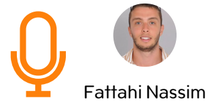
Rakuten Aug - 2024

Houda Ouazzani

Virgile Oger

Fattahi Nassim

Francis Guede



Fattahi Nassim

Sommaire

Fattahi Nassim

Introduction

Application de classification automatique multimodale des produits Rakuten

Contexte :

- **L**a catégorisation automatique des produits représente un défi majeur pour les plateformes de e-commerce
- Rakuten doit gérer efficacement des millions de produits dans son catalogue
- L'innovation réside dans l'utilisation combinée du texte ET de l'image pour une classification plus précise

Problématique

Comment développer un système de classification automatique qui exploite simultanément :

- Les descriptions textuelles des produits
- Les images associées
- Pour catégoriser précisément parmi 27 catégories distinctes

Objectifs du projet

Développer une solution basée sur l'IA combinant :

- Modèles classiques de Machine Learning
- Techniques avancées de Deep Learning

Explorer et optimiser différentes approches :

- Évaluer les performances des modèles traditionnels
- Adapter le modèle CLIP pour notre cas d'usage
- Comparer les résultats des différentes approches

Créer une application performante permettant :

- La classification en temps réel de nouveaux produits
- Une interface utilisateur intuitive
- Une visualisation claire des résultats

Francis Guede

Presentation du jeu de données 1/4

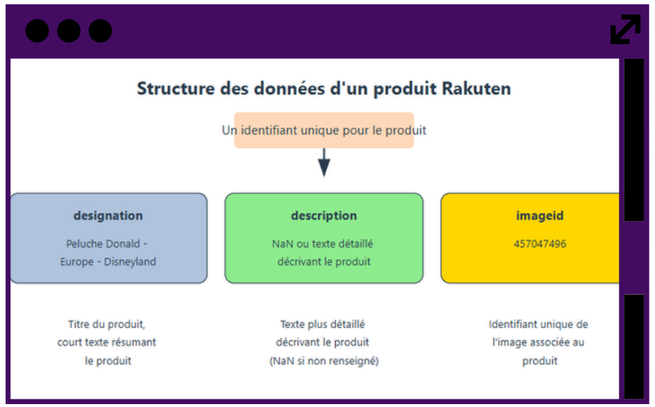
Vue d'ensemble

Dataset total: ~99K produits

- Ensemble d'entraînement: 84 916 produits
- Ensemble de test: 13 812 produits

Structure des données

- Données textuelles:
- Désignation: titre court du produit
- Description: texte détaillé (présent pour 64.9% des produits)
- Données visuelles:
- Images des produits
- Caractéristiques variables:
  - Tailles différentes
  - Qualité hétérogène
  - Angles de vue divers
  - Conditions d'éclairage variées



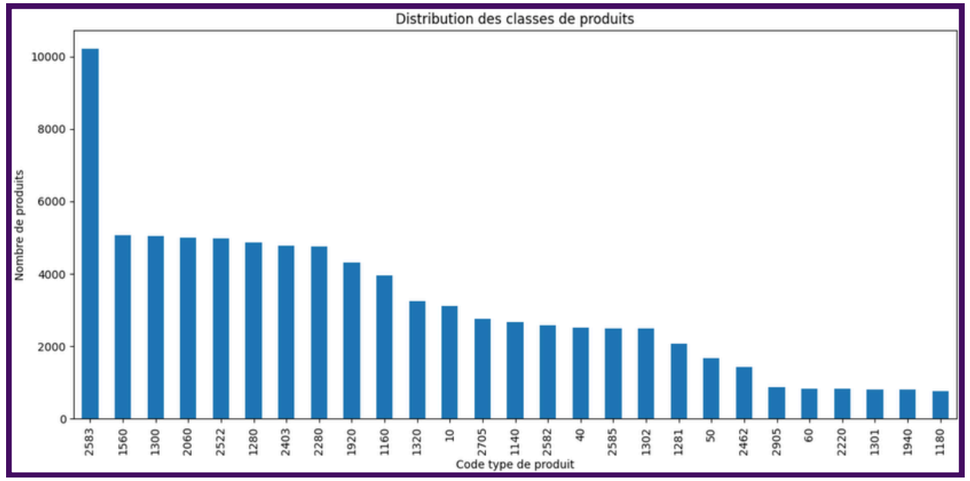
Francis Guede

Presentation du jeu de données 2/4

Distribution des Catégories

27 catégories distinctes couvrant divers domaines:

- Livres et presse
- Jeux vidéo et consoles
- Jouets et jeux
- Maison et jardin
- Et plus encore...



Points clés sur la distribution

- Distribution déséquilibrée
- Certaines catégories > 10 000 produits
- D'autres catégories < 1 000 produits

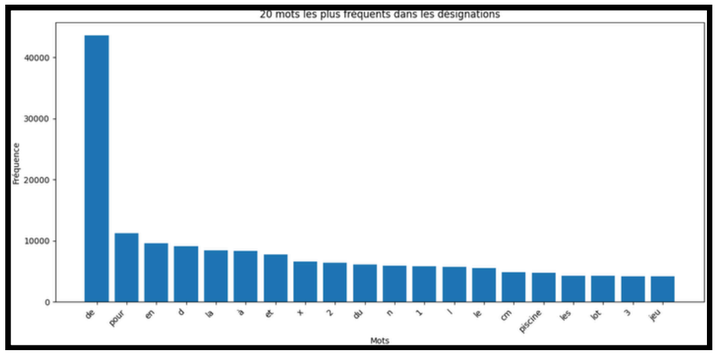
Francis Guede

Presentation du jeu de données 3/4

Particularities des données

Textuelle :

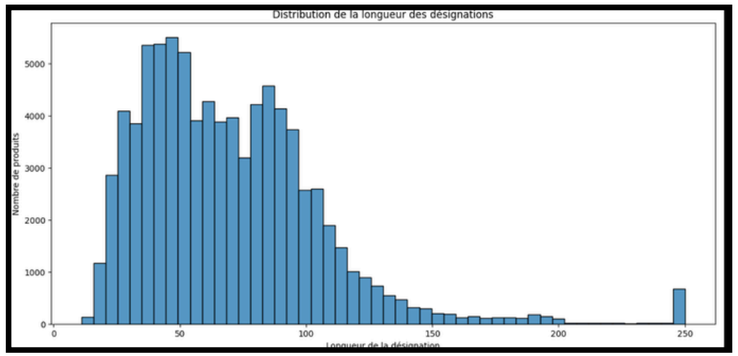
- Multilingue (français, anglais principalement)
- Longueur variable des désignations (25-100 caractères)
- 35% des produits sont sans description



L'analyse des 20 mots les plus fréquents dans les désignations montre :
Les mots les plus courants sont des articles et prépositions (par exemple, "de", "pour", "en")
Des termes spécifiques comme "jeu", "cm", "pièce" apparaissent fréquemment.

Images:

- Grande diversité de styles
- Qualité variable
- Multiple angles et présentations



L'analyse de la longueur des désignations de produits révèle : Une distribution asymétrique vers la droite La plupart des désignations ont entre 25 et 100 caractères Quelques désignations très longues (>200 caractères) sont présentes.

Francis Guede

Presentation du jeu de données 4/4

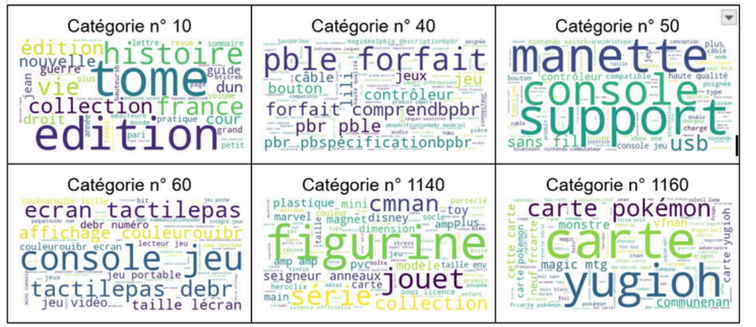
Analyse par categorie de produit

L'analyse détaillée des mots fréquents par catégorie de produit révèle :

- Des termes spécifiques à chaque catégorie (ex: "piscine" pour la catégorie 2583)
- Des variations de longueur de désignation selon les catégories.

Frequence de mots par catégories produtis

Les nuages de mots par catégorie montrent bien que certains mots sont plus présents dans les catégories respectives, ce qui permet d'identifier visuellement la type de produits correspondants.



Virgile Oger

Prétraitement des données et première approche NLP

Notre démarche initiale

Dans un projet de classification, la qualité du prétraitement est souvent aussi importante que le choix du modèle. Nous avons donc accordé une attention particulière à cette étape cruciale.

Prétraitement du texte

La diversité linguistique et la variabilité des descriptions produits nécessitent un prétraitement robuste. Notre pipeline transforme les textes bruts en données exploitables par les modèles.

Pipeline de prétraitement textuel :

- Nettoyage basique Conversion en minuscules pour standardiser
- Suppression des caractères spéciaux pour réduire le bruit
- "Cette première étape élimine les variations non significatives"

```python
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)
```

Ce code illustre notre approche de normalisation textuelle

Gestion des cas particuliers

Challenge des données manquantes

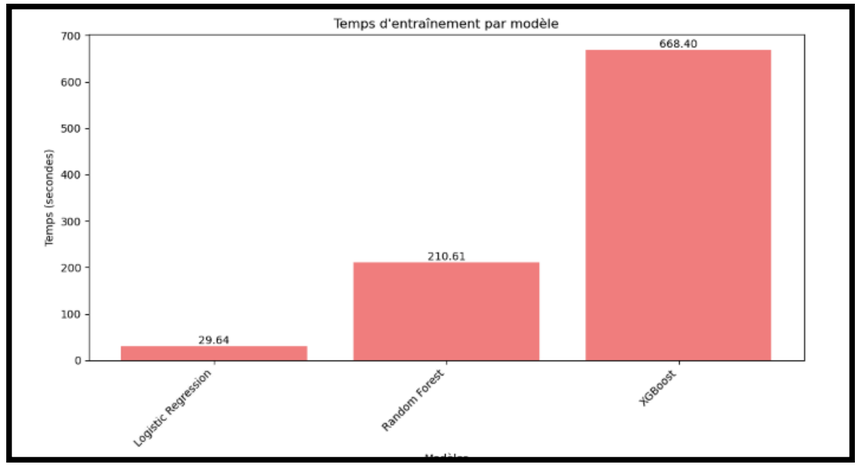
35.1% des produits n'ont pas de description détaillée. Pour maintenir la cohérence, nous avons développé une stratégie de substitution intelligente.

Notre solution :

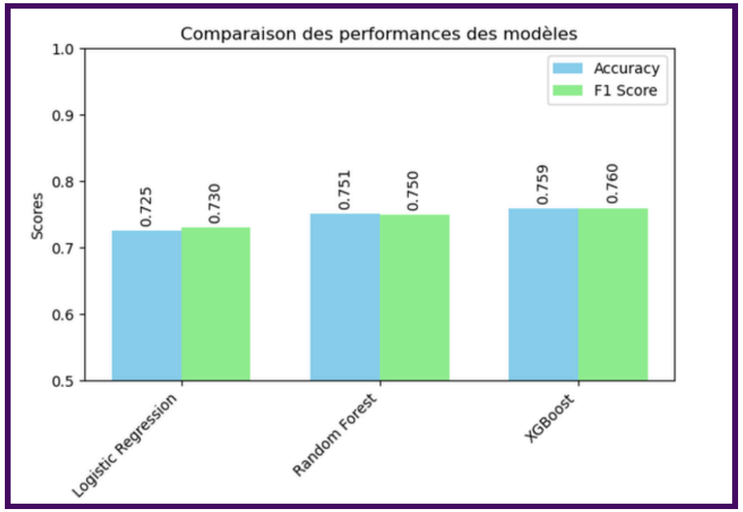
- Utilisation de la désignation comme substitut
- Cette approche permet de ne pas pénaliser les produits sans description tout en conservant l'information essentielle

Virgile Oger

Prétraitement des données et première approche ML Graph



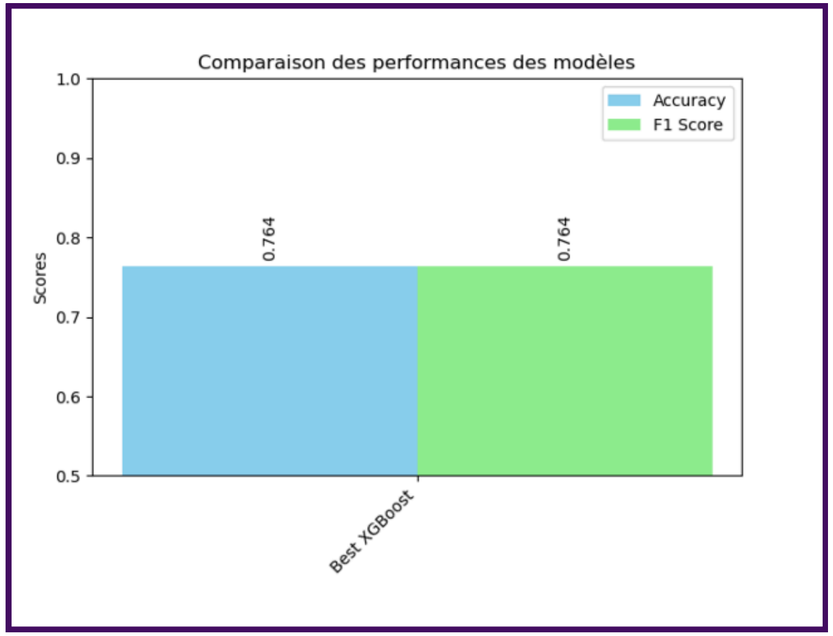
- On observe une augmentation significative du temps d'entraînement entre les modèles, allant de ~30 secondes à plus de 600 secondes
- Le troisième modèle XGBOOST nécessite beaucoup plus de temps d'entraînement (644.43s) que les deux autres, suggérant une complexité nettement supérieure



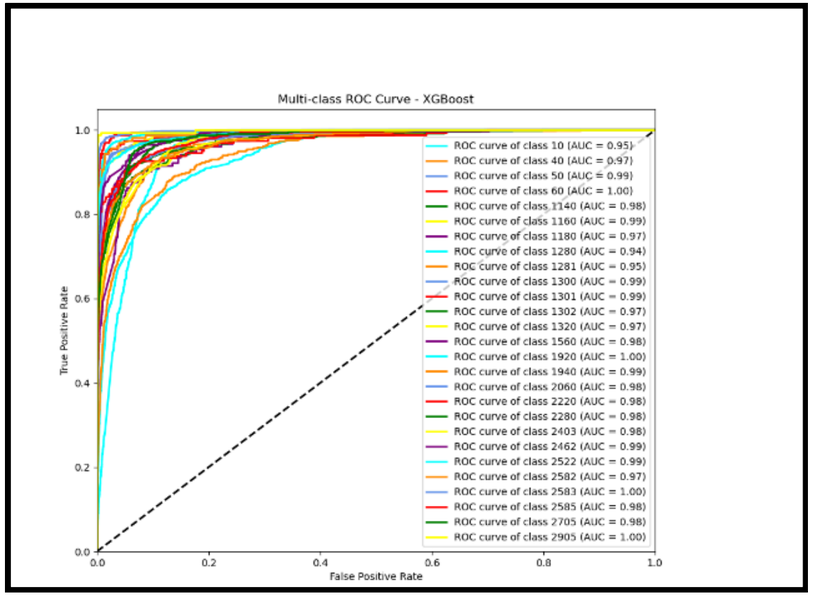
- Les trois modèles (Logistic Regression, Random Forest et XGBoost) montrent des performances similaires avec des scores autour de 0.75
- XGBoost performe légèrement mieux avec une accuracy de ~0.76, suivi de près par Random Forest et Logistic Regression

Virgile Oger

Prétraitement des données et première approche ML Graph



- L'optimisation par GridSearch avec XGBoost atteint une accuracy de 0.744 et un F1-score de 0.754
- Les performances sont relativement similaires à celles du XGBoost standard (vu dans le graphique du milieu), suggérant que l'optimisation des hyperparamètres n'a pas apporté d'amélioration significative dans ce cas



Virgil Oger

Première approche : Machine Learning classique

Pourquoi commencer par le ML classique ?

Avant de nous lancer dans le Deep Learning, nous avons voulu établir une baseline solide avec des modèles traditionnels. Cette approche nous permet de :

- Comprendre les défis spécifiques du dataset
- Établir des métriques de référence
- Identifier les points d'amélioration potentiels

Résultats et enseignements:

Ce que nous avons appris,
Chaque modèle nous a apporté des insights différents :

De la régression logistique

Les features textuelles sont discriminantes
Importance de la normalisation
Limites des approches linéaires

Du Random Forest

Certaines catégories sont naturellement plus difficiles à prédire
Importance des interactions entre features
Nécessité d'une approche plus sophistiquée

De XGBoost

Amélioration significative mais encore insuffisante
Plafonnement des performances
Signal clair pour explorer le Deep Learning

Modèles testés et justification

Régression logistique "Modèle simple mais efficace comme baseline"

- Avantages :
  - Rapidité d'entraînement
  - Facilité d'interprétation
  - Bon point de départ pour comprendre les données

Random Forest "Une évolution naturelle vers plus de complexité"

- Avantages :
  - Rapidité d'entraînement
  - Facilité d'interprétation
  - Bon point de départ pour comprendre les données

XGBoost "L'Etat de l'art du ML Classique "

- **Avantages :**
  - Rapidité d'entraînement
  - Facilité d'interprétation
  - Bon point de départ pour comprendre les données

Houda Ouazzani

Approche Deep Learning avec CLIP

Pourquoi passer au Deep Learning ?

Les limites des approches classiques nous ont poussé vers une solution plus sophistiquée

Limites rencontrées

- Difficulté à capturer la sémantique complexe des descriptions
- Performance insuffisante sur les données visuelles
- Besoin d'une meilleure généralisation

CLIP : Une solution innovante pour notre problème !

Qu'est-ce que CLIP ?

CLIP, développé par OpenAI, est une avancée majeure dans le traitement multimodal. Son nom signifie 'Contrastive Language-Image Pre-training', ce qui reflète sa capacité à comprendre simultanément le texte et les images.

Architecture révolutionnaire

Imaginez CLIP comme un traducteur universel qui parle à la fois le langage des images et celui du texte. Il utilise deux encodeurs spécialisés :

- Un encodeur de texte basé sur un transformer
- Un encodeur d'image basé sur une architecture Vision Transformer (ViT)

Ces deux encodeurs travaillent en tandem pour créer une compréhension unifiée du produit.

Houda Ouazzani

Pourquoi CLIP est parfait pour notre cas ?

Apport technique de CLIP pour notre classification

1. Architecture d'encodage avancée

L'architecture de CLIP apporte des avantages techniques cruciaux :

- Encodeur Vision Transformer (ViT) :
  - Découpage de l'image en patches de 32x32 pixels
  - Attention multi-têtes pour capturer les relations spatiales
  - Capacité à gérer des images de tailles variables (224x224 à 288x288)

```
# Configuration de notre ViT
patch_size = 32
hidden_dim = 768
num_heads = 12
```

2. Fusion multimodale optimisée

Notre implémentation exploite la fusion des embeddings :

```
def forward(self, input_ids, attention_mask, pixel_values):
    # Extraction des embeddings texte et image
    outputs = self.clip(
        input_ids=input_ids,
        attention_mask=attention_mask,
        pixel_values=pixel_values
    )
    # Fusion par concatenation
    pooled_output = torch.cat(
        [outputs.text_embeds, outputs.image_embeds],
        dim=1
    )
    # Dimension résultante : 2 * 512 = 1024
```

3. Prétraitement textuel:

```
processor_config = {
    'max_length': 77,  # Longueur maximale CLIP
    'truncation': True,
    'padding': True,
    'return_tensors': 'pt'
}
```

Cette configuration permet :

- Tokenization efficace des descriptions multilingues
- Padding/truncation automatique
- Normalisation des séquences

4.Prétraitement image

```
transforms = A.Compose([
    A.RandomRotate90(),
    A.Flip(),
    A.OneOf([
        A.GaussNoise(var_limit=(10.0, 50.0)),
        A.MultiplicativeNoise(multiplier=(0.9, 1.1))
    ], p=0.2),
    ToTensorV2()
])
```

Cette configuration permet :

- Rotations et retournements pour varier les orientations.
- Ajout de bruit et modifications d'éclairage pour simuler des imperfections.
- Conversion de l'image en tensor pour être utilisable avec PyTorch.

Houda Ouazzani

Notre Model Avec Clip

Optimisation des performances

Hyperparamètres critiques : Nos tests avec Optuna ont révélé des valeurs optimales précises.

```
best_params =
  'lr': 3.9205620014637784e-05,
  'dropout_rate': 0.29176375000178967,
  'hidden_size': 256,
  'batch_size': 64
```

- Learning rate précis pour éviter l'instabilité
- Dropout calibré pour la régularisation
- Taille de batch optimisée pour notre GPU Nvidia A100 google colab

```
optimizer = torch.optim.AdamW(
    model.parameters(),
    lr=best_params['lr'],
    weight_decay=0.01
)

scheduler = get_linear_schedule_with_warmup(
    optimizer,
    num_warmup_steps=0,
    num_training_steps=total_steps
)
```

Gestion de la mémoire

Batching efficace : Notre custom collate function optimise le traitement batch

```
def custom_collate(batch):
    batch_dict = {key: [] for key in batch[0].keys()}
    for b in batch:
        for key in batch_dict:
            batch_dict[key].append(b[key])

    # Padding dynamique pour optimiser la mémoire
    for key in ['input_ids', 'attention_mask']:
        batch_dict[key] = torch.nn.utils.rnn.pad_sequence(
            batch_dict[key],
            batch_first=True
        )
```

Adapation du Model pour Clip

En resumé : Nous n'avons pas utilisé CLIP tel quel. Nous l'avons adapté spécifiquement pour notre cas d'usage :

- Ajout de couches de classification personnalisées
- Optimisation pour nos 27 catégories spécifiques
- Intégration de mécanismes de régularisation pour éviter le surapprentissage

Fine-tuning intelligent

Notre approche de fine-tuning est similaire à l'apprentissage d'une nouvelle langue :

- On garde les connaissances de base de CLIP (comme le vocabulaire de base)
- On ajoute notre 'vocabulaire' spécifique à Rakuten
- On ajuste la 'grammaire' pour notre cas d'usage particulier

Fattahi Nassim

Synthèse des Performances

Évolution globale

Notre modèle CLIP adapté a montré une progression remarquable sur le dataset Rakuten. L'entraînement a révélé une amélioration constante des performances :"

- F1-Score : 0.49 → 0.97 (entraînement)
- Validation : jusqu'à 0.86
- Loss : 2.19 → 0.23

Efficacité opérationnelle

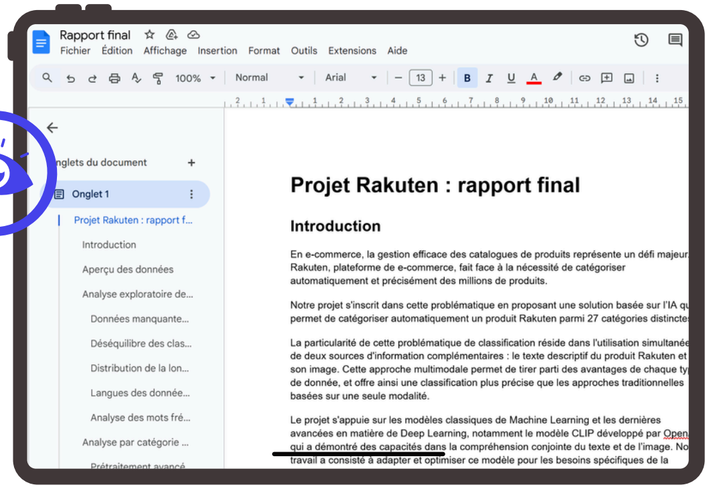
Le modèle démontre une excellente efficacité en production :"

- Temps moyen par itération : 1.95s
- Convergence stable
- Excellent équilibre apprentissage/généralisation

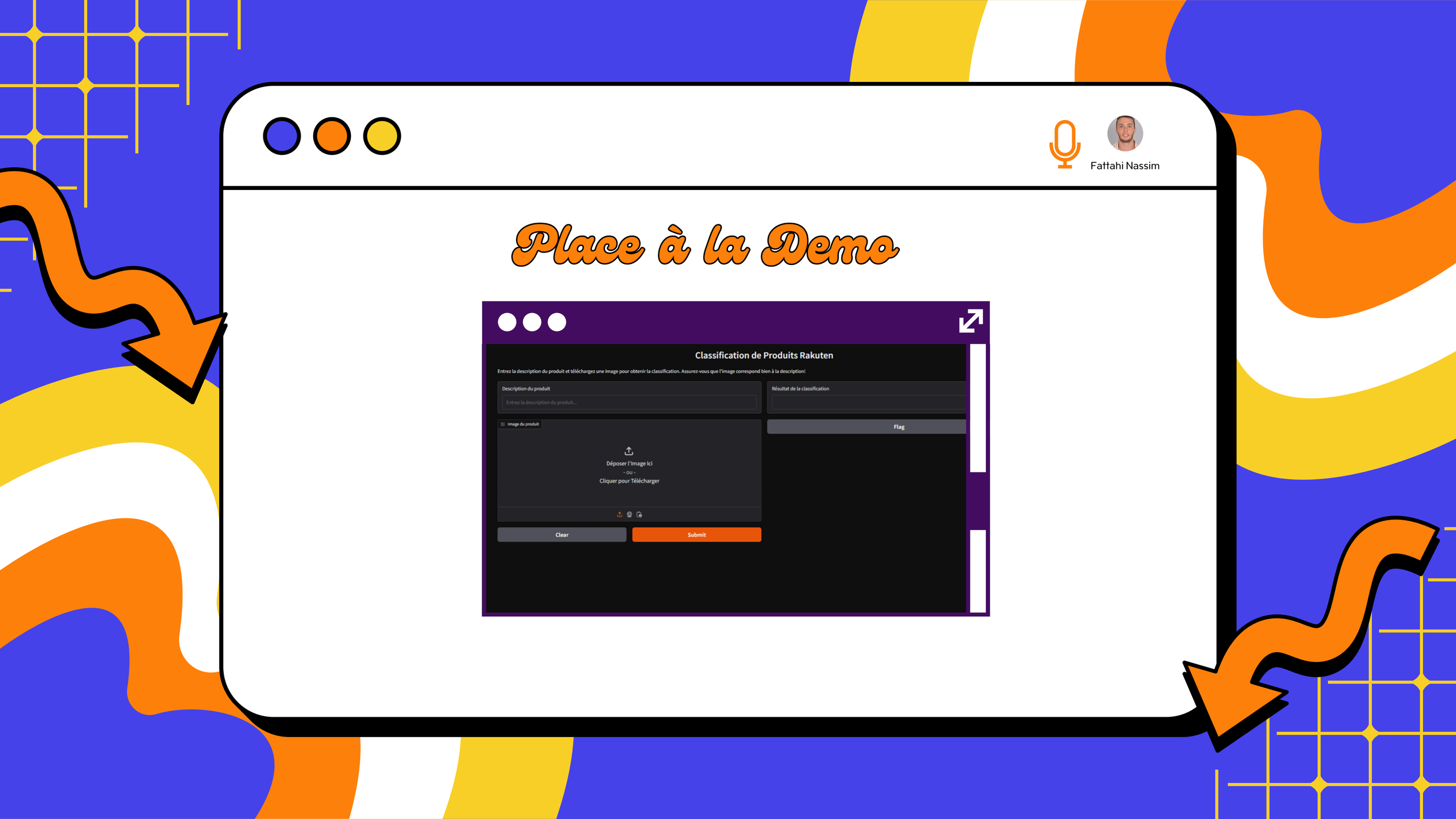
Configuration optimale

Ces résultats ont été obtenus grâce à une configuration fine des hyperparamètres :"

- Learning rate : 3.92e-05
- Dropout rate : 0.29
- Hidden size : 256
- Batch size : 64



NB : Pour une analyse approfondie des résultats et visualisations détaillées, se référer au rapport technique complet.


Fattahi Nassim
Place à la Demo
Classification de Produits Rakuten
Entrez la description du produit et téléchargez une image pour obtenir la classification. Assurez-vous que l'image correspond bien à la description!
Description du produit
Entrez la description du produit...
Résultat de la classification
Image du produit
Flag
Déposer l'image ici
- ou -
Cliquer pour Télécharger
Clear
Submit

Fattahi Nassim

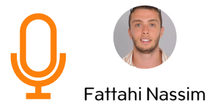
Conclusion

Les perspectives d'amélioration sont nombreuses :

- Extension à plus de catégories de produits
- Intégration d'un système de feedback utilisateur pour l'apprentissage continu
- Adaptation à d'autres langues et marchés
- Développement d'un système de recommandation basé sur notre architecture

Ces évolutions permettraient d'étendre l'utilité du système au-delà de la simple classification, vers une solution complète de gestion de catalogue e-commerce.

sys.exit()



Fattahi Nassim

Merci !