Rapport Projet Fil Rouge

Catégorisation et Matching de Produits e-commerce

Auteurs :
M. Jean-Nicolas Vizy
M. Valentin Phetchanpheng
M. Samuel Cohen-Solal
M. Pascal Lim

Référents Académiques :
M. Hamid Jalalzai
M. James Eagan

Référents Industriels :
M. Sidoine Kakeuh-Fosso
M. Dimitar Draganov

Version 1.0 du
2 juillet 2019

Table des matières

Table des figures

Introduction

Innoscape, jeune startup, a fait le constat que les entreprises ne tiraient pas suffisamment profit des données externes et notamment de l'Open Data dans leurs études de marché. En effet, il était assez compliqué et coûteux de suivre un produit sur les différentes plateformes de vente et sites e-commerce ou encore l'état des stocks des différents points de ventes. Innoscape considère que ces données représente une source importante d'information dans l'analyse stratégique et concurrentielle pour les marques et a construit une solution répondant à ces problématiques. Afin de fournir aux marques des outils technologiques toujours plus performants, et à l'heure du Big Data, l'utilisation d'algorithmes de machine learning pour le traitement automatique de ces données s'est présentée comme le nouveau défi d'Innoscape pour poursuivre son expansion et répondre à de nouvelles problématiques.

Comment identifier et catégoriser un produit à partir d'un descriptif non standardisé ?

Nous avons choisi de prendre part à ce défi et de chercher en collaboration avec Innoscape une solution en s'appuyant sur nos compétences dans le domaine du Big Data et du machine learning. Nous avons décomposé ce projet en plusieurs parties.

Dans un premier temps, nous allons exposer le contexte et les enjeux de ce projets. Puis nous identifierons les technologies et solutions existant à ce jour. Enfin, nous allons utiliser des algorithmes d'apprentissage afin de classifier les produits dans les bonnes catégories, et détecter si deux produits d'une même catégorie ayant des descriptifs différents sont identiques.

Chapitre 1

Contexte et objectifs

1.1 L'entreprise

Innoscape a été créée à partir du constat qu'aujourd'hui, les entreprises n'utilisent pas de sources d'information externes pour étudier l'état de leur marché. Innoscape ambitionne donc d'apporter de la valeur à partir de l'Open Data, en se concentrant sur le secteur du commerce digitalisé, défini comme les activités de vente au travers de points de vente physiques, accompagnées de dispositifs en ligne visant à informer les consommateurs et à proposer des services complémentaires comme le click and collect, et le retour de produits. En effet, les entreprises qui vendent leur produit par l'intermédiaire de distributeurs sont confrontées :

— D'une part, au poids du commerce physique, qui reste très important, et à la complexité du suivi du référencement de leurs produits sur de grands réseaux de distribution ;
— D'autre part, à l'importance croissante des sites d'e-commerce qui deviennent une source d'information privilégiée pour 40 à 60% des consommateurs (au travers des avis publiés en ligne), et qui complexifie la réalisation d'études de marché.

Innoscape fournit donc des solutions de marketing basées sur la data, visant à l'excellence opérationnelle en termes de réactivité, de flexibilité, d'interactivité, et de précision, afin de disrupter les acteurs traditionnels des études marketing (Nielsen, GFK), qui fournissent des informations avec des méthodes classiques, coûteuses et lentes.

Les principaux services proposés sont les suivants :
— Suivi du référencement dans les magasins physiques sur une zone géographique ;
— Outil de suivi des prix dans le temps ;
— Agrégation de notes de consommateurs ;
— Travail sur des produits de langues différentes ;

1.2 Les objectifs globaux

Les objectifs du projet se décomposent en trois problématiques :

Problématique 1 - Catégoriser les types de produit :

- Standardiser les types de produit (chaque site a une catégorisation et une architecture différente) pour les faire correspondre à des catégories de l'architecture interne Innoscape.

- Input : description des produits, catégories du produit dans les sites e-commerce

- Nous traiterons d'abord le problème à partir d'une langue, le français. Compte tenu de l'activité d'Innoscape, la solution devra intégrer la gestion de plusieurs langues à terme.

- Nous devront également prévoir un pré-traitement des données textuelles adapté.

Problématique 2 - Identifier les produits identiques :

- Mettre en place un pré-processing comme dans la partie précédente, afin de faire correspondre chaque produit au catalogue produits Innoscape

- Enrichir l'information à partir d'informations annexes (avis clients, prix etc)

- A titre indicatif, les méthodes actuelles de product matching d'Innoscape affichent une précision de l'ordre de 80% environ.

Problématique 3 – Gérer les changements de description des produits existants :

- Détecter les changements des produits existants sur une même source afin de réaffecter le produit à la bonne catégorie et minimiser le risque de perte d'information.

- Garder la trace du produit malgré les modifications.

Les objectifs étant fixés, nous allons nous intéresser au type de données manipulées.

1.3 Les données

Lors du crawling, plusieurs types de données sont récupérées à travers les sites e-commerce des fournisseurs. Le format principal de ces données est du texte, mais nous avons également les images produits qui pourront éventuellement être utilisés. A ce jour, nous avons deux datasets. Le plus important en terme de volumétrie contient environs 100 000 produits. Le second en contient environs 30 000. En terme de features, nous avons à peu près 10 informations liées au produit à disposition. Cela comprend par exemple les catégories du produit dans le site de distribution, le descriptif et la marque du produit.

Chapitre 2

État de l'Art

Dans cette partie nous allons voir les méthodes, technologies et solutions actuelles liées à notre objectif principal qui est de repérer à partir de données textuelles des similarités :

Le but de cet état de l'art n'est pas de faire un listing complet des méthodes existantes mais d'avoir un aperçu des différents types d'approche possibles et ainsi pouvoir faire un choix adapté à notre problématique.

On s'intéressera dans un premier temps aux types de préprocessing des données textuelles. Trouver de la similarité entre des mots qui ont le même sens peut être compliqué. En effet, les mots peuvent être écrits de manière différente selon s'ils sont au singulier ou au pluriel, au féminin ou au masculin et pourtant ils portent un sens commun. Il est donc important de travailler la donnée textuelle en amont pour apporter plus de pertinence à chaque corpus.

Dans un second temps, on étudiera les différents types de représentations des données textuelles. En effet, pour calculer des similarités et utiliser de algorithmes de classification, il faut passer à des données numériques. Il est donc commun d'utiliser des méthodes de représentation en vecteurs. Plusieurs méthodes existent, avec des objectifs et des propriétés différents.

Enfin, on examinera dans un dernier temps les différentes mesures de similarité qui nous permettront de créer des scores de similarité entre nos produits.

2.1 Préprocessing des données textuelles

L'étape de pré-traitement des données que l'on appelle aussi "preprocessing" consiste à nettoyer et normaliser les données textuelles avant de pouvoir les analyser et les classifier via un algorithme. Tout cela dans le but de rendre la modélisation plus précise, simple, et rapide, et sans perdre d'information importante.

Dans notre cas, cette étape de pré-traitement s'effectue en trois phases : la normalisation des données, le retrait des stop words ainsi que la racinisation/lemmatisation.

2.1.1 Reformatage des données

Afin de pouvoir comparer les données textuelles entre elles, il convient de normaliser les données par le biais de différents traitements :

- Gestion des NaN : Parmi les données que nous conservons, on supprime les lignes ne contenant que des NaN.
- Conversion des mots/lettres en minuscule : Deux mots identiques véhiculent la même information qu'un des deux mots commence par une majuscule ou non.
- Gestion des caractères spéciaux : Élimination des accents, suppression des caractères numériques, des signes de ponctuation et caractères spéciaux (" ?", "/", "@" etc).

2.1.2 Retrait des Stop Words

La seconde manipulation effectuée dans le traitement du texte est la suppression de ce qu'on appelle en anglais les "stop words". Ce sont les mots très courants dans la langue étudiée tels que les articles, prépositions... ("de", "et", "à", "le"... en français) et qui n'apportent pas de valeur informative pour la compréhension du "sens" d'un corpus. Ces "stop words" sont très fréquents et augmentent le temps de calcul, c'est pour ces raisons que nous décidons de les supprimer.

Afin de ne conserver que les mots ayant une signification représentative du texte, un autre traitement consiste à supprimer les mots faisant partie du vocabulaire "commun" qui est spécifique aux données, c'est à dire les mots les plus fréquents du corpus et qui n'apportent aucune information.

Pour définir formellement les stop words au sein d'un corpus de textes, plusieurs approches sont envisageables. On peut par exemple ne conserver que les mots qui ont un indice de Gini élevé : l'indice de Gini d'un mot w dans un corpus de textes à répartir dans k classes est donné par :

$$G(w) = \sum_{i=1}^{k} p_i(w)^2 \tag{2.1}$$

Où $p_i(w)^2$ est la probabilité conditionnelle d'appartenance d'un texte à la classe i, sachant que le mot w est présent dans le texte. Ainsi, un mot qui apparaît surtout dans une seule classe aura un coefficient de Gini élevé, alors qu'un mot également présent dans toutes les classes aura un coefficient de Gini égal à 1/k. D'autres métriques peuvent également être utilisées, telles que le gain d'information ou l'information mutuelle.

2.1.3 Lemmatisation et racinisation

Pour éviter que deux mots quasiment identiques ne soient traités comme deux mots différents par un algorithme (par exemple, un même nom au singulier et au pluriel, ou un même adjectif au masculin et au féminin), la "lemmatisation" et la "racinisation" sont des moyens permettant de traiter ces problèmes.

La lemmatisation consiste à représenter chaque mot par sa forme canonique. Ainsi, les verbes par leur forme à l'infinitif et les noms par leur forme au masculin singulier sont pris en compte. L'intérêt principal de la lemmatisation est la substitution des mots par leur racine ou leur lemme. Ce processus permet une réduction du nombre de descripteurs. Par exemple, le remplacement des mots "conductrice", "conducteur" par l'unique racine "conducteur" semble être avantageux pour des tâches de similarité tout comme le remplacement des formes conjuguées "franchit" et "franchi" par le lemme "franchir". Cette méthode présente un avantage car la lemmatisation utilise à la fois le contexte entourant le mot et des informations grammaticales supplémentaires. Pour certains mots comme « marteau », la racinisation et la lemmatisation produisent les mêmes résultats, indiquant qu'il s'agit d'un nom. Cependant, pour des mots comme « avions », qui peut définir soit un nom ou un verbe, la racinisation réduira le mot en « avion », mais la lemmatisation réduira le mot différemment en le détectant soit comme le verbe « avoir » ou le nom « avion » en fonction des mots à proximité.

Le stemming ou "racinisation" consiste à supprimer tous les suffixes, préfixes et autres désinences des mots afin de ne conserver que leur origine.

Plusieurs librairies Python proposent des stemmers (on peut notamment citer Snowball de NLTK ou encore Spacy). Il conviendra d'essayer plusieurs solutions afin de déterminer laquelle est la plus performante, de manière à éviter de choisir un stemmer qui raccourcisse trop les mots et crée ainsi des confusions entre des mots différents (par exemple en transformant « terrasse » et « terre » en « terr ») ou au contraire un stemmer trop peu efficace.

2.1.4 Réduction de la dimension

Comme on le verra dans la section suivante, en fonction de la représentation choisie pour les textes, on peut être amené à travailler dans des espaces de très grande dimension, et sur des matrices très "sparse". Différentes techniques de réduction peuvent être envisagées :

— TF-IDF (term frequency-inverse document frequency) : représente un texte par les fréquences des mots qui le composent. Les termes les plus fréquents dans le corpus, assimilables à des stop words, sont pénalisés. Cette technique peut permettre une réduction de la dimension, dans la mesure où elle s'accompagne souvent d'une

suppression des hapax (termes trop peu fréquents pour qu'un classifieur puisse en tirer un enseignement pertinent pour des prédictions futures).
— LSA (Analyse sémantique latente), qui consiste à effectuer une décomposition en valeurs singulières de la matrice des occurrences du corpus ;
— NMF : factorisation en matrices non négatives de la matrice des occurrences ;
— LDA (Latent Dirichlet Allocation) : méthode générative visant à estimer les distributions a priori des thèmes au sein d'un corpus, et des termes au sein de chaque terme ;
— Embedding générés à partir de réseaux de neurones : un auto-encodeur permet de réaliser une représentation ad hoc d'un corpus, dont la dimension sera déterminée par la taille de la couche cachée. Selon les besoins, une telle représentation peut être construite à partir du corpus qu'on étudie, ou bien être obtenue à partir de modèles pré-entraînés (tels que Word2Vec, FastText, GloVe, etc. . .).

Un bon choix de représentation permet de simplifier l'entraînement et la prédiction et d'éliminer les détails superflus, tout en préservant au maximum l'information contenue dans les textes.

2.2 Types de Représentation des Données

2.2.1 Bag-of-Words

Le choix traditionnel de représentation textuel est le bag-of-words. Les textes sont représentés sous forme de vecteurs de dimension égale au nombre de mots dans le vocabulaire global et à valeurs entre 0 (absence du mot dans le texte) et 1 (présence du mot). Dans tous les cas, une telle approche sera nécessairement simplificatrice, dans la mesure où elle ignorera l'ordre dans lequel les mots apparaissent dans un texte et leur nombre d'occurrences. Ainsi, «le chat mange la souris » et « la souris mange le chat » auront la même représentation dans un modèle bag-of-words.

2.2.2 TF-IDF

Définition

Le TF-IDF pour "Term Frequency - Inverse Frequency Document" est une méthode de mesure notamment utilisée dans la recherche d'information ("Information Retrieval") ou le Text Mining. Ce score permet d'évaluer l'importance d'un mot contenu dans un document, relativement à un ensemble de documents.

L'idée derrière ce score est que, plus un terme a une occurrence forte au sein d'un

document, plus il est significatif (TF). De plus, si un terme a une forte occurrence au sein d'un document mais aussi des autres documents alors ce terme n'est pas caractéristique d'un thème en particulier, et sa fréquence brute (TF) doit être pénalisée (IDF).

Formule mathématique du TF-IDF :

$$TFIDF(t,d) = TF * IDF \tag{2.2}$$

où t : terme et doc : document

Term Frequency :

Mesure la fréquence d'un mot au sein d'un document.

$$TF(t,d) = \frac{\text{count of t in doc}}{\text{number of words in doc}} \tag{2.3}$$

Inverse Document Frequency :

$$IDF(t) = ln(\frac{\text{total number of docs}}{\text{number docs t appears in}}) \tag{2.4}$$

2.2.3 Plongements lexicaux

Word2Vec et plongements lexicaux réalisés à partir de réseaux de neurones

Word2Vec constitue une représentation élaborée, qui permet notamment de placer les mots dans un espace de façon à ce que la distance entre deux mots dont le sens est proche soit inférieure à la distance entre deux mots dont les significations sont très différentes. En outre, il peut permettre de travailler dans un espace de dimension plus restreinte que le nombre total de mots constituant le vocabulaire du corpus étudié.

Ceci vient de ce que la représentation Word2Vec n'est pas directement basée sur la présence ou la fréquence d'apparition d'un mot dans un texte, mais sur la proximité entre deux mots telle qu'elle a été préalablement estimée par un réseau de neurones. On distingue deux types de Word2Vec : la modélisation Continuous Bag of Words (CBOW) et la modélisation SkipGram.

Plus généralement, les plongements lexicaux ("embeddings") construits à partir de réseaux de neurones constituent généralement l'état de l'art en matère de traitement du langage naturel.

Ces embeddings présentent plusieurs avantages notables par rapport aux représentations Bag of Words :

- Ils permettent, tout en travaillant sur de très gros corpus, de se ramener à des espaces de travail de quelques centaines de dimensions (en fonction du nombre de neurones choisis pour la couche cachée)

- Ils préservent la proximité sémantique entre les mots : dans l'espace de représentation, des mots de sens similaires seront proches les uns des autres. Ceci permet de représenter des relations sémantiques sous la forme d'opérations algébriques : par exemple, dans un modèle Word2Vec bien entraîné, le vecteur le plus proche de l'expression "France - Paris + Italy" est "Rome", ce qui illustre le fait que la notion de capitale d'un pays peut elle-même être représentée et déduite du modèle. Ceci serait impossible dans une simple approche bag of words.

- En pratique, des librairies Python telles que Gensim mettent à disposition des modèles Word2Vec pré-entraînés, ce qui peut s'avérer utile pour certaines applications.

Cependant, ces modèles présentent aussi certaines limites :

- Entraîner un tel modèle requiert un corpus de taille conséquente, sans quoi sa performance risque d'être nettement inférieure à celle d'un modèle entraîné dans une représentation bag of words dont la dimension aura été réduite par analyse en composante principale (courbe "LSA" sur la figure ci-dessous) ;

- Les modèles pré-entraînés disponibles via des librairies Python sont entraînés sur des corpus très généraux (typiquement, Wikipedia, ou Google News), et peuvent se révéler peu performants lorsqu'on travaille sur domaine spécifique associé à des termes peu utilisés dans le langage courant.

- La disponibilité et la qualité de modèles pré-entraînés est assez variable selon les langues.

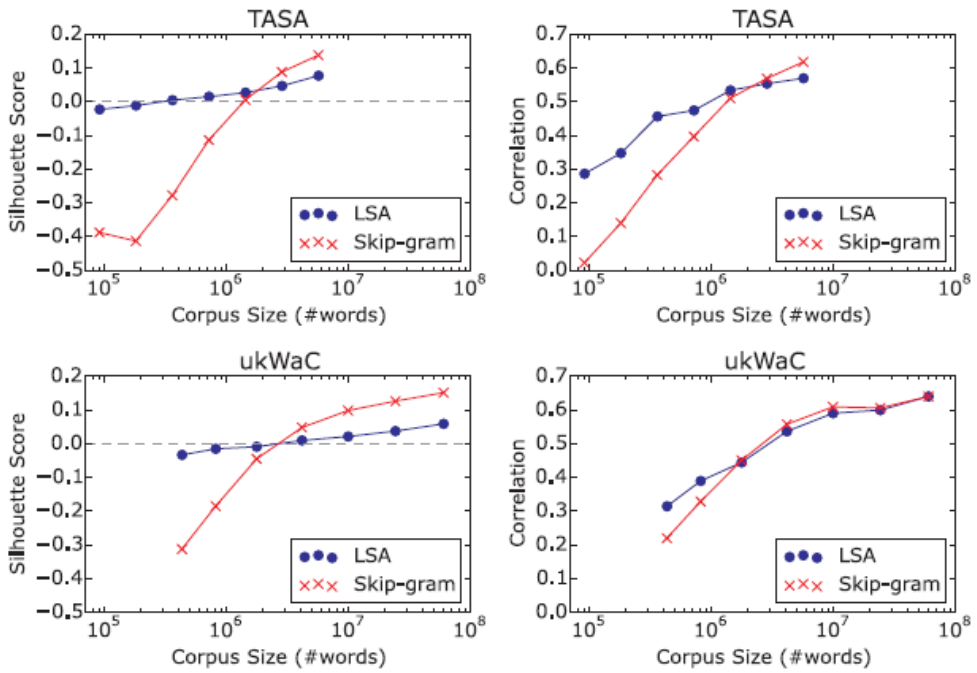

FIGURE 2.1 – Comparaison des scores de performance d'un modèle Word2Vec (SkipGram) et d'un modèle bag of words (LSA) en fonction de la taille du corpus étudié

Or, la problématique d'Innoscape se réfère précisément à un corpus de taille limitée (quelques dizaines de milliers de documents, pour quelques centaines de milliers de mots), qui utilise un vocabulaire spécialisé (lexique du bricolage et des travaux), dans plusieurs langues. Par conséquent, choisir a priori une représentation Word2Vec n'est pas forcément le bon choix pour notre cas d'usage. Par conséquent, nous avons dû nous pencher sur des représentations concurrentes, moins consommatrices en données.

2.2.4 LSA, NMF et LDA

L'analyse sémantique latente (LSA), l'allocation de Dirichlet latente (LDA) et la factorisation de matrices non négatives (NMF) permettent d'établir des relations entre un ensemble de documents et les termes qu'ils contiennent, en construisant des "topics" liés aux documents et aux termes. Ce processus est appelé extraction de topics.

Elles utilisent pour cela tout d'abord les matrices d'occurrences (BOW ou TF-IDF) des termes dans les documents et effectue des opérations différentes les unes des autres pour essayer d'extraire des topics représentatifs des données textuels.

Dans le cas de la LSA, une décomposition en valeur singulière est effectuée sur la matrice d'occurrences. La LDA se base sur un modèle génératif probabiliste où chaque document est considéré avoir un set de topics associé. La distribution des topics au sein du corpus,

et des termes au sein des topics, est réputée suivre une loi de Dririchlet. Enfin la NMF approxime la matrice d'occurrences par le produit de deux matrices : la première lie les documents aux topics et le deuxième lie les mots aux topics.

Ces méthodes de réduction de dimensions sont plus adaptées pour des petits jeux de données qui correspond à notre cas. Le choix de représentation s'orientera donc plus entre ces trois méthodes.

2.3 Mesures de Similarité

Les mesures de similarité les plus connues dans le domaine textuel sont l'indice de Jaccard et la mesure Cosine. En ce qui concerne les images, il existe également plusieurs solutions existantes. Pour notre problématique nous avons choisi de nous orienter vers les réseaux de neurones siamois (SCNN).

2.3.1 Indice de Jaccard

L'indice de Jaccard permet de calculer la similarité entre deux ensembles.

Soient deux ensembles de mots A et B :

$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$

Cette mesure est très intéressante pour calculer des similarités entre deux séquences binaires. Si on représente nos ensembles A et B en séquences de 1 et de 0, on obtient par exemple :

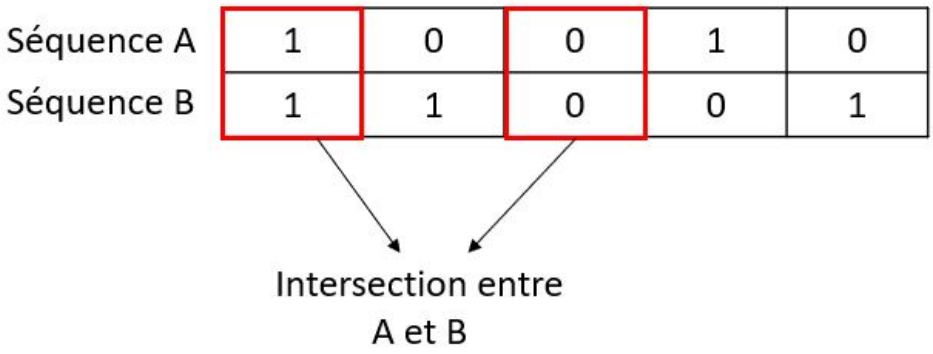

FIGURE 2.2 – Exemple de représentation de l'intersection entre deux séquences binaires représentant deux ensembles A et B

La mesure de Jaccard dans ce cas là est facilement calculable et est égale à : $J(A, B) = \frac{2}{5}$.

Cette représentation en séquences binaires correspond exactement à la représentation bag-of-words vue précédemment. Il est donc très aisé de mettre en place une mesure de similarité après avoir choisi cette représentation.

2.3.2 Similarité Cosine

La similarité Cosine correspond au cosinus de l'angle entre deux vecteurs. Il est calculé à partir d'un produit scalaire entre ces vecteurs divisé par le produit des normes des deux vecteurs :

$$\cos(\theta) = \frac{\mathbf{A}.\mathbf{B}}{||\mathbf{A}||.||\mathbf{B}||}$$

Cette mesure est très bien adaptée avec la représentation vectorielle des mots utilisée par pratiquement toutes les méthodes de représentation textuelles.

Chapitre 3

Réalisations

Dans cette partie, nous allons détailler les étapes réalisées pour répondre aux problématiques d'Innoscape.

3.1 Problématique 1 : Catégorisation des produits e-commerce

Les actions décrites dans toute cette partie concernent la problématique 1 qui est de catégoriser les produits dans les références Innoscape.

3.1.1 Analyse des Données

Pour la première problématique, nous avons travaillé à partir de 3 fichiers de données qui nous ont été partagé par Innoscape.

Données références Innoscape

Le premier data set qu'on appelle "données références Innoscape" contient donc l'architecture interne Innoscape, c'est à dire un référentiel avec l'ensemble des catégories dans lesquelles les produits doivent être affectés.

FIGURE 3.1 – Exemples de catégorie de la référence Innoscape

Ce référentiel contient seulement la catégorisation la plus fine que peut avoir un produit dans le référentiel Innoscape sans prendre en compte les niveaux de catégorisation plus macroscopiques.

Les catégories correspondent plus au monde du bricolage qui est le secteur prédominant dans le business d'Innoscape.

Le nombre de catégories est égal pour le moment à 97 mais est susceptible de varier selon les données vendeurs.

Données vendeurs

Les deux autres fichiers correspondent à des "données vendeurs" qui contiennent les données non structurées de produits provenant de différents sites de distributeurs et e-commerce et pour lesquels nous devons identifier la catégorie. Les données sont en plusieurs langues.

Pour notre étude on a d'abord commencé à étudier les données en français. On a donc adapté nos librairies de NLP pour correspondre à la langue étudiée. L'étude faite pourra être ensuite généralisée aux autres langues en adaptant les librairies.

Dans la table vendeurs, on a plusieurs informations qui peuvent nous aider à catégoriser les produits :
- On a d'abord les catégories initiales des produits dans les sites e-commerce avec les différentes hiérarchies de catégorisation (ex : Bricolage, Quincaillerie, Colles et adhésifs,

Colles à carrelage).
- On a ensuite la description de ces produits

Les données vendeurs, toutes langues confondues, correspondent à plus de 100 000 lignes.

Les données, comme décrites au-dessus, sont non structurées. Pour créer plus de pertinence sur ces données, il faut faire un travail de pré-processing.

3.1.2 Préprocessing

Cette étape de pré-traitement consiste à nettoyer et standardiser les données afin de ne conserver que les informations pertinentes. Ce travail est conditionné par le type de représentation des données et de la méthode de labelling pour préparer le train set comme cela sera décrit dans la partie suivante.

Durant cette phase de preprocessing, nous avons donc travaillé à partir des fichiers de données décrit plus haut. Dans un premier temps pour chaque data set, nous n'avons gardé que les champs pertinent pour l'analyse à ce stade :

- Pour les données de référence, nous n'avons gardé que la colonne contenant les catégories Innoscape ;

- Pour les données vendeurs, nous n'avons gardé que les champs indiquant les quatre niveaux hiérarchique des catégories ainsi que d'autres champs porteurs d'informations comme la description du produit.

Afin d'organiser nos traitements, nous avons implémenté différentes fonctions permettant de normaliser les données, retirer les stop words, ainsi que lemmatiser et raciniser les données textuelles. Nous faisons ensuite appel à ces fonctions au sein d'une fonction "preprocessing" que l'on applique à chaque data set. Nous détaillons ci-dessous les étapes de traitements.

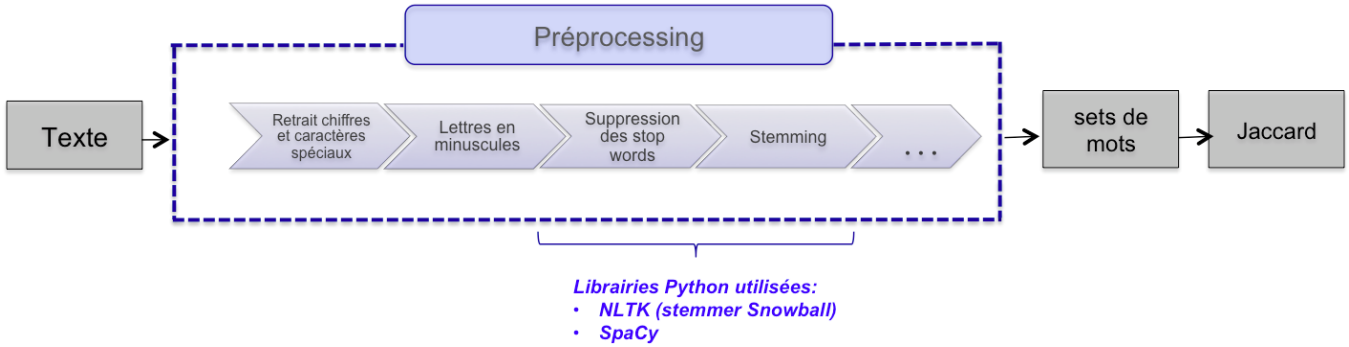

FIGURE 3.2 – Etapes de pré-processing

Normalisation

Au sein des données vendeurs, nous retirons toutes les lignes ne contenant que des NaN car sans information il est impossible pour ces lignes d'être catégorisées. Pour chaque dataset, nous retirons également des données textuelles les données quantitatives (chiffres, nombres, quantités d'un produit...) et caractères spéciaux qui, même si elles permettent de déterminer un produit, ne sont pas pertinentes pour la catégorisation. Enfin nous transformons l'ensemble des mots en minuscules afin d'uniformiser les données et éviter les distinctions de deux mêmes mots sur la base d'écriture en majuscule ou minuscule.

Retrait des stop words

Nous avons filtré les stop words des textes. En effet, il existe dans la librairie NLTK une liste par défaut des stop words dans plusieurs langues, notamment le français. Nous avons donc repris cette liste que nous avons enrichie de mots spécifiques aux données et jugé non pertinents. Ces mots contiennent par exemple la suppression des marques dans la description.

De la même manière, certains mots dont la longueur est inférieure à 3 lettres ne sont pas pertinents et il convient alors de les supprimer. C'est notamment le cas des mots exprimant des unités de mesure, ou encore des références de produits (après suppression des caractères numériques).

Racinisation (Stemming) - Lemmatisation

Nous avions racinisé (stemming) et lemmatisé les données en utilisant deux librairies python différentes, Snowball de NLTK et Spacy. Cependant ces librairies n'étaient pas satisfaisantes lors du traitement, et ce pour des raisons différentes. En effet, la librairie Snowball de NLTK nous semblait trop "radicale" dans le stemming (elle réduisait par exemple le mot "terrasse" à la racine "terr", ce qui pouvait donner lieu à des confusions avec des tuiles en terre cuite). Le lemmatiseur de Spacy était convaincant sur nos données mais prenait beaucoup de temps à transformer les données. Il a donc fallu trouver une autre solution alliant efficacité et vitesse du traitement.

Nous avons finalement utilisé la librairie python TreeTagger pour la lemmatisation, qui ne présentait pas de tels inconvénients.

productfamily_seller	productseller_name	productfamily_seller_clean	productseller_name_clean
Colles à carrelage	SikaCeram Xtra Colle à carrelage intérieur e...	colle carrelage	max blanc exterieur carreau xtra interieur col...
Mastic	Colle mastic pour joints d étanchéité	mastic	etancheite mastic colle joint
Cires	Sikagard Protection Sol MAT Imperméabilisant...	cire	mat sol impermeabiliser effet protection
Perceuse visseuse	Perceuse visseuse RYOBI 18V OnePlus 1 batter...	perceur visseur|visseuse	oneplus visseur|visseuse lithium perceur batte...
Produits d étanchéité	Sika Multiseal Bande d étanchéité autocollan...	etancheite produit	etancheite dechirure cuit terre bander autocol...

FIGURE 3.3 – Exemple de pré-processing pour les produits

3.1.3 Extraction de topics

Une fois tous les pré-traitements nécessaires effectués, nous avons recherché une représentation pertinente des données, dans un espace vectorielle de dimension relativement limitée (quelques centaines de dimensions tout au plus). A cet effet, nous avons testé différentes techniques de construction d'embeddings, que nous avons évaluées selon plusieurs critères :

— la cohérence et l'intelligibilité des topics, ie la présence d'un vocabulaire contenu dans les vecteurs issus de l'embedding ;
— la capacité de l'embedding à prendre en compte les spécificités du cas d'usage (lexique spécialisé, travail sur un corpus de taille restreinte) ;
— la propension de la technique d'embedding à préserver l'information pertinente pour la classification malgré la réduction de dimension.

Pour chacun des embeddings testés, nous avons également cherché à déterminer sa dimension optimale (qui n'est pas nécessairement la même pour toutes les techniques), et nous avons testé une série d'algorithmes pour compléter l'évaluation ci-dessus par une évaluation empirique de la performance des solutions testées. Les algorithmes ont été entraînés à partir d'un corpus dont la constitution est détaillée dans la section 3.1.4 Catégorisation supervisée. Les embeddings testés sont les suivants : Word2Vec, LSI (latent semantic indexing), LDA (latent Dirichlet allocation), NMF (nonnegative matrix factorisation).

Word2Vec

Nous avons d'abord testé l'embedding Word2Vec à 200 dimensions pour la langue française entraîné sur le corpus frWac. Il s'agit d'un embedding pré-établi que nous avons importé directement. Un tel embedding, réalisé à partir de corpus de textes très large a a priori l'avantage de bien préserver le vocabulaire des textes. De fait, lors de la conversion du corpus Innoscape dans cet embedding, seuls 73 mots, soit moins de 1% du vocabulaire total, n'ont pas été reconnus. Cependant, cette représentation n'est pas sans poser problème, comme l'illustre la figure ci-dessous, qui représente la catégorisation des produit obtenue en affectant chaque produit à la classe avec laquelle il est le plus colinéaire :

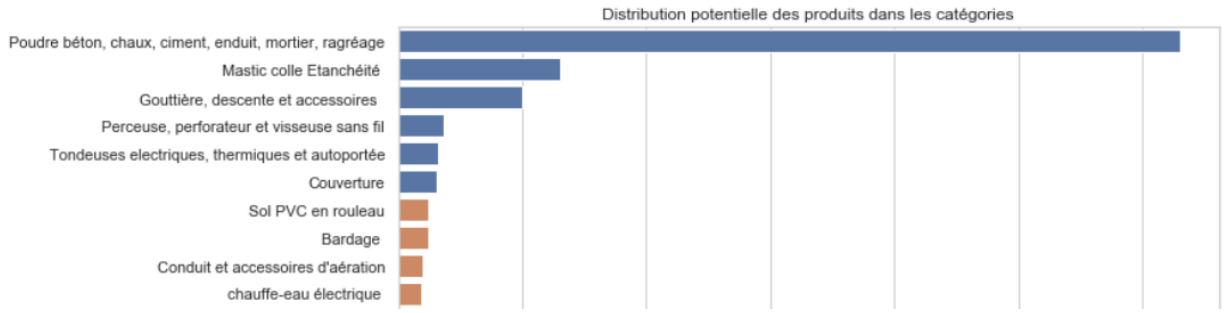

FIGURE 3.4 – Word2Vec - catégorisation des produits par la distance cosine

On observe clairement que la grande majorité des produits sont affectés à la famille 'Poudre béton, chaux, ciment, enduit, mortier, ragréage'. Or, un examen rapide du corpus révèle qu'au moins 40% des références produits concernent des tuiles, qui devraient être affectées à la famille 'Couverture'. Cette famille apparaît ici beaucoup plus minoritaire. Ceci est dû au fait que l'embedding utilisé semble attribuer à un même vecteur les termes relatifs aux tuiles et à la maçonnerie. Ainsi, la similitude cosine entre le mot 'tuile' et les mots 'béton', 'chaux', 'ciment' est de 0,75. Dans les faits, cet embedding rend donc difficile de distinguer deux catégories présentant des effectifs significatifs. Qui plus est, s'agissant d'un embedding pré-établi, il est difficile de le modifier pour l'adapter aux spécificités de notre corpus. Par conséquent, nous avons choisi de réaliser notre propre embedding.

Latent semantic indexing

La LSI est une technique de réduction de la dimension qui consiste tout simplement à effectuer une décomposition en valeurs singulières de la matrice termes-documents d'un corpus, de manière à ne garder que les composantes les plus significatives. A l'image de la réduction de dimension par analyse en composantes principales, le choix de la dimension et des composantes peut donc être réalisé en triant les vecteurs par ordre de valeurs propres décroissantes, et à sélectionner les n premiers vecteurs permettant d'expliquer l'essentiel de la variance (méthode du "coude"). C'est ainsi que nous avons procédé, comme illustré sur la figure ci-dessous.

Comme on peut le constater, les 150 à 200 premiers vecteurs propres suffisent à rendre compte de l'essentiel de la variance du corpus. Le contenu des topics est illustré sur la figure suivante :

On observe une certaine cohérence dans le vocabulaire des topics. Ainsi, les topics 0, 1, 2, et 4 font clairement référence à la notion de toiture au sens large, tandis que le topic 3 renvoie davantage aux notions de mastic, colle et étanchéité. Cependant, l'interprétation est compliquée par la présence de mots ayant des poids négatifs dans les topics (un topic est ainsi caractérisé par ce qu'il n'est pas, ce qui présente un intérêt limité d'un point de

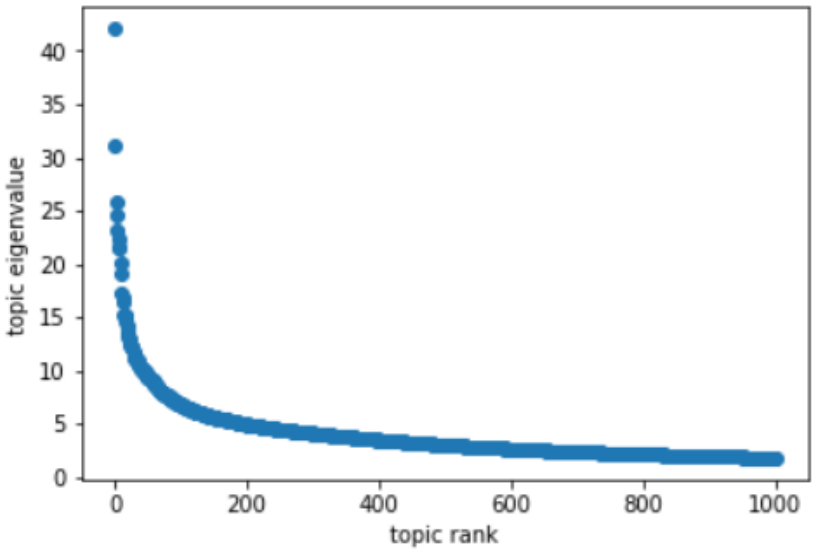

FIGURE 3.5 – LSI - Valeurs propres associées aux 1000 premiers vecteurs propres à gauche

```
[(0,
  '0.693*"tuile" + 0.338*"terre" + 0.321*"cuit" + 0.219*"ardoise" + 0.202*"couverture" + 0.178*"accessoire" + 0.155
*"rive" + 0.100*"rouge" + 0.090*"douille" + 0.087*"droit"'),
 (1,
  '0.598*"gouttiere" + 0.259*"toiture" + 0.248*"bardage" + 0.218*"accessoire" + 0.209*"raccord" + 0.191*"charpente"
+ 0.191*"eau" + 0.176*"materiau|materiaux" + 0.174*"descente" + 0.157*"evacuation"'),
 (2,
  '0.421*"terre" + 0.401*"cuit" + -0.379*"ardoise" + 0.283*"gouttiere" + -0.273*"accessoire" + -0.266*"tuile" + -0.
244*"charpente" + 0.203*"couverture" + -0.129*"toiture" + 0.125*"eau"'),
 (3,
  '0.467*"mastic" + 0.336*"adhesif" + 0.291*"outillage" + 0.282*"peinture" + 0.261*"electroportatif" + 0.255*"colle
" + 0.233*"quincaillerie" + 0.203*"etancheite" + 0.154*"coller" + 0.148*"perceur"'),
 (4,
  '-0.346*"charpente" + 0.342*"eau" + -0.327*"toiture" + -0.307*"bardage" + 0.217*"tuile" + 0.216*"evacuation" + 0.
200*"electroportatif" + 0.172*"outillage" + 0.167*"traitement" + 0.157*"chauffe"'),
```

FIGURE 3.6 – LSI - Contenu des 5 premiers topics

vue pratique). Par ailleurs, la multiplicité des topics relatifs à une même notion (ici la toiture) ne facilite pas non plus l'interprétation.

Factorisation non négative (NMF)

Pour pallier aux inconvénients de la LSI, et notamment la présence de mots avec des poids négatifs dans les topics, nous avons eu recours à la factorisation non négative. Pour déterminer la dimension optimale de l'embedding pour cette technique, nous avons cherché la dimension pour laquelle l'écart (au sens de la norme de Frobenius) entre la matrice termes-documents de départ et la matrice reconstituée à partir de la factorisation était le plus faible. Comme le montre la figure ci-dessous, cette erreur est minimale pour une dimension de 250 :

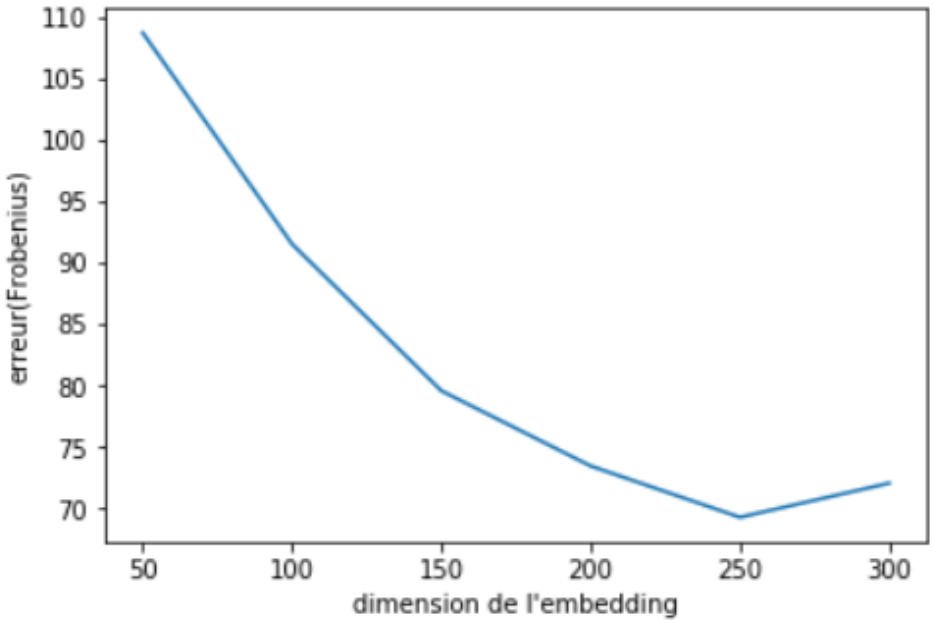

FIGURE 3.7 – NMF - distance de Frobenius entre la matrice de départ et la matrice reconstituée en fonction de la dimension

Le contenu des topics est illustré sur la figure suivante :

```
Topic #0: tuile 13.27 ardoise 0.41 couverture 0.23 huguenot 0.16
Topic #1: gouttiere 7.09 sol 2.23 bardage 2.13 revetement 2.11
Topic #2: terre 8.37 cuit 8.33 tuile 5.32 couverture 3.65
Topic #3: peinture 5.23 adhesif 4.35 mastic 3.61 quincaillerie 2.05
Topic #4: electroportatif 6.6 quincaillerie 3.6 outillage 2.5 charbon 0.37
Topic #5: charpente 5.24 bardage 4.12 toiture 3.97 menuiserie 2.13
Topic #6: tondeur 4.68 tondeuse 4.68 autoporte 2.82 tailler 1.25
Topic #7: evacuation 3.6 traitement 3.44 eau 3.25 accessoire 1.51
Topic #8: gros 4.17 œuvre 4.12 voirie 2.81 bpe 2.73
Topic #9: ite 4.19 finition 1.72 enduire 1.44 isolation 1.29
Topic #10: etancheite 6.71 toiture 1.96 materiau 0.93 materiaux 0.89
```

FIGURE 3.8 – NMF - Contenu des 10 premiers topics

Sans surprise, les mots avec des poids négatifs ont disparu. Qui plus est, les topics relatifs à la toiture semblent mieux rendre compte de certaines nuances (tuiles, par opposition à la charpente ou à l'étanchéité).

Latent Dirichlet Allocation (LDA)

La LDA est une méthode générative, qui suppose que la distribution des topics au sein d'un corpus et la distribution des termes au sein des topics suivent des lois de Dirichlet.

Pour déterminer la dimension optimale d'un tel embedding sur notre corpus, nous avons déterminé la dimension maximisant la cohérence des topics au sens de la mesure UCI. Cette mesure est définie comme suit :

$$C_{UCI} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} PMI(w_i, w_j) \tag{3.1}$$

avec :

$$PMI(w_i, w_j) = ln(\frac{P(w_i, w_j) + \epsilon}{P(w_i)P(w_j)}) \tag{3.2}$$

où le terme epsilon est destiné à éviter les erreurs numériques. Comme l'illustre la figure ci-dessous, la dimension optimale ressort donc à 450 :

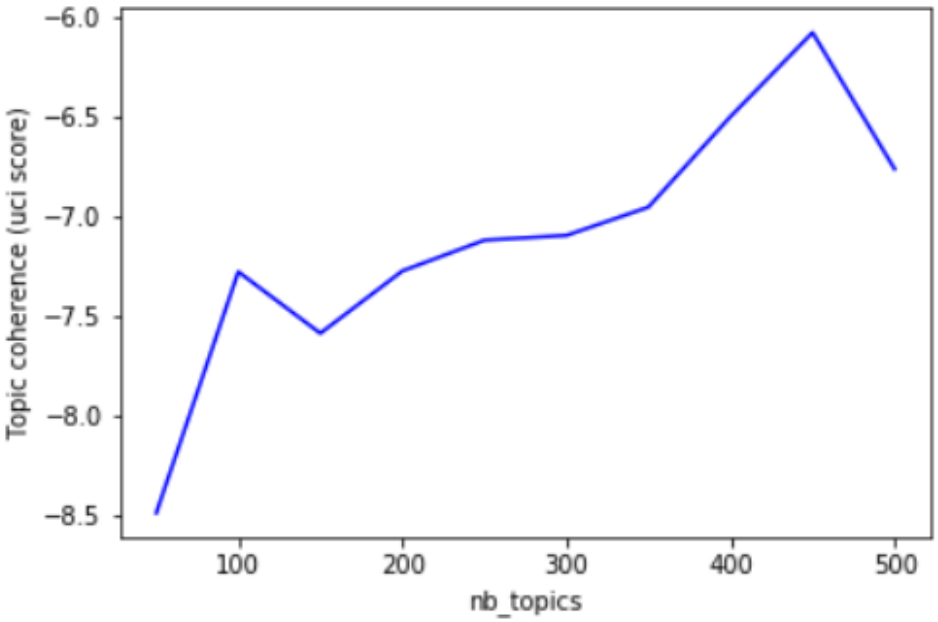

FIGURE 3.9 – LDA - cohérence des topics en fonction de la dimension de l'embedding

Malgré tout, les topics ainsi obtenus restent relativement difficiles à interpréter :

On est en particulier frappé par le déséquilibre entre certains topics dominés par un faible nombre de mots cohérents et avec un fort poids, et d'autres topics constitués d'un grand nombre de mots avec de très faibles scores.

```
 (256,
  '0.554*"sds" + 0.351*"max" + 0.003*"outillage" + 0.000*"main" + 0.000*"rpm" + 0.000*"hammers" + 0.000*"rotary" + 0.
000*"hammer" + 0.000*"perceur" + 0.000*"support"'),
 (21,
  '0.000*"silex" + 0.000*"specifique" + 0.000*"rech" + 0.000*"mmrouleaux" + 0.000*"retraire" + 0.000*"retrait" + 0.00
0*"maestro" + 0.000*"biocontrole" + 0.000*"cygne" + 0.000*"tsba"'),
 (420,
  '0.602*"pierre" + 0.147*"tyrolien" + 0.045*"gros" + 0.038*"œuvre" + 0.004*"construction" + 0.002*"materiau|materiau
x" + 0.000*"renovation" + 0.000*"materiau" + 0.000*"ciment" + 0.000*"mortier"'),
 (59,
  '0.466*"ref" + 0.148*"plus" + 0.130*"quincaillerie" + 0.090*"electroportatif" + 0.056*"outillage" + 0.045*"aspect"
+ 0.014*"batterie" + 0.010*"accessoire" + 0.008*"blanc" + 0.003*"gravillon"'),
 (334,
  '0.015*"etancheite" + 0.014*"toiture" + 0.012*"bander" + 0.007*"bardage" + 0.005*"gros" + 0.005*"œuvre" + 0.002*"lo
ng" + 0.001*"terrasse" + 0.001*"larg" + 0.001*"renovation"'),
```

FIGURE 3.10 – LDA - exemples de topics

Sélection de la représentation vectorielle

A l'issue de la comparaison de ces différentes techniques, nous avons pu valider à le type de représentation de nos données (TF-IDF+LSI, TF-IDF+NMF et TF-IDF+LDA), en comparant les performances de précision en classification sur le set d'entraînement établi.

Pour ce faire, nous avons décidé d'utiliser un algorithme de classification supervisée possédant des performances plutôt bonnes en général et qui sera très proche de celui qu'on va sélectionner. Plus précisément, il s'agit d'une forpet aléatoire (utilisant les hyperparamètres fixés par défaut dans scikit-learn).

Nous avons obtenu les résultats suivants à partir d'une validation croisée sur 10 folds :

Représentation vectorielle	Précision
TF-IDF + LSI (150 dimensions)	77.63%
TF-IDF + NMF (250 dimensions)	**78.34%**
TF-IDF + LDA (450 dimensions)	74.47%

Nous avons donc validé l'embedding TF-IDF + NMF qui transforme les données textuelles en vecteurs de faible dimension en se basant sur la méthode de factorisation de matrices non négatives.

3.1.4 Catégorisation supervisée

Pour effectuer une catégorisation supervisée ou semi-supervisée, nous avons besoin de données labellisées afin d'entraîner un classifieur.

Nous allons tout d'abord décrire comment nous avons procédé dans l'étape de labellisation des données puis dans un second temps expliquer le choix de notre algorithme de classification.

Constitution du train set

Comme nous l'avons indiqué plus haut, nous sommes tenus de classer les produits selon une typologie précise, définie par Innoscape, ce qui rend les démarches non supervisées a priori peu pertinentes. Cependant le jeu de données qui nous a été fourni ne compte aucun produit labellisé. Nous avons donc décidé de procéder à une étape de labellisation de données pour constituer le training set initial avec des données provenant de différentes marketplaces pour créer un set d'entraînement représentatif des sources utilisées par Innoscape.

Nous avons choisi de labelliser une partie des données de manière manuelle mais en nous aidant de recommandations sur les catégories. Pour ce faire, les données textuelles correspondant aux catégories des produits dans leur marketplace initial et les catégories Innoscape ont été plongées dans une représentation Word2Vec. Les recommandations ont ensuite été créées à partir d'une mesure de similarité Cosine entre les vecteurs provenant de la représentation Word2Vec.

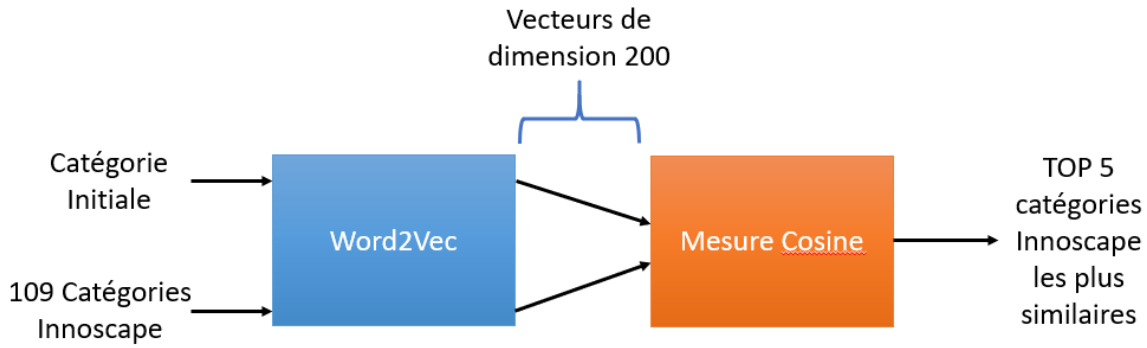

FIGURE 3.11 – Processus de mise en place de recommendations pour la labellisation

Nous récupérons ensuite le Top 5 des scores de similarité entre la catégorie initiale du produit et les catégories Innoscape. Les recommandations proposées par notre méthode facilitent la catégorisation manuelle mais ne permettent pas de remplacer l'action manuelle. En effet, il se retrouve souvent qu'on ne trouve pas la catégorie Innoscape qui convient dans le Top 5 des catégories les plus similaires. Il est donc nécessaire de vérifier les informations du produit directement sur le site d'origine. Nous avons finalement réussi à labelliser 3000 lignes de données de cette manière. Pour disposer d'encore plus de données, nous nous sommes appuyés sur le fait que le site Leroy-Merlin a un plan de classement des produits proche de celui d'Innoscape, ce qui permettait d'identifier les catégories d'un certain nombre de produits grâce à l'indice de Jaccard entre la catégorie Leroy-Merlin et la catégorie Innoscape. Nous avons ainsi obtenu 4000 lignes supplémentaires.

Cette première étape de labellisation va permettre d'entraîner ensuite un classifieur qui

pourra automatiquement labelliser une part plus importante du dataset restant automatiquement.

Sur-échantillonage

Suite à l'analyse de ces données d'entraînement labellisées, nous avons remarqué que les classes sont très déséquilibrées. Cela risque de poser un problème lors de l'entraînement d'un algorithme sur ces données.

Plusieurs méthodes existent, notamment les suivantes :
- SMOTE / SMOTE Borderline
- ADASYN

L'algorithme SMOTE a été testé au cours projet afin de limiter les déséquilibres en termes d'effectifs entre les catégories. Cependant, cette méthode permet seulement de compléter des données dans les catégories en se calquant sur des données déjà présentes. Pour rendre compte de la diversité d'une catégorie, il faut passer par une labellisation classique. Qui plus est, dans les faits, le recours à cette technique n'a pas permis d'améliorer les performances des classifieurs testés.

Retrait des produits ambigus

Compte tenu du fait que les données labellisées dont nous disposons pourraient malgré tout contenir des erreurs d'attribution, nous avons entrepris de retirer les produits potentiellement ambigus, qui pourraient altérer les frontières de décision d'un classifieur. Pour ce faire, nous avons défini les produits ambigus comme suit :

— un produit peut être ambigu sa représentation dans l'embedding utilisé est répartie sur un trop grand nombre de topics (auquel cas il peut se référer à un grand nombre de thématiques et donc de familles de produits) ;
— un produit peut être ambigu si les principaux topics qui composent sa représentation dans l'embedding sont eux-mêmes intrinsèquement ambigus, ie si ces topics intègrent un trop grand nombre de mots (auquel cas ils peuvent se référer à des thématiques trop floues pour être utiles à la classification) ;

Pour identifier ces produits (resp. ces topics), nous avons procédé comme suit :

— nous avons trié les topics (resp.) les mots du produit (resp. du topic) par ordre de poids décroissant ;
— pour chaque topic (resp. mot), on calcule le quotient du poids de ce topic (resp. de ce mot) par rapport à celui du topic (resp. du mot) précédent ;
— si ce ratio est supérieur à un certain seuil (par exemple, 0,5), on considère que les deux topics (resp. mots) ont des poids comparables, et jouent un rôle significatif

dans la composition du produit (resp. du topic) ;
— on reproduit cette démarche jusqu'à identifier le premier topic (resp. mot) qui ne joue pas un rôle significatif dans la composition du produit (resp. du topic). On peut alors déterminer le nombre de topics (resp. de mots) significatifs dans la composition du produit (resp. du topic).

Une fois le nombre de topics (resp. de mots) pertinents identifiés, on peut retirer du training set les produits qui comptent trop de topics significatifs, ou dont les topics significatifs sont des topics comptant trop de mots significatifs.

Bien entendu, les deux seuils évoqués dans la méthode ci-dessus (seuil de ratio de poids de deux topics/mots successifs, et nombre maximal de topics/mots significatifs admissibles) sont des hyperparamètres dont la valeur ne doit pas être fixée de manière aléatoire. Une comparaison des performances en classification nous a permis de déterminer que les couples de paramètres les plus efficaces sont les suivants :
— pour l'identification des produits comptant trop de topics significatifs : 3 topics significatifs admissibles, avec un seuil de ratio de poids de 0,66 ;
— pour l'identification des topics comptant trop de mots significatifs : 6 mots significatifs admissibles, avec un seuil de ratio de poids de 0,66.

Sélection du modèle de classification

Pour les algorithmes de classification, deux pistes se sont ouvertes à nous :

- Les algorithmes semi-supervisés qui paraissaient prometteurs pour notre problématique, sachant qu'on ne possède que très peu de labels.

- Les algorithmes supervisés qui nécessitent des données conséquentes pour commencer à avoir des résultats intéressants.

Concernant les algorithmes semi-supervisés, nous avons testé pour l'instant que l'algorithme Label Propagation de scikit-learn qui a donné des résultats inférieurs à 50%, très en-deçà des résultats obtenus lors de la validation de l'embedding (supérieurs à 70%). Les résultats étant très faibles, nous avons préféré nous tourner vers des algorithmes supervisés.

Les algorithmes d'ensemble learning ont donné les meilleurs scores pour notre problème de classification, notamment l'algorithme ExtraTrees qui a donné les meilleures performances de précision :

Représentation vectorielle	RandomForest	ExtraTrees
TF-IDF + LSI	77.63%	78,84%
TF-IDF + NMF	78.34%	**79,26%**
TF-IDF + LDA	74.47%	75,02%

Les paramètres de l'algorithme ExtraTrees qui ont permis d'obtenir ces résultats cor-

respondent à un bagging à partir de 2000 estimateurs et une profondeur maximale des arbres de décision à 100.

Conclusion et actions entreprises pour améliorer la précision du modèle

Il ressort des essais d'embeddings réalisés que la solution la plus performante est un embedding TF-IDF puis NMF à 250 dimensions. Cependant, pour atteindre des niveaux de performance satisfaisants, nous avons entrepris les actions suivantes :

— élargissement du dataset : pour accroître la quantité de données, nous avons ajouté aux 3000 lignes labellisées 4000 lignes supplémentaires labellisées de manière semi-automatique au moyen de leur score de Jaccard. Les lignes ajoutées ont un score élevé (supérieur à 0,5), offrant un haut niveau de fiabilité.

— filtrage des hapax préalablement à l'embedding : Nous avons remarqué que la taille du vocabulaire maximal retenu lors du preprocessing jouait un rôle important sur la qualité de l'embedding obtenu. Ainsi, en ne gardant que les 500 mots les plus fréquents du corpus (hors stop words), les performances en classification peuvent être améliorées ;

— simplification de la typologie de classement : Nous avons retiré et/ou fusionné avec d'autres les familles de produits dont les effectifs étaient insuffisants pour l'entraînement d'un classifieur (typiquement, moins de 20 références)

— retrait des produits ambigus : Voir plus haut le détail de la méthode employée.

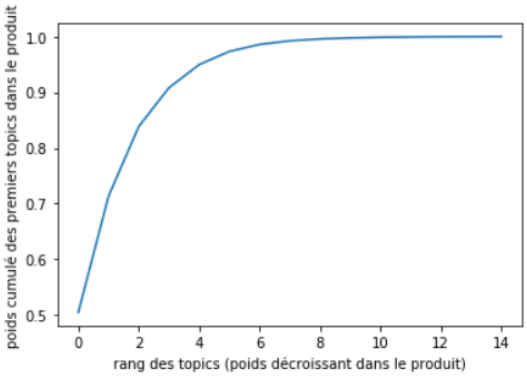

FIGURE 3.12 – LDA - poids cumulés moyens des principaux topics au sein d'un descriptif de produit

Ces différentes techniques nous ont permis d'augmenter sensiblement les performances

du modèle, pour atteindre 98,03% de précision. Cependant, compte tenu des grands déséquilibres au sein du jeu de données (les effectifs par famille vont de 20 à plus de 500), nous avons voulu compléter cette mesure de performance brute par une autre mesure. L'accuracy brute avantage un modèle performant sur les catégories nombreuses, au détriment des catégories rares. Pour remédier à cette difficulté, nous avons cherché à déterminer un seuil de confiance sur les probabilités d'appartenance d'un produit à une classe, de manière (i) à limiter le risque d'erreur pour les catégories avec peu d'effectifs et (ii) à permettre que les produits difficiles à classer soient identifiés et validés par un opérateur humain.

La mesure de performance auxiliaire que nous avons choisie a donc été le nombre de types d'erreurs de classification constatées ; autrement dit, le nombre de coefficients non nuls hors de la diagonale de la matrice de confusion. Nous avons voulu déterminer, pour différents seuils de confiance, combien de types d'erreurs existaient, et quelle proportion des prédictions réalisées par le modèle pouvaient être jugées fiables (voir figures ci-dessous). Ceci permet de déterminer qu'en imposant qu'un produit ne puisse être affecté par le modèle à une catégorie que si la probabilité estimée d'appartenance à cette catégorie est d'au moins 60%, le nombre de types d'erreurs passe de 14 à 3, tandis que seuls 5% des produits doivent faire l'objet d'une vérification manuelle.

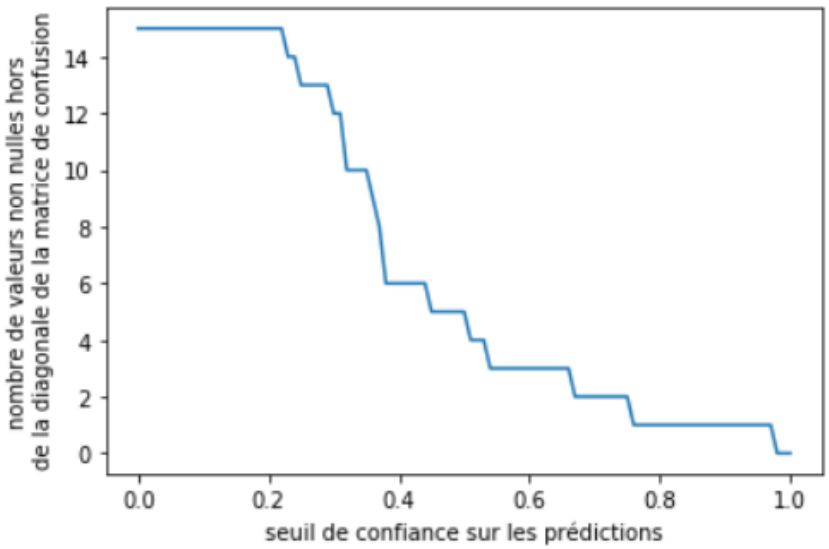

FIGURE 3.13 – Nombre de types d'erreurs réalisées en fonction du seuil de confiance

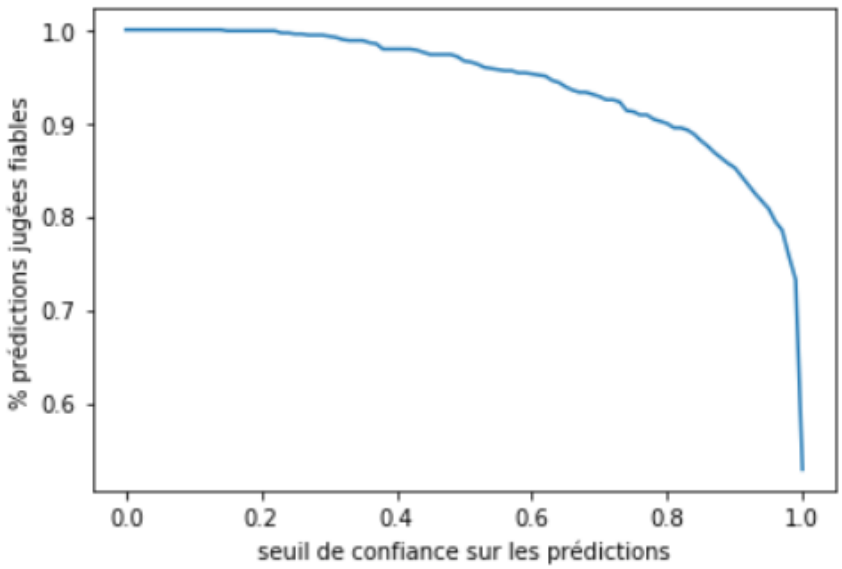

FIGURE 3.14 – Proportion de prédictions fiables en fonction du seuil de confiance

Pour conclure sur cette première problématique, on peut résumer la démarche suivie par le schéma suivant :

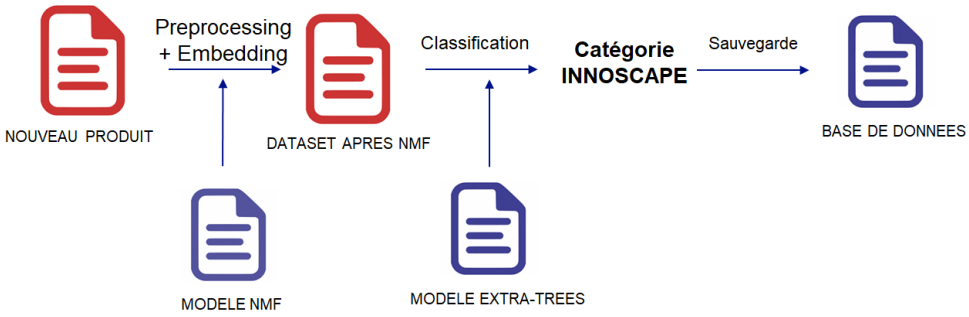

FIGURE 3.15 – Schéma du pipeline de classification des produits

3.2 Problématique 2 : Product Matching

Une fois les produits classés selon les catégories, l'objectif est ici de pouvoir identifier si des produits sont similaires ou identiques au sein des catégories. Par exemple dans la catégorie "Perceuse, perforateur et visseuse filaire", être en mesure de pouvoir identifier qu'un modèle spécifique de perceuse vendu sur Amazon est identique à une perceuse vendue sur un autre site vendeur comme Bricorama.

3.2.1 Données

Nous disposons toujours des mêmes données vendeurs, en prenant en compte pour cette problématique le prix et la marque des produits. Nous avons également la catégorie Innoscape des produits que nous avons établie dans la problématique précédente.

Comme lors de la problématique 1, nous nous heurtons ici à un problème de données non étiquetées. Notre objectif étant d'identifier les produits similaires, nous nous intéressons donc à un niveau de classification plus fin, au niveau produit. Il est alors nécessaire d'établir un catalogue produit dans lequel ces derniers sont identifiés selon un code ou label propre à Innoscape. Disposant déjà de la catégorie des produits, nous avons réfléchi à une manière de définir la codification des produits. Par exemple, dans la catégorie des "Outillage à main" un produit pourrait être définit selon le code OM-XXXX.

Afin de définir le catalogue produit, nous affecterons un code unique (le label produit) à un produit d'une marque donnée et ayant des caractéristiques spécifiques via un algorithme que nous définirons. Plus explicitement, si les caractéristiques d'un produit donné "match" avec celles d'un produit recensé dans le catalogue produit alors celui-ci sera labellisé selon ce code. Sinon un nouveau code sera affecté à ce produit donné, synonyme d'un nouveau type de produit au sein du catalogue.

3.2.2 Crawling sur les données du web

Les données dont on dispose, comme présenté précédemment, concernent seulement les catégories des produits e-commerce et non leurs références produits. Il faut donc pouvoir identifier les produits identiques et leur attribuer une référence.

Pour cela, nous avons décidé d'utiliser un site de comparateur de prix pour les produits e-commerce (idealo.fr) qui possèdent une fiche produit et une page unique pour chaque produit.

En utilisant leurs bases de données de fiches produits à travers la barre de recherche, nous avons réussi à donner une référence unique (page unique d'Idealo) à chacune de nos données catégorisées. Cependant, les résultats de la recherche affichaient parfois plusieurs fiches produits différentes ou bien aucun résultat. Dans le premier cas la recherche n'est pas assez précise et dans le deuxième cas le produit n'est pas référencé dans la base de données ou bien la description est beaucoup trop détaillée.

De plus, le site étant un comparateur de prix, il propose pour une fiche produit plusieurs produits provenant de différents sites e-commerce. Ces différents produits liés à une même référence produit permettront d'établir notre set d'entraînement pour notre modèle de matching de produits par la suite. Les données récupérées sur ces produits sont essentiellement le prix, la description produit et la marque.

3.2.3 Extraction des features / NER

A partir des données vendeurs nous avons retenu les features suivantes : la marque, la description et le prix.

La marque est une donnée importante car elle permet filtrer et identifier les produits similaires, car deux produits identiques sont nécessairement de la même marque. De même que pour la problématique 1 et après traitement, la description contient les termes pertinents pour caractériser un produit et identifier des similarités. Enfin le prix est aussi un indicateur pour identifier un produit.

Nous avons également extrait différents attributs caractérisant les produits comme la couleur, le poids, le volume, le voltage... à partir de la description pour les associer à de nouvelles features par l'utilisation de techniques de NER (Name Entity Recognition). La reconnaissance d'entités nommées est une sous-tâche de l'extraction d'information consistant à rechercher des objets textuels (mots ou groupe de mots) pouvant être classés dans des catégories prédéfinies au sein d'un corpus. Il existe plusieurs méthodes de NER comme l'approche par dictionnaire ou par expressions régulières (regex).

L'approche par dictionnaire consiste à retrouver les catégories d'entités nommées contenues dans le dictionnaire. Cette approche est utile pour reconnaître des catégories d'entités nommés précises. Nous avons retenu cette approche pour extraire la marque de la description. En effet pour certains produits le champ marque n'était pas renseigné, cependant il était possible de retrouver la marque à partir de la description. L'approche par dictionnaire convenait alors pour ce cas de figure car les marques sont des entités spécifiques. Nous avons donc recensé de manière la plus exhaustive possible l'ensemble des marques de notre corpus en les normalisant, c'est à dire que les produits "SIKAFLEX" ou "SIKACERAM" sont normalisés en "SIKA". Il convient aussi de mettre à jour le dictionnaire des marques dans le cas où des produits de nouvelles marques font leur apparitions.

Une autre méthode de NER et d'extraction des entités que nous avons utilisée est fondée sur les expressions régulières. L'utilisation de regex est particulièrement efficace pour extraire les attributs suivant certains patterns, comme les attributs quantitatifs. Par exemple, pour les attributs quantitatifs, on retrouve souvent le pattern suivant : chiffre ou nombre suivi de l'unité de mesure. Nous avons ainsi pu extraire différents attributs comme le poids, les dimensions, le volume, la puissance, le voltage, la couleur, le modèle etc.
Enfin nous avons normalisé les unités de mesure (poids en gramme, volume en litre etc) afin que les données de chaque attribut aient la même unité. L'extraction de features depuis la description par regex nous a donc permis d'obtenir plus d'informations et caractéristiques concernant les produits ainsi que de créer de nouvelles features.

Bien entendu tous les attributs ne sont pas pertinents pour tous les produits. En effet, le voltage ne sera pas un attribut renseigné pour un produit "pot de peinture" et ce champ sera donc vide. La gestion des données manquantes sera donc un enjeu auquel nous

FIGURE 3.16 – Exemple d'extraction des attributs à partir de la description

répondrons plus tard dans un paragraphe consacré à la matrice de similarité.

Enfin après extraction des features nous avons retiré tous les attributs retrouvés par NER de la description, puis nous avons repris le preprocessing de la problématique 1 (tokenization, retrait des stop words et lemmatization), de sorte qu'il ne reste plus que les termes pertinents pour caractériser un produit. Nous avons alors représenté les données textuelles de la description dans un espace d'embedding TF-IDF + NMF (250 topics) comme dans la problématique 1.

A partir de la description et de ces caractéristiques nous pourrons alors étudier les similarités lors de la phase de Product Matching.

3.2.4 Métriques

Afin d'étudier les similarités entre chaque attribut de chaque produit nous avons utilisé et implémenté les distances suivantes :
- Cosine
- Levenshtein
- Boolean
- Euclidienne normalisée

Nous avons utilisé la mesure de similarité Cosine "cosine similarity" de la librairie scikit learn pour calculer la similarité entre les représentations vectorielles (TF-IDF + NMF) des descriptions produits.

Pour mesurer la similarité entre les modèles, la distance de Levenshtein s'est avérée la plus adaptée. La distance de Levenshtein entre 2 mots ou chaînes de caractères A et B correspond au nombre minimal de remplacements, ajouts et suppressions de lettres pour passer du mot A au mot B. Dans notre cas, deux produits dont les modèles extraits sont "680k" et "rsds680k" auront une certaine similarité et donc une distance plus faible car il faut ajouter les caractères "r,s,d,s" pour passer de "680k" à "rsds680k". Ainsi cette mesure fournit une indication sur le degré de ressemblance de ces chaînes.

Pour les variables catégorielles comme la marque et la couleur, nous avons implémenté une distance "Boolean" de la manière suivante :
Dans le cas où les deux variables sont renseignées , si elles sont identiques alors la distance est égale à 0 sinon la distance est égale à 1. Et si au moins une des deux variables n'est

pas renseignée alors la distance est égale à 0.5.
Par exemple deux produits de même marque "Ryobi" auront une distance égale à 0.

Pour les variables quantitatives comme le prix, le poids, les dimensions, la puissance, le voltage... nous avons implémenté une distance "Euclidienne normalisée" de la manière suivante :

$$distance(x_i, x_j) = \sqrt{(\frac{x_i - x_j}{x_i x_j})^2} \tag{3.3}$$

Pour les deux distances implémentées, nous nous sommes assurés que ce sont bien des distances, c'est à dire des applications positives respectant les propriétés de symétrie, séparation et inégalité triangulaire.

3.2.5 Matrice de similarité

Grâce aux distances définies précédemment, on peut créer notre matrice de distances qui va correspondre aux distances entre chaque paire possible de produits et pour chaque attribut. On définit cette matrice pour chacune de nos catégories. Les distances calculées correspondront à nos différentes features et le label au fait que la paire de produits est oui ou non identique. Le problème qu'on rencontre dans la création des matrices de similarité est qu'on ne peut pas toujours calculer les distances pour chaque paire de produit. Ce problème est dû à trois causes principales :

— Dans un premier temps, il peut s'expliquer par le fait que les produits ne sont pas concernés par certains des attributs. Par exemple l'attribut voltage ne sera pas rempli pour des pots de peinture.
— Dans un second temps, la cause est liée à la richesse de la description. Parfois certains produits sont renseignés par les vendeurs avec beaucoup de détails sur les caractéristiques, et d'autres non. On ne peut donc pas récupérer tous ces attributs avec les expressions régulières.
— Dans un dernier temps, il arrive que nos techniques de NER ne permettent pas de récupérer toutes les informations liées aux attributs car elles ne sont pas adaptées à toutes les patterns des descriptions vendeurs.

Concernant le premier cas, on se retrouve avec des attributs complètement non remplis à l'intérieur de chaque catégorie, par exemple le voltage dans la catégorie "Peinture". Il n'est donc pas pertinent de garder ces attributs pour notre set d'entraînement. Nous avons décidé de mettre en place un seuil sur le taux de remplissage des attributs. Si le taux de remplissage se retrouve inférieur à notre seuil, on décide de ne pas considérer la feature pour notre set d'entraînement. Parfois l'attribut est très intéressant à considérer pour une catégorie donnée, cependant il se peut que quelques produits dans la catégorie n'aient pas

d'informations sur celui-ci. On décide dans ce cas de ne pas retirer la feature mais affecter pour les quelques produits concernés la valeur médiane de l'attribut (pour les attributs de nature quantitative) dans la catégorie. On obtient ainsi un set d'entraînement pertinent pour entraîner un classifieur et prédire les produits identiques pour chacune des catégories.

3.2.6 Apprentissage & Classfication

Le schéma global du process d'apprentissage pour notre modèle de classification est le suivant :

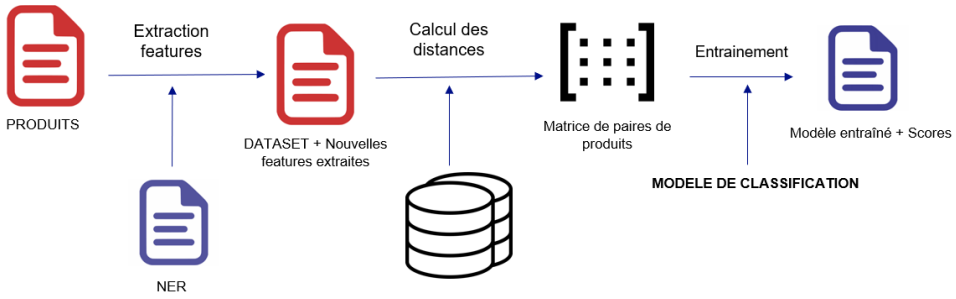

FIGURE 3.17 – Schéma du pipeline d'apprentissage pour le matching de produits

Nous avons testé plusieurs modèles sur ce set d'entraînement en privilégiant les modèles ensemblistes qui donnent très souvent les meilleures performances. Nous avons également tenté différentes représentations vectorielles pour notre description.

Pour évaluer nos modèles, nous nous sommes surtout basés sur la métrique F1-score car nos classes sont très déséquilibrées. En effet, nous avons construit nos datasets en calculant les distances entre chaque paire de produit possible. En prenant cette approche on arrive avec beaucoup de paires dissemblables donc le label « non identique » ou « 0 » est prépondérant.

L'évaluation de nos modèles sur le set de test a donné les résultats suivants :

Représentation vectorielle	Modèle	Rappel	Précision	F1-score
TF-IDF	ExtraTrees	84.0%	82.2%	82.7%
TF-IDF + NMF	LightGBM	85.4%	79,8%	81.9%
TF-IDF + NMF	ExtraTrees	88.4%	84.1%	**85.8%**

On utilise ensuite le modèle entraîné pour prédire la similarité entre les nouveaux produits et les produits déjà existants dans notre base de données. Si le produit est classé comme identique à un des produits existants, on lui associe la référence de ce produit sinon on lui attribue une nouvelle référence produit.

Chapitre 4

Perspectives

4.1 Axes d'amélioration

4.1.1 Critères de sélection des features pour la problématique 2

En l'état, le préprocessing employé pour le traitement de la problématique 2 repose sur deux choix méthodologiques forts :

— Le choix d'ignorer, pour une famille de produits donnée, les features qui sont renseignées pour moins de 60% des produits ;

— Le choix d'utiliser par défaut, lorsqu'une feature pertinente n'est pas renseignée pour un produit donné, la médiane de cette feature sur les produits connus.

Ces partis pris méthodologiques, bien qu'ils puissent naturellement faire l'objet de débats, nous ont paru préférables à des approches fondées sur des règles métiers empiriques, qui, pour chaque famille, prioriseraient les attributs de manière distincte.

Par ailleurs, conscients de ce que l'importance des attributs était susceptible de varier fortement entre les familles, nous avons choisi d'entraîner des classifieurs utilisant des arbres de décision et des méthodes d'ensembles, de telle sorte que le classifieur de chaque famille puisse identifier les attributs les plus pertinents.

Cette méthode nous a permis d'obtenir des résultats satisfaisants dès lors qu'une famille comporte un nombre suffisant de paires de produits pour l'entraînement (de l'ordre de quelques centaines de produits et de quelques dizaines de paires de produits identiques).

Toutefois, on pourrait sans doute améliorer encore les résultats, en ajoutant des features supplémentaires en entrée des classifieurs utilisés pour la problématique 2. Ces features seraient en nombre égal à celui des autres features, et prendraient (par exemple) la valeur 1 si un attribut est renseigné pour les deux produits d'une paire, et la valeur 0 dans le cas contraire. En procédant ainsi, et en retirant les deux choix méthodologiques susmention-

nés, on pourrait alors sélectionner les attributs pertinents par famille de manière moins arbitraire, tout en utilisant au mieux l'information disponible.

4.1.2 Intégration des résultats à une base produits Innoscape

Pour que la solution que nous proposons puisse être pleinement opérationnelle, il est nécessaire que l'expérience acquise en classifiant des produits soit enregistrée et vienne enrichir le jeu de données utilisé pour l'entraînement des produits et la base de comparaison aux produits déjà connus. Ceci est particulièrement important pour fiabiliser les prédictions effectuées sur des familles de produits pour lesquelles on dispose de peu d'échantillons.

A cet effet, trois tâches sont donc nécessaires pour parachever le travail réalisé :

— mettre en place un système de clefs produits propre à Innoscape : en effet, jusqu'à présent, pour effectuer des comparaisons de produits, nous avons utilisé l'url des pages produits obtenues d'Idealo. Il conviendrait de remplacer ce système par un système d'attribution automatique de clefs produits, qui pourraient être de la forme "sku_"+"key_source_sku_"+"matching" (par exemple).
— ajouter de nouveaux éléments à la base produits Innoscape au fur et à mesure ;
— mettre en place une API adaptée aux besoins d'Innoscape, facilitant les corrections manuelles en cas de doute sur la nature ou la référence d'un produit.

4.1.3 Définition de seuils de confiance par famille pour la problématique 2

Comme pour la problématique 1, il est important de signaler tout matching douteux de deux produits, en étudiant la probabilité qu'ils soient identiques, telle que fournie par le classifieur. Il est à noter que ce seuil est à déterminer pour chaque famille, car il dépend notamment de l'importance accordée par les clients d'Innoscape au suivi de telle ou telle famille de produits. Pour ce faire, on pourra utiliser deux approches complémentaires :

— D'une part, une analyse classique de la courbe ROC permet de déterminer un tel seuil, en trouvant le meilleur compromis entre minimisation des taux de faux positifs, et maximisation de vrais positifs (cf figure ci-dessous) :
— D'autre part, il peut être intéressant d'étudier la distribution des probabilités d'identité parmi (i) les vrais positifs et (ii) les faux positifs, l'objectif étant de déterminer dans quelle mesure il est possible de discriminer ces deux catégories de paires de produits sur la base des probabilités d'identité (cf figure ci-dessous).

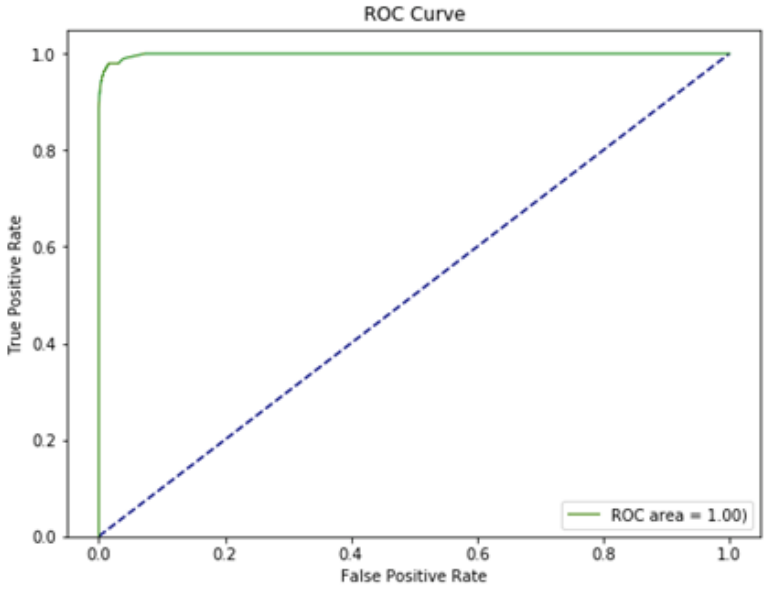

FIGURE 4.1 – Exemple de courbe ROC obtenue pour le classifieur de la problématique 2 pour les perceuses sans fil

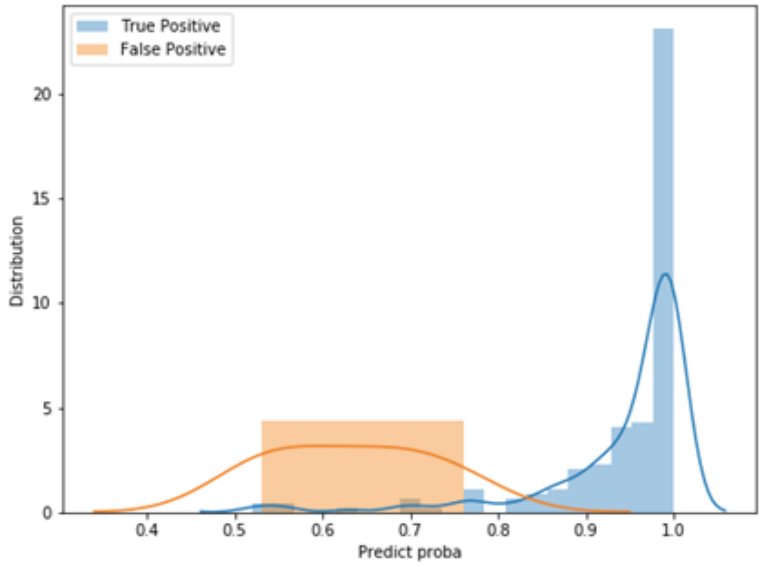

FIGURE 4.2 – Distributions des probabilité d'identité de deux produits analysés par le classifieur - focus sur les vrais positifs (bleu) et les faux positifs (orange)

4.2 Problématique 3 : Détection des changements d'informations sur les produits et mises à jour

Pour rappel, la problématique 3 concerne la prise en compte des modifications des informations dans les marketplaces. L'enjeu pour Innoscape est de détecter ces changements afin de mettre à jour les informations des produits concernés dans les bases de données Innoscape.

En raison de la relative difficulté de se procurer des données labellisées, il ne nous a pas été possible de tester la robustesse de la solution proposée en cas de changement de descriptifs de produits sur des sites e-commerce. Il convient cependant de noter que la solution que nous avons proposée pour la collecte de données labellisées en vue de traiter la problématique 2 pourrait aisément permettre, en effectuant un crawling régulier de comparateurs de produits tels qu'Idealo, de constituer un jeu d'entraînement pour un classifieur destiné à détecter de tels changements de description.

Les modèles entraînés doivent également se mettre à jour face à ces modifications afin de pouvoir prendre en compte les corrections et ajouts d'informations des marketplaces.

Cependant, indépendamment de la méthode choisie pour collecter les données, il sera nécessaire :

— de déterminer une fréquence pertinente de mise à jour des données produits, en lien avec les besoins opérationnels d'Innoscape ;
— et, le cas échéant, d'identifier quelles nouvelles informations doivent être prises en compte : faut-il ne conserver que la dernière version d'un descriptif, au risque de perdre l'information déjà connue ? Si non, comment gérer les erreurs éventuelles commises par les sites d'e-commerce ?

4.3 Multilinguisme

Innoscape exploite les données de sites e-commerces en plusieurs langues, et a formulé le besoin de d'obtenir une solution capable de traiter ces informations. Deux approches peuvent être envisagées :

— développer une solution pour le français, et recourir à la traduction automatique pour les autres langues ;
— développer un modèle par langue.

Compte-tenu du volume des données qui nous ont été fournies (au maximum, 1500 produits par langue étrangère), il ne nous paraît pas possible, en l'état, de développer un modèle par langue, et ce d'autant plus que le vocabulaire employé dans les descriptifs produits est très spécifique et se prête mal à l'utilisation de modèles de langage pré-entraînés.

L'approche fondée sur la traduction en français (ou éventuellement en anglais) paraît donc plus pertinente en l'état. Plusieurs algorithmes de traduction sont disponibles, les plus connus et performants étant Google Translate et DeepL. Cependant ces deux solutions sont payantes à partir d'une certaine quantité de traductions. D'autres modèles gratuits sont mis à disposition mais ne permettent pas d'obtenir les traductions sur toutes les langues avec lesquelles on travaille dans le projet.

Néanmoins, le traitement des langues étrangères introduit quelques difficultés supplémentaires, dont nous ignorons pour l'instant l'impact sur les performances des modèles évoqués plus haut :

— Il faut tout d'abord reconnaître la langue dans laquelle un produit est décrit, et donc entraîner un premier classifieur pour la détection de langue, dont les erreurs éventuelles pourraient peser sur la performance du modèle général.
— Par ailleurs, il faudra vraisemblablement ajuster les seuils de confiance sur les prédictions de chaque famille, pour chaque langue.

Conclusion

Dans le cadre du besoin exprimé par Innoscape, il nous a été demandé dans un premier temps, de catégoriser des produits provenant de sites e-commerce à partir de leur descriptif au sein d'une hiérarchie commune. Ce problème fait l'objet de recherches actives en matière de machine learning, et un grand nombre de techniques existent et sont développées pour le traiter. Au cours de cette première phase du projet, les données textuelles brutes ont été traitées puis transformées afin qu'elles puissent être utilisées pour entraîner un modèle de classification. L'entraînement du modèle a nécessité la mise en place d'un jeu d'entraînement contenant des produits pour permettre au modèle de pouvoir catégoriser tout type de produit. Grâce à une sélection pertinente de l'embedding, du classifieur et à un enrichissement du jeu d'entraînement, on arrive à des performances de classification satisfaisantes (98% de précision, 94% de F1-score). Certains produits restent très complexes à catégoriser pour le modèle et nécessitent encore une aide humaine. Cependant, plus le jeu d'entraînement grandira, plus le modèle deviendra performant pour la tâche de catégorisation.

Dans un second temps, nous devions mettre en place un algorithme de product matching pour identifier les produits qui sont identiques. Cette problématique est un prolongement de la première problématique. En effet, le travail de similarité s'effectue directement au sein des catégories de produit. Cette phase du projet a nécessité le crawling de données supplémentaires afin d'enrichir et avoir plus de paires de produits dans le jeu d'entraînement. Nous avons également utilisé des méthodes de NER (regex) pour extraire et travailler avec des attributs caractérisant les produits. Après un choix de métriques adaptées pour chaque attribut nous avons construit une matrice de similarité dans laquelle on calcul les scores de similarité de chaque attribut pour chaque paire de produit tout en prenant en compte certaines contraintes pour la sélection des attributs. Nous avons alors pu entraîner un classifieur et obtenir de bonnes performances moyennes (85% de F1-score moyen sur l'ensemble des catégories).

Enfin, la dernière problématique concerne la détection des changements d'informations sur les sites e-commerce. Les modèles qui seront mis en place au niveau des deux premières problématiques doivent être entraînées sur des données fiables. Il est donc nécessaire de pouvoir récupérer les données mises à jour qui correspondent souvent à des corrections et

de ré-entraîner nos modèles sur ces données.

Enfin ce projet fil rouge nous a permis de mener à bien un projet Data Science répondant à une problématique métier réelle et universelle. D'un point de vue technique nous avons pu explorer, apprendre et mettre en pratique les méthodes de NLP pour le traitement des données textuelles et la représentation de ces données dans un espace vectoriel. Plus largement cela nous a permis de mettre en perspective notre travail qui s'inscrit dans un processus métier, en réfléchissant par exemple aux métriques d'évaluations les plus adaptée aux besoins métiers.

Remerciements

Nous remercions chaleureusement toutes les personnes qui nous ont accompagnés au cours de ce projet pour leur temps et leurs conseils, et tout particulièrement MM. Kakeuh-Fosso, Draganov, et Jalalzai, ainsi que M. Eagan.

Bibliographie

— T.Mikolov, K.Chen, G.Corrado, J.Dean, Efficient Estimation of Word Representations in Vector Space, arXiv :1301.3781v3 [cs.CL] 7Sep 2013

— T.Mikolov, I.Sustkever, K.Chen, G.Corrado, J.Dean, Distributed Representations ofWords and Phrases and their Compositionality, arXiv :1310.4546v1 [cs.CL] 16 Oct 2013

— X.Rong, word2vec Parameter Learning Explained, arXiv :1411.2738v4 [cs.CL] 5 Jun 2016

— A.Dai, Q.Le, Semi-supervised Sequence Learning, arXiv :1511.01432v1 [cs.LG] 4 Nov 2015

— E.Altszyler, S.Ribeiro, M.Sigman, D Fernandez Slezak, The interpretation of dream meaning : Resolving ambiguity using Latent Semantic Analysis in a small corpus of text, n.56, 2017

— C. Aggarwal, C. Zhai, Mining Text Data, Springer, ISBN 978-1-4614-3222-7, 2012

— Blei, David M. ; Ng, Andrew Y. ; Jordan, Michael I (January 2003). Lafferty, John, ed. "Latent Dirichlet Allocation". Journal of Machine Learning Research.

— S.Essid, A.Ozerov, Unsupervised data decompositions.
— Rodrigo Caye Daudt, Bertrand Le Saux, Alexandre Boulch, Yann Gousseau Détection dense de changements par réseaux de neurones siamois