SCIENCES SORBONNE UNIVERSITÉ

FACULTÉ DES SCIENCES ET INGÉNIERIE

MASTER 2 INFORMATIQUE
PARCOURS DONNÉES, APPRENTISSAGE ET CONNAISSANCES DAC

APPROCHES NLP POUR L'ENRICHISSEMENT DE MÉTA-DONNÉES DANS UN DATALAKE

RAPPORT DE STAGE

ÉTUDIANT :

AKNOUCHE ANIS

TUTEURS :

JULIEN BURET (Zeenea)
BERND AMANN (LIP6)

2021 \ 2022

1 Remerciements

Je tiens à remercier profondément mes deux tuteurs de stage M.Amann côté LIP6 et M.Buret côté Zeenea, pour leur disponibilité et leur suivi. Ils ont tous les deux su me donner des directives et des conseils issus des deux mondes de la recherche et de l'industrie. Et c'est grâce à eux que j'ai pu mener à bien ce stage.

Je tiens aussi à remercier l'ensemble de l'équipe BD du LIP6, notamment M.Naacke pour sa réactivité et ses différentes réflexions constructives. Sans oublier, la merveilleuse équipe de Zeenea qui a su répondre à mes différents besoins au sein de l'entreprise.

Contents

2 Introduction

2.1 Contexte

Zeenea est une entreprise informatique émergente et spécialisée dans le domaine de la gestion des grosses masses de données. Son histoire commence en 2017 où la société à été fondée par Stéphane Jotic, Julien Buret, Guillaume Bodet et Luc Legardeur grâce à l'expérience terrain qu'ils ont pu acquérir en tant qu'experts data.

L'objectif chez Zeenea est de proposer à ces clients une plateforme et des services afin de faciliter l'exploration et l'exploitation des grosses masses de données. Cet ensemble de données massives est désigné par le mot "datalake" qui signifie lac de données. Un datalake correspond à un emplacement de stockage centralisé qui contient des données hétérogènes (bases de données, documents, images, etc) de sources diverses (Amazon s3, BigQuery, Azure Data factory, etc). Ces données désignent l'ensemble des informations produites par l'entreprise ou récoltées chez les clients. Les métadonnées sont des informations qui permettent de décrire d'autres données. L'enrichissement de métadonnées dans un datalake permet de faciliter la compréhension des données pour ainsi les exploiter et les explorer de façon efficace.

Zeenea fournit à ces clients les outils nécessaires pour faciliter l'exploration et l'exploitation de leur patrimoine de données. Deux solutions cloud et complémentaires sont proposées.

La première solution proposée est ***Zeenea Explorer*** qui est une plateforme qui facilite l'exploration des données aux formats divers issues de sources hétérogènes. Un scanner est déployé dans l'infrastructure du client et permet de charger des métadonnées issues des différentes bases de données du client telles que les schémas de données, les noms des tables, les noms des colonnes, les descriptions textuelles des colonnes sans tenir compte des données. Un traitement est fait sur ces métadonnées pour générer des parcours d'exploration personnalisée afin de gagner en efficacité. La plateforme propose via son moteur de recherche des résultats optimisés selon le profil et les préférences des utilisateurs ainsi qu'une suggestion des similarités entre les différents concepts du catalogue de données. La plateforme permet aussi une priorisation des informations en fonction des équipes, car les attentes diffèrent selon les besoins de chaque équipe (un data scientist et un chargé de marketing ne vont pas travailler sur les même données). Zeenea Explorer permet aussi de partager les connaissances sur les différents items du catalogue de données et de leurs cas d'usages à l'aide de fonctionnalités collaboratives diverses et un accès à une documentation qui est mise à jour en temps réel.

La deuxième solution proposée est ***Zeenea Studio*** qui est une plateforme qui permet de facilement gérer, maintenir et enrichir la documentation du patrimoine de données d'un client. La plateforme permet de créer un inventaire de divers types d'objets du catalogue de données (datasets, data processes, business terms, fields, visualizations, etc.). Elle permet aussi de faire un enrichissement de la documentation en détectant les objets similaires dans le catalogue de données. L'ensemble des liens et similitudes générés sur le catalogue de données permet de construire un graphe de connaissance pour améliorer la contextualisation sémantique des objets du catalogue de données et faciliter son exploration.

Le sujet de stage consiste à proposer et développer de nouvelles solutions pour l'enrichissement d'un catalogue de métadonnées textuelles extraites à partir du contenu d'un datalake (documentation techniques, documents, messages, schéma de données, etc). Cet enrichissement consiste dans la construction d'un référentiel métier et l'annotation des métadonnées avec les concepts et les propriétés de ce référentiel.

2.2 Problématique

La plateforme ***Zeenea Studio*** est principalement utilisée pour la gestion de la documentation des différents concepts du catalogue de données. Dans la partie enrichissement de la documentation, des liens de similitudes sont détectées entre les différents objets du catalogue de données pour simplifier leur agrégation . La problématique consiste à suggérer des liens entre des business-items et l'ensemble des colonnes contenues dans le patrimoine de données. Un business-item est défini comme étant un objet contenant un nom et une description textuelle. Il est conçu pour un cas d'usage précis et contient des liens vers divers objets du catalogue de données.

Une des méthodes utilisée par ***Zeenea*** se base sur la similarité syntaxique des noms de business-items et des noms de colonnes du patrimoine de données. Pour cela la distance de Levenshtein est calculée pour mesurer la différence entre deux chaînes de caractères. Elle correspond au nombre minimal de caractères qu'il faut insérer, substituer ou supprimer afin de faire correspondre une chaîne à une autre. Dire que deux mots sont similaires syntaxiquement signifie qu'ils partagent un nombre de caractères positionnés dans un certain ordre. Tandis que la similarité sémantique entre ces deux mots correspond au sens et aux différents concepts qui y sont liés.

2.3 Contributions

L'objectif de ce stage est de développer une méthode qui permet de générer des suggestions de linked-items (liens entre 2 entités) entre les objets du catalogue de données qui partagent les mêmes informations au niveau sémantique. Nous nous intéressons plus précisément à la détection de liens de similarité sémantique entre des *business-items* et les colonnes des différents datasets qui composent le patrimoine de données. Les colonnes des différents datasets sont toutes définies par un nom de colonne et certaines contiennent des descriptions textuelles comme le montre la figure 2.

	table_catalog	table_schema	table_name	column_name	field_path	data_type	description
0	bigquery-public-data	chicago_taxi_trips	taxi_trips	unique_key	unique_key	STRING	Unique identifier for the trip.
1	bigquery-public-data	chicago_taxi_trips	taxi_trips	taxi_id	taxi_id	STRING	A unique identifier for the taxi.
2	bigquery-public-data	chicago_taxi_trips	taxi_trips	trip_start_timestamp	trip_start_timestamp	TIMESTAMP	When the trip started, rounded to the nearest ...
3	bigquery-public-data	chicago_taxi_trips	taxi_trips	trip_end_timestamp	trip_end_timestamp	TIMESTAMP	When the trip ended, rounded to the nearest 15...
4	bigquery-public-data	chicago_taxi_trips	taxi_trips	trip_seconds	trip_seconds	INT64	Time of the trip in seconds.

Figure 2: Métadonnées de la table Chicago_taxi_trips issue bigquery-piblic-data.

Pour traiter cette problématique, nous avons utilisé des méthodes basées sur la Semantic Textual Similarity (STS) ainsi que quelques concepts de recherche d'informations.

La démarche proposée consiste à créer des clusters de colonnes similaires sémantiquement et chaque cluster correspondra à un topic avec une représentation textuelle. Pour créer ces clusters de colonnes similaires, deux approches sont proposées. La première approche (Section 4) consiste à concaténer l'ensemble des noms de colonnes avec leurs descriptions textuelles (si elle sont fournies) qui vont servir à générer des vecteurs embeddings pour le clustering. La deuxième approche (Section 5) consiste à d'abord utiliser que les descriptions textuelles des colonnes pour générer les clusters, ensuite les autres colonnes sans descriptions textuelles sont ajoutées aux clusters correspondant de manière incrémentale. Une fois que les clusters de colonnes similaires sont crées, le modèle va retourner les clusters les plus similaires à chaque business-item. Et grâce aux représentations textuelles des topics, la navigation entre les clusters les plus pertinents sera plus précise. Et pour chaque cluster on retourne les colonnes les plus similaires sémantiquement avec les business-item. Deux niveaux de recherches sont mis en place, le premier permet de retrouver les clusters les plus pertinents et le deuxième niveau permet de retourner les colonnes les plus pertinentes au sein d'un cluster. Enfin, l'utilisateur pourra valider les colonnes les plus pertinentes et créera des linked-items entre ces colonnes et les business-items correspondants.

Le rapport est organisé en neuf sections qui sont décrites ci-dessous:

- La Section 2 (Introduction) présente l'entreprise avec les solutions qu'elle propose et introduit la problématique et le sujet du stage.
- La Section 3 (État de l'art) présente un état de l'art sur des techniques de schema matching, des méthodes d'embedding de mots et de documents ainsi que la détection de topics.
- La Section 4 (Approche 1) présente la première approche (énoncée plus haut) pour générer les clusters de colonnes.
- La Section 5 (Approche 2) présente la deuxième approche (énoncée plus haut) pour générer les clusters de colonnes.
- La Section 6 (Génération de représentations de topics) présente la méthode suivie pour la génération des représentations textuelles des topics.
- La Section 7 (Évaluation) présente la partie évaluation des suggestions de liens générées.
- La Section 8 (Cas d'usage) présente un cas d'usage de l'application (en utilisant l'approche 2) pour un business-item défini.
- La Section 9 (Déploiement) présente le coté déploiement de l'application, avec l'infrastructure utilisée.
- La Section 10 (Conclusion) présente un résumé du travail qui a été effectué ainsi que quelques perspectives.

3 État de l'art

3.1 Schema matching

Le schema matching [2] est un concept général qui décrit les tâches qui permettent de trouver des similarités dans les bases de données en se basant sur la structure des données (le schéma) ainsi que le contenu. Dans ce stage, on s'intéresse aux techniques qui permettent de trouver des attributs (Colonnes) qui sont similaires et qui portent la même information sémantique.

Il y existe de nombreux travaux sur cette problématique et nous nous sommes intéressées à une solution particulière qui était adaptée à la problématique du stage. L'article [12] présente deux méthodes qui permettent de générer des mappings entre les attributs d'un dataset A et les attributs d'un dataset B. La première méthode *(Centroid Method)* consiste à générer des clusters contenant les attributs similaires du dataset A. Ensuite, chaque attribut du dataset B est affecté au cluster le plus similaire contenant les attributs du dataset A et ainsi des liens de mapping sont crées entre les attributs des deux dataset A et B. La deuxième méthode *(Combined Method)* consiste à mélanger les attributs des deux datasets A et B et générer des clusters contenant à la fois des attributs du dataset A et du dataset B. La première méthode basée sur les centroïdes est celle qui a été reprise dans ce stage, car elle assure de faire correspondre à chaque attribut du dataset B un cluster d'attributs du dataset A, contrairement à la deuxième méthode qui peut générer des clusters contenant des attributs issus uniquement d'un seul dataset ce qui ne permettrait pas de faire du mapping entre les attributs des deux datasets.

L'approche proposée dans ce stage s'inspire de l'approche précédente qui est basée sur les centroïdes *(Centroid Method)*.

3.2 Embeddings de mots et de documents

Les approches proposées dans ce stage sont dirigées par l'utilisation de métadonnées textuelles (noms des colonnes et descriptions textuelles). Pour suggérer des linked-items entre les business-items et les différentes colonnes du patrimoine de données, un score de similarité sémantique doit être utilisé. Pour cela, nous avons utilisé le score de similarité cosinus (Définit dans la Section 4.3) qui calcule un score de similarité entre deux vecteurs de réels. C'est pour cela qu'il faut faire correspondre aux colonnes et aux business-items des vecteurs embeddings de dimension définie. Le vecteur embedding est une représentation vectoriel d'une donnée textuelle qui fait en sorte que les mots partageant des contextes similaires possèdent des vecteurs embeddings qui sont relativement proches.

Une des premières représentations vectorielle proposée dans le domaine de la recherche d'information et de la classification de documents est la représentation en *sac de mots (Bag-of-words en anglais)* [18]. Avec cette méthode de vectorisation un texte est représenté par un ensemble de mots et le vecteur correspondant contient les occurrences des mots dans le texte. La taille de ce vecteur dépend de la taille du vocabulaire utilisé. Une variante de cette représentation vectorielle utilise à la place des occurrences des mots une autre méthode de pondération qui permet d'évaluer l'importance d'un mot contenu dans un document par rapport à une collection de documents, qui est le *TF-IDF* pour *term frequency - inverse document frequency* [8]. Le problème avec cette représentation est qu'elle ne tient pas en compte l'ordre des mots dans un texte ainsi que la similarité sémantique entre les mots. En 2014, [6] propose *Word2Vec* qui est un modèle de langue utilisé pour générer des représentations vectorielles des mots qui peuvent encapsuler différentes relations telles que la synonymie ou l'antonymie. L'article [6] présente en détail les deux algorithmes d'entraînement du modèle *Word2Vec* qui sont *Continuous bag of words (CBOW)* et *Skip-Gram*. Le *CBOW* permet de prédire un mots à partir de son contexte qui est représenté par les mots qui l'entoure. Le *Skip-Gram* quant à lui apprend à prédire un ensemble de mots qui représente un contexte à partir d'un seul mot. L'algorithme *Skip-Gram* est plus lent mais donne de meilleurs résultats que le *CBOW*. Peu de temps après, [9] reprend le modèle *Word2Vec* et présente le *Doc2Vec* qui ajoute un vecteur (Paragraph ID) qui correspond au vecteur embedding du document.

(a) Doc2Vec PV-DM model issu de [9]

(b) Doc2Vec PV-DBOW model issu de [9]

Figure 3: Algorithmes d'entraînement de Doc2Vec.

La figure 3a est une extension du modèle *CBOW* qui au lieu d'utiliser que des mots pour prédire un mot, utilise aussi un autre vecteur qui est propre au document. Cette méthode est appelée *Distributed Memory version of Paragraph Vector PV-DM*. La figure 3b correspond au *Words version of Paragraph Vector PV-DBOW* qui est similaire à l'algorithme *Skip-Gram* de *Word2Vec*. Cet algorithme permet de prédire un ensemble de mots à partir du vecteur embedding du document. *Doc2Vec* est un algorithme performant qui est utilisé dans plusieurs taches de traitement automatique du langage naturel mais qui a été dépassé rapidement avec l'arrivée des transformers en 2017.

Un transformers [16] est un modèle d'apprentissage profond qui utilise le mécanisme d'attention qui permet d'identifier les liens qui existent entre les mots, pour ainsi avoir une meilleure compréhension de la structure de la langue et générer des vecteurs embeddings représentatifs de l'information contenue dans les données textuelles. Des modèles basés sur les transformers ont émergé et atteint l'état de l'art dans plusieurs taches en traitement automatique du langage, telles que la traduction de langue, la recherche d'information, l'analyse de sentiment, etc. On peut citer *Bert* [5] pour *Bidirectional Encoder Representation from Transformers* qui est un modèle développé par Google en 2018, un encodeur basé sur les transformers et pré-entraîné sur un large corpus de données comme wikipédia.

Le modèle *MPNet* [13] reprend la méthode d'entraînement masked language modeling (MLM) utilisé avec Bert et la méthode permuted language modeling (PLM) utilisé avec XLNet. [13] présente le MPNet comme une nouvelle méthode de pré-entraînement qui tire bénéfice des deux méthodes MLM et PLM. Le masked language modeling est une méthode qui consiste à masquer des tokens aléatoirement et d'entraîner un modèle à les prédire, le défaut de cette méthode est que les tokens masqués sont prédis indépendamment. Le permuted language modeling consiste à permuter les positions des tokens dans une séquence de mots, positionner les tokens à prédire en fin de séquence et entraîner le modèle à prédire ces mots étant donné un contexte de manière autorégressive, ce qui fait que le modèle ne conditionne que sur le contexte précédant le mot à prédire et n'a aucune information sur la séquence globale. MPNet regroupe ces deux méthodes d'entraînement, ce qui permet de prédire les tokens masqués en prenant compte le contexte qui précède le mot à prédire y compris les tokens prédits et les tokens masqués (en utilisant leurs positions). La prédiction des tokens masqués est conditionnée par l'ensemble des tokens de la séquence, ce qui fait que le MPNet prend plus d'information lors de la prédiction d'un token masqué. Dans la partie expérimentale de [13], le $Bert_{Base}$ a été repris, constitué de 12 couches de transformers, état caché de dimension 768. Le modèle dépasse Bert, XLNet et RoBERTa dans plusieurs taches de compréhensions du langage naturel *(Natural Language Understanding - NLU)*.

Dans la partie expérimentale, nous avons comparé la génération d'embeddings avec Word2Vec, une représentation sous forme de sac de mots, MPNet et quelques modèles distillés issus de BERT selon la qualité des clusters générés.

3.3 Détection de thématiques

La génération des clusters de colonnes issues du patrimoine de données consiste à regrouper des colonnes selon les thématiques (topics) correspondantes. Chaque cluster correspond à un ensemble de documents (colonnes) partageant la même thématique. Ces thématiques sont extraites selon le contenu textuelle des colonnes (nom de colonne ou description textuelle).

Latent Dirichlet Allocation (LDA) [3] est une méthode fréquemment utilisée pour cette tâche. Il s'agit d'une méthode probabiliste qui modélise à la fois la distribution des thématiques dans un document et la distribution des mots utilisés pour représenter une thématique. Elle prend en entrée un ensemble de documents textuels sous forme de sac de mots (BoW).

En particulier la représentation des mots sous BoW reste symbolique et ne prends pas en compte les similarités sémantiques entre deux données textuelles ainsi que l'ordre des mots. Par exemple, avec la représentation BoW une description textuelle a le même vecteur embedding selon n'importe quel ordre des mots. Tandis qu'avec un modèle de langue, pour chaque ordre

des mots le vecteur embedding est différent. BerTopic est un modèle qui tire parti des word embeddings générés à l'aide de modèles de langue basés sur les transformers. [7] présente le BERTopic comme un modèle de détection de thématiques qui génère des clusters de documents textuels partageant le même topic. Le modèle reprend la même architecture que Topic2Vec [11], en utilisant UMAP (Uniform Manifold Approximation and Projection) [10] pour la réduction de dimension et HDBSCAN (Hierarchical Density-Based Spatial Clustering for Application with Noise) [4] pour la génération des clusters. BerTopic se base sur le *Cluster-based TF-IDF* pour la génération des représentations de topics contrairement au Topic2Vec qui est basé sur la proximité des word embeddings des mots avec les vecteurs centroïdes des clusters aux quels ils appartiennent.

L'architecture du modèle BertTopic a été reprise dans ce stage afin d'améliorer la qualité des clusters générés et avoir des représentations textuelles des thématiques détectées.

4 Approche 1 : Clustering des noms de colonnes concaténés avec leurs descriptions textuelles

4.1 Description de l'approche 1

Dans cette première approche, nous avons pris en considération le noms des colonnes avec leurs descriptions textuelles. Une colonne sera représentée comme la concaténation du nom de colonne et de sa description textuelle si elle est fournie. Un nettoyage est effectué sur ces données textuelles (partie détaillée dans la Section 4.2). À l'aide d'un modèle de langue, un vecteur embedding est généré pour chaque colonne (En utilisant sa représentation textuelle). Puis, nous avons appliqué plusieurs méthodes pour partitionner ces données et générer des clusters de colonnes. Un cluster doit contenir un ensemble de colonnes appartenant à la même thématique. Ainsi chaque cluster sera représenté par une thématique.

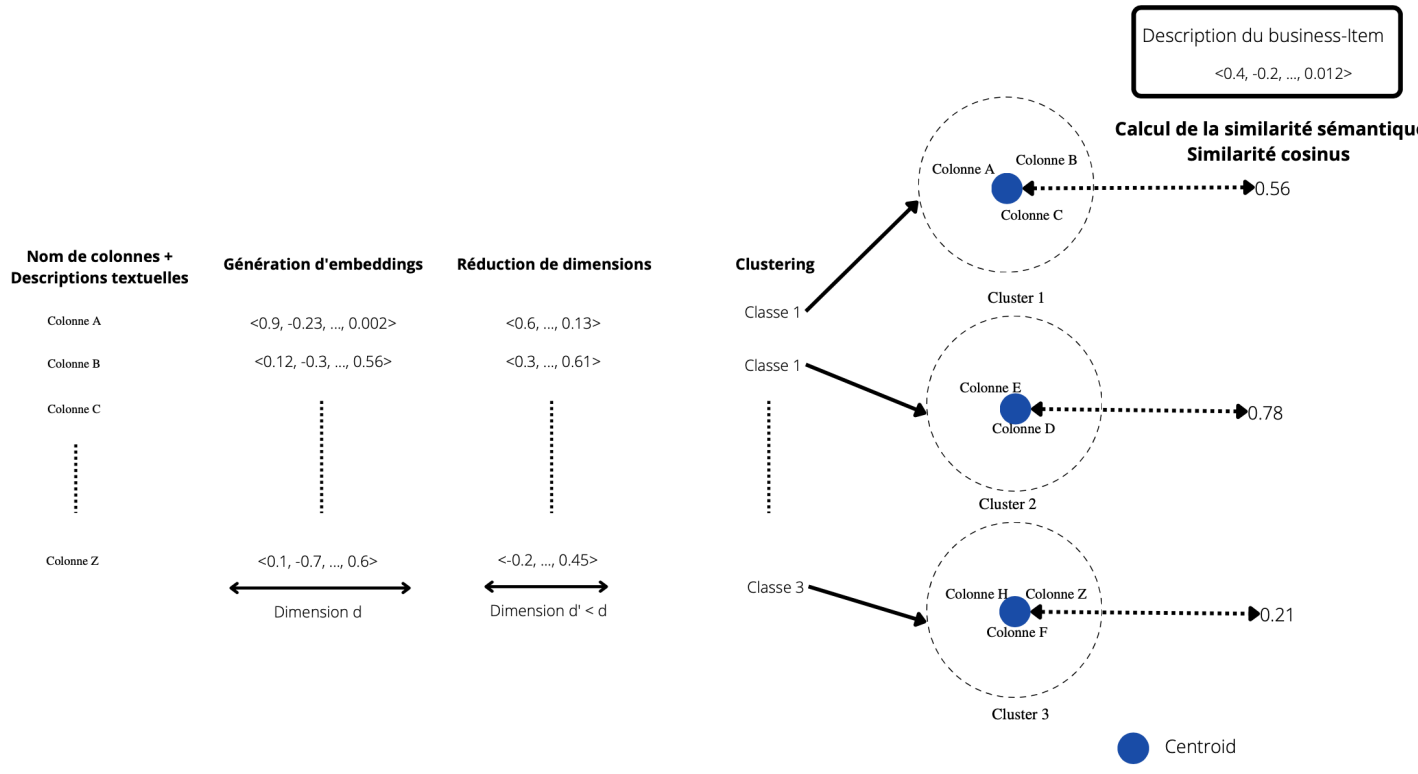


Figure 4: Schéma de l'approche 1 (Clustering et création des linked-items).

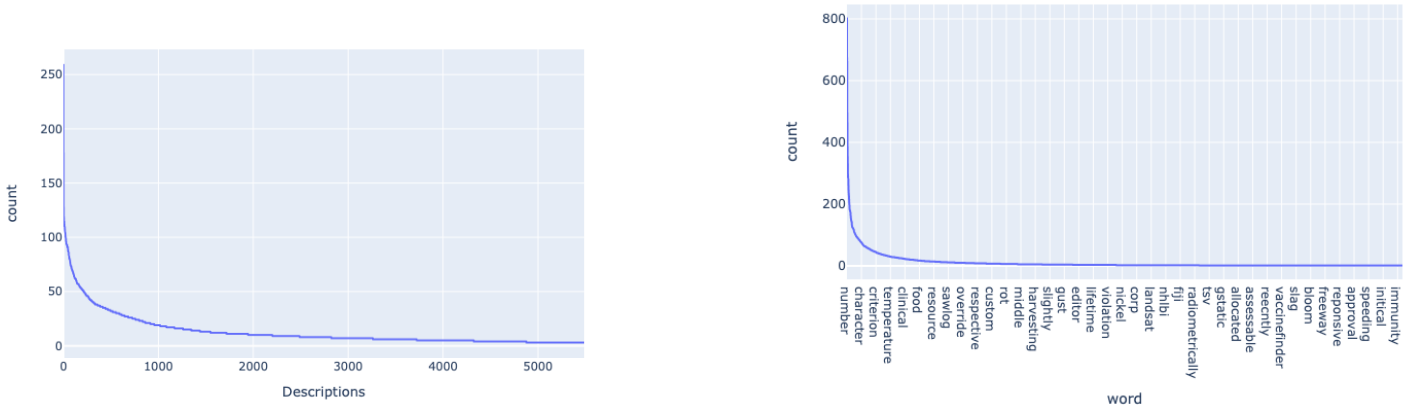
Après, le partitionnement des colonnes , on génère des vecteurs centroïdes pour chaque cluster. Un vecteur centroïde de cluster correspond à la moyenne des vecteurs embeddings des colonnes qui sont contenues dans ce cluster. On génère aussi un vecteur embedding pour le *business-item* à l'aide de sa description textuelle. Puis, on calcule un score de similarité sémantique avec la similarité cosinus entre le vecteur embeddings du *business-item* et les vecteurs centroïdes des clusters. On retourne les **N** clusters les plus similaires avec le *business-item*. Et pour chaque cluster, on renvoie les **M** colonnes qui ont un score de similarité cosinus avec le *business-item* supérieur à une valeur **V**. Les colonnes retournées correspondent aux

colonnes similaires sémantiquement avec le *business-item*. Enfin, l'utilisateur sélectionne les colonnes les plus pertinentes parmi celles retournées et crée ainsi des linked-items entre ces colonnes et le *business-item*.

4.2 Dataset et statistiques

Les données utilisées sont issues de la plateforme BigQuery (https://cloud.google.com/bigquery) . BigQuery est un entrepôt de données et un service de Google Cloud Plateforme GCP (https://cloud.google.com). Cette plateforme contient des projets qui sont décomposés en datasets contenant des données et des méta-données à accès et utilisation libre. Pour nos expériences, nous avons choisi 9 projets qui font un total de 257 datasets et 258 813 colonnes.

On s'intéresse plus particulièrement au noms de colonnes et aux descriptions textuelles. Après un nettoyage des données qui consiste en la suppression des noms de colonnes et descriptions dupliquées ou vides, un traitement NLP a été appliqué aux noms de colonnes et aux descriptions textuelles qui consiste en la mise en minuscule, un filtrage pour récupérer uniquement les mots, suppression des stopwords et lemmatization. En gardant aussi que les descriptions qui contiennent plus de deux mots. Après cette étape de nettoyage, on obtient 5488 colonnes avec descriptions textuelles et 12524 colonnes sans descriptions textuelles.

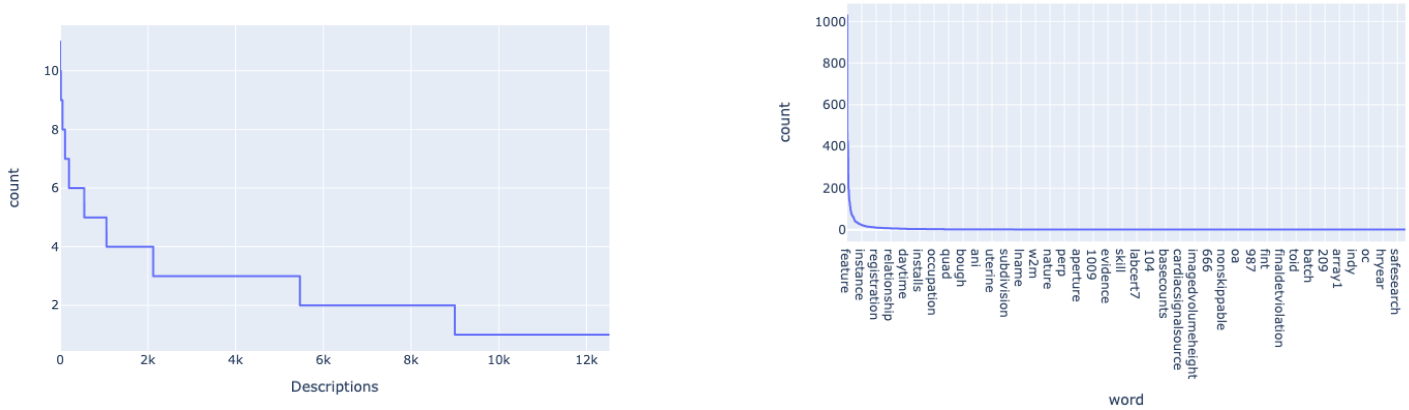


(a) Nombre de mots par description.

(b) Fréquence des mots du vocabulaire.

Figure 5: Quelques statistiques sur les mots des descriptions textuelles.

De la Figure 5a, on remarque qu'environ 95 % des descriptions textuelles contiennent moins de 50 mots. Le nombre de mots maximum dans une description textuelle est de 234, le minimum est de 3 mots et la moyenne est de 13 mots par description. La taille du vocabulaire des descriptions textuelles est de 5866 et on remarque sur la Figure 5b que la fréquence des mots suit bien la loi de Zipf [19] qui stipule que la fréquence d'apparition des mots dans un texte est liée inversement à son rang.



(a) Nombre de mots nom de colonne.

(b) Fréquence des mots du vocabulaire.

Figure 6: Quelques statistiques sur les mots des noms de colonnes sans descriptions textuelles.

Sur la Figure 6a, on remarque que plus de 83 % des colonnes sans descriptions textuelles contiennent un nombre de mots dans leur nom de colonnes qui est égal ou inférieur à 3 mots. Le nombre maximum de mots dans le nom d'une colonne est de 11, le minimum est de 1 et la moyenne est de 2 mots. La taille du vocabulaire des mots des noms de colonnes sans descriptions textuelles est de 7288 et on remarque sur la Figure 6b que la fréquence des mots de ce vocabulaire respecte aussi la loi de Zipf.

4.3 Métriques de mesure de qualité des clusters

Les métriques utilisées pour comparer les différents partitionnements et mesurer leurs qualités :

- **Similarité cosinus :**
  Permet de mesurer la similarité sémantique entre 2 données textuelles grâce à leurs vecteurs embeddings.

$$cos(\theta) = \frac{< X_1 , X_2 >}{\|X_1\|_2 . \|X_2\|_2} \in [0, 1]$$

  Plus la valeur est proche de 1 plus les embeddings sont similaires et inversement.

- **Inertie Intra-cluster :**
  Permet de mesurer l'homogénéité des données dans les clusters. Elle correspond à la moyenne des moyennes des similarités cosinus des embeddings avec leur centroïde (moyenne des embeddings contenus dans un cluster). Plus la valeur est proche de 1 plus les données dans les clusters sont homogènes et similaires sémantiquement et plus la valeur est proche de 0 plus les données dans les clusters sont hétérogènes et non similaires sémantiquement.

$$\frac{1}{N_c} . \sum_{i=1}^{N_c} \frac{1}{N_i} . \sum_{j=1}^{N_i} Sim\_Cos(C_j , \frac{1}{N_i} \sum_{k=1}^{N_i} C_k) \in [0, 1]$$

Sim_Cos : Représente la similarité cosinus
N_c : Nombre de clusters
N_i : Nombre d'éléments dans le cluster i
C_j : Colonne j

- **Inertie Inter-cluster :**
  Permet de mesurer la séparabilité des clusters. Elle correspond à la moyenne des scores de similarité cosinus entre les centroïdes des clusters et le point d'inertie de tout l'ensemble des embeddings. Le point d'inertie correspond à la moyenne de l'ensemble des vecteurs embeddings. Plus la valeur est proche de 0 plus les clusters seront bien séparés (ils ne partagent pas de similarité sémantique) et plus la valeur est proche de 1 plus les clusters sont similaires sémantiquement, ce qui signifie que les clusters ne sont pas bien séparés.

$$\frac{1}{N_c} . \sum_{i=1}^{N_c} Sim\_Cos(\frac{1}{N_i} \sum_{k=1}^{N_i} C_k , \frac{1}{N} \sum_{j=1}^{N} C_j) \in [0, 1]$$

Sim_Cos : Représente la similarité cosinus
N_c : Nombre de clusters
N_i : Nombre d'éléments dans le cluster i
N : Nombre d'éléments dans tous les clusters
C_x : Colonne x

- **Rapport d'inertie :**

  Représente la valeur du rapport et permet de fusionner les deux inerties *intra-cluster* et *inter-cluster*, de telle sorte que plus la valeur du rapport est grande, plus le partitionnement est bon. Le rapport augmente lorsque le score d'inertie intra-cluster augmente et le score d'inertie inter-cluster diminue.

$$\frac{Inertie\, Intra - cluster}{Inertie\, Inter - cluster}$$

- **Taux d'outliers :**
  Représente le taux d'éléments classés comme étant des valeurs aberrantes ou du bruit selon un algorithme de clustering.

- **Nombre de clusters :**
  Le nombre de clusters est lié à la taille du dataset et au nombre de type de relations que partage les données dans le dataset. Un grand nombre de clusters permet de déduire que les données sont hétérogènes et peuvent être décomposées selon plusieurs groupes.

4.4 Expérimentations (Clustering) et résultats

Dans cette section, on va expérimenter quelques méthodes pour le partitionnement des colonnes selon l'approche 1 :

4.4.1 K-MEANS

Le K-Means est un algorithme de clustering qui utilise la distance euclidienne pour faire correspondre à des vecteurs le cluster qui correspond. Le nombre de clusters doit être spécifié en entrée au modèle.

Dans cette expérience, on va d'abord itérer le K-Means pour un nombre de clusters allant de 2 à 500 clusters sur les vecteurs embeddings de l'ensemble des colonnes du patrimoine de données. Puis, on va appliquer la méthode de l'Elbow qui retourne le nombre de clusters optimal selon un score d'inertie.

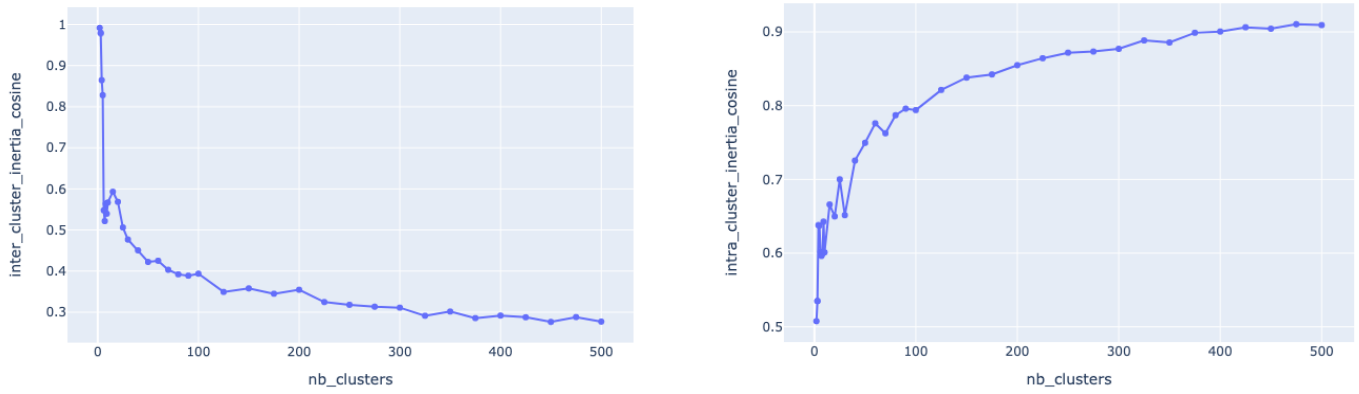
Les vecteurs embeddings des colonnes (Nom de colonne + description textuelle si elle est fournie) ont été générés par le modèle Doc2Vec (nombre de dimensions : 384, nombre d'epochs : 40, algorithme d'entraînement : distributed memory (PV-DM)).

K-Means : Issu de Scikit-Learn (sklearn.cluster.KMeans)

Méthode de l'Elbow : Issue de Yellowbrick qui étend l'api Scikit-Learn (yellowbrick.cluster.KElbowVisualizer)

4.4.1.1 Clustering avec K-MEANS (Un nombre de cluster allant de 2 à 500)

On observe l'évolution des scores d'inertie (inter-cluster et intra-cluster) en utilisant le K-MEANS pour différents nombre de clusters :



(a) Score d'inertie inter-cluster.

(b) Score d'inertie intra-cluster.

Figure 7: Évolution des scores d'inertie selon le nombre de cluster.

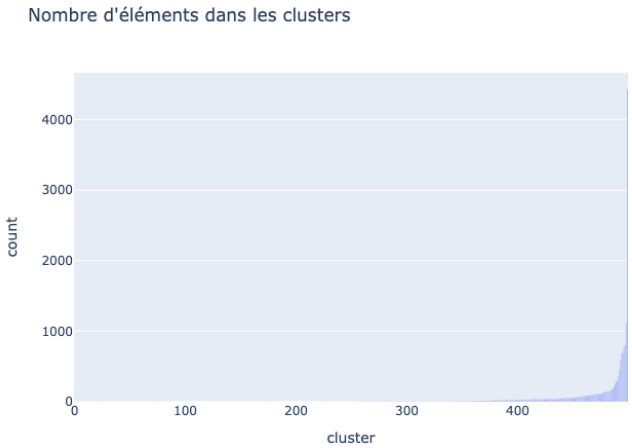


Figure 8: Nombre d'éléments dans chaque cluster pour un partitionnement avec K = 500 clusters. (Ordre croissant)

- Cardinalité minimale : 1
- Cardinalité maximale : 4432
- Cardinalité moyenne : 36

- Inertie inter-cluster : 0.28
- Inertie intra-cluster : 0.90

On peut remarquer sur la figure 7a et 7b que plus le nombre de clusters augmente, plus les performances sont meilleures en termes de score d'inertie inter-cluster (0.28) et intra-cluster (0.90), obtenue avec k = 500 clusters. Si on suit ce raisonnement, on atteindra les meilleures performances lorsque le nombre de clusters sera égale au nombre de colonnes, et donc on aura un nombre d'éléments par cluster qui va converger vers 1. Selon la figure 8 on peut voir clairement qu'il y a environ 450 clusters qui contiennent un très faible nombre d'éléments. La majeure partie des données est répartie sur environs 50 clusters.

4.4.1.2 Méthode de l'Elbow pour trouver le nombre optimal de clusters (KElbowVisualizer de yellowbrick)

La méthode de l'Elbow consiste à trouver le nombre optimal de cluster K pour lequel l'algorithme des K-Means obtient le meilleur partitionnement. Un score de distorsion (ou autre) est calculé selon un certain nombre de clusters. Ensuite, l'algorithme renvoie la valeur k qui correspond au nombre optimal de clusters formé par le coude de la courbe de l'évolution du score de distorsion selon le nombre de clusters.

Le score de *distorsion* correspond à la moyenne des différences quadratiques entre les vecteurs embeddings d'un cluster et le vecteur centroïde du cluster.

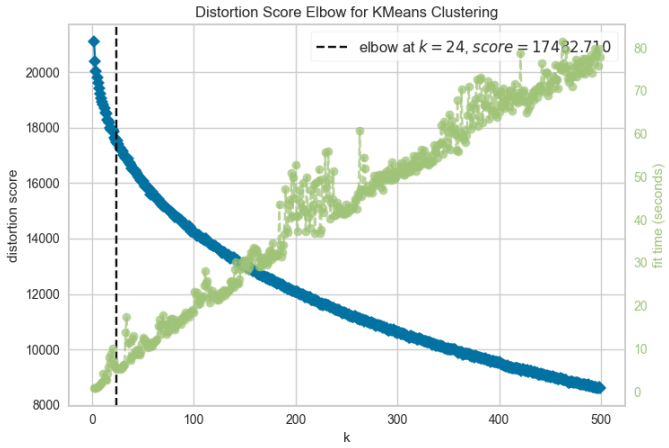


Figure 9: Évolution du score de distorsion (blue) et du temps d'exécution (vert) selon le nombre de clusters .

Selon la figure 9, plus le nombre de clusters augmente, plus le score de distorsion diminue. Cela signifie que la distance entre chaque élément est son centroïde diminue.

Le temps d'exécution du clustering K-Means évolue linéairement pour un nombre de clusters allant de 2 jusqu'à 500 clusters.

Selon la méthode de l'Elbow le nombre optimal de clusters est de 24 clusters dont le score moyen de distorsion est de 17 482.710.

Clustering K-Means avec k = 24 clusters :

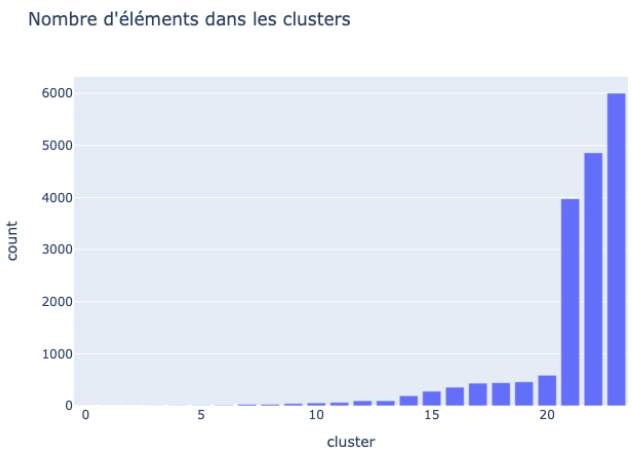


Figure 10: Nombre d'éléments dans chaque cluster.

- Cardinalité minimale : 1
- Cardinalité maximale : 6001
- Cardinalité moyenne : 750

- Inertie inter-cluster : 0.52
- Inertie intra-cluster : 0.68

Pour k = 24 clusters, le nombre d'éléments dans chaque cluster n'est pas uniforme. Et on peut remarquer qu'il y a 3 clusters qui contiennent plus de 80% des éléments. Avec un score d'inertie intra-cluster de 0.68 qui signifie que les données dans les clusters sont assez homogènes et un score d'inertie inter-cluster de 0.52 qui signifie que les clusters ne sont pas très bien séparés.

L'algorithme K-Means n'est pas adapté aux données non linéairement séparable. En plus du fait qu'il y a une forte influence des données aberrantes.

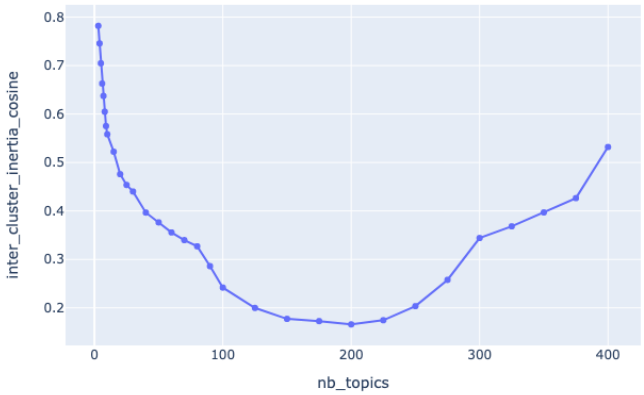
4.4.2 Latent Dirichlet Allocation LDA

Le Latent Dirichlet Allocation est un modèle probabiliste et génératif qui est utilisé pour la détection de thématiques (Topic modeling) dans un ensemble de documents. Il permet de modéliser la distribution des topics dans un document ainsi que la distribution des mots dans un topic. LDA utilise la représentation vectorielle sous forme de sac de mots (BoW) des données textuelles. Le nombre de topics doit être spécifié en entrée au modèle.

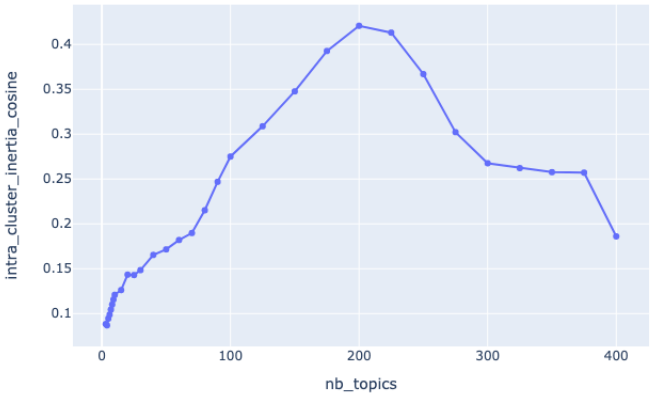
Dans cette partie, nous utilisons le LDA pour générer des clusters de colonnes. Où on fait correspondre chaque cluster à un topic. Et chaque colonne est affectée au cluster correspondant au topic le plus présent dans la représentation textuelle de la colonne (Nom de colonne + description textuelle si elle est fournie).

LDA : Issu de Scikit-Learn (sklearn.decomposition.LatentDirichletAllocation)

Avec un nombre de topics allant de 2 à 500 topics, on peut observer les scores d'inertie suivants pour tirer le nombre optimal de clusters :



(a) Score d'inertie inter-cluster.



(b) Score d'inertie intra-cluster.

Figure 11: Évolution des scores d'inertie selon le nombre de topics.

Sur la figure 11, on remarque qu'avec un nombre de topics égale à 200, on obtient les meilleurs scores en terme d'inertie intra-cluster et inter-cluster. Avec un score d'inertie inter-cluster de 0.16 qui signifie que les clusters sont très bien séparés et un score d'inertie intra-cluster de O.42 qui signifie que les clusters ne sont pas très homogènes.

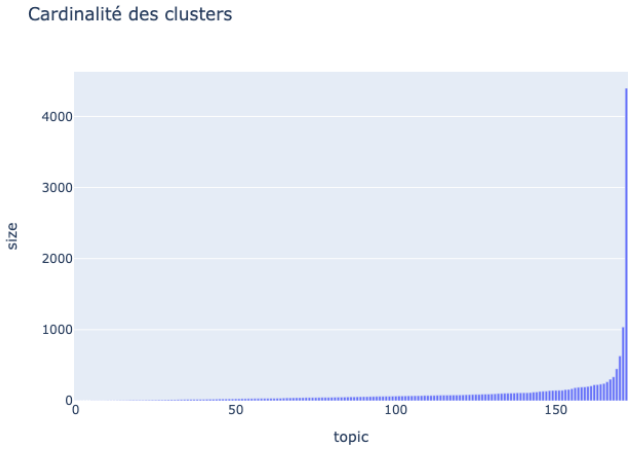


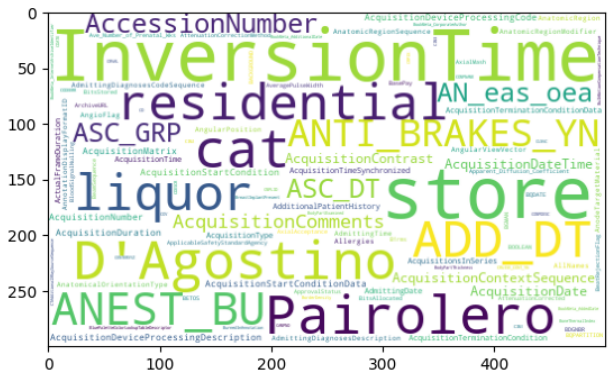
Figure 12: Évolution du nombre d'éléments dans un topic.

- Cardinalité minimale : 1
- Cardinalité maximale : 4397
- Cardinalité moyenne : 104

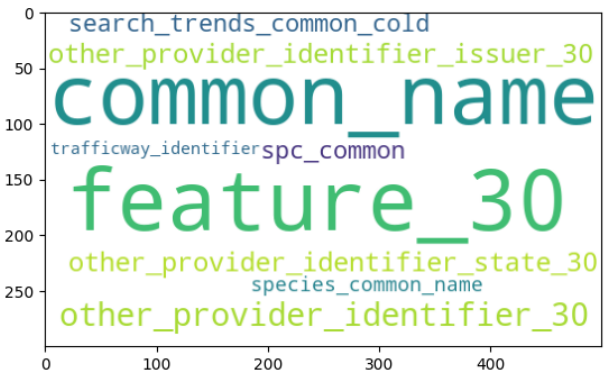
- Inertie inter-cluster : 0.16
- Inertie intra-cluster : 0.42

Sur la figure 12, on observe qu'il y a beaucoup de clusters qui contiennent un nombre d'éléments très faible et très peu de clusters qui contiennent un nombre important d'éléments. Cela est dû au fait que le dataset utilisé contient des données de divers domaines issus de plusieurs datasets hétérogènes.

Word Cloud de 2 topics :



(a) WordCloud topic 1



(b) WordCloud topic 2

Figure 13: Contenu de 2 topics sous WordCloud.

On peut très bien voir sur la figure 13 que les deux topics sont bien séparés, pas de colonnes partagées entre ces deux topics. Par contre, le contenu des clusters n'est pas très homogène, les colonnes dans le même cluster ne partagent pas une forte similarité, cela est dû au données bruit qui ne sont pas traitées. Ce qui reflète les scores d'inertie précédemment obtenus.

4.4.3 BerTopic

Bertopic est utilisé pour la génération de clusters et détection de thématiques. Après la génération des vecteurs embeddings des données textuelles à l'aide de transformers (Des modèles basés sur Bert), une réduction de dimension est appliquée avec UMAP (Uniform Manifold Approximation and Projection) à 5 dimensions. Puis un clustering hiérarchique HDBSCAN (Hierarchical Density-Based Spatial Clustering for Application with Noise) est appliqué aux embeddings de dimension réduite. Chaque cluster correspond à un topic. Les représentations des topics sont générées en utilisant le Cluster-based TF-IDF qui est un score TF-IDF basé sur les clusters (décrit dans la Section 6).

UMAP pour Uniform Manifold Approximation and Projection est une méthode de réduction de dimensions. La méthode peut être vue comme une amélioration de la TSNE (T-Distributed Stochastic Neighbor Embedding) [15], de telle sorte que UMAP préserve mieux les structures globales et locales des données projetés et le temps d'exécution est beaucoup réduit. L'algorithme prend en entrée 2 paramètres importants qui sont le nombre minimum de points à considérer lors du parcours de voisinage et la distance minimale des éléments dans l'espace de projection. UMAP génère une sorte de graphe qui lie des formes géométriques qu'on appelle simplexe. Un simplexe est défini comme une généralisation du triangle avec une dimension quelconque (1-simplexe : segment, 2-simplexe : triangle, 3-simplexe : tétraèdre, etc). Le n-simplexe est l'objet géométrique dans un plan à n dimensions formé en partant d'une origine vers n points. Une fois que le graphe des simplexes de l'espace initial est généré, UMAP essaie de retrouver ces simplexes dans l'espace de projection tout en gardant la structure globale des éléments.

Nous avons utilisé la même version de UMAP présente dans BerTopic qui est issue de la bibliothèque UMAP-LEARN (https://umap-learn.readthedocs.io/en/latest/). Nous avons aussi utilisé un nombre de voisins égale à 15 avec une distance minimale entre les points dans l'espace de projection qui est de 0 (Selon les paramètres utilisés dans BerTopic).

HDBSCAN pour Hierarchical Density-Based Spatial Clustering for Application with Noise. C'est un algorithme de clustering basé sur l'estimation de densité. Le coeur de l'algorithme est le Single-linkage clustering qui est une méthode de clustering hiérarchique qui correspond au fait de grouper des clusters de manière ascendante, de telle sorte que chaque paire de clusters qui sont proches vont être fusionnés pour en faire un seul cluster (En partant de l'état où chaque élément correspond à un cluster). C'est une méthode qui est très sensible au bruit, c'est pour cela qu'il faut définir un moyen de reconnaître des zones bruitées du reste des données. Ci-dessous le fonctionnement du modèle :

- **Définition des métriques à utiliser :** Les zones de forte densité correspondent aux zones où les éléments sont très proches et les zones de faible densité correspondent aux zones où les éléments sont très distants les uns des autres.

Pour cela une distance d'accessibilité mutuelle a été définie (Équation (1)) avec $core_k(x)$ qui correspond au rayon du cercle de centre l'élément x qui contient les k plus proches voisins de x, tandis que $d(a, b)$ correspond à la distance euclidienne entre a et b. Ce qui fait que la distance d'accessibilité mutuelle entre deux points dispersés va correspondre à leur distance euclidienne et entre deux points denses va correspondre à une des $core_k(x)$ distance. Et plus le nombre de voisins k augmente plus les zones denses vont s'élargir ce qui va réduire le taux de données bruits et des outliers.

$$d_{mreach-k}(a, b) = max\{core_k(a), core_k(b), d(a, b)\} \quad (1)$$

- **Construction de l'arbre couvant de poids minimal :** Les données sont représentées sous forme de graphe pondéré où chaque donnée correspond à un noeud, avec une arête entre chaque 2 noeuds avec un poids qui correspond à la distance d'accessibilité mutuelle entre ces 2 noeuds. Le graphe est ensuite crée et contient le nombre minimum d'arêtes de telle sorte qu'en enlevant une arête, le graphe n'est plus connexe, en plus du fait que la somme des poids des arêtes doit être minimale. Et cette structure de graphe correspond à un arbre couvrant de poids minimal.
- **Clustering hiérarchique :** L'algorithme utilise le single-linkage clustering pour créer une hiérarchie de clusters en utilisant l'arbre couvrant de poids minimal. Un tri descendant est effectué sur les poids des arêtes du graphe. Au début, il considère l'ensemble des noeuds du graphe connexe comme appartenant à un seul cluster. Puis, en supprimant l'arête de poids maximal de l'arbre couvrant, deux composantes connexes peuvent être générées, et chacune d'elle représente un cluster. Ainsi de suite jusqu'à itérer sur l'ensemble des arêtes et obtenir un nombre de clusters égal aux nombre de noeuds de l'arbre couvrant utilisé.
- **Condenser l'arborescence des clusters :** Cette étape reprend l'arborescence des clusters crée précédemment et enlève de l'arborescence les clusters ayant un nombre minimum d'éléments inférieur à un seuil qui est prédéfini. Pour ne garder que les clusters les plus pertinents et considérer les données qui ne font pas partie d'un cluster comme étant du bruit.
- **Extraction des clusters :** Arriver à cette étape, le but est de sélectionner le partitionnement le plus pertinent et contenant les clusters les plus persistants. Une nouvelle métrique est définie et correspond à $\lambda = \frac{1}{distance}$. Grâce à cette métrique, on peut définir la persistance d'un cluster avec λ_{birth} correspondant au moment de création du cluster et λ_{death} correspondant au moment où le cluster est décomposé en d'autres clusters. Selon différentes distances de seuil dans l'arbre couvrant, des noeuds se détachent de certaines composantes connexes et donc ne font plus partie de certains clusters. λ_p est défini comme le moment où le noeud p quitte un cluster. Pour chaque cluster, un score de stabilité est calculé et correspond à la somme des durées de vies des points dans les clusters $\sum_{p \in cluster}(\lambda_p - \lambda_{birth})$. Les clusters les plus stables sont retournés.

L'algorithme HDBSCAN prend en entrée un paramètre important qui est le nombre d'éléments minimum dans un cluster. Dans les expérimentations qui suivent, nous avons utilisé un nombre d'éléments minimum dans un cluster égale à 15 éléments. Dans l'implémentation du BerTopic (https://github.com/MaartenGr/BERTopic) le nombre d'éléments minimum correspond à 10 éléments (Valeur par défaut). Plus le nombre d'éléments minimum dans un cluster augmente, plus le nombre de clusters diminue ce qui réduit le bruit.

4.4.3.1 Modèles de génération de documents embeddings

Nous avons choisi des modèles de langue issus de sentence-transformers (https://www.sbert.net/index.html) pour la génération des vecteurs embeddings pour chaque colonne. Sentence-transformers est un framework python qui regroupe les modèles qui ont atteint l'état de l'art dans des tâches utilisant les embeddings de phrase, texte et image.

Quatre modèles de langue ont été utilisés pour générer les documents embeddings des colonnes. Ils sont tous entraînés de façon à générer des vecteurs embeddings contextualisés, afin de faire du clustering ou de la recherche d'information. Trois modèles de langue distillés et basés sur Bert (all-MiniLM-L12-v2, paraphrase-multilingual-MiniLM-L12-v2, all-distilroberta-v1) qui sont légers et rapides. Ainsi que le modèle all-mpnet-base-v2 qui utilise la méthode d'entraînement MPNet (décrite dans la section 3.2).

Ces modèles sont issus de la bibliothèque *HugginFace* (https://huggingface.co). HugginFace est une communauté et une plateforme de science des données. Elle permet de construire, entraîner et déployer des modèles de machine learning et deep learning.

Les modèles qui contiennent au début de leur nom **"all-"** sont entraînés sur environs 1 milliard de paires de séquences pour des tâches de similarité sémantique textuelle STS (Avec des datasets pour le Question-Answering, le Paraphrasing, etc). Les modèles sont décrits ci-dessous :

- **all-MiniLM-L12-v2** : C'est un modèle distillé et basé sur le MiniLM-L12-H384 qui contient 12 couches de transformers avec un état caché de taille 384, le modèle de base est présenté dans l'article "MINILM: Deep Self-Attention Distillation

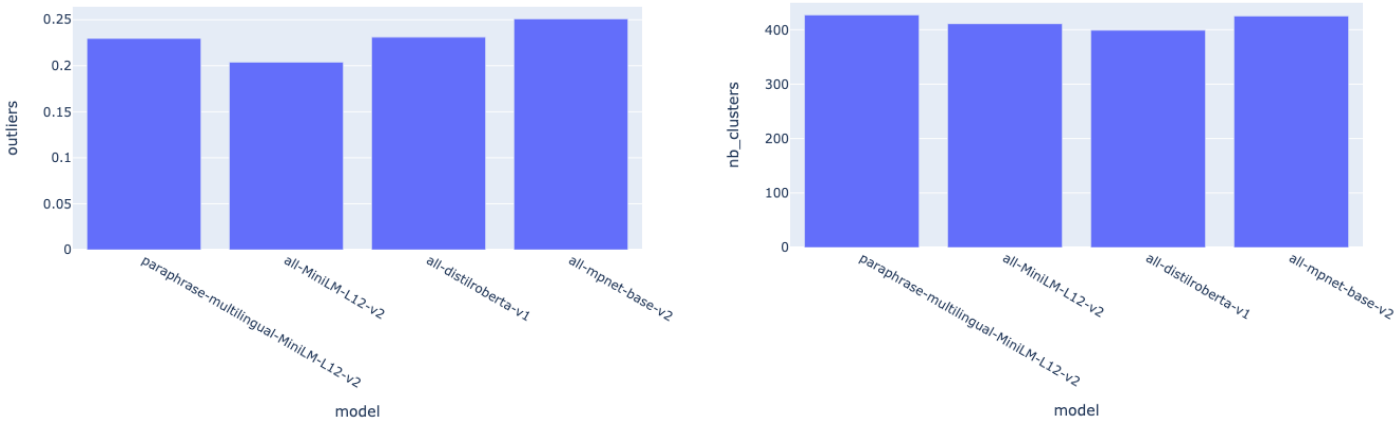
for Task-Agnostic Compression of Pre-Trained Transformers" [17]. Il génère des vecteurs embeddings de taille 384 dimensions. Le modèle est le résultat d'un transfert de connaissances entre la dernière couche du modèle $Bert_{Base}$ (teacher model) (préentraîné sur wikipédia et d'autres corpus de données) vers toutes les couches du modèle MiniLM (student model). Cela est fait grâce à une Kullback Leibler Divergence KLD loss entre les distributions de la self-attention de la dernière couche de $Bert_{Base}$ et les distributions de la self attention des différentes couches du student model.

- **paraphrase-multilingual-MiniLM-L12-v2** : Le modèle est entraîné en utilisant comme Teacher: paraphrase-MiniLM-L12-v2 et Student:Multilingual-MiniLM-L12-H384. Il génère des vecteurs embeddings de taille 384 dimensions. C'est un modèle distillé basé sur le MiniLM. Le paraphrase-MiniLM-L12-v2 est basé sur le MiniLM-l12-v2 et entrainé sur des datasets de paraphrases. Paraphraser un texte consiste à le reformuler syntaxiquement tout en gardant la même information sémantique. le Multilingual-MiniLM-L12-H384 est entrainé de la même manière que le MiniLM-L12-H384 sauf qu'il est basé sur $XLM-RoBERTa_{BASE}$ qui est entraîné sur les données issues de la plateforme Common Crawl sur 100 langues différentes.
- **all-distilroberta-v1** : Génère des vecteurs embeddings de taille 768. Le modèle est basé sur le distilroberta-base. C'est une version distillée de RoBERTa qui est elle-même une version améliorée de BERT avec quelques modifications dans la partie entraînement (Entraînement plus long, des batchs plus grands et sur plus de données - Pas d'entraînement avec le Next Sentence Prediction NSP - Entraînement sur des séquences plus longues, etc). Le modèle contient 6 couches de transformers, un état caché de taille 768 dimensions avec 12 head-attentions. Le modèle a été préentraîné sur OpenWebTextCorpus. Il reprend le même entraînement que DistilBert.
- **all-mpnet-base-v2** : Le modèle est basé sur le $BERT_{BASE}$ qui est constitué de 12 couches de transformers et un état caché de taille 768 qui correspond à la dimension des vecteurs embeddings générés. Le modèle est entraîné avec la méthode MPNET qui regroupe le Masked Language Modeling et le Permuted Language Modeling présenté dans [13]. Préentraîné sur Wikipedia, BooksCorpus, OpenWebText, CC-NEWS, etc.

Pour chaque modèle on récupère le dernier état caché qui contient les word embeddings des tokens données en entrée. Un average pooling (Moyenne des words embeddings) est appliqué pour générer le vecteur embeddings de la séquence données en entrée.

4.4.3.2 Clustering avec HDBSCAN

Pour chaque colonnes (Nom de la colonne avec sa description textuelle si elle est disponible) et pour chaque modèle de langue, on génère les vecteurs embeddings correspondants. Après, on effectue un clustering avec HDBSCAN et on compare les résultats des partitionnements pour chaque modèle de langue selon les scores d'inertie précédemment définis (Section 4.2).



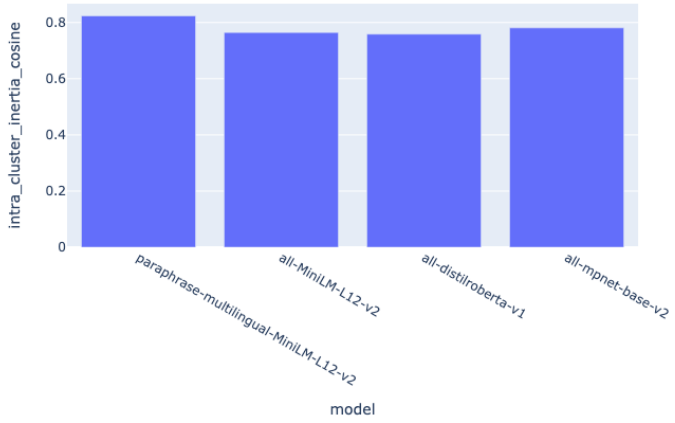
(a) Taux d'outliers selon le modèle

(b) Nombre de clusters selon le modèle

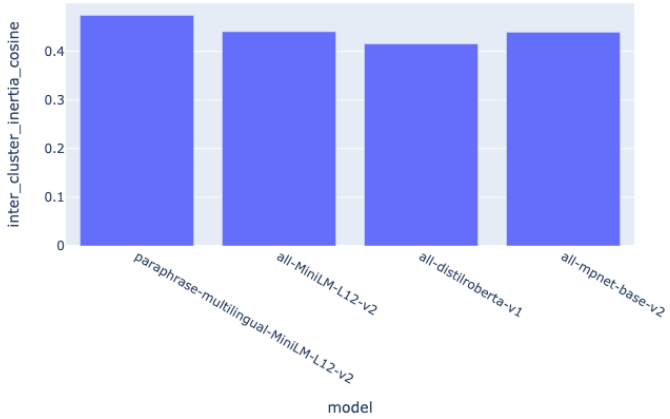
Figure 14: Le taux d'outliers et le nombre de clusters des partitionnements correspondants aux embeddings générés par chaque modèle.

Depuis la Figure 14a, on remarque que le plus grand taux d'outliers est obtenu avec le partitionnement des embeddings générés par le modèle all-mpnet-base-v2 25%. Tandis qu'avec les autres modèles, les partitionnements contiennent des taux d'outliers au tour de 21%.

Sur la Figure 14b, on observe qu'avec le modèle paraphrase-multilingual on obtient le plus grand nombre de clusters qui est de 427 clusters. Avec les vecteurs embeddings du modèle all-mpnet-base-v2, le partitionnement génère 425 clusters. Tandis qu'avec les autres modèles, les partitionnements sont au autour de 400 clusters.



(a) Inertie intra-cluster selon les modèles de langue



(b) Inertie inter-cluster selon les modèles de langue

Figure 15: Les scores d'inertie intra-cluster et inter-cluster des partitionnements correspondants aux embeddings générés par chaque modèle.

La Figure 15a représente les scores d'inertie intra-cluster obtenus selon les partitionnements des vecteurs embeddings générés avec les différents modèles de langue. Un très grand score d'inertie intra-cluster signifie que le contenu des clusters est très homogène. On remarque qu'avec le modèle paraphrase-multilingual on obtient le meilleur score d'inertie intra-cluster qui est de 0.82. Avec le all-mpnet-base-v2 on obteint un score de 0.78. Avec les deux autres modèles, on obtient des scores d'inertie intra-cluster qui sont autour de 0.75 .

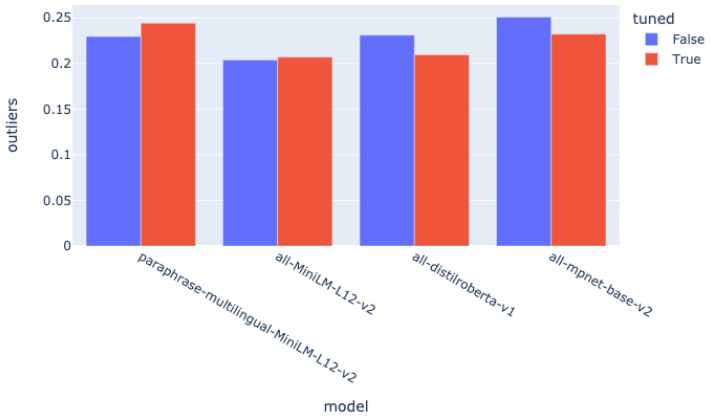
Sur la Figure 15b sont représenté les scores d'inertie inter-cluster. Un faible score d'inertie inter-cluster signifie que les clusters sont très hétérogènes et très bien séparés. Avec le modèle paraphrase-multilingual on obtient un partitionnement avec un score d'inertie inter-cluster de 0.47, tandis que les autres modèles sont autour de 0.42 .

4.4.3.3 Domain Adaptation Le *Domain adaptation* est une méthode qui permet aux modèles préentraînés sur des données d'un domaine initial et général de s'adapter aux données d'un domaine cible. Cette méthode permet d'obtenir des modèles avec de meilleurs performances sur le domaine cible.

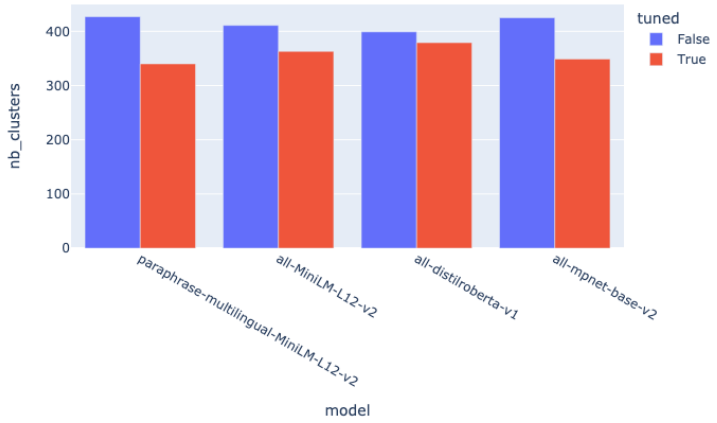
Pour ce faire, nous avons réentraîné les modèles de langues sur les descriptions textuelles des colonnes avec le Masked Language Modeling MLM. Afin de permettre aux modèles d'apprendre des informations contextuelles sur les nouvelles données textuelles qu'ils vont encoder.

Pour le réentraînement des modèles avec la méthode du Masked Language Modeling (MLM), nous avons utilisé une probabilité de 0.15 pour qu'un mot soit remplacé par le token ['MASK'], un nombre d'epochs égale à 100 avec un batch de taille 64. Le réentraînement est fait sur les serveurs de la faculté qui sont équipés de GPUs (RTX 2080).

4.4.3.4 Clustering avec HDBSCAN des vecteurs embeddings générés par les modèles réentrainés Dans cette partie, on génère les vecteurs embeddings des colonnes qui sont représentées par un nom de colonne concaténé avec sa description textuelle si elle est fournie. Et pour cela on utilise les modèles de langue réentrainés avec la méthode du MLM. Ensuite, on compare les scores d'inertie obtenus après un partitionnement avec HDBSCAN :



(a) Taux d'outliers selon le modèle

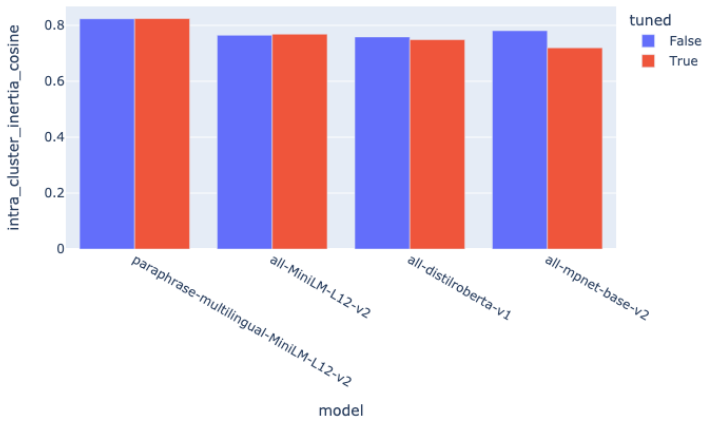


(b) Nombre de clusters selon le modèle

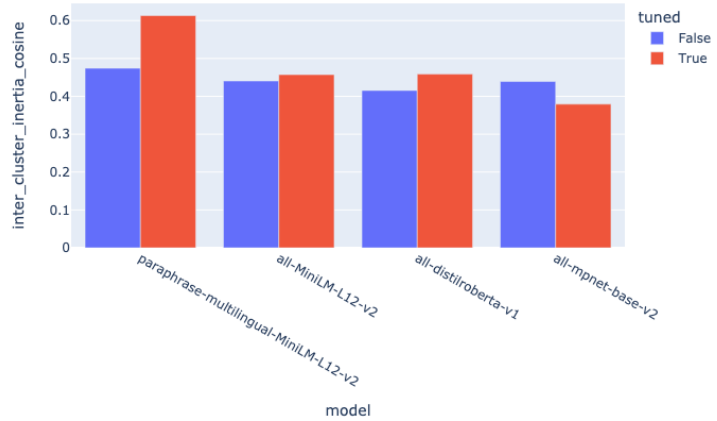
Figure 16: Le taux d'outliers et le nombre de clusters obtenus après un partitionnement en utilisant des vecteurs embeddings de chaque modèle de langue (En rouge ceux qui sont réentraîné avec le MLM sur les descriptions textuelles et en bleu ceux qui ne le sont pas).

Après le réentraînement des modèles avec le MLM, on remarque sur la Figure 16a que le taux d'outliers à diminuer pour les partitionnement des vecteurs embeddings générés avec les deux modèles all-mpnet-base-v2 (0.23%) et all-distilroberta-v1 (0.209%). Tandis que les taux d'outliers en utilisant les des deux autres modèles ont augmenté, 0.24% pour le paraphrase-multilingual et le all-MiniLM-L12-v2 0.207%.

On remarque aussi sur la Figure 16b que le nombre de clusters a diminué en utilisant les modèles réentraînés. Le nombre moyen de clusters est passé de 415 à 357 clusters après le réentraînement des modèles.



(a) Inertie intra-cluster selon les modèles de langue



(b) Inertie inter-cluster selon les modèles de langue

Figure 17: Les scores d'inertie intra-cluster et inter-cluster selon les partitionnements obtenus en utilisant les vecteur embeddings de chaque modèle de langue (En rouge ceux qui sont réentraîné avec le MLM sur les descriptions textuelles et bleu ceux qui ne le sont pas).

La Figure 17a montre les scores d'inertie intra-cluster. On remarque qu'il n y'a pas de grande différence entre les scores d'inertie intra-cluster des partitionnements des vecteurs embeddings générés par les modèles réentraînés et ceux qui ne le sont pas, sauf pour le all-mpnet-base-v2 qui passe d'un score d'inertie intra-cluster de 0.78 à 0.71 après le réentraînement.

La Figure 17b monte les scores d'inertie inter-cluster. On remarque que le score augmente légèrement pour en utilisant le all-MiniLM-L12-v2 (0.44 à 0.45) et le all-distillroberta-v1 (0.41 à 0.45) et considérablement pour le paraphrase-multilingual (0.47 à 0.61). Tandis que le partitionnement des vecteurs embeddings générés avec all-mpnet-base-v2 obtient le meilleur score en passant de 0.43 à 0.37.

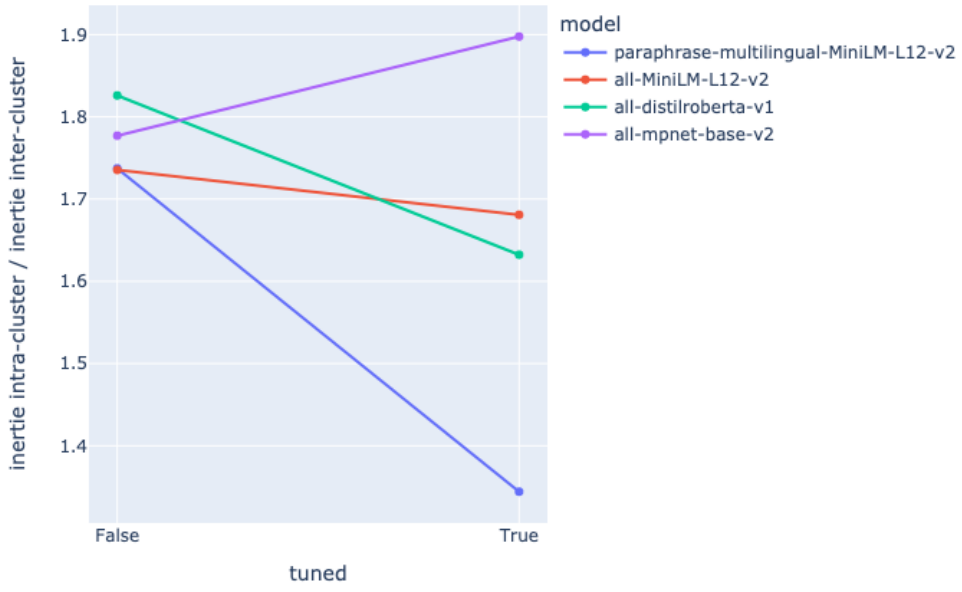


Figure 18: Évolution du rapport des inerties $\frac{inertie\,intra-cluster}{inertie\,inter-cluster}$ des partitionnements des vecteurs embeddings générés par des modèles non réentraînés (tuned à False) et réentraînés (tuned à True).

Sur la Figure 18, on remarque que le meilleur rapport d'inertie est obtenu avec le modèle all-mpnet-base-v2 réentraîné qui est de 1.89, et le pire rapport d'inertie est obtenu avec le paraphrase-multilingual réentraîné qui est de 1.34. Plus le rapport est grand, plus le modèle est meilleur, car cela correspond à une forte inertie intra-cluster et une faible inertie inter-cluster.

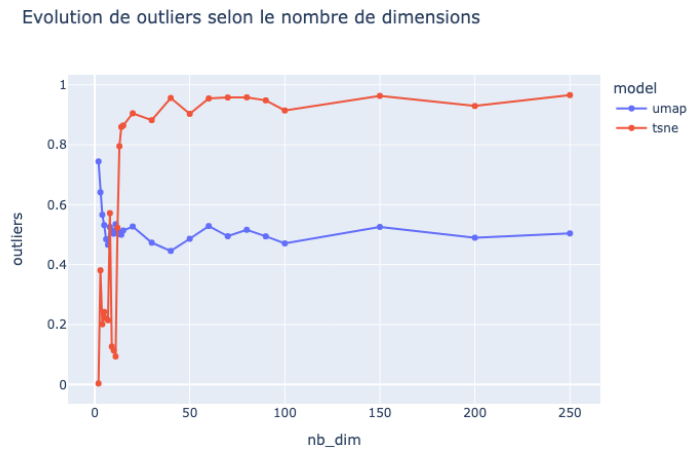
Le modèle basé sur le MPNet et réentraîné sur les descriptions textuelles des colonnes obtient le meilleur score en terme de rapport d'inertie. Cela est dû au fait que le MPNet a pu capturer plus d'informations contextuelles lors de la phase d'entraînement. Ce modèle a été repris dans la suite des expérimentations pour la génération de vecteurs embeddings.

4.4.4 Réduction de dimensions suivie d'un clustering avec HDBSCAN

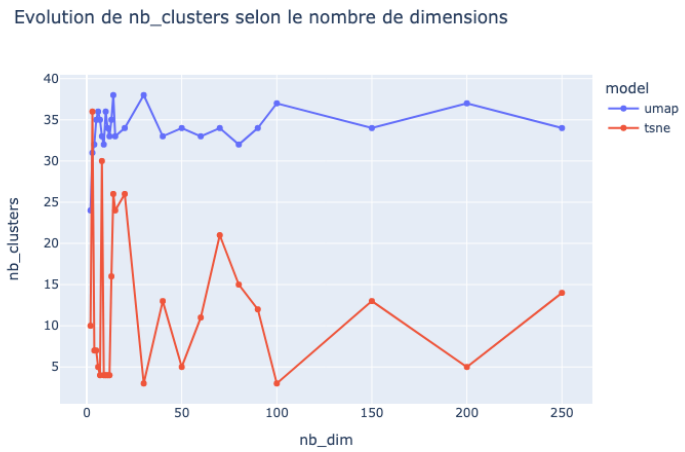
4.4.4.1 TSNE et UMAP suivies de HDBSCAN (Dataset de taille 2K) Dans cette expérience, le dataset utilisé contient 2k colonnes concaténées avec leurs descriptions textuelles dans le cas où elles sont fournies. À partir de ces données textuelles, des vecteurs embeddings de taille 768 dimensions ont été générés grâce au modèle de langue all-mpnet-base-v2 réntrainé avec le MLM sur l'ensemble des descriptions textuelles. Une réduction de dimensions a été faite avec UMAP et TSNE selon plusieurs nombres de dimensions sur les vecteurs embeddings des colonnes. Ensuite, un clustering avec HDBSCAN est appliqué sur les embeddings de dimensions réduites afin de comparer entre UMAP et TSNE selon quelques métriques de mesure de qualité des partitionnements (Scores d'inertie).

TSNE pour T-Distributed Stochastic Neighbor Embedding [15] est un algorithme de réduction de dimension. L'algorithme modélise la dispersion des données dans l'espace initial sous forme de distribution de probabilités de la loi normale. Entre chaque paire de points une distribution de probabilité est définie et fait correspondre la probabilité que le point puisse être choisi comme voisin de l'autre point. Ensuite, l'algorithme approche cette distribution de probabilités avec une autre distribution de probabilités issues de la Loi de Student dans l'espace de projection. Pour cela une décente de gradient est utilisée pour minimiser la divergence de Kullback Leibler entre les deux distributions de probabilités. L'algorithme prend en entrée un paramètre important qui est la perplexité, qui représente le nombre minimum d'éléments voisins à prendre en considération lors du calcul du graphe de voisinage dans l'espace initial. Plus la perplexité est grande, plus on va s'intéresser aux structures globales que forment les données et plus la perplexité est faible, plus on va s'intéresser aux structures locales.

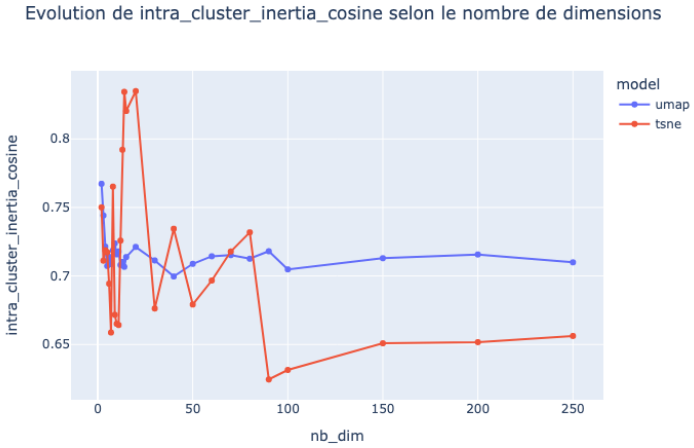
Nous avons utilisé la version disponible dans Scikit-Learn (https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html). Avec un nombre minimum d'éléments voisins égale à 15.



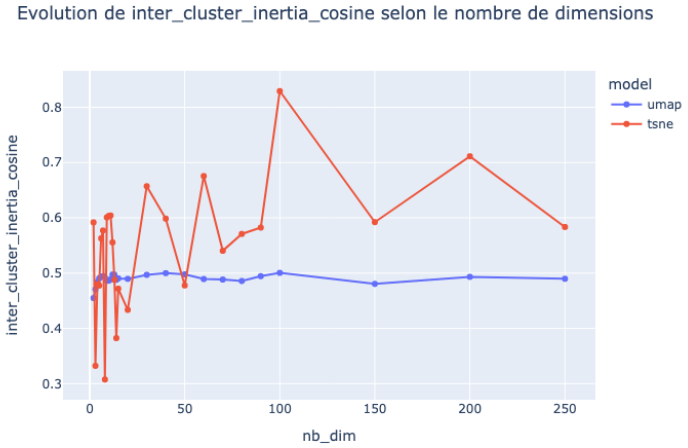
(a) Taux d'outliers



(b) Nombre de clusters



(c) Inertie intra-cluster



(d) Inertie inter-cluster

Figure 19: Quelques métriques de mesures de qualité des partitionnements selon le nombre de dimensions des vecteurs embeddings utilisés et le modèle de réduction de dimension.

Sur la Figure 19, on remarque qu'avec la TSNE on peut atteindre de très bons scores d'inertie intra-cluster (0.83 avec 14 dimensions) et de très bons scores d'inertie inter-cluster aussi (0.30 avec 8 dimensions). Tandis que UMAP atteint un score d'inertie intra-cluster de 0.76 et un score d'inertie inter-cluster de 0.45 avec 2 dimensions. Cela est dû au fait qu'avec la TSNE le nombre d'outliers est très élevé, ce qui fait que le nombre de clusters est faible et les scores d'inertie sont très bons comparés au cas de UMAP.

4.4.4.2 Évolution du temps d'exécution des modèles de réduction de dimensions

Après la génération des vecteurs embeddings par un modèle de langue vient l'étape de la réduction de dimension où UMAP est utilisé dans le cas de BerTopic. Dans cette section l'idée est de tester différents modèles de réduction de dimensions, linéaires comme la PCA et non linéaires comme UMAP, TSNE ou ISOMAP pour ainsi comparer leurs temps d'exécution. Ci-dessous les descriptions de la PCA et ISOMAP :

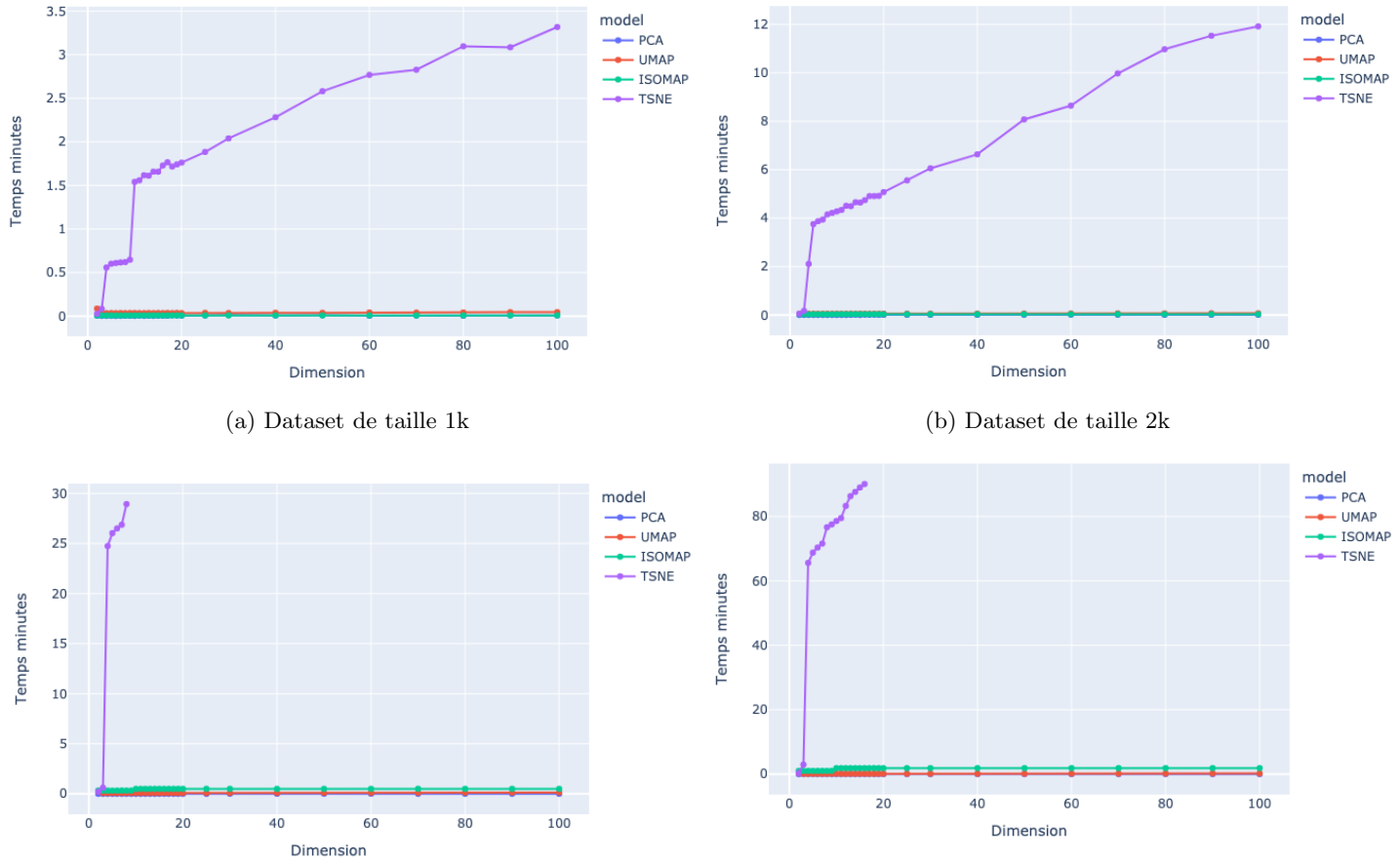
PCA pour Principal Component Analysis [1] est une méthode de réduction de dimensions linéaire. Elle est utilisée pour la visualisation de données ou pour la réduction de bruit (dans les dimensions) des données avant d'appliquer dessus un algorithme de machine learning.

Comme première étape, les différentes dimensions des vecteurs de l'espace initial sont standardisées pour donner la même importance à l'ensemble des dimensions des vecteurs de l'espace initial. Cette standardisation permet d'éviter que certaines dimensions avec des valeurs plus grandes dominent les dimensions qui ont des valeurs plus petites. Ensuite, une matrice de covariance est calculée entre l'ensemble des dimensions, pour détecter les dimensions qui sont le plus corrélées. Après,

les vecteurs propres avec leurs valeurs propres associées sont générés (Pour la matrice de covariance). Les vecteurs propres représentent les composantes principales. Et les valeurs propres décrivent la variance et le taux d'information que portent les vecteurs propres associés. Un nombre n de composantes principales est retourné ayant une variance supérieure à un seuil. Le produit de la transposée des composantes principales avec les vecteurs de l'espace initial du dataset génère les nouveaux vecteurs correspondant dans l'espace de projection.

ISOMAP pour Isometric Mapping [14] est une autre méthode de réduction de dimensions non-linéaire qui conserve les longueurs dans l'espace de projection. Un graphe de voisinage est généré entre tous les nœuds du dataset de l'espace initial. Une matrice d'adjacence est calculée selon la distance géodésique qui est définie comme étant la somme des poids des arêtes le long du chemin le plus court entre deux nœuds. Ensuite, les vecteurs propres de la matrice des distances géodésiques sont calculés et les n vecteurs supérieurs de la matrice sont séléctionnés. La transposée de la matrice des n vecteurs propres multipliée par les coordonnées initiales des noeuds génère les coordonnées dans le nouvel espace de projection.

Dans cette expérimentation, nous avons repris les vecteurs embeddings de taille 384, générés précédemment par Doc2Vec (Section 4.4.1). Le but de cette partie est de comparer le temps d'exécution des différentes méthode de réduction de dimensions selon la taille du dataset et le nombre de dimensions dans l'espace de projection.



(a) Dataset de taille 1k

(b) Dataset de taille 2k

(c) Dataset de taille 6k

(d) Dataset de taille 10k

Figure 20: Évolution du temps d'exécution des modèles de réduction de dimensions selon le nombre de dimensions et la taille du dataset.

De ces figures (fig. 20a, fig. 20b, fig. 20c, fig. 20d), on remarque que le temps d'exécution des modèles de réduction de dimensions tels que UMAP, ISOMAP, PCA est constant et très faible, même en augmentant le nombre de dimensions et la taille du dataset.

Par contre le temps d'exécution de la TSNE évolue de façon exponentielle en augmentant le nombre de dimensions et la taille du dataset. (Avec la TSNE pour un dataset de taille 1k et une réduction à 10 dimensions, le temps d'exécution est d'environ 1.5 minutes tandis que pour un dataset d'une taille de 10k le temps d'exécution est d'environ 80 minutes).

Notes :
Bien que l'utilisation de la TSNE suivie de HDBSCAN donne de très bons scores d'inertie dans le cas d'un petit ensemble de données (dataset de taille 2k) comparé à UMAP, la solution ne reste pas adaptée pour les gros volumes de données pour les raisons suivantes :

- Le temps d'exécution de la TSNE augmente de façon exponentielle en augmentant la taille du dataset. L'architecture des bases de données change régulièrement (ajout et suppressions de colonnes ou de tables), le partitionnement obtenu après un temps considérable en utilisant la TSNE peut ne pas correspondre à l'état actuel du datalake.
- Les versions de la TSNE qui fonctionnent sous GPU ne supporte que la réduction à 2 dimensions. (TSNE de cuML de RAPIDS.AI)
- UMAP préserve mieux les structures locales et globales en créant des zones à forte densité avec une bonne séparation. Tandis que la TSNE permet seulement de préserver les structures locales, en mettant plus de poids entre chaque point et ses voisins les plus proches.

4.4.4.3 Clustering des colonnes (Réduction de dimensions avec différents modèles suivie d'un clustering avec HDBSCAN) Dans cette partie, nous avons testé différentes méthodes de réduction de dimensions (Umap, PCA, ISOMAP) sur les vecteurs embeddings générés par le modèle all-mpnet-base-v2 (768 dimensions dans l'espace initial) réentraîné sur les descriptions textuelles avec la méthode MLM. Pour chaque nombre de dimensions des vecteurs embeddings réduits (2 à 700 dimension), un partitionnement est appliqué sur ces vecteurs avec HDBSCAN et les métriques d'inertie sont calculées :

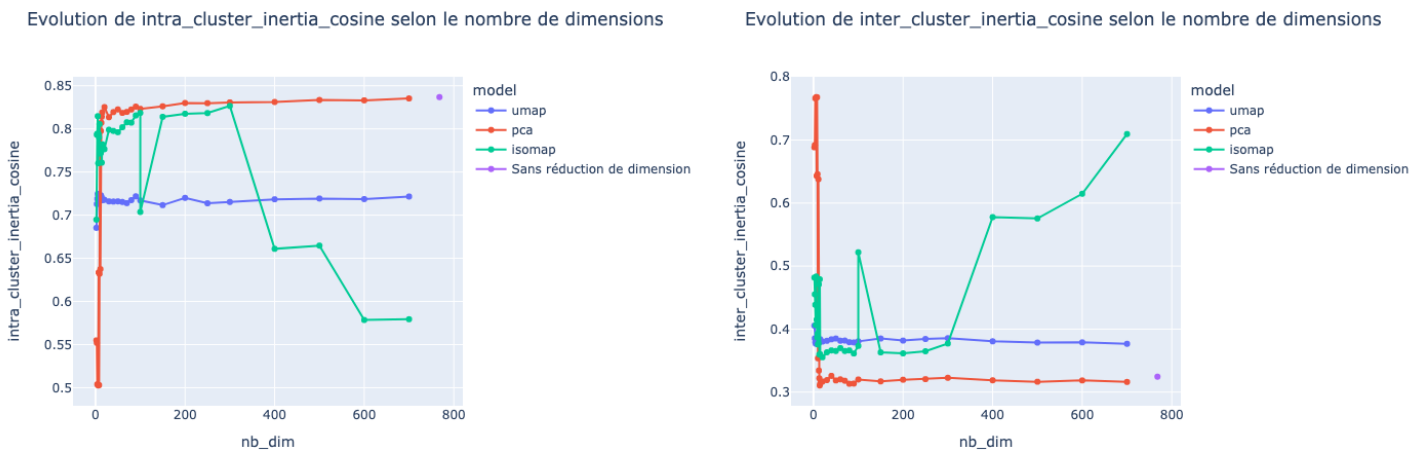
(a) Taux d'outliers selon le modèle

(b) Nombre de clusters selon le modèle

Figure 21: Le taux d'outliers et le nombre de clusters obtenus après un partitionnement avec HDBSCAN et une réduction de dimensions selon différents modèles.

De la Figure 21a, on remarque que le taux d'outliers obtenu après un partitionnement des vecteurs embeddings de dimensions réduites avec la PCA est autour de 67% et 70% sans réduction de dimensions. Ces taux sont très élevés comparé aux taux d'outliers obtenu avec UMAP qui est de 30%. Avec ISOMAP le taux d'outliers reste plus au moins élevé entre 8 et 400 dimensions (environ 76%) mais qui diminue aux alentours de 1% pour un nombre de dimensions allant de 400 à 700.

De la Figure 21b, on observe que le nombre de clusters obtenus avec UMAP est entre 200 et 250 clusters, contrairement à l'utilisation de la PCA, ISOMAP et sans réduction de dimensions où le nombre de clusters est plus faible. ISOMAP arrive à détecter un nombre maximum de clusters égale à 108. En utilisant la PCA et sans réduction de dimensions, on obtient un nombre de cluster aux alentours de 61. Cela est dû aux taux d'outliers très élevés.



(a) Inertie intra-cluster selon les modèles de langue

(b) Inertie inter-cluster selon les modèles de langue

Figure 22: Les scores d'inertie intra-cluster et inter-cluster obtenus après un partitionnement avec HDBSCAN et une réduction de dimensions selon différents modèles.

De la Figure 22a, on constate que les scores d'inertie intra-cluster obtenus avec UMAP sont stables selon différents nombres de dimensions utilisés et sont en moyenne de 0.71. Pareil avec les scores d'inertie inter-cluster qui sont aux alentours de 0.38 pour UMAP. Avec les autres modèles, les scores d'inertie intra-cluster sont plus élevés, et les scores d'inertie inter-cluster sont plus faibles, et cela est dû au fait que le taux d'outliers est très élevé et le nombre de clusters est très faible. (à part avec ISOMAP qui à partir de 250 dimensions obtient des scores d'inertie intra-cluster plus faibles et des scores d'inertie inter-cluster plus élevés).

Une réduction de dimension avec UMAP suivie d'un clustering avec HDBSCAN permet de détecter le plus grand nombre de clusters. Ce qui permet d'obtenir divers topics. Et cette richesse de topics permettra de faire un choix plus spécifiques des clusters et des colonnes qui sont en relation avec un business-item.

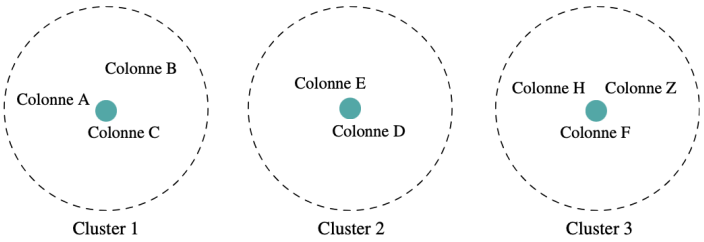
5 Approche 2 : Clustering des colonnes avec descriptions textuelles et clustering incrémental des colonnes sans descriptions textuelles

5.1 Description de l'approche 2

Les descriptions textuelles des colonnes apportent plus d'informations que les noms de colonnes seuls. Dans cette deuxième approche, nous nous intéressons d'abord au partitionnement des colonnes ayant une description textuelle. Ensuite, on s'intéressera aux colonnes sans descriptions textuelles.

Dans la première approche, l'utilisation d'une réduction de dimensions suivie d'un HDBSCAN a obtenu les meilleurs résultats en termes de scores d'inertie comparé aux clustering en utilisant le K-Means ou le LDA. C'est pourquoi dans cette deuxième approche, on va reprendre la même méthode qui consiste à générer des vecteurs embeddings pour toutes les colonnes ayant une description textuelle grâce à un modèle de langue, ce qui sera suivi par une réduction de dimensions sur ces vecteurs embeddings et enfin un HDBSCAN sera appliqué sur ces vecteurs embeddings afin d'obtenir un partitionnement des colonnes ayant une description textuelle.

- Clustering des colonnes avec description textuelle :



- Affectation des colonnes sans description textuelle aux clusters :

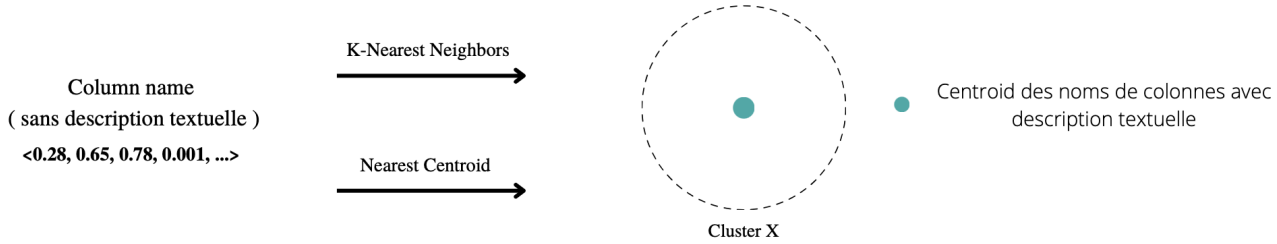


Figure 23: Schéma de l'approche 2 (Clustering des colonnes).

Après que le partitionnement des colonnes avec descriptions textuelles est fait, deux méthodes vont être utilisées pour faire correspondre aux colonnes sans descriptions textuelles le cluster le plus similaires sémantiquement (Clustering incrémental):

- **K-Nearest Neighbors (KNN) :**
  La méthode consiste à projeter les vecteurs embeddings des noms de colonnes avec descriptions textuelles et des noms de colonnes sans descriptions textuelles dans le même espace de projection. Les vecteurs embeddings des noms de colonnes avec descriptions textuelles sont déjà annotés par le cluster auxquels ils appartiennent. Chaque vecteur embedding de nom de colonne sans descriptions textuelles est affecté au cluster le plus majoritaire parmi les K colonnes avec description textuelle les plus proches (En terme de distance euclidienne). Et ainsi, on aura fait correspondre aux colonnes sans descriptions textuelles un cluster généré préalablement avec le partitionnement des colonnes avec descriptions textuelles.
- **Nearest Centroid (NC) :**
  La méthode consiste à trouver pour chaque colonne sans description textuelle le cluster de colonnes avec description textuelle le plus similaire sémantiquement. D'abord, il faut calculer un vecteur centroïde pour chaque cluster. Le vecteur centroïde correspond à la moyenne des vecteurs embeddings des noms de colonnes avec description textuelle contenus dans le cluster. Une fois que ces vecteurs centroïdes ont été générés, on calcule un score de similarité sémantique (similarité cosinus) entre ces centroïdes et chaque vecteur embedding de nom de colonne sans description textuelle pour ainsi l'affecter au cluster le plus similaire sémantiquement.

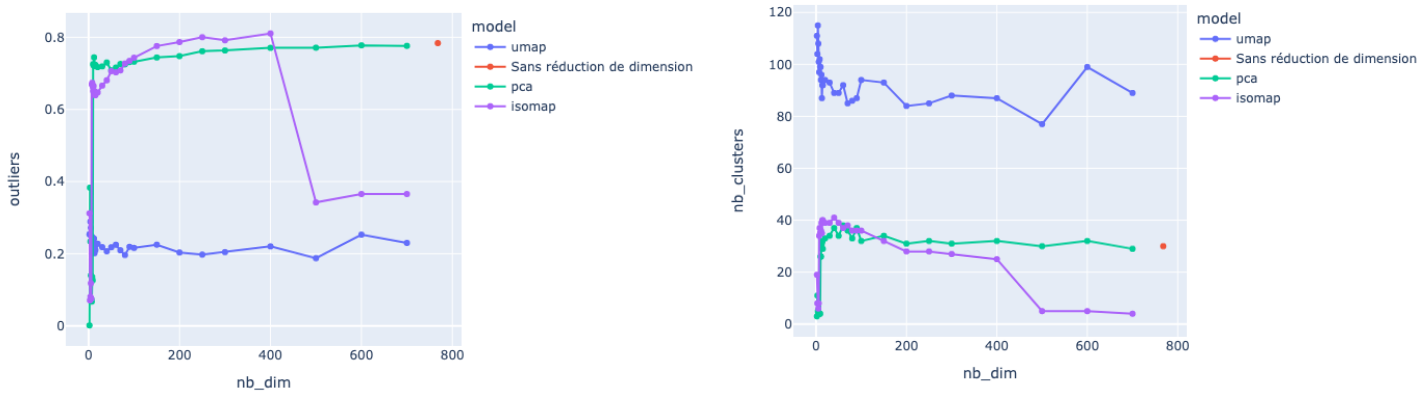
Après que toutes les colonnes sans descriptions textuelles soient affectées aux clusters similaires sémantiquement, on obtient le partitionnement final de toutes les colonnes avec et sans description textuelle. Et pour chaque cluster, on recalcule un vecteur centroïde qui va correspondre à la moyenne des vecteurs embeddings de tous les noms de colonnes appartenant au cluster. Et pareil qu'avec la première approche, on génère un vecteur embedding pour le *business-item* à l'aide de sa description textuelle et on calcule un score de similarité sémantique (similarité cosinus) entre le vecteur embedding du *business-item* et les vecteurs centroïdes des clusters. Ensuite, on retourne les **N** clusters les plus similaires avec le *business-item* et pour chaque cluster on retourne les **M** colonnes qui ont un score de similarité cosinus supérieur à une valeur **V** avec le *business-item*. Les colonnes retournées vont correspondre aux colonnes similaires sémantiquement avec le *business-item*. Ensuite, l'utilisateur pourra sélectionner les colonnes les plus pertinentes parmi celles retournées et ainsi créer des *linked-items* entre ces colonnes et le *business-item*.

5.2 Expérimentations (Clustering) et résultats

Dans cette section, nous avons expérimenté quelques méthodes pour le partitionnement des colonnes selon l'approche 2 :

5.2.1 Clustering des colonnes avec descriptions textuelles (Réduction de dimensions avec différents modèles suivie d'un clustering avec HDBSCAN)

Dans cette partie, on va s'intéresser au partitionnement des colonnes qui contiennent des descriptions textuelles. Chaque colonne est représentée pas sa description textuelle. On utilise le modèle all-mpnet-base-v2 réentrainé sur ces descriptions textuelles avec le Masked Language Modeling MLM (vecteurs embeddings de dimensions 768). En utilisant différents modèles de réduction de dimensions (Umap, PCA, ISOMAP) selon des dimensions de l'espace de projections allant de 2 à 700 dimensions, on applique sur ces vecteurs embeddings de dimensions réduites un partitionnement avec HDBSCAN. Pour chaque partitionnement qui correspond à l'utilisation d'un modèle de réduction de dimensions et pour chaque nombre de dimensions, des mesures d'inertie sont retournées.



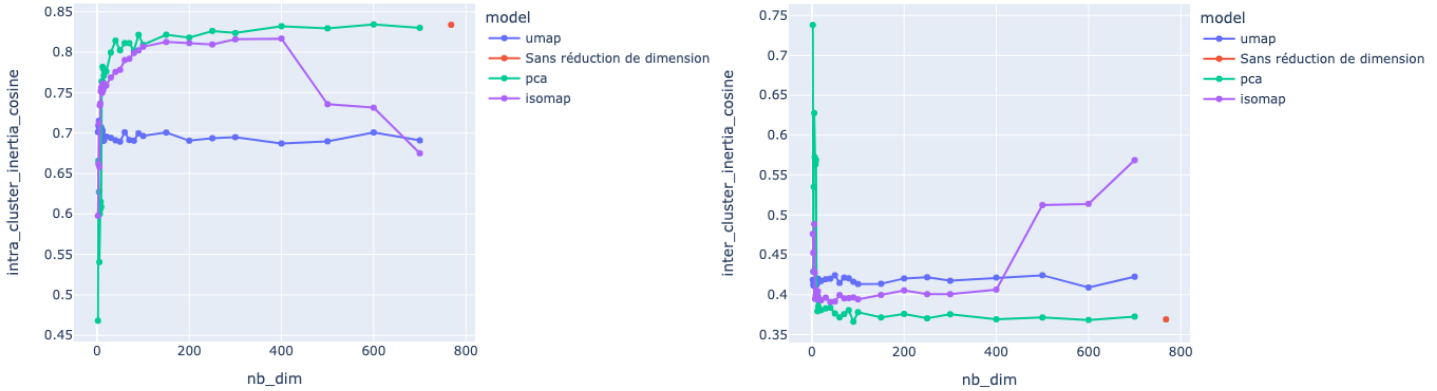
(a) Taux d'outliers

(b) Nombre de clusters

Figure 24: Le taux d'outliers et le nombre de clusters obtenus après un partitionnement avec HDBSCAN des vecteurs embeddings de dimensions réduite selon quelques modèles de réduction de dimensions.

De la Figure 24a, on remarque que le taux d'outliers des partitionnements obtenus en utilisant UMAP pour réduire la taille des vecteurs embeddings est d'environs 20%, avec la PCA le taux est d'environ 70% et ISOMAP atteint les 80% d'outliers avant de redescendre à 34% pour un nombre de dimensions supérieur à 500. Sans réduction de dimensions, le HDBSCAN obtient un taux d'outliers de 78%. Umap obtient le plus faible taux d'outliers.

De la Figure 24b, on remarque qu'en utilisant UMAP on obtient le plus grand nombre de clusters qui est entre 77 et 115 clusters. Tandis qu'avec les autres modèles (PCA, ISOMAP) le nombre de clusters est inférieur à 41 et est égale à 30 clusters dans le cas où il n'y a pas de réduction de dimensions. Ce faible nombre de clusters est dû au taux d'outliers élevé.



(a) Inertie intra-cluster

(b) Inertie inter-cluster

Figure 25: Les scores d'inertie intra-cluster et inter-cluster obtenus après une réduction de dimensions des vecteurs embeddings selon différents modèles suivie d'un partitionnement avec HDBSCAN

De la Figure 25, on remarque que les scores d'inertie obtenu avec UMAP sont plus stables comparé aux autres modèles.Le score moyen obtenu avec UMAP en inertie intra-cluster est de 0.70 et en inertie inter-cluster de 0.41. Avec ISOMAP et la PCA les scores intra-cluster sont plus élevés et les scores inter-cluster sont plus faibles (Sauf dans le cas d'ISOMAP avec un nombre de dimensions supérieur à 500 où les scores d'inertie inter-cluster augmentent). Cela est dû au fait qu'une réduction de dimensions avec ces modèles suivie d'un clustering avec HDBSCAN génère de grands taux d'outliers, ce qui est aussi le cas d'un clustering sans réduction de dimensions. Par contre, l'utilisation de UMAP permet de réduire le taux d'outliers et de générer plus de clusters.

5.2.2 Clustering incrémental des colonnes sans descriptions textuelles (K-Nearest Neighbors et Nearest Centroid)

Un partitionnement HDBSCAN est appliqué sur les embeddings des descriptions textuelles des colonnes de différentes dimensions générées avec UMAP. On choisit le partitionnement qui obtient le meilleur score de rapport d'inertie (Rapport entre l'inertie intra-cluster et l'inertie inter-cluster) :

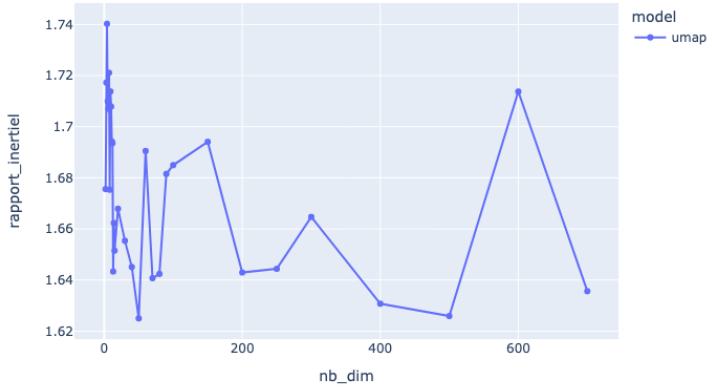
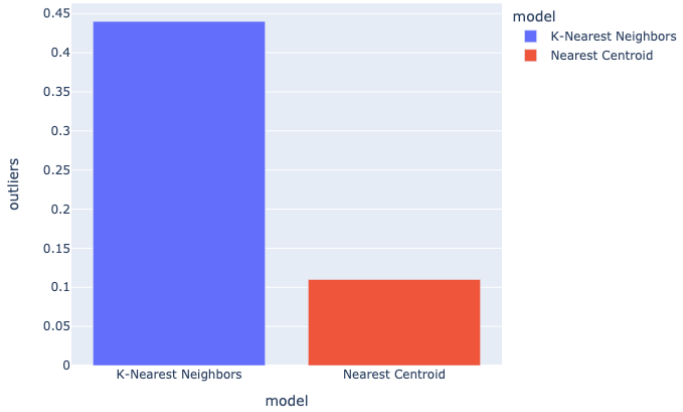


Figure 26: Évolution du rapport d'inertie après une réduction de dimensions avec UMAP suivie d'un clustering HDBSCAN.

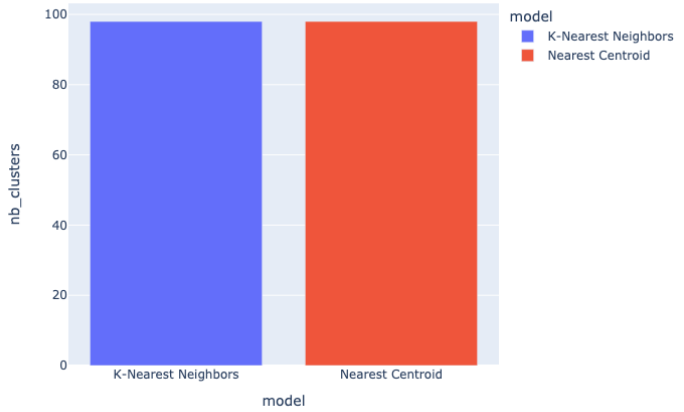
De la Figure 26, on remarque qu'avec une réduction de dimensions avec UMAP à 4 dimensions donne le meilleur rapport d'inertie qui est de 1.74. Cela correspond au partitionnement avec les meilleurs scores d'inertie (0.71 en score d'inertie intra-cluster et 0.40 en score inter-cluster).

Après que le nombre de dimensions idéal qui donne un partitionnement avec le meilleur rapport d'inertie (Maximum) est trouvé, on considère que le partitionnement des colonnes avec descriptions textuelles est fait. Ensuite, on passe à l'étape où on affecte les colonnes sans descriptions textuelles aux clusters correspondants. Et pour cela, deux méthodes ont été présentées dans la Section 5.1 . la méthode du K-Nearest Neighbors et celle du Nearest Centroid. Les deux méthodes

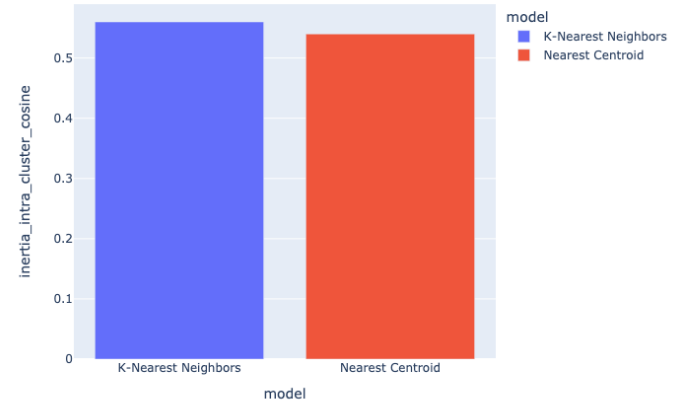
partagent le même principe qui consiste à faire correspondre à chaque colonne sans description textuelle un cluster, sachant que les clusters ont été générés par un partitionnement des colonnes contenant des descriptions textuelles.



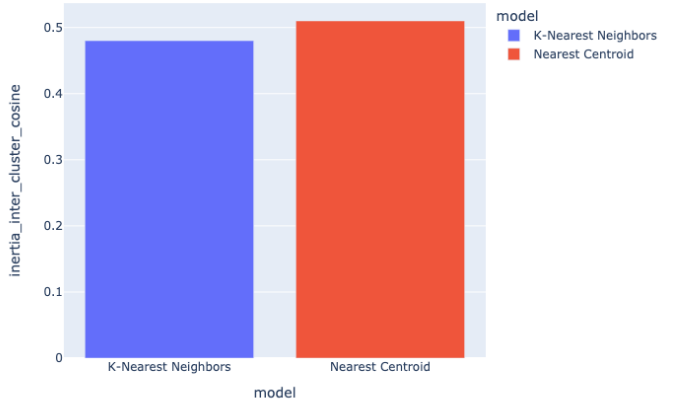
(a) Taux d'outliers



(b) Nombre de clusters



(c) Inertie intra-cluster



(d) Inertie inter-cluster

Figure 27: Qualité des partitionnements après ajout des colonnes sans descriptions textuelles avec la méthode K-Nearest Neighbors et Nearest Centroid.

De la Figure 27a, on remarque qu'avec la méthode KNN le nombre d'outliers est passé de 20% (Partitionnement des colonnes avec descriptions textuelles) à 44% d'outliers. Tandis qu'avec la méthode du Nearest Centroid, en ajoutant les colonnes sans descriptions textuelles le taux d'outliers est descendu à 11%. Depuis la Figure 27b Le nombre de clusters est resté inchangé avec les 2 méthodes car il n'y a pas eu de nouveau répartitionnement mais plutôt un clustering incrémental.

Depuis les figures fig. 27c et fig. 27b, on remarque que les scores d'inerties ont régressé comparé à l'état initial (Clustering des colonnes avec descriptions textuelles). Le K-Nearest Neighbors obtient un score d'inertie intra-cluster de 0.56 et 0.48 en inertie inter-cluster. Tandis que le Nearest Centroid obtient un score d'inertie intra-cluster de 0.54 et 0.51 en inertie inter-cluster.

Le taux d'outliers a diminué en utilisant la méthode du Nearest Centroid car le nombre de colonnes a augmenté considérablement dans les différents clusters autres que celui des outliers. Et cela est dû au fait qu'un colonne est considérée comme outlier seulement dans le cas où elle est similaire sémantiquement au vecteur centroïde représentant l'ensemble des éléments outliers. Les données considérés comme des outliers et du bruit sont dispersés sur l'ensemble de l'espace de projection, et tout cette ensemble n'est représenté que par un seul vecteur (le vecteur centroïde). Contrairement au KNN où une colonnes qui se trouve au milieu d'autres colonnes considérées comme des outliers sera considérée aussi comme tel (à condition d'avoir un nombre de voisins k considérés comme outliers).

6 Génération des représentations des topics

Une fois que le clustering est fait. On génère pour chaque cluster une représentation textuelle en se basant sur la méthode du Cluster-based TF-IDF décrite dans [7]. Le but de cette représentation de topic est d'essayer de donner un sens sémantique aux différentes colonnes qui sont dans un cluster.

$$W_{t,c} = tf_{t,c} \cdot \log(1 + \frac{A}{tf_t})$$

t : Un terme
c : Un cluster qui est défini comme la concaténation de tous les documents appartenant au cluster (Noms de colonnes et descriptions textuelles dans notre cas)
$W_{t,c}$: Le poids du terme t dans le cluster c
$tf_{t,c}$: La fréquence du terme t dans le cluster c
tf_t : La fréquence du terme t dans tous les clusters
A : Le nombre de mots moyen dans tous les clusters

Formule de calcul des poids des termes dans les clusters selon la méthode du Cluster-based TF-IDF.

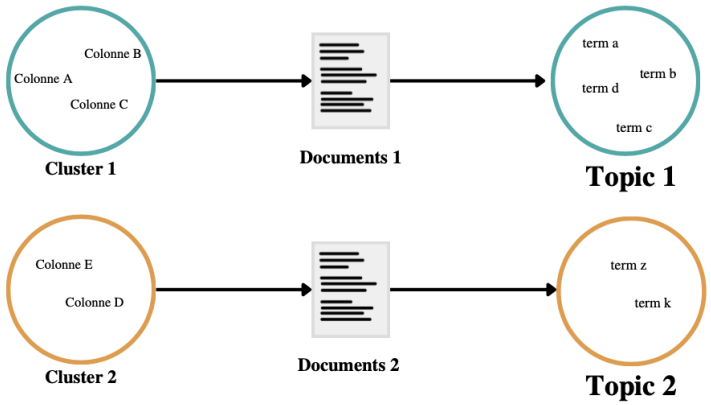


Figure 28: Génération de représentations de topics pour 2 clusters .

Pour pouvoir générer la représentation textuelle d'un cluster, il faut concaténer touts les noms de colonnes avec leurs descriptions textuelles (si elles sont disponibles) pour générer un seul document. Ensuite, il faut calculer le poids $W_{t,c}$ pour chaque terme t du document représentatif du cluster c. Enfin, on retourne tous les termes dont le poids est supérieur à un certain seuil. Ainsi, on aura la représentation du cluster sous forme d'un document ne contenant que les mots les plus pertinents.

7 Évaluation

Dans cette section, nous avons évaluer la pertinence des colonnes suggérées pour chaque business-item en utilisant les deux approches étudiées, qui sont l'approche 1 (Clustering de l'ensemble des colonnes) et l'approche 2 (Clustering des colonnes avec descriptions textuelles et clustering incrémental des colonnes sans descriptions textuelles avec les méthodes K-Nearest Neighbors **KNN** et Nearest Centroid **NC**).

Trois métriques pour mesurer la qualité des résultats retournés par les différentes approches :

- **Précision :** Représente le nombre d'éléments pertinents retournés (*VraisPositifs*) par rapport au nombre total d'éléments pertinents retournés (*VraisPositifs*) et non pertinents retournés (*FauxPositifs*).

$$Précision = \frac{VraisPositifs}{VraisPositifs + FauxPositifs}$$

- **Rappel :** Représente le nombre d'éléments pertinents retournés (*VraisPositifs*) par rapport à l'ensemble total d'éléments pertinents retournés (*VraisPositifs*) et pertinents non retournés (*FauxNégatifs*).

$$Rappel = \frac{VraisPostifs}{VraisPositifs + FauxNegatifs}$$

- **F1-score :** Représente une mesure qui combine entre le **Rappel** et la **Précision**.

$$F1\text{–}score = \frac{2 * Précision * Rappel}{Précision + Rappel}$$

On prend aléatoirement 1000 colonnes avec descriptions textuelles qu'on va considérer comme étant des business-items. Pour chaque business-item on crée des linked-items avec les 25 colonnes les plus proches sémantiquement de l'ensemble des données. Et pour chaque business-items et chaque approche on va retourner les N topics les plus proches avec ce business-item et dans chaque topic on va retourner les M colonnes les plus proches sémantiquement. On calculera une précision, un rappel et un f1-score pour chaque business items et pour chaque nombre top-N topics et top-M colonnes.

Ensuite, en fait varier le nombre de topics N entre 2 et 10 ainsi que le nombre de colonnes M entre 5 et 15. Une moyenne pour chaque métriques précision, rappel et f1-score est calculée pour chaque approche et pour l'ensemble des combinaisons (N, M) avec N $\in$ [2, 10] et M $\in$ [5, 15].

On obtient les résultats suivants :

Approches	Précision	Rappel	F1-score
Approche 1	0.19	0.72	0.298
Approche 2 (KNN)	0.27	0.73	0.392
Approche 2 (NC)	**0.28**	**0.79**	**0.410**

Table 1: Précision, Rappel et F1-score selon les résultats de l'approche 1 et de l'approche 2 avec les deux méthodes KNN et NC.

On peut observer à partir de la Table 1 que l'approche 2 qui utilise la méthode du Nearest Centroid (NC) obtient les meilleurs scores (Un score de 0.28 en précision, 0.79 en rappel et 0.41 en f1-score). Pour cette approche et en utilisant la méthode NC, la précision reste faible (0.28) car le nombre de colonnes retournées varie de 10 colonnes (2 topics et 5 colonnes) à 150 colonnes (10 topics et 15 colonnes) pour un nombre de colonnes pertinentes égale à 25 (Nombre de positifs) par business-item. Par contre, le rappel est assez élevé (0.79), ce qui signifie que le modèle retourne une grande partie des colonnes pertinentes. La combinaison du rappel et de la précision donne un très bon f1-score (0.41) comparé à celui de l'approche 1 et de l'approche 2 avec le K-Nearest Neighbors (KNN).

De ces résultats, on peut déduire que faire un clustering uniquement sur les colonnes avec descriptions textuelles suivies ensuite d'un clustering incrémental des colonnes avec la méthode du Nearest Centroid permet d'obtenir des bons résultats en terme de f1-score comparé aux autres approches. Cela est dû au fait que le clustering est de meilleure qualité sachant que les descriptions textuelles apportent plus d'informations sur l'ensemble des thématiques présentes dans tout le patrimoine de données contrairement aux noms de colonnes.

8 Cas d'usage

Le but de cette section est d'inférer des linked-items entre un Business-item et les différentes colonnes du patrimoine de données qui sont similaires sémantiquement.

Nous avons utilisé un business-item issu d'une base de donnée d'un client. La description du business-item est assez longue (51 mots). Ce business-item fait référence à une position géographique ainsi qu'à des informations financières et institutionnelles.

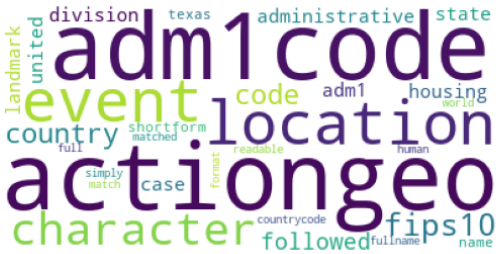
8.1 Approche 2 - NC : Clustering des colonnes avec descriptions textuelles et clustering incrémental des colonnes sans descriptions textuelles avec la méthode du Nearest Centroid

Après le clustering de toutes les colonnes avec descriptions textuelles, on génère des centroïdes pour chaque cluster qui correspondent à la moyenne des embeddings de colonnes contenues dans un cluster. Ensuite, on affecte chaque colonne sans description textuelle au cluster le plus similaires sémantiquement. Après, un score de similarité sémantique est calculé entre l'embedding de la description textuelle du business-item et les différents centroïdes en utilisant la similarité cosinus. Et on

renvoie les top-N topics les plus similaires avec le business-item. Enfin, pour chaque topic on retourne les top-M colonnes ayant une similarité cosinus supérieure à un seuil **V** avec le business-item.

Pour un nombre de topic **N** = 3, un nombre de colonnes **M** = 10 et un seuil de similarité cosinus **V** = 0.20, On obtient les résultats suivants :

(a) Topic 118 (similarité cosinus avec le business-item 0.47)



(b) Topic 162 (similarité cosinus avec le business-item 0.33)

(c) Topic 164 (similarité cosinus avec le business-item 0.32)

Figure 29: Représentations des 3 topics les plus proches du business-item avec leurs scores de similarité cosinus.

De la figure 29, on observe que les trois représentations sont assez liées avec le business-item. Le nuage de mots 29a contient des mots tels que bank, institution, federal et commercial qui sont en lien avec le coté finance de ce business-item. Tandis que les nuages de mots 29b et 29c contiennent des mots qui font plus référence au coté région et position géographique mentionnés dans la description textuelle du business-item.

Les top-10 colonnes les plus similaires au business-item dans les top-3 topics retournés. (Avec un score de similarité cosinus au business-item):

Topic 118 :

- institution_class: 0.47
- bank_charter_class: 0.47
- insured_commercial_bank: 0.43
- iba: 0.42
- chartering_agency: 0.40
- insured_savings_institute: 0.40
- qbp_region: 0.40
- federal_charter: 0.39
- top_holder: 0.38
- docket: 0.38

Topic 162 :

- Actor1Geo_ADM1Code: 0.34
- ActionGeo_ADM1Code: 0.32
- Actor2Geo_Type: 0.32
- Actor1Geo_Type: 0.31
- Actor2Geo_ADM1Code: 0.30
- ActionGeo_FullName: 0.24
- ActionGeo_CountryCode: 0.24
- Actor2Geo_CountryCode: 0.23
- Actor1Geo_CountryCode: 0.21
- Actor2Geo_FullName: 0.20

Topic 164 :

- facility_sub_region_2: 0.35
- sub_region_3: 0.34
- state_province_inc: 0.33
- country_subdivision_secondary: 0.32
- facility_country_region_code: 0.30
- facility_sub_region_1: 0.30
- country_subdivision_primary: 0.29
- country_iso_code_2: 0.29
- province_abbreviation: 0.29
- countries_and_territories: 0.27

Arriver à cette étape, l'utilisateur peux valider les linked-items suggérés entre les colonnes de ces topics avec le business-item.

9 Déploiement

Le but de cette section est de mettre en avant la partie déploiement de l'application. *Zeenea* propose des solutions qui sont totalement basées sur le cloud. Les solutions sont déployées sur des instances d'Amazon Web Services (AWS).

Afin de tester notre solution sur les données des clients, nous avons utilisé Docker pour créer deux conteneurs qui seront déployés. Docker est une technologie qui permet de lancer des applications dans des conteneurs isolés et qui peuvent s'exécuter sur n'importe quel environnement.

Une plateforme de développement Elastic Compute Cloud (EC2) d'AWS a été mise en place pour valider les testes ainsi que les connexions avec des Buckets Simple Storage Service (S3) pour le chargement et le stockage des données. Deux conteneurs ont été crées:

- **Container 1** : Sert à charger les données d'un client depuis un fichier JSON du Bucket S3 correspondant. Ensuite, vient l'étape de nettoyage et structuration des données (Noms de colonnes et descriptions textuelles) qui seront stockés en local sur l'environnement de production. Enfin, un modèle de langue est réentraîné sur les descriptions textuelles des colonnes, puis est stocké en local.
- **Container 2** : Sert à appliquer l'approche 1 pour des données où il n'y a pas beaucoup de colonnes avec descriptions textuelles et l'approche 2 qui utilise le Nearest Centroid dans le cas où il y a beaucoup de colonnes avec descriptions textuelles. Pour chaque approche, un gridsearch est appliqué pour trouver le meilleur paramétrage du modèle de réduction de dimensions pour ensuite générer des clusters avec HDBSCAN afin d'obtenir les meilleurs scores d'inertie. Les deux approches servent à générer des suggestions de linked-items entre des business-items et les colonnes du patrimoine de données. Pour évaluer la pertinence des linke -items suggérés, on calcule une précision, rappel et f1-score en les avec les vrais linked-items générés par le client. Enfin, les résultats des inerties des clusters crées par les deux approches sont stockés dans un Bucket S3 ainsi que les métriques qui évaluent le retour des deux approches (précision, rappel et f1-score).

Une dernière étape avant la mise en production consiste à valider qu'il n y a pas d'exfiltration de données lors de l'exécution des deux conteneurs.

10 Conclusion

Dans ce stage, deux approches ont été présentées pour la suggestion de linked-items entre les business-items et les colonnes d'un patrimoine de données. La première approche se base sur le clustering des vecteurs embeddings générés par la concaténation des noms de colonnes avec leurs descriptions textuelles dans le cas où elles sont disponibles. Le clustering est fait avec HDBSCAN précédé par une réduction de dimensions avec UMAP. la deuxième approche consiste à appliquer dans un premier temps une réduction de dimension avec UMAP suivie d'un clustering avec HDBSCAN sur les colonnes avec descriptions textuelles. Une fois que les clusters sont générés, un clustering incrémental est effectué sur les colonnes sans descriptions textuelles pour leurs affectées un cluster. Grâce à la méthode du Cluster-based TF-IDF, des représentations textuelles sont générées pour chaque thématique (cluster). Ces représentations textuelles permettent de facilement naviguer entre les différents clusters pour choisir les colonnes les plus pertinentes.

En termes de précision, rappel et f1-score les résultats de la deuxième approche sont meilleurs que ceux de la première approche. Cela est dû au fait que les descriptions textuelles des colonnes offrent plus d'informations sémantiques que les noms des colonnes. Et dans la deuxième approche la méthode du Nearest Centroid a montré de meilleurs résultats comparé aux résultats de la méthode K-Nearest Neighbors car avec l'utilisation du KNN le taux d'outliers augmente contrairement au Nearest Centroid où il diminue. Et plus le taux d'outliers augmente plus le nombre d'éléments dans les clusters autres que celui des outliers diminue. Ce qui fait que le nombre d'éléments pertinents qui peuvent correspondre à un business-item diminue aussi.

Ces approches peuvent être généralisées et appliquées pour générer des suggestions de linked-items entre les autres concepts métiers du catalogue de données et les tables et datasets du patrimoine de données. Cela permettrait aussi de générer et d'enrichir les descriptions textuelles des différents objets du catalogue de données grâce aux descriptions textuelles des topics auxquels ils appartiennent ou qui leur sont similaires sémantiquement.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Philip A Bernstein, Jayant Madhavan, and Erhard Rahm. Generic schema matching, ten years later. *Proceedings of the VLDB Endowment*, 4(11):695–701, 2011.

[3] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Advances in neural information processing systems*, 14, 2001.

[4] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[7] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[8] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[9] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[10] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[11] Liqiang Niu, Xinyu Dai, Jianbing Zhang, and Jiajun Chen. Topic2vec: learning distributed representations of topics. In *2015 International conference on asian language processing (IALP)*, pages 193–196. IEEE, 2015.

[12] Tanvi Sahay, Ankita Mehta, and Shruti Jadon. Schema matching using machine learning. In *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 359–366. IEEE, 2020.

[13] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

[14] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[15] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[17] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

[18] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1(1):43–52, 2010.

[19] George Kingsley Zipf. The meaning-frequency relationship of words. *The Journal of general psychology*, 33(2):251–256, 1945.