Projet 6

Classifiez automatiquement des biens de consommation

Camille BRODIN

Problématique / Données / Modélisations / Conclusions

Problématique

Missions confiées par Place de marché

Base de données :

- Données sur 1050 produits avec photos associées.
- **Table au format .csv et images au format .jpg.**

Trois missions :

- Etude de faisabilité d'un moteur de classification d'articles.
- Réaliser une classification supervisée.
- Tester une API pour élargir la gamme de produits.

Données : Nettoyage

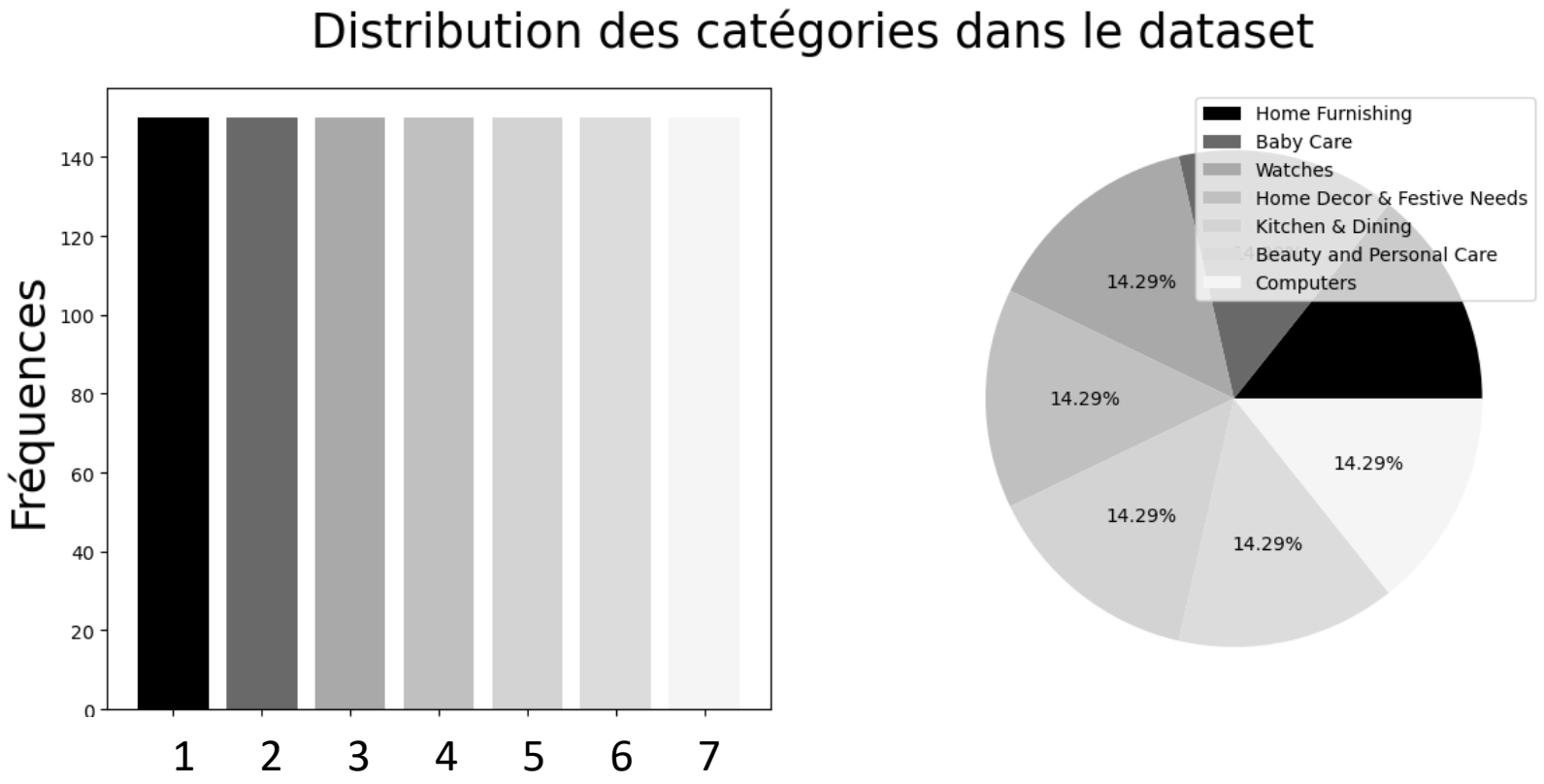
Etapes	Méthodes	Justifications
1. Suppression des champs inutiles	data.drop(columns=[“x”])	'crawl_timestamp’, 'product_url’, 'pid’, 'retail_price’, 'discounted_price’, 'is_FK_Advantage_product’, 'product_rating’ 'overall_rating’, 'product_specifications’
2. Vérifications doublons et NaN	data[‘brand'].fillna(“ ")	Pas de « doublons », imputation NaN brand avec « »
3. Traitement des catégories - Séparation de l’arborescence - Fusion des descriptions	'["Watches >> Wrist Watches >> Maserati Time Wrist Watches"]’ : def get_categories_from_cell avec cell.strip('[]“’) et split(" >> ") corpus = data['descriptions'] = data['product_name'] + ' ' + data['description'] + ' ' + data['brand']	Intérêt de travailler avec 1ere catégorie à 7 modalités Intérêt d’avoir un corpus riche (description générale mais précise des produits)

unique_id	image	categorie1	descriptions	brand	product_name
identifiant	nom du fichier	categorie generale	descriptif	marque	nom du produit
object	object	object	object	object	object

➢ **1050 lignes et 15 variables ->1050 et 3 sur le jeu de données nettoyé (+ 1 variable ID)**

Données : Présentation et exploration

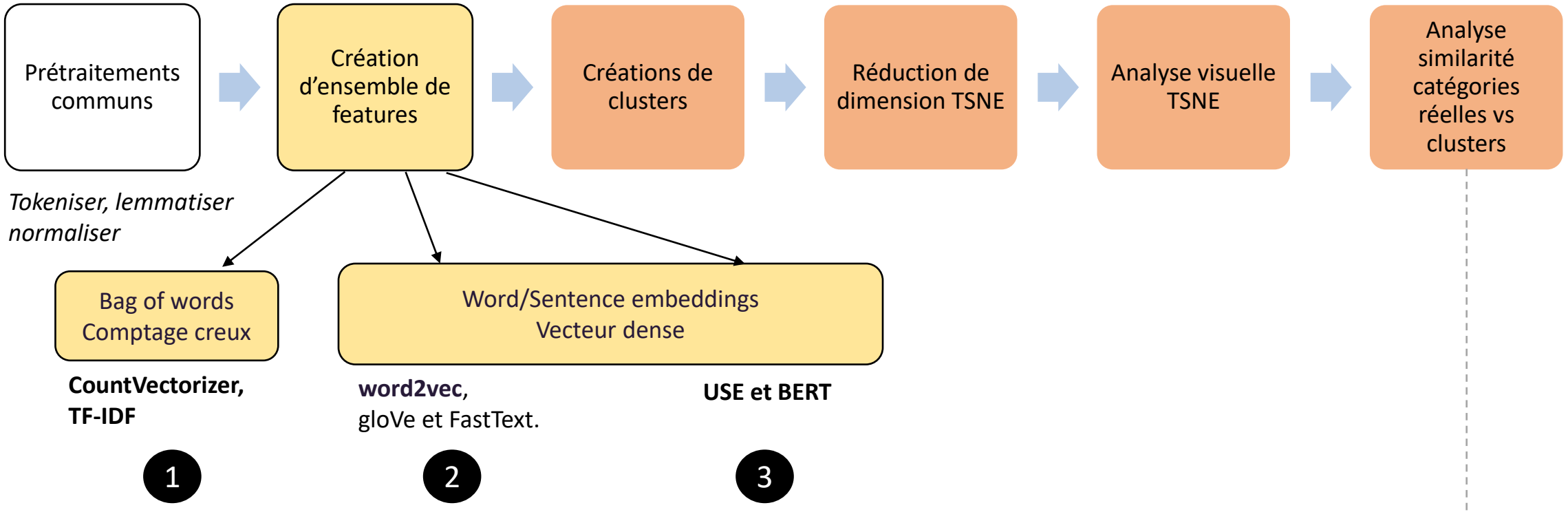
1. **Home Furnishing**
2. **Baby Care**
3. **Watches**
4. **Home Decor & Festive Needs**
5. **Kitchen & Dining**
6. **Beauty and Personal Care**
7. **Computers**



➢ **150 produits dans chacune des 7 catégories principales -> labels à prédire**

Données : Etapes réalisées pour l'étude de faisabilité

❖ **Données textes :**

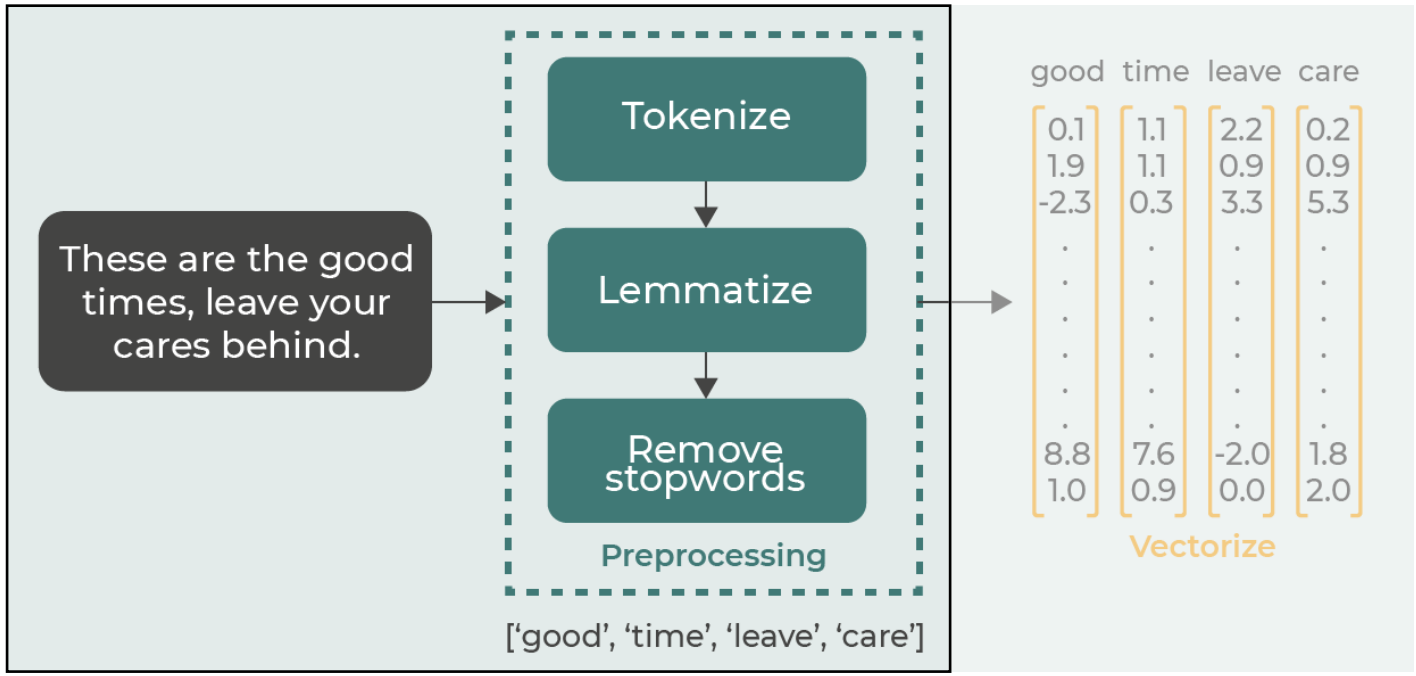


➢ **Métriques adaptées à la problématique métier =**

L'ARI, indice proportionnel au nombre de paires d'échantillons dont les étiquettes sont les mêmes + temps d'ajustement.

Données textes: Prétraitements communs

❖ **0. Prétraitement des données textuelles :**



Exemple d'une phrase traitée : 'camerii wm64 elegance analog watch for men boys'

Données textes : Extraction de features « bag of words » -> count et Tf-idf

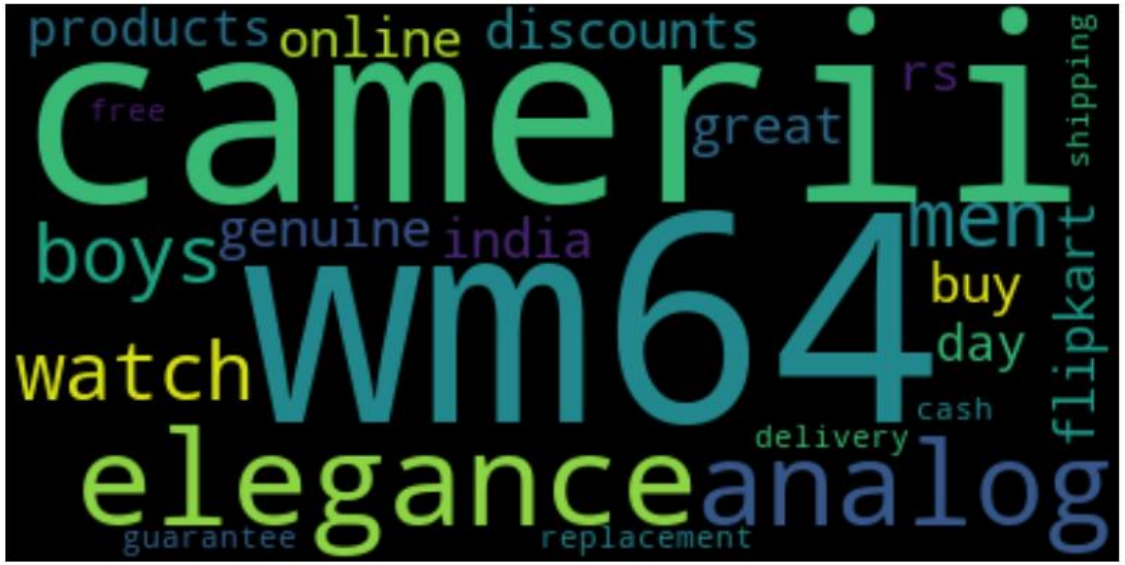
❖ **1. Extraction de features « bag of words » -> count et Tf-idf**

-> Générer des features (matrice creuse) à partir de mots

CountVectorizer()

(1050, 5324)

ARI : 0.4683
time : 11.0



TfidfVectorizer()

(1050, 5324)

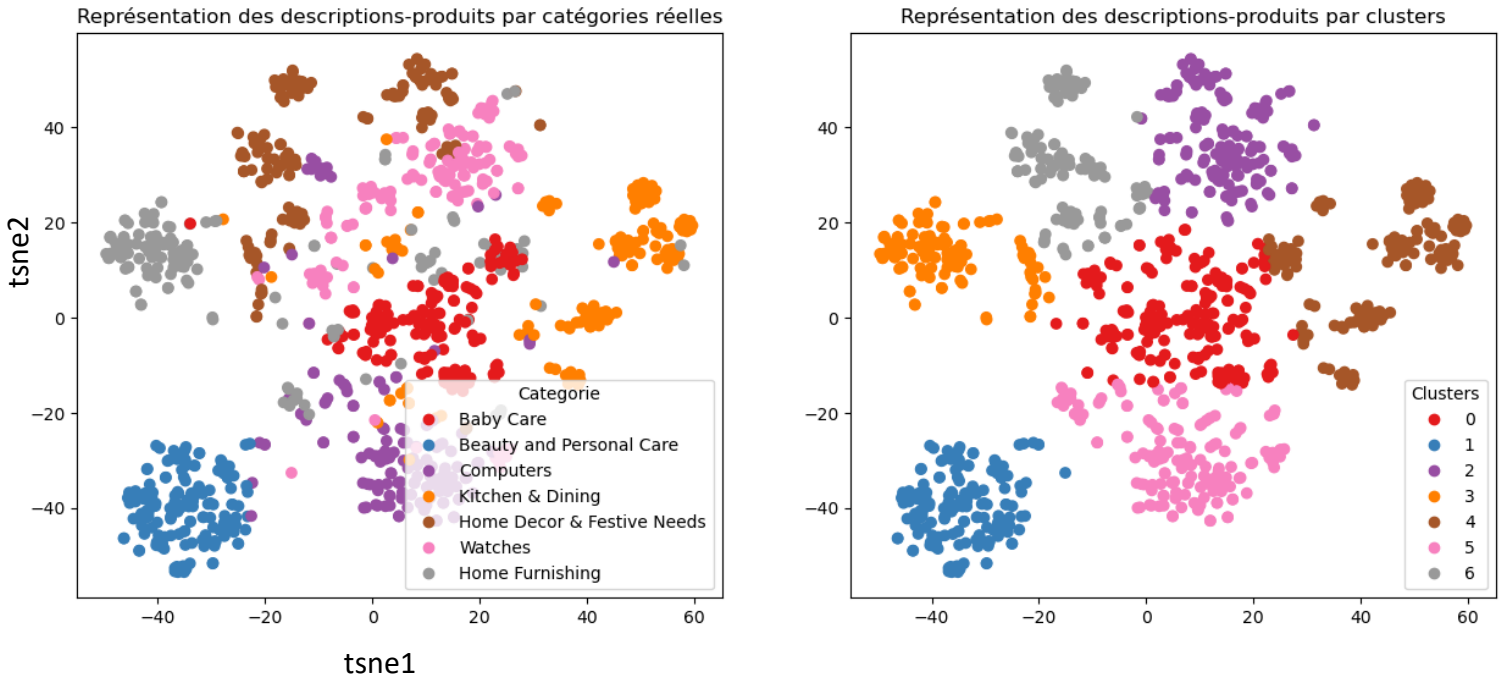
ARI : 0.5494
time : 9.0

- **Taille du vocabulaire : 5 324 mots (+ 4 269 stop-words)**
- **Encodage par vecteurs creux de 5 324 dimensions**

Données textes: Extraction de features « bag of words » -> Tf-idf

❖ 1. Extraction de features « bag of words » -> TF-IDF:

TF-IDF (Term Frequency-Inverse Document Frequency) calcule la pertinence d'un mot dans une série ou un corpus par rapport à un texte. ***(1050, 5324)***



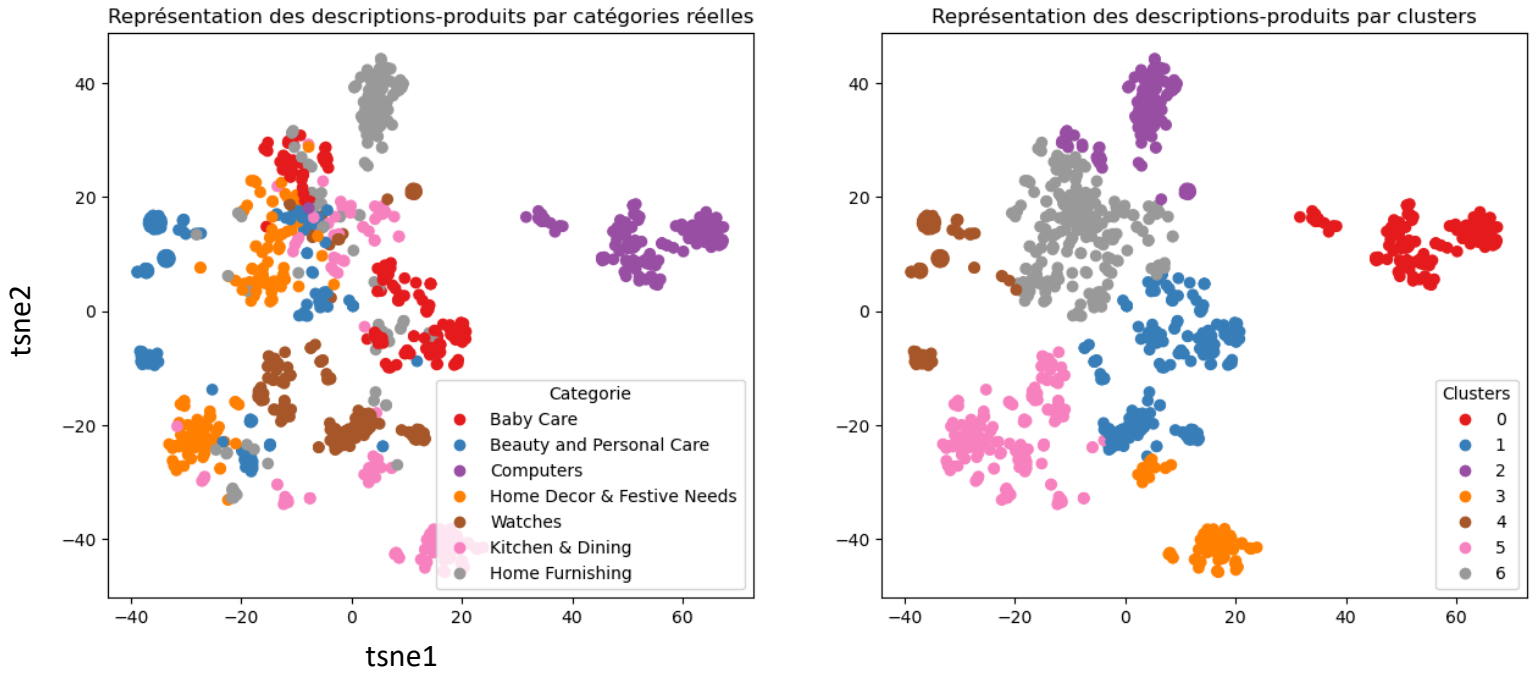
ARI : 0.5494
time : 9.0

➢ **A gauche la représentation 2D par TSNE -> quelques incohérences entre description-produits et catégories réelles. A droite la classification obtenue par kmeans est assez fidèle aux catégories réelles à gauche.**

Données textes: Extraction de features word embedding -> Word2Vec

❖ 2. Extraction de features « Word Embeddings » classique : Word2Vec

Word2Vec = réseau neuronal pour prédire les mots cibles dans les phrases et prend le coefficient de la dernière couche du réseau neuronal comme éléments du vecteur de mots. **Embedding matrix: (4713, 100)**



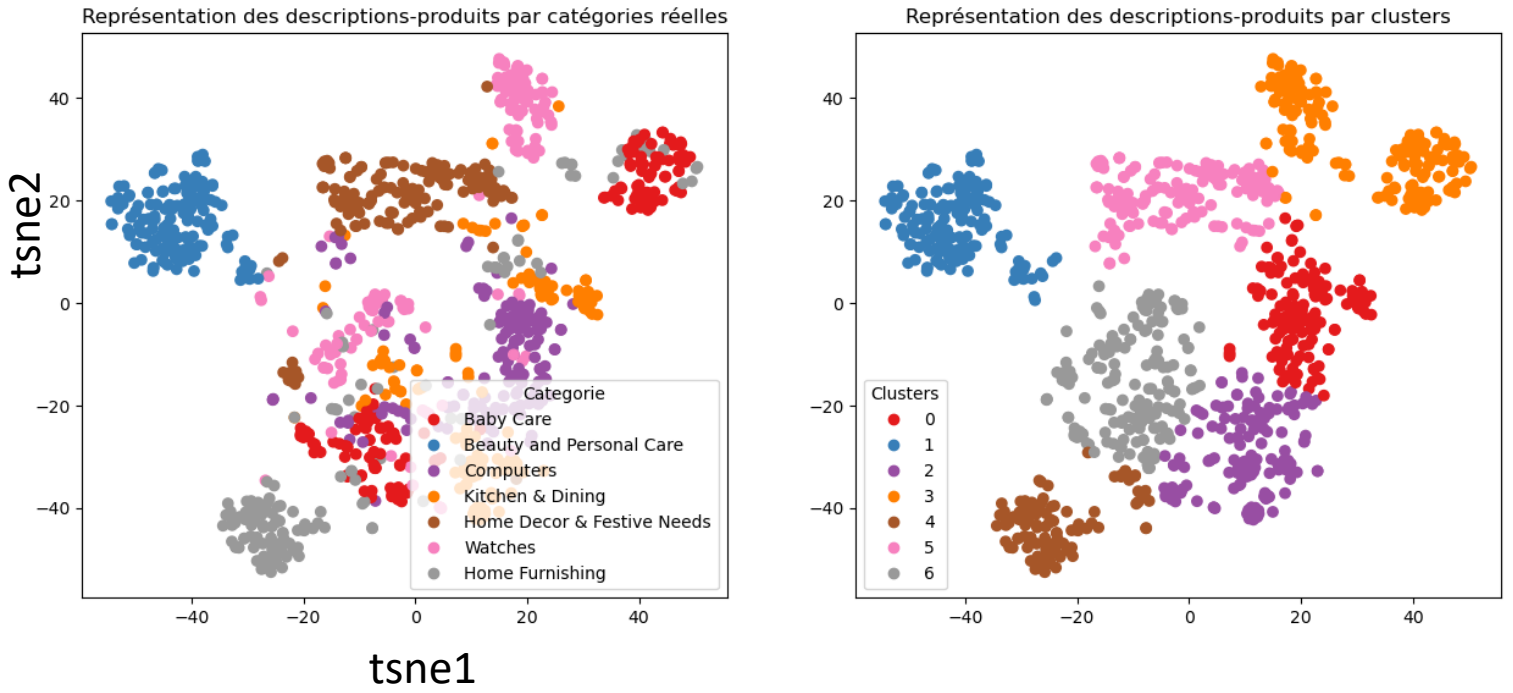
ARI : 0.3722
time : 9.0

➢ **A gauche la représentation 2D par TSNE -> incohérences plus fortes entre description-produits et catégories réelles. A droite, la classification obtenue par kmeans est assez peu fidèle aux catégories réelles**

Données textes: Extraction de features word embedding -> BERT

❖ 2. Extraction de features « Word Embeddings » : BERT

BERT : Bidirectional Encoder Representations from Transformers. Modèle procède de façon bi-directionnel, ce qui lui permet d'avoir une bien meilleure compréhension du texte.



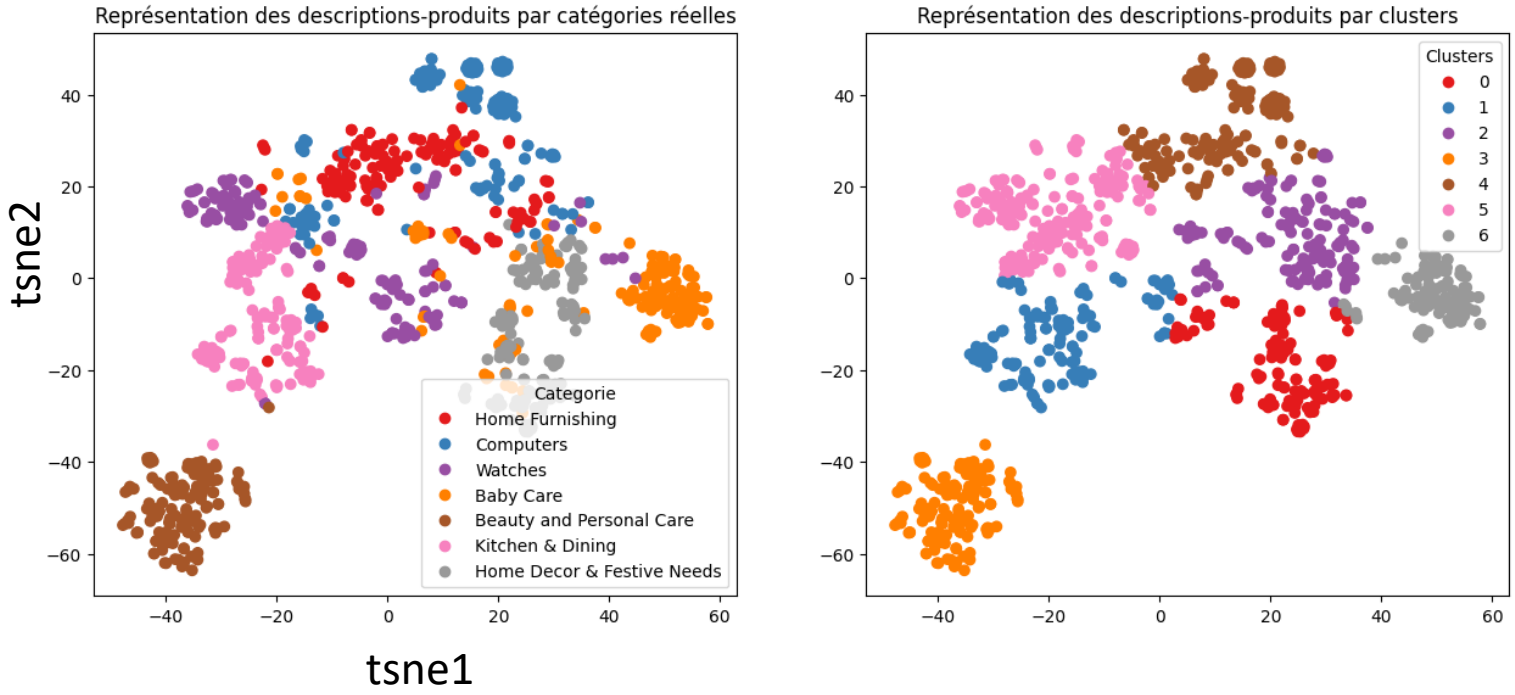
ARI : 0.428
time : 9.0

- **A gauche la représentation 2D par TSNE -> quelques incohérences entre description-produits et catégories réelles.**
- **A droite la classification obtenue par kmeans est assez peu fidèle aux catégories réelles à gauche.**

Données textes : Extraction de features sentence embedding -> USE

❖ **2. Extraction de features « Sentence Embeddings » : <u>USE</u>**

USE : Encodage du texte par réseau neuronal (transfer learning) **Universal-sentence-encoder**



ARI : 0.4136
time : 21.0

- **A gauche la représentation 2D par TSNE -> quelques incohérences entre description-produits et catégories réelles.**
- **A droite la classification obtenue par kmeans est assez peu fidèle aux catégories réelles à gauche.**

Données : Etapes réalisées pour l'étude de faisabilité

✓ **Données textes**

❖ **Données images :**

Prétraitements communs → Création de descripteurs → Créations de clusters de descripteurs → Création histogramme par image → Réduction de dimension TSNE → Analyse visuelle TSNE → Analyse similarité catégories réelles vs clusters

OpenCV

Bag of images

Algorithme SIFT

1

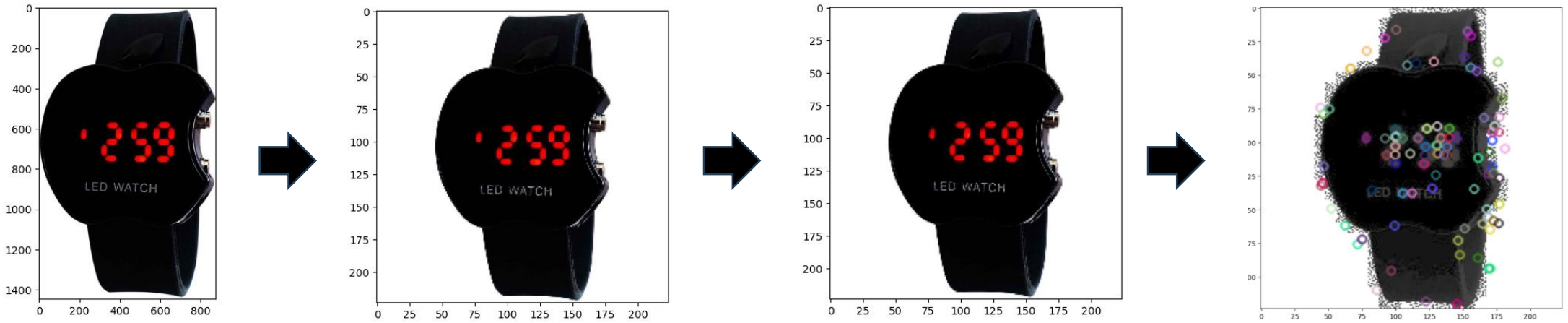
Etude de faisabilité = Algorithme de Transfer Learning basé sur des réseaux de neurones

VGG16

2

Données images : Prétraitements OpenCV

❖ **0. Prétraitement des données images et extraction de features « Bag of images » : SIFT**



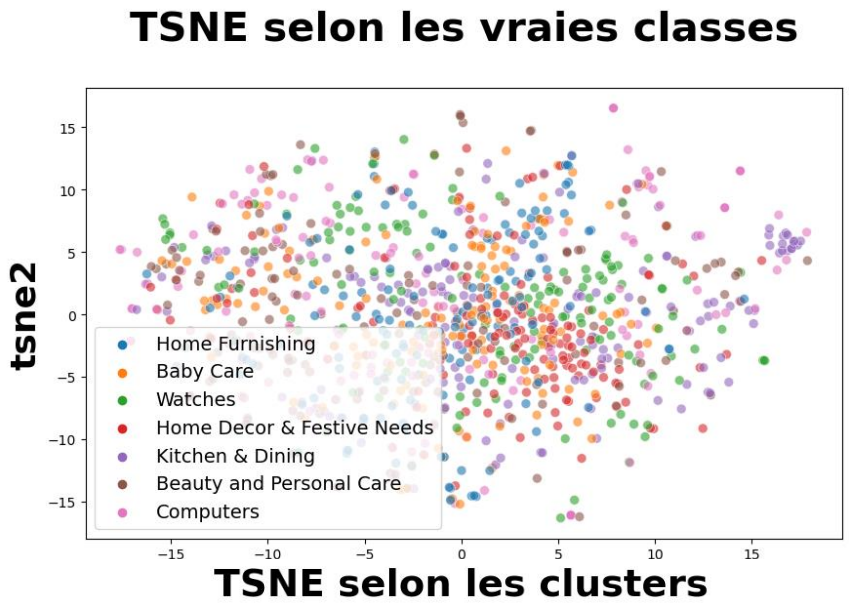
Redimensionnement 224x224(x3)

Optimisation de luminosité et contraste

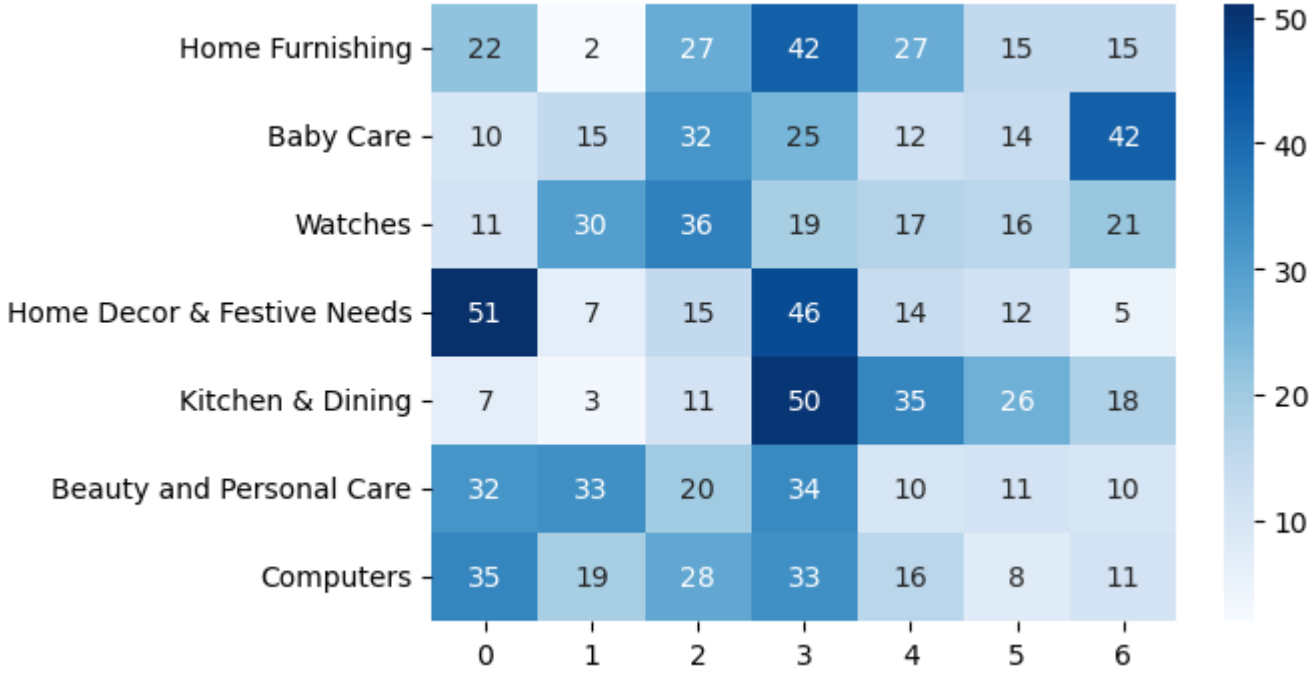
Passage en niveau de gris et égalisateur d'histogramme

- **Descripteur = vecteur qui décrit le voisinage de la feature à laquelle il est associé. Il est utilisé pour repérer les paires de features qui se ressemblent le plus dans deux images.**
- **Dimension des descripteurs : (237954, 128),** Nombre de clusters : squrt(237954) = 488

Données images : Extraction de features par algorithme de type SIFT

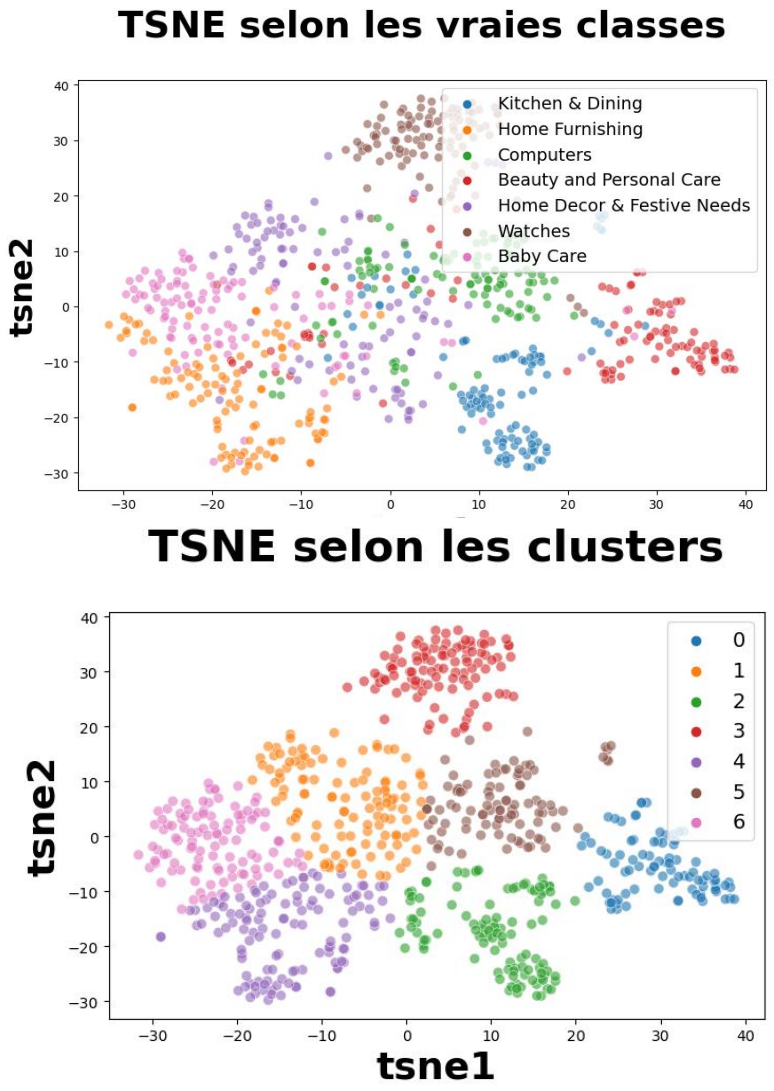


❖ **1. Suite extraction de features « Bag of images » : <u>SIFT</u>**

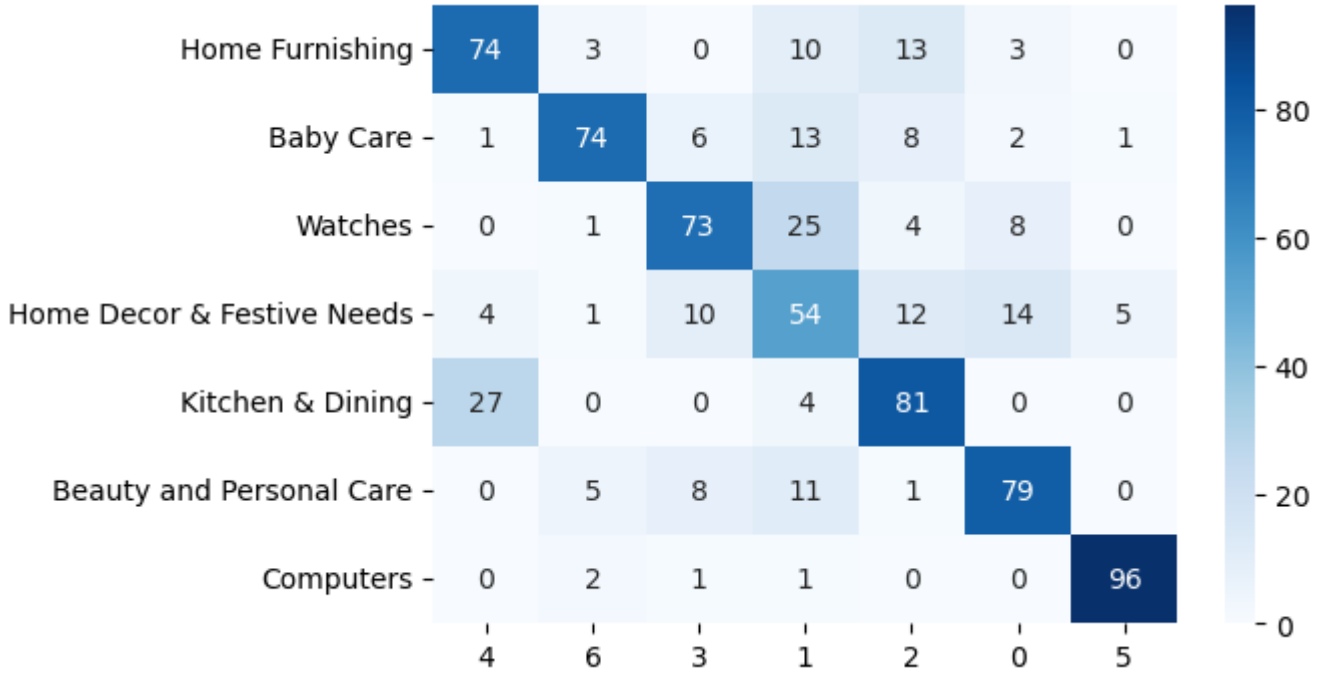


ARI : 0.0326
time : 6.0

Faisabilité : Extraction de features par algorithme CNN



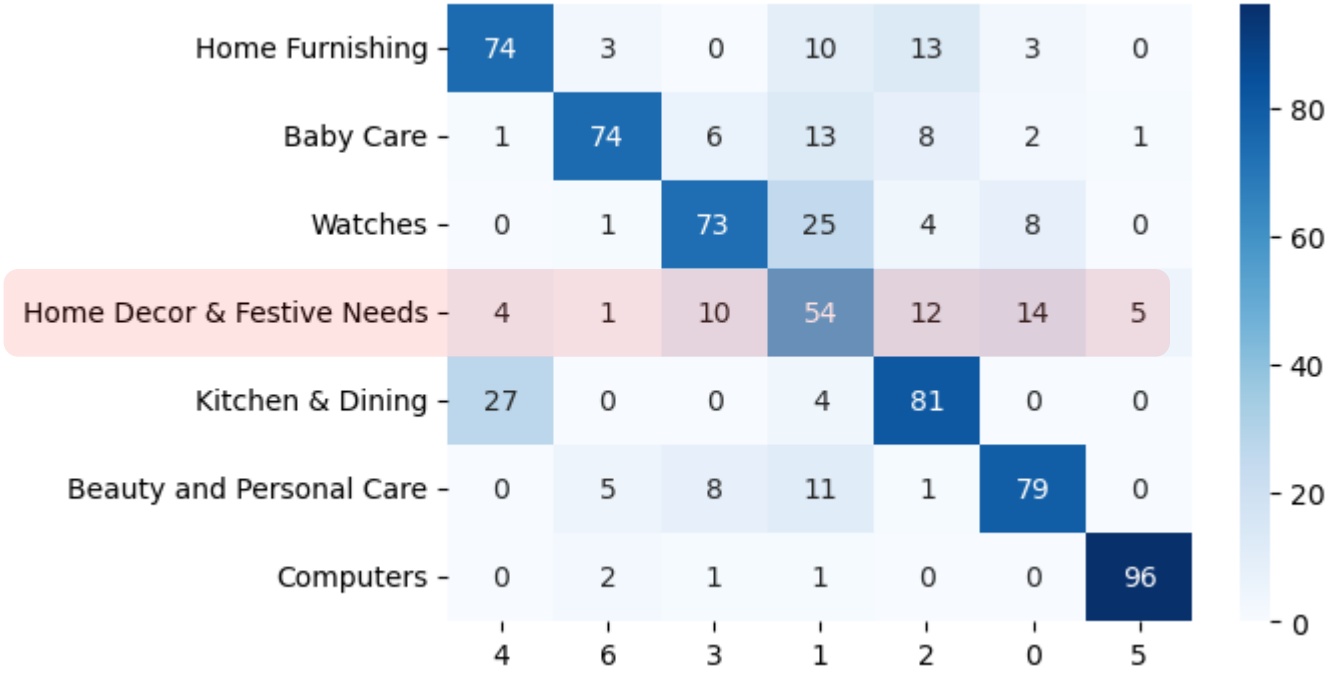
❖ **2. Extraction des caractéristiques par Algorithme de Transfer Learning basé sur des réseaux de neurones**

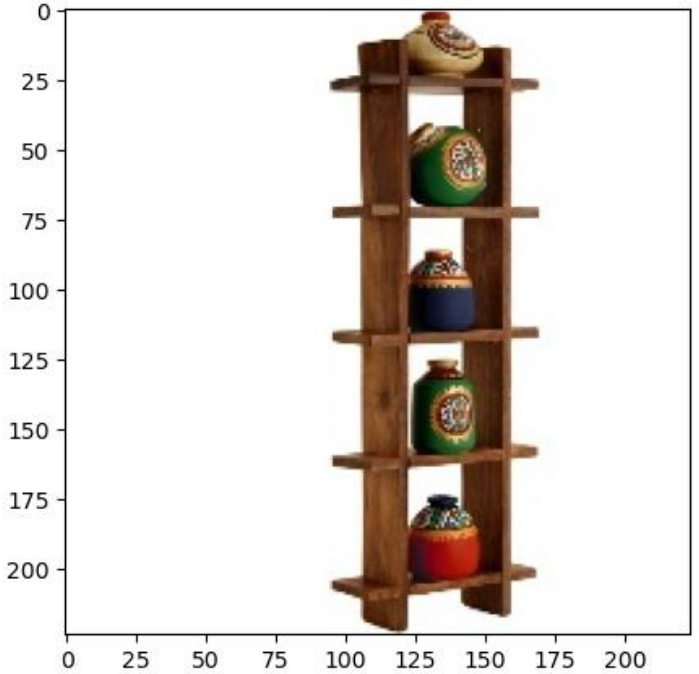


➢ **Dimension des features = (735, 4096), après PCA (735, 581), TSNE (735,2)**

Faisabilité : Extraction de features par algorithme CNN

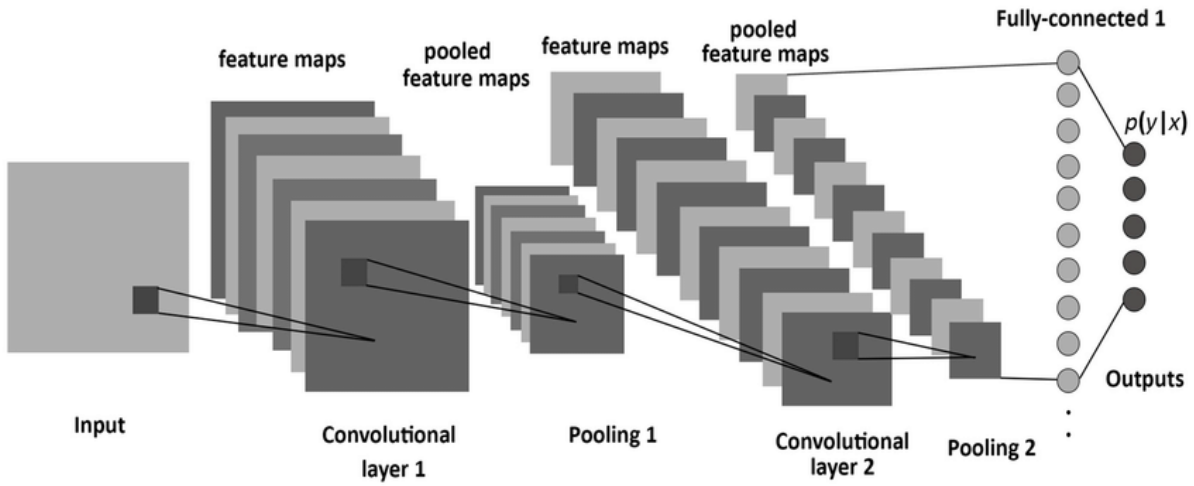
❖ **2. Extraction des caractéristiques par Algorithme de Transfer Learning basé sur des réseaux de neurones**





Classification supervisée images avec data augmentation

❖ **Classification supervisée avec data augmentation : <u>VGG16 avec Imagenet</u>**



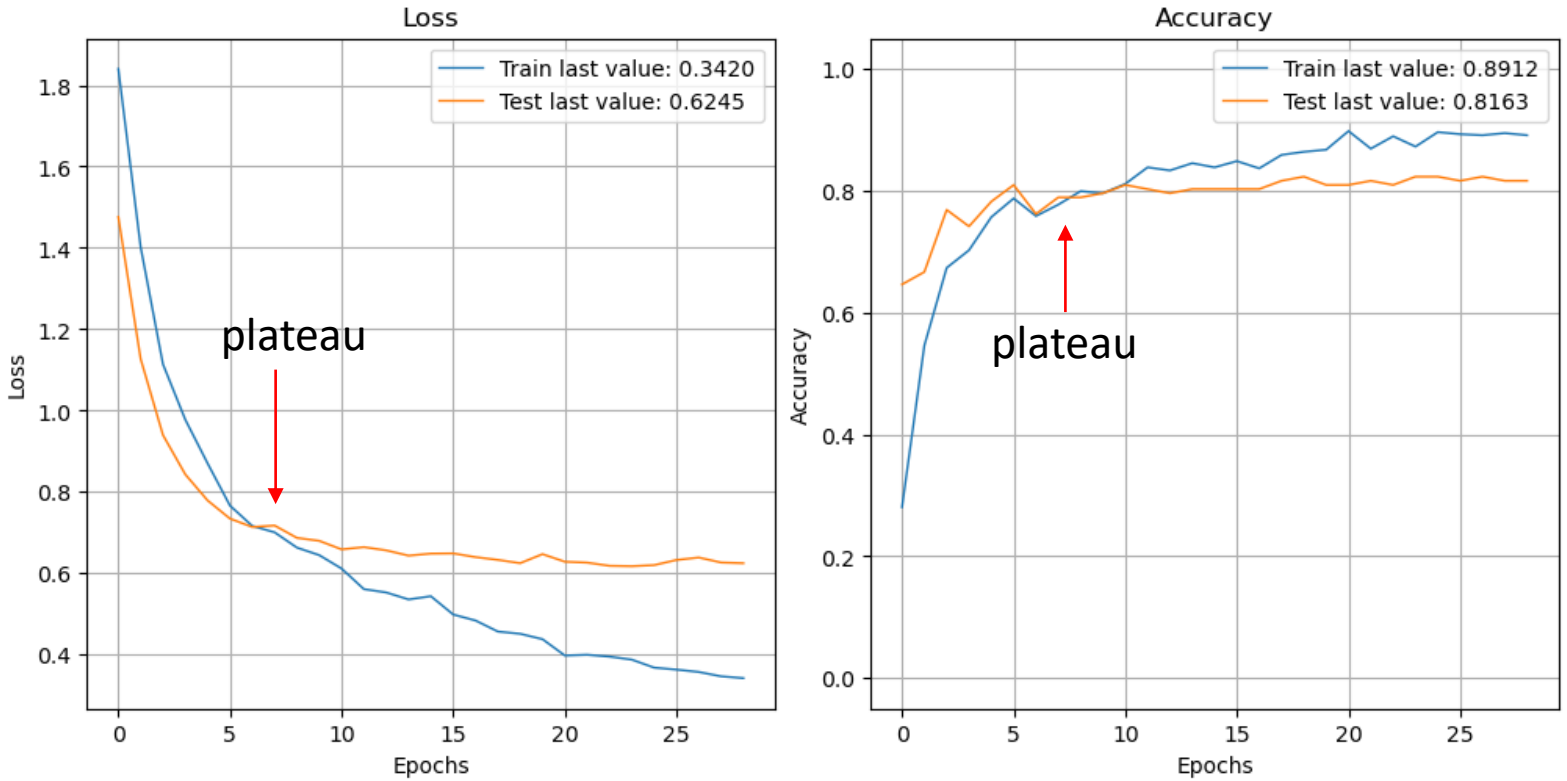

```
# Récupération modèle pré-entraîné
model_base = VGG16(include_top=False, weights="imagenet",
                   input_shape=(224, 224, 3))
for layer in model_base.layers:
    layer.trainable = False
```

- **Réseau VGG-16 pré-entraîné sur ImageNet (paramètre weights=« imagenet »).**
- **On retire la dernière couche fully-connected (include_top = False)**
- Ce réseau tronqué calcule la représentation de chaque image en entrée à partir des features déjà apprises lors du pré-entraînement.
- On entraîne alors un classifieur, initialisé aléatoirement, sur ces représentations pour résoudre le nouveau problème.

Classification supervisée images avec data augmentation

❖ **Classification supervisée avec data augmentation :**

Optimisation d'un hyperparamètre du modèle (le nombre d'Epochs)



 # Score du dernier epoch:

19/19 - loss: 0.2434 - accuracy: 0.9388

Training Accuracy : 0.9388

5/5 - loss: 0.6369 - accuracy: 0.8095

Validation Accuracy : 0.8095

 # Score de l'epoch optimal:

Validation Accuracy : 0.8163

Test Accuracy : 0.7975

- Précision satisfaisante du modèle de classification supervisée images avec **Epochs** = 30
- La data augmentation est une vraie plus-value dans notre contexte.

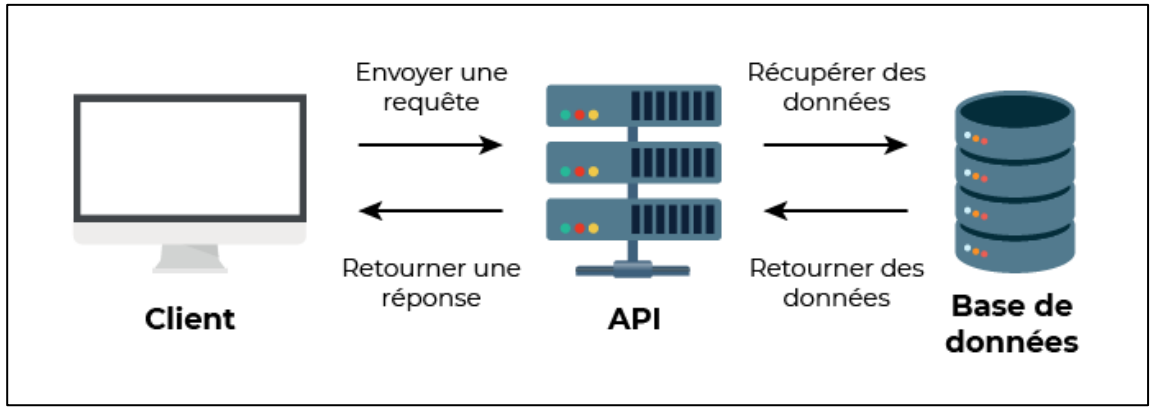
Test avec API : Elargissement gamme de produits, épicerie fine

Edamam Food and Grocery Database

By Edamam | Updated a month ago | Food

❖ **Intérêt de l'utilisation d'une API :**

10 produits à base de "champagne" via l'API Edamam dans un fichier ".csv", foodId, label, category, foodContentsLabel, image.



❖ **Requête Python :**

```
import requests

url = "https://edamam-food-and-grocery-database.p.rapidapi.com/api/food-database/v2/parser"

querystring = {"ingr":"champagne"}

headers = {
    "X-RapidAPI-Key": 
    "X-RapidAPI-Host": "edamam-food-and-grocery-database.p.rapidapi.com"
}

response = requests.get(url, headers=headers, params=querystring)

print(response.json())
```

Test avec API : Elargissement gamme de produits, épicerie fine

❖ **Requête Python :** *# foodId, label, category, foodContentsLabel, image*

dict_keys
text
parsed
hints
_links

20 items dont "food" qui est compose des features suivantes :

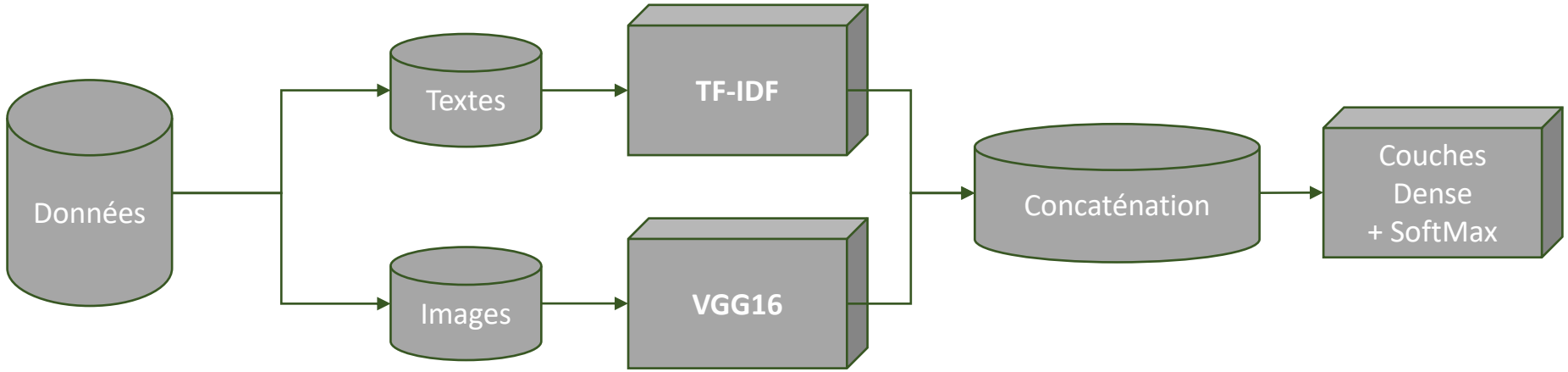
```
{'foodId': 'food_b753ithamdb8psbt0w2k9aquo06c',
'label': 'Champagne Vinaigrette, Champagne',
'category': 'Packaged foods',
'foodContentsLabel': 'OLIVE OIL; BALSAMIC VINEGAR; CHAMPAGNE VINEGAR; GARLIC;
DIJON MUSTARD; SEA SALT.'}
```

```
df.head(10)
```

✓ 1.1s Python

	foodId	label	category	image	foodContentsLabel
0	food_a656mk2a5dmqb2adiamu6beihduu	Champagne	Generic foods	https://www.edamam.com/food-img/a71/a718cf3c52...	NaN
1	food_b753ithamdb8psbt0w2k9aquo06c	Champagne Vinaigrette, Champagne	Packaged foods	NaN	OLIVE OIL; BALSAMIC VINEGAR; CHAMPAGNE VINEGAR...
2	food_b3dyababjo54xobm6r8jzbghjgqe	Champagne Vinaigrette, Champagne	Packaged foods	https://www.edamam.com/food-img/d88/d88b64d973...	INGREDIENTS: WATER; CANOLA OIL; CHAMPAGNE VINE...

Conclusion faisabilité et recommandations



Caractéristique	TF-IDF	CountVect	Word2Vec	BERT	USE	SIFT	VGG16
ARI	**0.5494**	**0.4683**	**0.3722**	**0.428**	**0.4136**	**0.0326**	**0.4887**
Temps de calcul	**9.0 sec**	**11.0 sec**	**9.0 sec**	**9.0 sec**	**21.0 sec**	**6.0 sec**	**109.0 sec**

Missions remplies :

✓ Nous avons démontré la faisabilité de regrouper automatiquement des produits de même catégorie avec les données txt et images. **Réseau neuronal avec apprentissage conjoint sur texte et images.**

Conclusion faisabilité et recommandations

Missions remplies :

- ✓ Nous avons réalisé **une classification supervisée avec data augmentation** afin d'optimiser le modèle.
- ✓ Nous avons testé une API pour élargir la gamme de produits à l'épicerie fine.

Merci pour votre attention

ANNEXES

Model	Advantages	Limitation
Weighted Words	* Easy to compute * Easy to compute the similarity between 2 documents using it * Basic metric to extract the most descriptive terms in a document * Works with an unknown word (e.g., New words in languages)	* It does not capture the position in the text (syntactic) * It does not capture meaning in the text (semantics) * Common words effect on the results (e.g., “am”, “is”, etc.)
TF-IDF	* Easy to compute * Easy to compute the similarity between 2 documents using it * Basic metric to extract the most descriptive terms in a document * Common words do not affect the results due to IDF (e.g., “am”, “is”, etc.)	* It does not capture the position in the text (syntactic) * It does not capture meaning in the text (semantics)
Word2Vec	* It captures the position of the words in the text (syntactic) * It captures meaning in the words (semantics)	* It cannot capture the meaning of the word from the text (fails to capture polysemy) * It cannot capture out-of-vocabulary words from corpus

```python
def transform_bow_fct(desc_text):
    word_tokens = tokenizer_fct(desc_text)
    sw = stop_word_filter_fct(word_tokens)
    lw = lower_start_fct(sw)
    # lem_w = lemma_fct(lw)
    transf_desc_text = ' '.join(lw)
    return transf_desc_text

# Fonction de préparation du texte pour le bag of words avec lemmatizatior


def transform_bow_lem_fct(desc_text):
    word_tokens = tokenizer_fct(desc_text)
    sw = stop_word_filter_fct(word_tokens)
    lw = lower_start_fct(sw)
    lem_w = lemma_fct(lw)
    transf_desc_text = ' '.join(lem_w)
    return transf_desc_text

# Fonction de préparation du texte pour le Deep learning (USE et BERT)


def transform_dl_fct(desc_text):
    word_tokens = tokenizer_fct(desc_text)
    # sw = stop_word_filter_fct(word_tokens)
    lw = lower_start_fct(word_tokens)
    # lem_w = lemma_fct(lw)
    transf_desc_text = ' '.join(lw)
    return transf_desc_text
```