) sont générées sur tout le dataset.

c) Intégration dans la fusion tardive globale

Les sorties softmax du MLP booster sont ensuite **ajoutées** à la concaténation globale des logits de tous les meilleurs modèles texte et image.

Ainsi, la représentation finale d'entrée du méta-modèle regroupe :

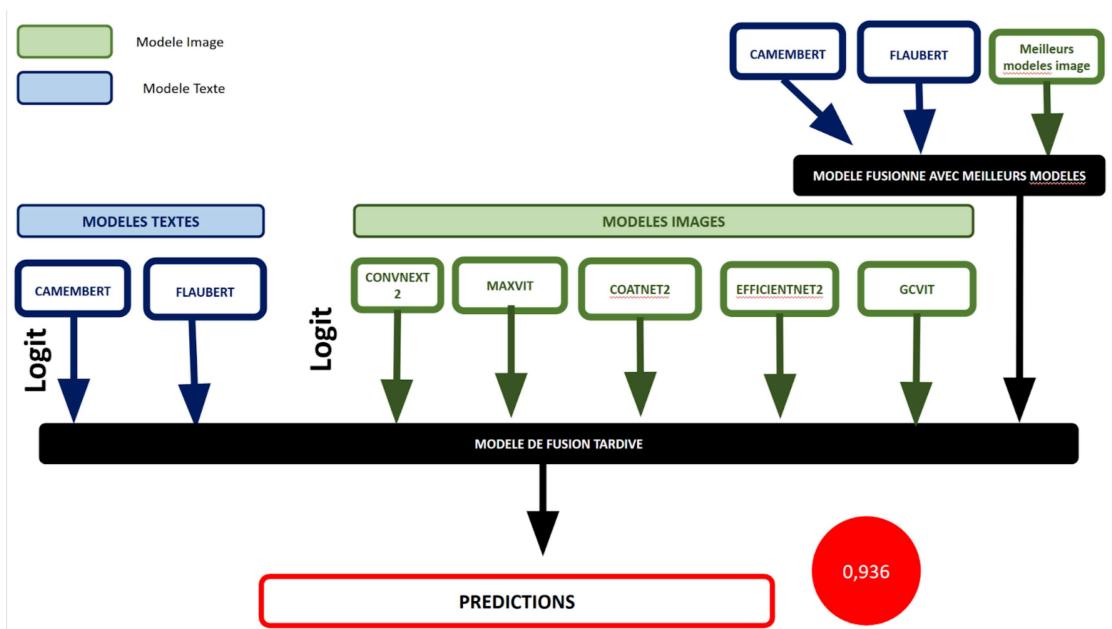
- Les logits individuels des modèles texte (CamemBERT, FlauBERT),
- Les logits individuels des modèles image (ConvNeXt2, EfficientNet2, Maxvit, CGViT...),
- **Les softmax issus du MLP de fusion**, qui apportent une synthèse boostée des informations multimodales.

d) Méta-fusion classique et optimisation

Sur cette représentation ultra-riche, nous entraînons une méta-fusion classique utilisant LightGBM et XGBoost. La combinaison pondérée de ces deux modèles est optimisée par validation croisée stratifiée sur un paramètre alpha, assurant un équilibre optimal entre leurs forces respectives.

Ce pipeline complet garantit :

- Une exploitation maximale de la richesse des données multimodales,
- Une modélisation fine des interactions complexes par le MLP booster,
- Une robustesse et une généralisation améliorées via la méta-fusion classique,
- Une évaluation rigoureuse grâce à la standardisation et au découpage identique.



Architecture multimodale



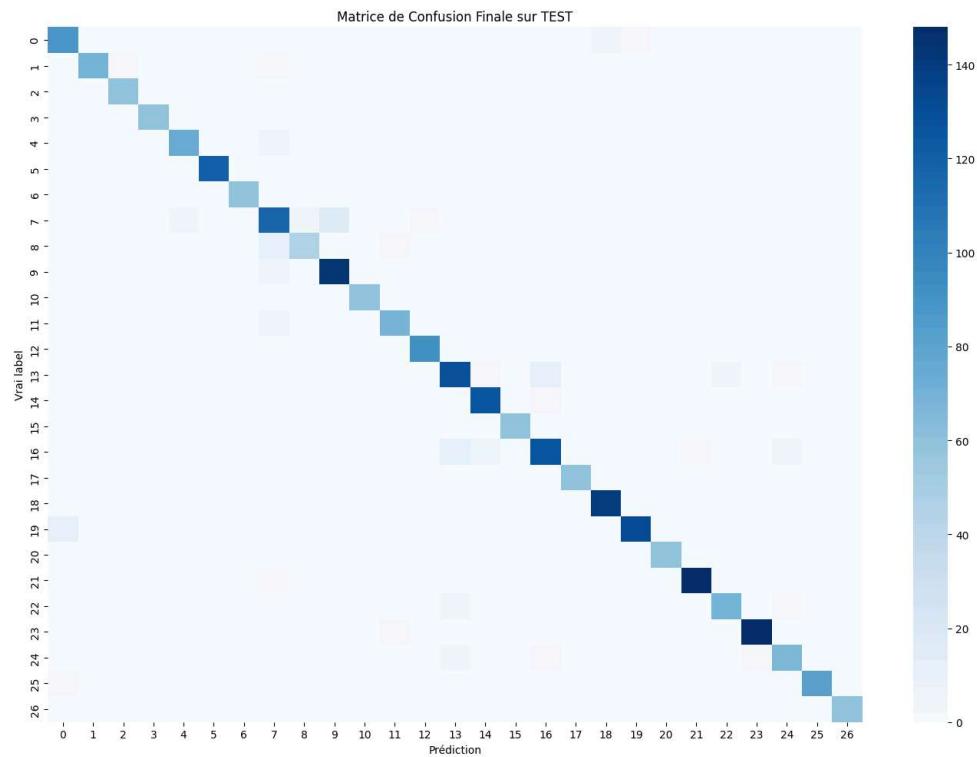
3. Résultats et perspectives

Cette architecture hiérarchique et modulaire nous a permis d'atteindre des performances de pointe dans le challenge, avec un F1-score pondéré 92,10%, nettement supérieur aux modèles "mono-modaux" et à la simple concaténation brute.

L'ajout du MLP booster est un élément clé, apportant une couche d'apprentissage supplémentaire capable d'exploiter la synergie entre texte et image avant la métá-fusion finale.

Cette stratégie est également très évolutive, permettant d'intégrer de nouveaux modèles ou modalités à l'avenir sans remettre en cause l'ensemble du pipeline.

Nous détaillons ci-dessous la démarche méthodologique, les architectures utilisées, les traitements des données et les résultats obtenus pour chacun de ces métá-modèles.



Matrice de confusion du modèle multimodal

```
[LightGBM] [Warning] No further splitting possible
[LightGBM] [Warning] No further splitting possible
Prédiction finale sur TEST...
/usr/local/lib/python3.11/dist-packages/lightgbm/_internal.py:105: UserWarning: warnings.warn(
Résultats finaux sur TEST :
Accuracy: 0.9353
F1 Macro: 0.9416
F1 Weighted: 0.9350
```

Meilleur score obtenu avec le modèle méta classifieur: 94%

Ce score sera toutefois ramené à 92,10% suite à la rationalisation de l'architecture et le renforcement de sa robustesse.

E. Interprétabilité

1. Modalité Texte

Pour mettre en évidence les éléments qui font la décision, dans le cadre de la modalité Texte, nous avons implémenté une interprétation locale à l'aide de la bibliothèque LIME, appliquée aux modèles FlauBERT et CamemBERT.

Nous avons collecté, sur un site de e-commerce analogue à Rakuten, la description de 3 produits différents vendus sur la plateforme représentant respectivement les catégories Livres, Jeux de société et Accessoires jeux vidéos, catégories qui existent au sein du set dont nous disposons pour le projet Rakuten.

Nos modèles ont donc été entraînés pour reconnaître la catégorie de ces types de produits.

Voici les résultats obtenus pour nos deux modèles et pour ces trois échantillons :

Interprétabilité			Prédiction FlauBERT		Prédiction CamemBERT	
Catégories	Description		Livres	eBooks	Jeux de société	Accessoires jeux vidéos
Livres	Transcendant les limites du temps et de l'espace, cette histoire vraie commença un 25 juillet 2022, lorsqu'Elena fut invitée par l'ancien membre d'équipage de Thor Han, Celadion, pour un voyage surprise vers Neptune qui se transforma en une aventure interdimensionnelle de deux mois et en une quête pour rencontrer les mystérieux « Architectes ». Ce livre est d'un niveau incroyable, émotionnel. Vous pleurerz une seconde, aurez la chair de poule l'instant d'après, serez submergé par l'amour et rirez aux éclats l'instant d'après.		Livres	eBooks	Jeux de société	Accessoires jeux vidéos
Jeux de société	JEU DE DEDUCTION ET DE MÉMOIRE : TRIO est un jeu de déduction malin et passionnant qui peut se jouer en solo ou en équipe !, Turtle Beach Rematch Core blanc/vert - Manette de jeu filaire - Licence Xbox Series XS, Xbox One et PC - touches audio, moteurs de vibration et gâchettes d'impulsion, câble de 3 m		Jeux de société	Jeux de société	Accessoires jeux vidéos	Accessoires jeux vidéos
Accessoires jeux vidéos						

Dans les résultats que produit LIME, il nous est possible d'observer plus précisément quels ont été les éléments déterminants. Voici comment les modèles se comportent pour nos trois échantillons :

CamemBERT	FlauBERT
Objet : Livre	
<pre>Top contributions LIME: livre → AIDE +0.2519 Ce → AIDE +0.2173 voyage → AIDE +0.1362 cette → AIDE +0.1356 histoire → AIDE +0.1335 instant → AIDE +0.1295 fut → AIDE +0.1165 amour → AIDE +0.1165 instant → AIDE +0.0992 deux → AIDE +0.0979</pre>	<pre>Top contributions LIME: livre → AIDE +0.3264 qui → AIDE +0.1011 temps → AIDE +0.1006 équipage → AIDE +0.0927 émotionnel → AIDE +0.0915 Ce → AIDE +0.0860 de → AIDE +0.0858 vraie → AIDE +0.0858 du → AIDE +0.0822 cette → AIDE +0.0805</pre>
Objet : Jeux de société	
<pre>Top contributions LIME: déduction → AIDE +0.0668 JEU → AIDE +0.0630 jeu → AIDE +0.0457 peut ← NUIT -0.0393 malin → AIDE +0.0355 DEDUCTION → AIDE +0.0314 équipe → AIDE +0.0236 TRIO ← NUIT -0.0221 jouer → AIDE +0.0204 solo ← NUIT -0.0188</pre>	<pre>Top contributions LIME: déduction → AIDE +0.2263 malin → AIDE +0.1708 jeu → AIDE +0.1243 de → AIDE +0.1157 JEU → AIDE +0.1103 un → AIDE +0.0871 DEDUCTION → AIDE +0.0767 TRIO → AIDE +0.0669 jouer → AIDE +0.0510 équipe → AIDE +0.0405</pre>

Objet : Accessoires jeux vidéos	
Top contributions LIME: 'Manette' → AIDE +0.2865 'jeu' → AIDE +0.2255 'Xbox' → AIDE +0.2171 'Beach' ← NUIT -0.1691 'touches' → AIDE +0.1508 'audio' → AIDE +0.1448 'XS' → AIDE +0.1268 'Xbox' → AIDE +0.0894 'et' → AIDE +0.0838 'Series' → AIDE +0.0785	Top contributions LIME: 'Manette' → AIDE +0.4282 'filaire' → AIDE +0.1489 'câble' → AIDE +0.1469 'vibration' → AIDE +0.1433 'Core' → AIDE +0.1265 'Xbox' → AIDE +0.1132 'Xbox' → AIDE +0.1081 'XS' → AIDE +0.1063 'PC' → AIDE +0.0985 'Turtle' → AIDE +0.0983

Si nous prenons l'exemple de l'objet Livre, on note que les deux modèles ont placé le token "livre" comme le plus déterminant pour la prise de décision. Mais si FlauBERT a répondu correctement, CamemBERT a identifié la catégorie eBook alors qu'il s'agissait bien d'un livre. Mais difficile, ici, de voir ce qui a provoqué cette erreur de classification.

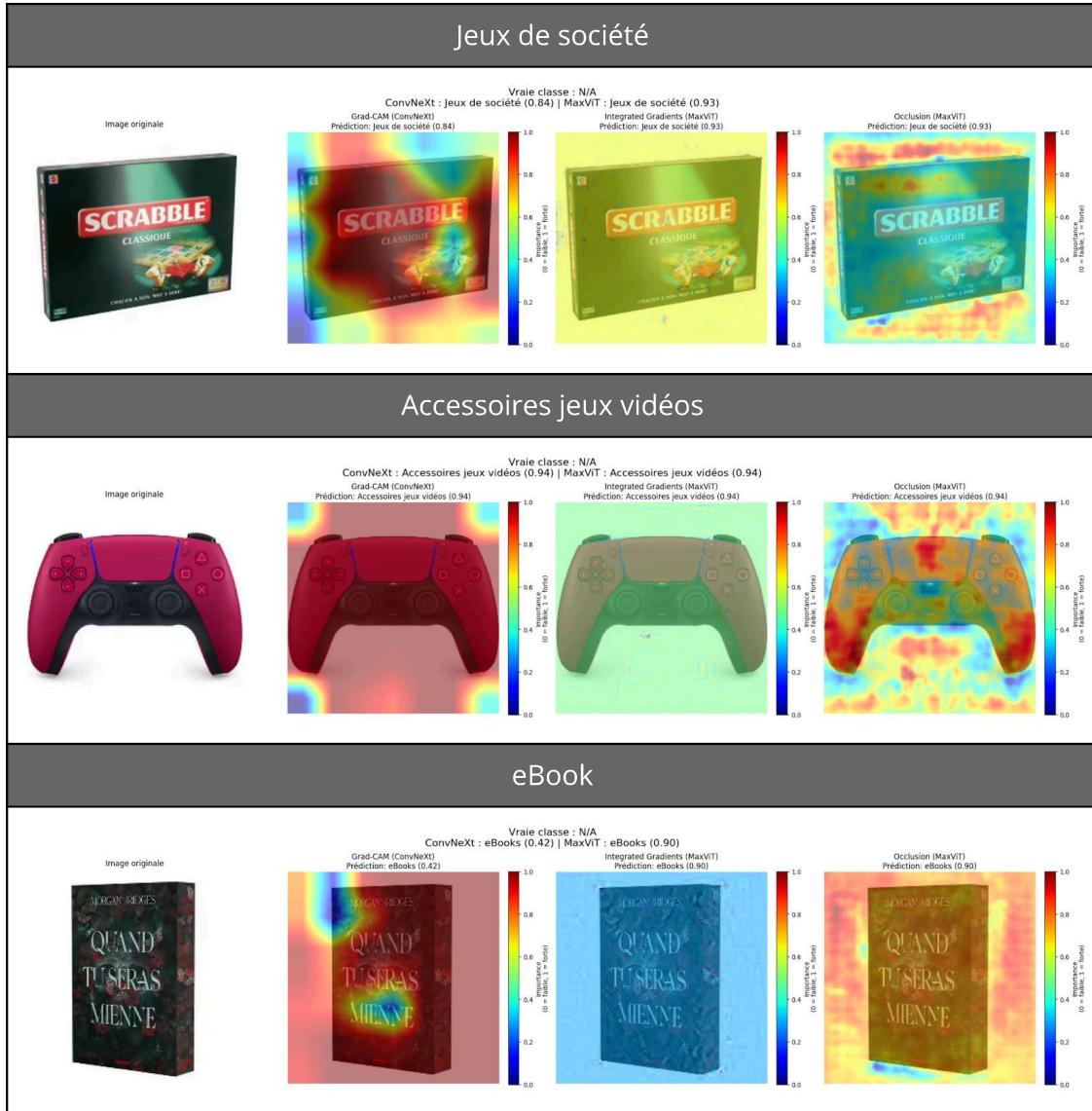
On note également, qu'à part le token "livre" pour lequel les deux modèles sont d'accord, les autres tokens retenus sont bien différents. Signe d'une complémentarité entre les deux modèles, qui explique certainement les meilleurs résultats obtenus par notre méta classifieur textuel combinant ces deux modèles !

2. Modalité image

S'agissant de l'interprétabilité de la modalité image, nous avons pris trois échantillons depuis le site de Rakuten, parmi les catégories jeux de société, accessoires de jeux vidéo et ebook.

Nous avons mis au banc d'essai les modèles convnext et maxvit selon 3 techniques différentes : Grad-CAM et sa carte de chaleur, Integrated gradients, qui matérialise dans l'image les pixels qui ont fait la décision et enfin, la technique de l'occlusion qui mesure l'impact dans la prédiction du masquage de différentes zones de l'image.

Voici, ci-après, les résultats produits par les modèles d'interprétabilité



Si nous analysons le dernier exemple, nous notons que Grad-CAM semble se focaliser principalement sur le titre, bien que le modèle montre une incertitude notable, avec une confiance assez faible (42%). S'agissant des deux autres techniques, la zone d'attention couvre l'ensemble de l'image. Aucune partie ne semble véritablement écartée et l'activation est assez diffuse, avec une confiance élevée.

Pour les 2 premiers exemples, la confiance de l'ensemble des modèles est très bonne et les techniques Grad-CAM et Occlusion semblent plus démonstratives.

VI. Conclusion

A. Retours d'expériences

En choisissant le projet Rakuten, nous avions tous anticipé la haute valeur ajoutée de ce sujet pour notre apprentissage. Et nous n'avons pas été déçus ! Mettre en œuvre les techniques de NLP, sur lesquelles reposent notamment les stars de l'IA que sont les chatGPT, Gemini, Claude, Mistral et tant d'autres, a été véritablement fascinant pour chacun de nous ! S'essayer aux technologies de computer vision et aux approches de transfert learning, de meta modeling, ajoutèrent à notre excitation !

Mais tout ceci a un prix : des heures et des heures de labeur, de la sueur, des larmes parfois (se rendre compte qu'on a oublié de sauver le best model sur son drive depuis la session Google Colab qui vient de tomber, après 18 epoch / 20 !) et du sang. Non, là c'était pour la formule :

Plus sérieusement, à travers le projet Rakuten, nous avons exploré une grande quantité d'approches, de bibliothèques, de modèles, nous nous sommes posés des centaines de questions, nous avons fait, défait, refait, expérimenté, épluché les sites spécialisés, avec, in fine, le sentiment d'avoir appris énormément et en même temps de n'avoir sondé qu'une infime partie de cet univers en perpétuel mouvement qu'est l'IA !

Si l'étude que nous livrons à travers ce projet nous rend assez fier, sa gestation n'aura pas été un long fleuve tranquille, loin s'en faut ! En effet, outre les contraintes et limites liées à l'exploitation de la plateforme Google Colab (quotas de ressources non garantis, déconnexion intempestives nécessitant sauvegardes intermédiaires, capacité de reprise sur crash, logs, ...), il a parfois fallu déployer des trésors d'ingénierie pour contourner les limites d'utilisation de certaines bibliothèques (notamment *deep translator*) et maîtriser la stabilité durant l'exploitation. Extrait de contraintes et des solutions de contournement :

- interdiction des robots, impliquant de simuler,
- taille max de l'entrée à traduire, nécessitant un découpage des entrées en blocs avec itérations,
- chasse aux NaN dans les entrées, qui impliquait un plantage systématique de la traduction.

La quête de la performance absolue a, elle aussi, produit son lot de difficultés et frustrations :

- Des ajustements de pipeline et d'hyper paramètres qui, contre toute attente, minorent les performances (standardisation, équilibrage des classes avec SVM, ...). Nous comprendrons par la suite que certaines de ces techniques sont moins adaptées au deep learning,
- L'utilisation de techniques d'entraînement nécessitant des jours de calculs (GridSearchCV, BayesSearchCV), avec des plantages à la clé, sans résultats ...
- Plus largement, entre la diversité des modèles testés, des matrices d'hyper paramètres, des différentes approches de fusion des logits de modèles, ce sont des centaines de combinaisons qui ont été explorées se traduisant par des jours de calculs CPU et GPU, au bilan carbone peut flatteur, il est vrai,
- La complexité de la solution livrée avec ses multiples itérations sur 3 méta classifieurs et tuning Optuna délicat pour chacun d'eux,
- La découverte tardive de la nécessaire harmonisation des sorties des modèles combinés (certains permettant de capturer les logits quand d'autres sortent les probas), obligeant à reprendre une partie du pipeline,

Ceci ne constituant qu'un extrait des difficultés rencontrées, la liste exhaustive relevant de la gageure. Mais ces frustrations, échecs ou difficultés sont, à n'en pas douter, à la source du terreau fertile de nos enseignements !

B. Applications métier

1. Trouver !

La solution de classification multiples à deux modalités, que nous avons étudiée dans le cadre de ce projet, traite d'une problématique clé pour une plateforme BtoC telle que Rakuten !

En effet, le succès de ces plateformes, et la croissance de leur chiffre d'affaires et de celui de leurs vendeurs, est étroitement lié à la capacité des acheteurs à trouver les produits qu'ils convoitent ! Si les produits sont bien "rangés", bien organisés au sein d'une arborescence de catégories claires et bien distinctes, c'est déterminant pour accompagner efficacement la recherche de produits pour les clients potentiels !

Les catégories peuvent être comparées aux rayons d'une grande surface, qui organisent les produits pour faciliter leur recherche par les clients. Qui ne s'est jamais retrouvé à faire des vas et viens d'un rayon à l'autre sans trouver le produit convoité et fini par renoncer à l'acheter !

Ainsi, associer un produit à la bonne catégorie, c'est favoriser la recherche par le client et la vente pour le vendeur tiers.

2. Une catégorisation de qualité !

Nous avons pu le constater dans le jeu de données, la qualité des annonces de produits est très variable. Certaines annonces sont richement décrites, accompagnées d'une photo soignée et sans ambiguïté, quand d'autres se contentent d'un titre sommaire, tout au plus quelques mots, sans description et d'une photo grossière, pixelisée ou encore mal cadrée ou surexposée.

Ces entrées, non structurées et non homogènes rendent difficiles à priori toute catégorisation pertinente !

Ajoutons à cela la pluralité des langues dans lesquelles lesannoncent peuvent être rédigées.

La solution offre ici une réponse efficace en assistant l'enregistrement de l'offre par le vendeur en lui proposant la catégorie la plus probable pour son produit. La puissance du modèle, entraîné sur un jeu de données conséquent, tant sur la modalité textuelle que sur l'image, peut ainsi aider à sélectionner la bonne catégorie en préconisant celle qui est la plus adaptée en fonction de l'observation des entrées fournies par le vendeur.

L'architecture de notre modèle est, en outre, capable de traiter la plupart des langues en entrée sans compromettre la performance des prédictions. Ceci via le choix d'une traduction systématique en langue française sur laquelle les modèles sont spécialisés !

3. Navigation assistée

Par sa capacité à traiter la modalité textuelle, la solution peut proposer au client, lors de sa recherche par mots clé, une sélection des catégories les plus probables associées, lui permettant de resserrer sa recherche et ainsi trouver plus rapidement les produits recherchés.

C. Et après

1. Optimisation de la performance

a) Par la technique

Bien qu'étant déjà d'un bon niveau, il apparaît possible de pousser encore le score des modèles livrés, notamment au niveau de l'entraînement du méta modèle. En effet, actuellement, le script utilise une recherche par grille pour tester toutes les combinaisons de modèles. Il serait possible de remplacer cette approche par un stacking direct où tous les logits des modèles de base sont utilisés comme features, avec des hyper paramètres optimisés par Optuna. Nous n'avons hélas pas eu le temps de tester cette dernière approche.

b) Par une qualité accrue du référencement

Le référencement d'un produit pourrait prévoir d'intégrer plusieurs images (le produit pris sous différents angles, des mises en situation, ...) qui enrichirait les données d'entrées pour l'entraînement des modèles images et favoriserait la performance de classification.

2. Expérience client

Une autre façon d'exploiter la capacité de reconnaissance du modèle image serait d'offrir au client potentiel, la possibilité de soumettre (ou prendre en live) une photo d'un produit. Le modèle analyserait l'image fournie, en déduirait sa catégorie et proposerait en retour l'ouverture de la page d'accueil de la catégorie et/ou des produits similaires recommandés.

3. Transposition métier

Enfin, la solution produite pourrait aisément être transposée, dans son exploitation, à d'autres domaines métiers. On peut citer, comme exemple :

- La dermatologie : une photo d'un bouton suspect enrichie d'un descriptif à caractère médical (description symptômes, antécédents, ...) permettrait une classification en différents niveaux d'alertes.
- Industrie / contrôle qualité : à partir de la photo d'une pièce, de son descriptif et des côtes de tolérance, la pièce serait mise au rebut ou non,
- Bâtiment : classification de matériaux à partir d'une photo et/ou d'une description,
- etc ...

Bien évidemment, ces domaines d'applications nécessiteraient de réentraîner les modèles sur les jeux de données adaptés, en transfert learning.

VII. Bibliographie

[BERT, flauBERT, camemBERT](#) : la base de nos modèles textuels

[Hugging Face](#) : comment ne pas le citer pour son immense contribution à notre projet !

[Medium](#) : un portail centré sur l'IA avec une mine d'information

[geeksforgeeks](#) : un autre portail avec cours et tutos sur la Data Science

[Arxiv](#) : fourni de nombreuses publications scientifiques notamment en science de la données

[StackOverFlow](#) : l'incontournable de tout bon développeur en galère

[Google Colab](#) : une plateforme salutaire pour faire ses griffes !

[Fusion de modèle](#)

[Planifier le taux d'apprentissage](#) dans PyTorch

[Optuna](#) pour l'optimisation automatique des [hyperparamètres](#)