


M1 Science Cognitives
Composante Sciences numériques, en Sciences Cognitives, traitement automatique des langues et innovation

Projet Tutoré Partie II : Réalisation

Étude de la qualité des données synthétiques pour les séries temporelles

Oscar Bidou, Anais Mignot–Charvillat, Youcef Namoun, Enzo Verbanaz, Rania Ait Chabane, Ayoub Maanni, Othmane Taoussi

Encadrant de projet : Azim Roussanaly

Laboratoire Ouvert en Learning Analytics
Laboratoire Lorrain de Recherche en Informatique et ses Applications
Institut des sciences du Digital, Management et Cognition
Université de Lorraine

November 3, 2025

Contents

List of Figures

List of Tables

Voici un lien vers le GitHub du projet .

1 Présentation du sujet

1.1 Contexte et Problématique

Dans le domaine actuel de la science des données et de l'intelligence artificielle, les données sont une ressource précieuse et essentielle pour l'avancement technologique. Cependant, il existe des situations où les données réelles sont rares, difficiles à obtenir, ou sensibles en raison des problèmes liés à la confidentialité de celles-ci. Ce projet vise à évaluer la qualité des données synthétiques générées comme solution à ces défis. Les données synthétiques sont des données artificiellement générées qui imitent les propriétés des données réelles sans contenir d'informations sensibles, offrant ainsi une alternative pour les analyses et les tests tout en respectant l'anonymisation des données.

1.2 Objectifs du Projet

L'objectif principal du projet est de développer une méthode efficace pour générer des données synthétiques de séries temporelles puis d'implémenter et d'appliquer différentes méthodes d'évaluations qui permettent de :

- Maintenir les propriétés statistiques des données réelles.
- Assurer la confidentialité des informations sensibles.
- Pouvoir être utilisées pour des analyses et des prédictions afin obtenir des scores équivalents à des données réelles.

1.3 Cadre du Projet

Le projet est réalisé en collaboration avec le Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA), qui a développé la plateforme LOLA (Laboratoire Ouvert en Learning Analytics). Cette plateforme centralise des données éducatives, permettant aux chercheurs et aux institutions de tester et d'évaluer des algorithmes sur une base de données commune. Les données utilisées dans ce projet proviennent du CNED (Centre national d'enseignement à distance) et sont sous forme de séries temporelles enregistrant les interactions des utilisateurs avec la plateforme éducative.

1.4 Définition des Séries Temporelles

Les séries temporelles sont des séquences de données discrètes recueillies ou mesurées à intervalles réguliers ou irréguliers au fil du temps. Elles sont cruciales pour l'analyse des tendances et des comportements sur une période donnée. Dans ce projet, les séries temporelles enregistrent des actions telles que les connexions des utilisateurs, les consultations de cours, et d'autres interactions avec la plateforme du CNED.

Exemple de donnée temporelle :

```
Timecode            | Acteur        | Verbe     | Objet
--------------------|---------------|-----------|--------------------
2015-02-04 17:54:00 | Gaëtan Martin | Connexion | Cours de Statistique
```

Idéalement, ces données doivent rester pertinentes dans l'axe de la temporalité. Ce qui signifie que les modèles de génération doivent non seulement reproduire les valeurs individuelles, mais aussi capturer les dynamiques et les dépendances temporelles entre ces valeurs.

1.5 Enjeux et Défis

1.5.1 Confidentialité des Données :

Les données éducatives contiennent des informations sensibles sur les utilisateurs, nécessitant des mesures strictes pour assurer la confidentialité. Le RGPD (Règlement Général sur la Protection des Données) impose des règles strictes sur la manière dont les informations personnelles doivent être traitées, ce qui rend crucial le développement de méthodes robustes pour générer des données anonymes sans compromettre leur utilité.

1.5.2 Qualité des Données Synthétiques :

La qualité des données synthétiques est évaluée en fonction de leur capacité à imiter les données réelles. Cela veut dire que les données synthétiques doivent avoir des propriétés statistiques similaires aux données réelles.

1.5.3 Utilité des Données Synthétiques :

L'utilité implique que les données synthétiques doivent permettre de réaliser des analyses et d'entraîner des modèles prédicatifs aussi bien que sur des données réelles. Elles permettront ainsi à un plus large publique d'étudier et de comprendre comment les utilisateurs interagissent cognitivement avec les plateformes éducatives pour proposer des solutions innovantes.

1.5.4 Méthodologies :

Le projet c'est divisé en deux étapes principales :

1. Génération des données synthétiques : Utiliser et comparer différentes techniques pour produire des données synthétiques.
2. Évaluation de la qualité des données synthétiques : Développer des méthodes pour évaluer la fidélité, l'utilité et la confidentialité des données générées.

Ce projet vise donc à explorer et à implémenter des techniques avancées pour la génération et l'évaluation de données synthétiques, avec un accent particulier sur les séries temporelles. Les résultats attendus contribueront à la recherche en intelligence artificielle et fourniront des solutions pratiques aux défis de l'anonymisation et de la disponibilité des données dans le domaine éducatif.

2 Travail réalisé

2.1 Description des données

Avant tout il est nécessaire de bien comprendre avec quelle donnée nous travaillons. Nous avons à notre disposition plus de 88 fichiers au format .json, avec 1000 exemples en moyenne dans chaque fichier. Ce qui nous permet de travailler avec un corpus d'environ 88 000 exemples. En voici un :

Afin de manipuler les données plus aisément nous les avons nettoyé. Les attributs "version", "objectType" et "parent" étant les mêmes sur tous les exemples, nous avons décidé de supprimer

```
{
  "version": "1.0.3",
  "actor": {
    "objectType": "Agent",
    "mbox": "mailto:285698@moodle-example.com"
  },
  "verb": {
    "id": "http://activitystrea.ms/schema/1.0/submit",
    "display": {
      "en": "submit"
    }
  },
  "object": {
    "id": "http://moodle-example.com/11",
    "objectType": "Activity"
  },
  "timestamp": "2018-05-18T00:00:00",
  "context": {
    "contextActivities": {
      "parent": [
        {
          "id": "http://moodle-example.com/course_1562",
          "objectType": "Activity"
        }
      ]
    }
  }
}
```

Figure 1: Exemple d'un individu en format Json

ces informations redondantes ne permettant pas d'inférer quoi que ce soit. Nous avons suivi la norme xAPI et avons donc créer un tableau à quatres colonnes.

Timestamp	Actor	Verb	Object
2018-05-18T00:00:00	285698@moodle-example.com	submit	http://moodle-example.com/11

Table 1: Exemple de donnée xAPI

Pour des raisons pratiques et techniques, nous avons simplifié encore une fois l'affichage. Le timestamp a été formaté afin de résoudre le problème de fuseau horaire. De plus, le timestamp a été transformé en epoch pour la génération avec le GAN afin de coller au modèle. C'est-à-dire que nous avons le nombre de secondes écoulées depuis le premier enregistrement. Pour les colonnes *Actor* et *Object*, nous avons enlevé les informations superflues.

Timestamp	Actor	Verb	Object
16530.0	285698	submit	11

Table 2: Donnée simplifiée xAPI

Notre jeu de donnée globale est donc un dataframe de 87231 lignes et 4 colonnes. Nous avons bien mis les données en type *object* pour bien les traiter comme des données catégorielles.

Timestamp	float64
Actor	object
Verb	object
Object	object
dtype: object	

Table 3: Type des valeurs du tableau

Voici les premières et dernières lignes de notre jeu de données prêt à être utilisé pour entraîner nos modèles de génération et comparer les jeux de données synthétiques :

Timestamp	**Actor**	**Verb**	**Object**
0.0	317406	scored	3
3423.0	331663	submit	resource_113547
16530.0	321189	viewed	1
18263.0	198693	submit	course_1562
28414.0	279052	submit	9
...	...	...	...
28813450.0	321706	scored	course_1562
28813808.0	335904	scored	1
28817324.0	330042	viewed	course_1562
28825299.0	286338	submit	7
28829718.0	325828	viewed	11

Table 4: Première Traitement

Les distributions de chaque colonne nous permet de mieux comprendre le jeu de donnée.

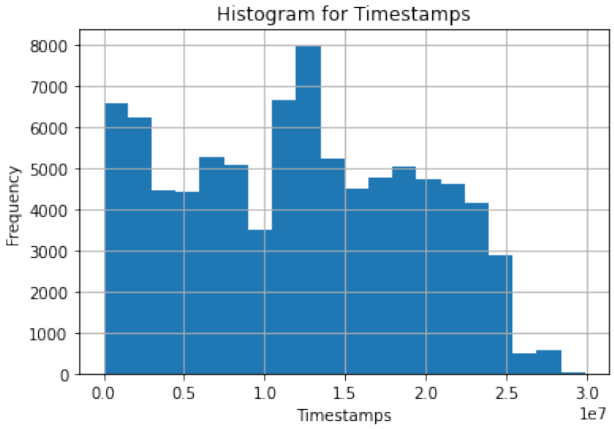

Figure 2: Distribution des TimeStamp des données réelles

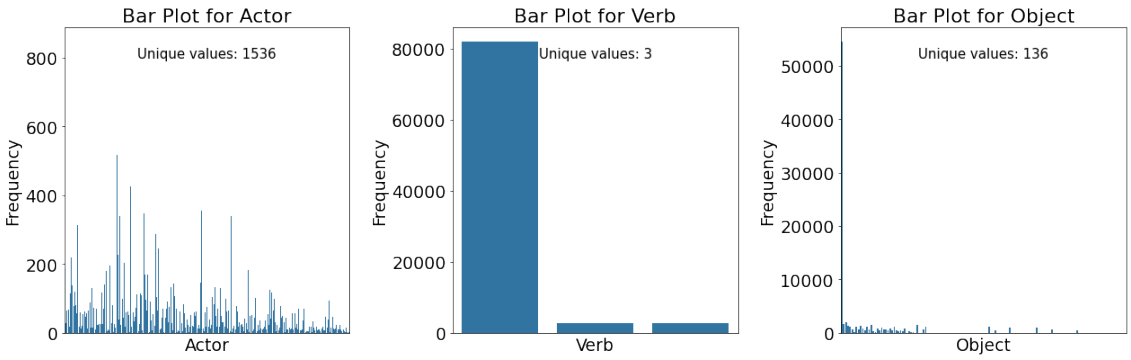

Figure 3: Nombre d'individus unique par catégorie pour les données réelles

2.2 Génération de données

La première partie de notre travail a consisté à générer des données. Nous savions que les données à répliquer seraient sous forme de séries temporelles. C'est pourquoi, après nos recherches bibliographiques, nous nous sommes orientés vers un GAN (Generative Adversarial Network) dans une variante temporelle appelée TimeGAN, qui permet de générer des données en intégrant une dimension temporelle.

2.2.1 TimeGAN

Pour appliquer TimeGAN à nos données, nous avons d'abord encodé les données en utilisant l'outil LabelEncoder de la bibliothèque scikit-learn. Le LabelEncoder attribue un entier unique à chaque valeur distincte dans une colonne, créant ainsi une représentation numérique des catégories. Nous l'avons appliqué aux trois colonnes catégorielles "actor", "verb" et "object". Concernant la colonne "timestamp", nous l'avons convertie en secondes pour maintenir la dimension temporelle des séries chronologiques.

Voici notre implémentation :

```
# Split the data
x_train, x_test = df[:int(0.8 * len(df)], df[int(0.8 * len(df):]

# Fit the model to the training data
model = TimeGAN(
    x=x_train,
    timesteps=20,
    hidden_dim=64,
    num_layers=3,
    lambda_param=0.1,
    eta_param=10,
    learning_rate=0.001,
    batch_size=16
)

# Train the model with the training data
model.fit(
    epochs=500,
    verbose=True
)

# Reconstruct the test data
x_hat = model.reconstruct(x=x_test)

# Generate the synthetic data
x_sim = model.simulate(samples=len(x_test))
```

Nous avons généré des données de ce type:

Timecode	Acteur	Verbe	Objet
12054577	296	2	6
12054592	296	2	58
12054819	296	2	60
12054892	967	2	59
12054964	296	2	6

Table 5: Données générée par TimeGAN

Cependant, nous n'avons pas été satisfaits par les données générées. En effet, une chose que nous avons éludée jusque-là est que nous avons dû encoder nos données catégorielles en données numériques pour qu'elles puissent entrer dans le modèle. Pour cette partie, nous avons d'abord encodé les données en utilisant l'outil LabelEncoder de la bibliothèque scikit-learn. Le LabelEncoder attribue un entier unique à chaque valeur distincte dans une colonne, créant ainsi une représentation numérique des catégories. Nous l'avons appliqué aux trois colonnes catégorielles "actor", "verb" et "object".

Sauf qu'en réalité, un paramètre que nous n'avions pas pris en compte est que nos données sont à l'origine des données tabulaires catégorielles, et qu'elles n'ont rien à voir avec les données que le modèle TimeGAN peut accepter. Il en résulte que l'encodage numérique des catégories crée une relation ordinale artificielle entre les valeurs, qui n'existe pas dans les données d'origine.

Par exemple, si "Verb" est encodé comme 0, 1, 2 pour les valeurs "viewed", "submit", "scored" respectivement, le modèle TimeGAN peut interpréter que ces valeurs ont une relation d'ordre ou de distance entre elles (comme si 1 était quelque part entre 0 et 2), ce qui est incorrect. Par

conséquent, les données générées ne respectent pas les relations originales entre les catégories. Les relations contextuelles et sémantiques essentielles sont perdues, ce qui rend les données générées inexploitables.

2.2.2 CTGAN Model

Nous avons donc entrepris de chercher un autre modèle de génération de données qui puisse prendre en entrée des données catégorielles tabulaires. Nous avons finalement jeté notre dévolu sur un modèle nommé CTGAN, qui est parfaitement adapté à ce type de données.

CTGAN (Conditional Tabular Generative Adversarial Networks) (Xu et al. 2019) est un modèle de réseau de neurones génératif inspiré de l'architecture des GANs (Generative Adversarial Networks) (Goodfellow et al. 2020). Le principe sous-jacent implique l'entraînement d'un générateur (G) et d'un classificateur (appelé discriminateur D), où le générateur tente de produire des données aussi proches que possible de la réalité afin de tromper le classificateur D.

Ce processus permet de générer des données synthétiques très réalistes et très similaires à celles de l'ensemble de données réel. Cependant, CTGAN n'est pas spécifiquement conçu pour intégrer la dimension temporelle des données. Pour résoudre cette lacune, nous avons converti les timestamps en secondes, préservant ainsi la structure temporelle des séries chronologiques.

Nous évaluerons ensuite si cela s'est avéré vrai ou faux. Après la phase d'entraînement, ce modèle devrait normalement être capable de générer des données tabulaires catégorielles similaires aux données originales en entrée. Pour ce faire, nous avons procédé à l'encodage de la colonne "timestamp" en la convertissant en secondes, puis nous avons trié notre ensemble de données en fonction de cette colonne. Cela nous permet de préserver la notion temporelle des séries chronologiques.

Pour implémenter le modèle CTGAN, nous avons utilisé la librairie SDV (The Synthetic Data Vault),z une bibliothèque open-source qui fournit des outils pour générer des données synthétiques de manière efficace et contrôlée.

Voici notre implémentation :

```
# Define metadata and constraints
metadata = SingleTableMetadata()
metadata.detect_from_dataframe(df)

# Fit the model to the training data
ctgan = CTGANSynthesizer(metadata, epochs=200)

# Train the model with the training data
ctgan.fit(df)

# Generate the synthetic data
synthetic_data = ctgan.sample(len(df))
```

Nous avons généré des données de ce type:

Timecode	Acteur	Verbe	Objet
401989	200851	viewed	resource_113514
2777793	254924	viewed	course_1562
24147602	279789	scored	11
9302319	205456	viewed	course_1562
6908414	13726	viewed	resource_113519

Table 6: Données généré par CTGAN

2.2.3 PAR Model

En documentant la librairie SDV nous avons découvert le modèle PAR qui est un modèle de génération de données spécialement conçu pour des données temporelles.

PAR (Probabilistic AutoRegressive) est une implémentation avancée d'un modèle probabiliste autorégressif spécialement conçu pour l'apprentissage des données de séries temporelles multivariées et multi-types. Ce modèle spécialement conçu pour des données timseries permet de capturer les dynamiques et les relations complexes entre les différentes variables au sein des séries temporelles.

Une fois entraîné, le modèle PAR peut générer de nouvelles données temporelles synthétiques conditionnées aux propriétés de l'entité à laquelle ces données de séries temporelles seraient associées.

Pour implemté ce modèle, nous avons utilisé la bibliothèque SDV (Synthetic Data Vault). SDV fournit une implémentation simple et rapide de PAR ainsi que des paramètres de configuration flexibles pour adapter le modèle à nos besoins spécifiques.

Nous avons défini les paramètres suivants :

- `entity_columns`: identifient les associations entre les lignes et des entités externes. Dans notre cas, nous n'avons pas d'entités externes, donc `entity_columns` = [].
- `context_columns`: fournissent des informations sur les entités associées aux séries temporelles et peuvent conditionner leur évolution. Dans notre cas, nous n'avons pas de telles colonnes, donc `context_columns` = [].
- `sequence_index`: ordonne les lignes dans les ensembles de données de séries temporelles pour refléter les dépendances entre elles. Dans notre cas, les timestamps ordonnent nos donnes, donc `sequence_index`='Timestamp'.

Voici notre implémentation :

```
# Define entity, context and sequence index columns
entity_columns = []
context_columns = []
sequence_index = 'Timestamp'

# PAR Model
model = PAR(
    entity_columns=entity_columns,
    context_columns=context_columns,
    sequence_index=sequence_index,
)

# Fit the model to the training data
model.fit(df)

# Generate the synthetic data
synth_data = model.sample(1)
```

Nous avons généré des données de ce type:

Timecode	Acteur	Verbe	Objet
2017-09-05 07:49:52	209424	viewed	course_1562
2017-09-05 10:05:53	301517	scored	3
2017-09-05 17:16:33	336434	submit	5
2017-09-06 15:33:07	357715	viewed	course_1562
2017-09-06 18:05:38	353102	viewed	resource_113539

Table 7: Données généré par PAR

Bien que le modèle PAR soit très prometteur, nous avons rencontré un problème de taille lors de la génération. En raison de la taille de notre dataset, il était impossible d'entraîner le modèle sur les 88 fichiers. Nous nous sommes donc contentés de 10 fichiers, soit environ 10 000 lignes, pour la base d'entraînement, et nous avons généré un jeu de données d'environ la même grandeur. Nous avons utilisé les 10 derniers fichiers qui contenaient toute la variété de données. Nous pensons qu'avec une infrastructure suffisamment grande, le modèle PAR aurait pu être plus performant en utilisant toutes les données pour l'entraînement.

2.2.4 Alternative à la génération

Face au problème que nous avons rencontré lors de la génération des données, nous avons essayé une approche différente afin de nous permettre d'avancer sur les méthodes d'évaluation. Nous avons divisé le jeu de données en deux de façon aléatoire. La première moitié des données serait les données "réelles" et la seconde "synthétiques". Cette méthode, bien qu'imparfaite en de nombreux points, permet d'avoir une uniformité de répartition des différentes catégories. La division en deux jeux ne se veut ni réaliste ni même crédible. Elle a uniquement pour but de permettre l'avancement du projet. De cette façon, tous les résultats des tests devraient être très performants. Nous appellerons ce jeu de données *False*. Il servira de base comparative pour les tests des autres générations.

2.3 Évaluation

2.3.1 Confidentialité des données synthétiques

- Corrélation inter-table

  Il est intéressant de regarder à quel point les données originales et synthétiques se ressemblent. Si nous arrivons à un point où les données sont si proches qu'il est impossible de discriminer si un individu a été généré ou s'il était présent dans le jeu de données original, cela nous assure que la confidentialité des données est respectée.

  Nous comparons donc la répartition de nos trois jeux de données générés avec celle du jeu de données original, visualisable sur la figure **??**, pour les trois colonnes catégorielles. En effet, il n'est pas pertinent, à notre sens, de comparer la génération des timestamps. Ceux-ci auraient d'ailleurs pu ne pas être générés. Il est tout à fait envisageable de se concentrer uniquement sur la génération des données catégorielles.

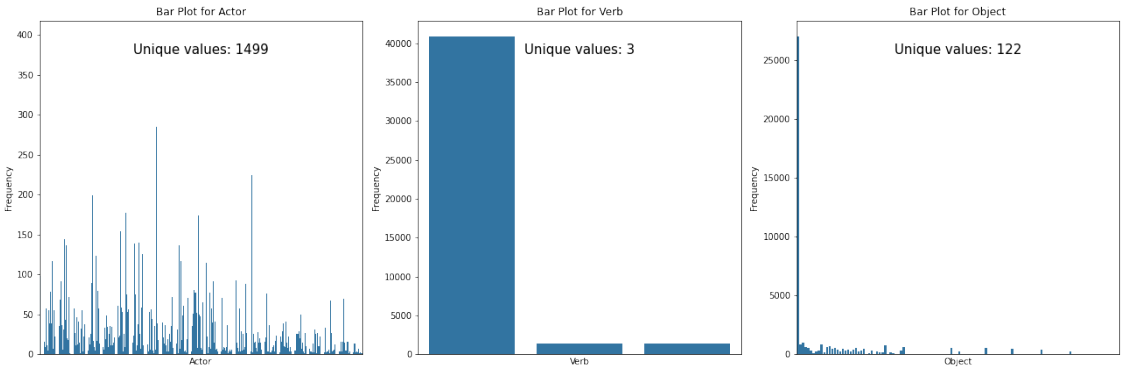

Figure 4: Nombre d'individus unique par catégorie pour les fausses données synthétiques

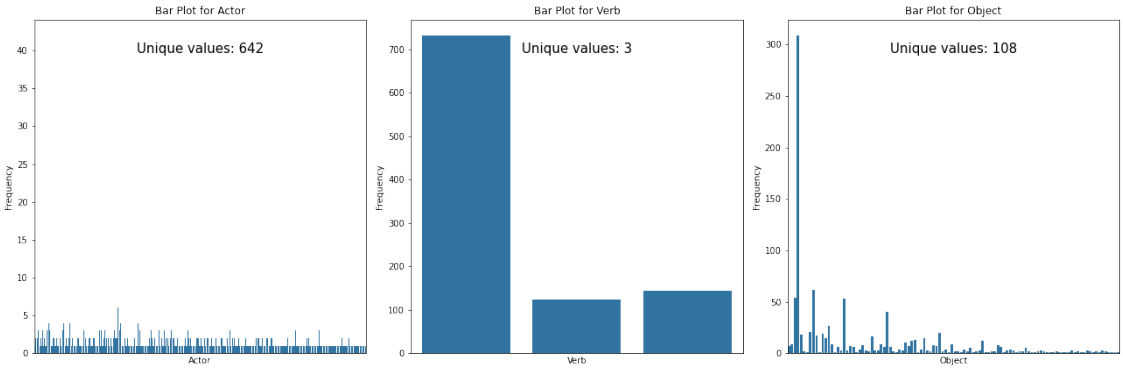

Figure 5: Nombre d'individus unique par catégorie pour les données CTGAN

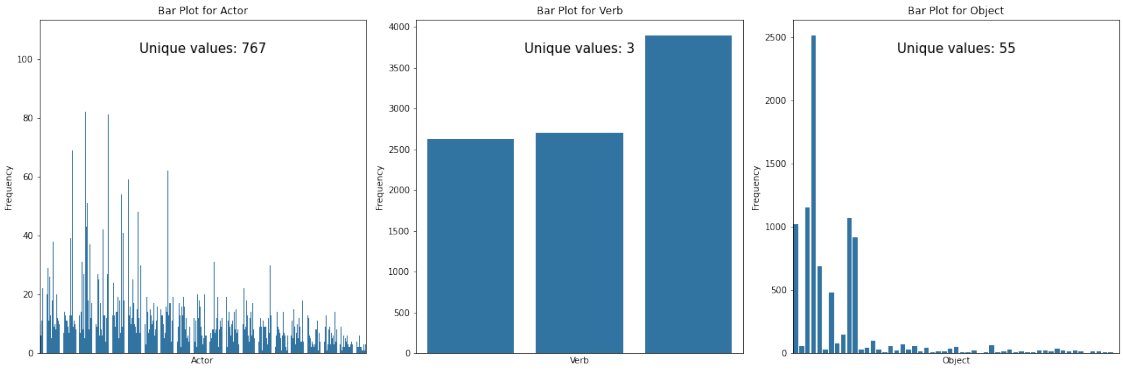

Figure 6: Nombre d'individus unique par catégorie pour les données PAR

Nous remarquons qu'il n'est pas pertinent de visualiser les données telles quelles. Le nombre de valeurs uniques n'est pas équivalent selon les jeux de données, car nous n'avons pas pu les générer de manière équivalente en raison de complications techniques. Par exemple, nous observons 124 valeurs d'objets dans les données originales, 108 valeurs dans les données générées avec le GAN et uniquement 55 valeurs dans les données synthétiques générées avec le modèle PAR. Les diagrammes en barres ne sont donc pas un indicateur de qualité dans cette

configuration, et il est préférable d'effectuer des tests statistiques plutôt qu'une évaluation visuelle. C'est pourquoi nous allons directement afficher les résultats des tests statistiques effectués sur les données normalisées.

χ^2 et χ^2 de contingence

Avec le test de χ^2, nous cherchons à déterminer si les deux jeux de données (original et synthétique) suivent la même loi de probabilité pour les données catégorielles.

Premièrement, nous examinons colonne par colonne si les catégories comportent le même nombre d'exemplaires dans les données originales et synthétiques, et cela pour les trois jeux de données synthétiques normalisés.

Pour la colonne *Actor*, seuls les 642 premiers exemplaires ont été conservés afin de pouvoir comparer des jeux de données de même taille, 642 étant la taille minimale parmi les quatre jeux de données.

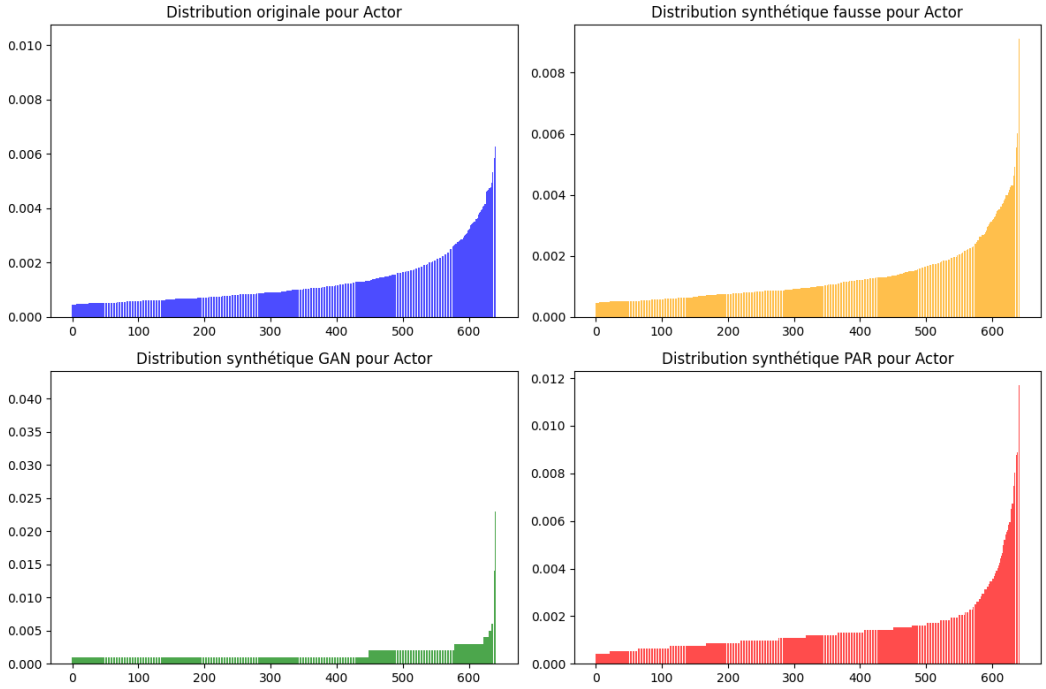

Figure 7: Répartition des valeurs selon les quatre jeux de données pour la colonne *Actor*

Résultat des tests :

Nous avons donc effectué trois test du χ^2 sur la colonne des acteurs.
Comparaison entre données originales et fausses données générées.
Comparaison entre données originales et données générées avec GAN.
Comparaison entre données originales et données générées avec PAR.

Cependant, une erreur est apparue lors de la réalisation des tests : *"Erreur lors de la comparaison des données synthétiques avec celles des données originales pour la colonne Actor : Les sommes des fréquences observées et attendues sont trop proches."*

Cela a été le cas pour les trois tests. Les répartitions étant trop similaires, le test n'a pas pu être effectué. D'un côté, cela nous montre que les données sont si proches qu'elles sont indis-

cernables, ce qui peut être un indicateur de confidentialité. Cependant, cela peut également indiquer que les jeux de données sont en réalité équivalents, et qu'aucune anonymisation n'a donc été faite.

En outre, nous pouvons observer visuellement que la répartition faite par le modèle GAN semble moins fidèle aux autres. D'autres tests seront effectués dans la section concernée pour approfondir cette observation.

Pour la colonne *Verb* les trois attributs ont été conservés.

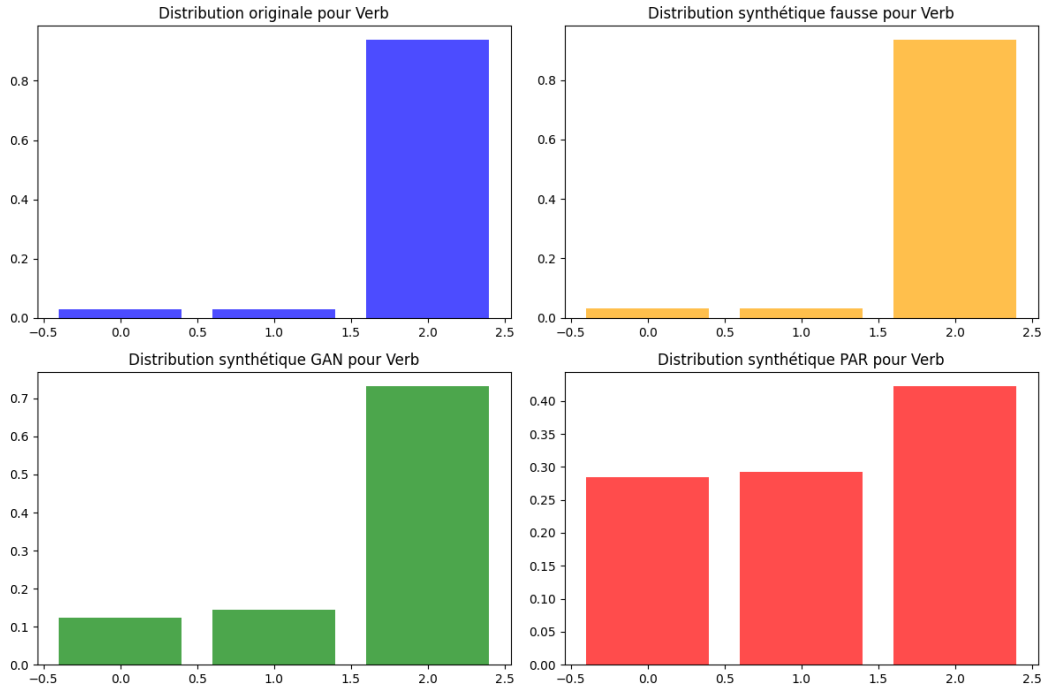

Figure 8: Répartition des valeurs selon les quatre jeux de données pour la colonne *Verb*

Résultat des tests :

Nous avons donc effectué trois test du χ^2 sur la colonne des verbes.
La comparaison entre les données originales et les fausses données générées donne une p_{valeur} égal à 0.9999317475103215.
La comparaison entre les données originales et les données générées avec GAN donne une p_{valeur} égal à 0.8958520037866342.
La comparaison entre les données originales et les données générées avec PAR donne une p_{valeur} égal à 0.5776878216716175.

Les $p_{valeurs}$ concordent bien avec les observations faites sur les graphiques. Les répartitions du jeu de données original et de celui faussement synthétique suivent la même loi de probabilité avec une p_{valeur} très élevée. La répartition du jeu de données généré par le GAN semble visuellement moins fidèle au jeu de données original que le faux jeu de données. Cette intuition se confirme lors du test avec une p_{valeur} plus faible, bien qu'encore élevée. Enfin, la distribution du jeu de données synthétique généré grâce à PAR semble visuellement peu en lien avec la loi de probabilité de la distribution originale, et la p_{valeur} est encore plus faible.

L'importante différence entre le jeu de données original et le jeu de données synthétique généré avec le modèle PAR soulève un problème. Dans le cadre de la confidentialité, si les deux jeux de données ne se confondent pas, il est très aisé de les différencier, et l'anonymat des personnes n'est donc plus certain. Cependant, ce manque de discernement est un bon signe pour la fidélité des données.

Pour la colonne *Object*, de la même manière que pour la colonne *Actor*, les résulats n'ont pas pu être calculé.

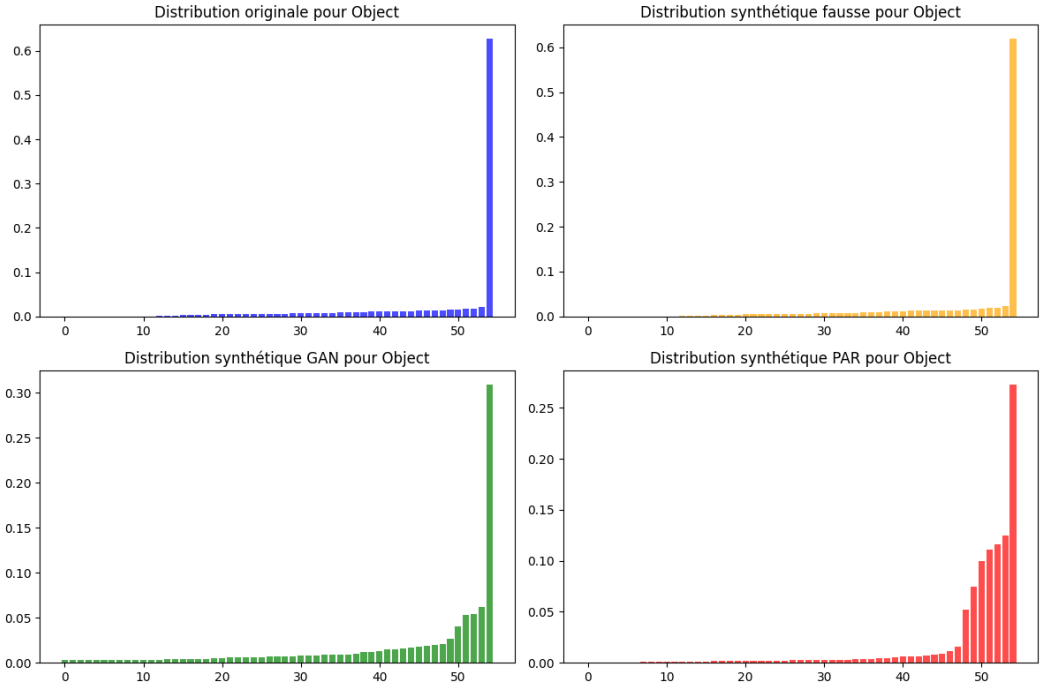

Figure 9: Répartition des valeurs selon les quatre jeux de données pour la colonne *Object*

– χ^2 **Contingence**

Nous utilisons le test du χ^2 pour examiner plus en détail les liens entre les variables grâce aux tables de contingence. Cela permet de déterminer les dépendances et les relations entre deux caractères généralement qualitatifs. Nous analysons l'indépendance entre les trois catégories, à savoir Actor, Verb et Object, comme précédemment. Nous vérifions que ces dépendances sont les mêmes dans le jeu de données original et dans les jeux de données synthétiques.

Data original			
Verb	0	1	2
Actor			
100543	2	0	0
100688	50	0	0
100723	11	3	3
100913	2	0	0
100933	7	2	1
...	..	..	..
98965	3	0	0
99299	46	0	0
99521	89	0	0
99583	6	0	0
99621	25	0	0

Synthetic data			
Verb	0	1	2
Actor			
100543	6	0	0
100688	40	0	0
100723	16	1	1
100913	2	0	0
100933	10	1	2
...	..	..	..
98965	4	0	0
99299	49	0	0
99521	75	0	0
99583	3	0	0
99621	24	0	0

Table 8: Tableaux de contingence entre Verb et Actor pour le jeu de donnée original (à gauche) et le faux jeu de donnée synthétique (à droite)

De la même manière, nous avons créé les tableaux de contingence pour toutes les paires de colonnes dans tous les jeux de données. Ensuite, nous avons effectué un test du χ^2 sur ces tableaux de contingence afin de déterminer s'il y avait une dépendance entre les variables au sein d'un jeu de données. Les résultats des différents tests réalisés sont affichés dans le tableau ci-dessous :

Comparaison	Données originales	Fausse données synthétiques	Données synthétiques CTGAN	Données synthétiques PAR
Actor vs Verb	0.0	0.0	0.002396	0.560720
Actor vs Object	0.0	0.0	1.0	0.002284
Verb vs Object	0.0	0.0	0.0	0.1742569

Table 9: P-value des χ^2 - arrondis à six décimales près

La dernière étape consiste donc à évaluer si les dépendances sont bien les mêmes entre les jeux de données. Concernant les données originales et les données synthétiques générées, il n'est pas surprenant de retrouver de fortes dépendances entre les variables dans les deux jeux de données, avec une p-valeur proche de zéro.

Le jeu de données synthétique généré avec le GAN n'a pas du tout réussi à capturer la dépendance entre les acteurs et les verbes, ce qui nuit à sa capacité à générer des données synthétiques de qualité. Il se peut que le lien entre ces deux variables soit temporel, ce que les modèles GAN ne sont pas particulièrement capables de générer.

Le générateur PAR a, quant à lui, réussi à capturer la dépendance entre les acteurs et les objets, mais a été moins performant globalement. Les autres p-valeurs étant supérieures à 0,05, on ne peut pas considérer qu'il a réussi à capter la dépendance entre les variables *Actor* et *Verb*, ainsi qu'entre *Verb* et *Object*.

– Coefficient de contingence

Tester directement la similitude des distributions entre les données originales et synthétiques à l'aide du test du χ^2 est pertinent mais limité, comme nous venons de le constater. C'est pourquoi nous évaluons maintenant la force des associations entre les variables dans chaque ensemble de données. Cette force d'association est calculée grâce à un coefficient de contingence. Un coefficient proche de 1 indique une forte association entre les variables, mais il n'indique pas directement si les tables de contingence sont similaires ou différentes entre les ensembles de données.

Voici les résultats obtenus :

- Coefficient de contingence pour les données initiales : 0.939

- Coefficient de contingence pour les données synthétiques false : 0.936

- Coefficient de contingence pour les données synthétiques gan : 0.993

- Coefficient de contingence pour les données synthétiques par : 0.962

Nous observons alors que les coefficients de contingence des données originales et des fausses données synthétiques sont extrêmement proches. De plus, il est intéressant de noter que les modèles génératifs ont tous les deux exacerbé les dépendances entre les variables.

V de Cramer

Pour effectuer le test du V de Cramer sur une table de contingence, il faut que toutes les catégories soient les mêmes. Nous avons donc décidé de conserver uniquement les catégories qui apparaissent dans les deux jeux de données. Un V de Cramer proche de zéro équivaut à une corrélation nulle. À l'inverse, la corrélation est considérée comme trop forte dès que le V de Cramer dépasse 0,2.

- V de Cramer pour False: 0.01962910993480217
- V de Cramer pour CTGAN: 0.11896733555389764
- V de Cramer pour PAR: 0.0

La corrélation entre PAR et les données réelles (0.0) est étonnamment bien supérieure à celle entre CTGAN et les données réelles (0.11). C'est en réalité le cas si les données générées par PAR sont totalement différentes de celle du jeu de donnée initial.

- DCR

  La distance au plus proche enregistrement permet d'évaluer la qualité de la confidentialité de nos données. Un algorithme ayant un bon score de confidentialité devrait avoir une probabilité moyenne de proximité supérieure sur le jeu de données réelle qu'au jeu de données HoldOut. Une probabilité de 0.5 serait le pire score possible.

  Ce test a été effectué uniquement sur les données *False*. Les autres générateurs étant trop complexes et trop longs pour générer une nouvelle fois des données avec uniquement la moitié des données originales.

Groupe	DCR
All	0.498
Object	0.499
Actor	0.502
Verb	0.5

Table 10: DCR

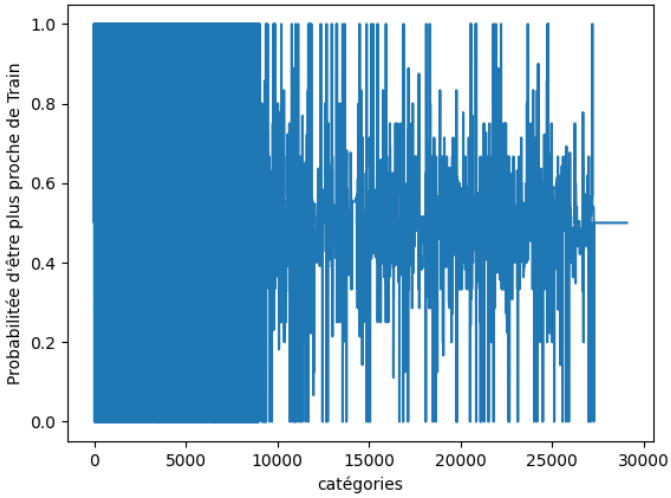

Figure 10: Probabilité d'être plus proche des données réelles que des données HoldOut

La probabilité de proximité moyenne est similaire entre les données réelles et HoldOut (≈ 0.5). *False* est ce qu'il existe de pire en confidentialité, ce qui était attendu.

2.3.2 Utilité des données

- Score de propension (pMSE) / F1 score / MAPE (Rosenbaum and Rubin 1983)

  Le score de propension est une estimation de la capacité d'un modèle à différencier les données réelles des données synthétiques. Plus l'algorithme est capable de faire cette distinction, moins les données synthétiques sont performantes. De même, le F1 score et la MAPE donnent une idée globale de l'utilité des données générées.

  Nous ne pouvons pas donner d'évaluation précise de l'utilité, car le classificateur utilisé est propre à l'utilisateur. Nous avons donc décidé d'implémenter des classificateurs classiques sur nos données *False* afin de représenter cette mesure. Ils ne servent qu'à titre indicatif. Les classificateurs utilisés sont : GaussianNB, DecisionTreeClassifier, KNeighborsClassifier et LogisticRegression.

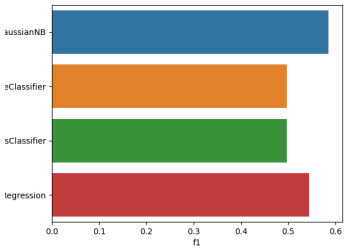

Figure 11: F1 score

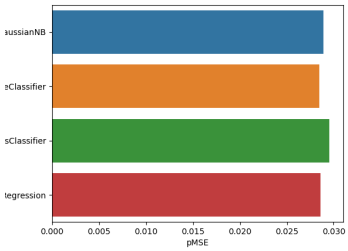

Figure 12: pMSE

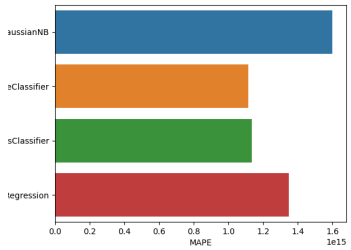

Figure 13: MAPE

Figure 14: Score des classeurs sur *False synthétique*

2.3.3 Fidélité des données

Nous avons décidé d'effectuer les tests ci-dessous sur les données totales *All*, mais aussi sur les données regroupées par colonnes (*Object*, *Actor*, *Verb*). De cette façon, nous pouvons mieux comprendre les différences de répartitions des données générées.

- TVC: Variation totale de la corrélation

  Après avoir mesuré la TVC pour chaque type de données différent, nous obtenons un score de 1 pour toutes les catégories. Le score 1 est le score maximal, indiquant une corrélation linéaire parfaite entre nos données réelles et synthétiques. Toutes les catégories de tous nos jeux de données synthétiques sont parfaitement corrélées avec les données réelles.

méthode de génération	All	Object	Actor	Verb
CTGAN	0.11	0.62	0.0	0.79
PAR	0.25	0.46	0.0	0.48
False	0.69	0.98	0.91	0.99

Table 11: TVC

- MSE: Erreur quadratique moyenne

  Bien qu'étant uniquement une mesure spatiale et non temporelle, l'application d'une MSE sur les répartitions de nos données peut s'avérer pertinente pour comparer la distance entre le nombre de données par catégorie.

méthode de génération	All	Object	Actor	Verb
CTGAN	290	21785143	8815	2197576717
PAR	290	20018232	8815	2025350140
False	4	1115	57	7928

Table 12: MSE

  Les résultats de la MSE montrent d'importantes divergences dans les répartitions, notamment lorsque nous regroupons les catégories par les *Object* et les *Verb*. Cependant, la distance moyenne est beaucoup plus faible pour toutes les catégories (*All*) et pour les catégories *Actor*. Cette mesure reflète les différences positionnelles et le nombre de valeurs au sein des séries temporelles de chaque catégorie. Il semble donc que CTGAN et PAR soient très similaires sur ce point, mais assez éloignés de la distribution réelle.

- KL: Divergence de Kullback-Leibler (Kullback and Leibler 1951)

  La divergence de KL est une mesure plus pertinente que la MSE pour évaluer la similarité spatiale de nos données. Un score de 0 équivaut à des données similaires. Cette mesure étant uniquement applicable aux données numériques, nous n'avons pas pu l'appliquer sur nos séries (qui sont catégorielles). Nous avons donc mesuré la divergence de KL sur les répartitions des données. Si la divergence est proche de 0, alors les répartitions sont très similaires.

méthode de génération	All	Object	Actor	Verb
CTGAN	0.0000476	0.0069	0.0005	0.31
PAR	0.000033	0.0056	0.0015	0.25
False	0.0002	0.0001	0.00004	0.000026

Table 13: Divergence de KL

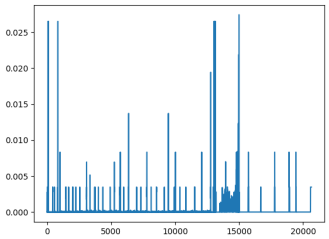

Figure 15: KL gan

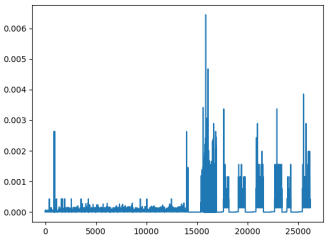

Figure 16: KL par

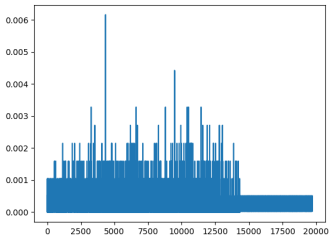

Figure 17: KL fasles

Figure 18: Divergence de KL sur toutes les catégories *False synthetique*

Les résultats obtenus par la divergence de KL sur les répartitions montrent une similitude très importante, de l'ordre de 10^{-5}. De manière similaire à la MSE, la divergence est la moins performante sur les *Verb*. Les jeux de données CTGAN et PAR ont une divergence de leur distribution très faible, ce qui est un très bon indicateur de la similitude des données synthétiques.

- Entropie croisée (Jamin and Humeau-Heurtier 2020)

  Similairement à la divergence de KL, nous avons calculé l'entropie sur les distributions des données. Plus l'entropie est forte, plus les distributions sont similaires.

méthode de génération	All	Object	Actor	Verb
CTGAN	30.5	2.46	34.4	0.42
PAR	18.2	3.05	34.4	0.88
False	12.69	2.04	6.79	0.27

Table 14: Entropie croisée

Les résultats de l'entropie sont très cohérents avec la divergence de KL, ce qui est attendu. Nous pouvons toutefois relever que le CTGAN a une entropie croisée générale presque deux fois supérieure à celle de PAR (30.5 pour le CTGAN / 18.2 pour le PAR). Alors que les entropies croisées sur les différentes catégories sont très similaires. CTGAN aurait alors une meilleure synthétisation des inter-relations entre les catégories.

- Visualisation des distributions

  Afin de visualiser la répartition de nos données, nous avons pour objectif d'effectuer une Analyse en Composantes Principales Temporelle (t-ACP). Pour ce faire, il est nécessaire de

numérisée nos données au préalable. Les *Verb* ont été labellisés ('viewed': 0, 'scored': 1, 'submit': 2), et les *Actor* ainsi que les *Object* ont été hashés en 5 dimensions chacune. La première étape consiste à étudier la variance expliquée en fonction du nombre de fenêtres temporelles choisies. Nous avons donc effectué une t-ACP sur nos données en fonction de 10 nombres de fenêtres différents : 1, 11, 21, 31, 41, 51, 61, 71, 81 et 91.

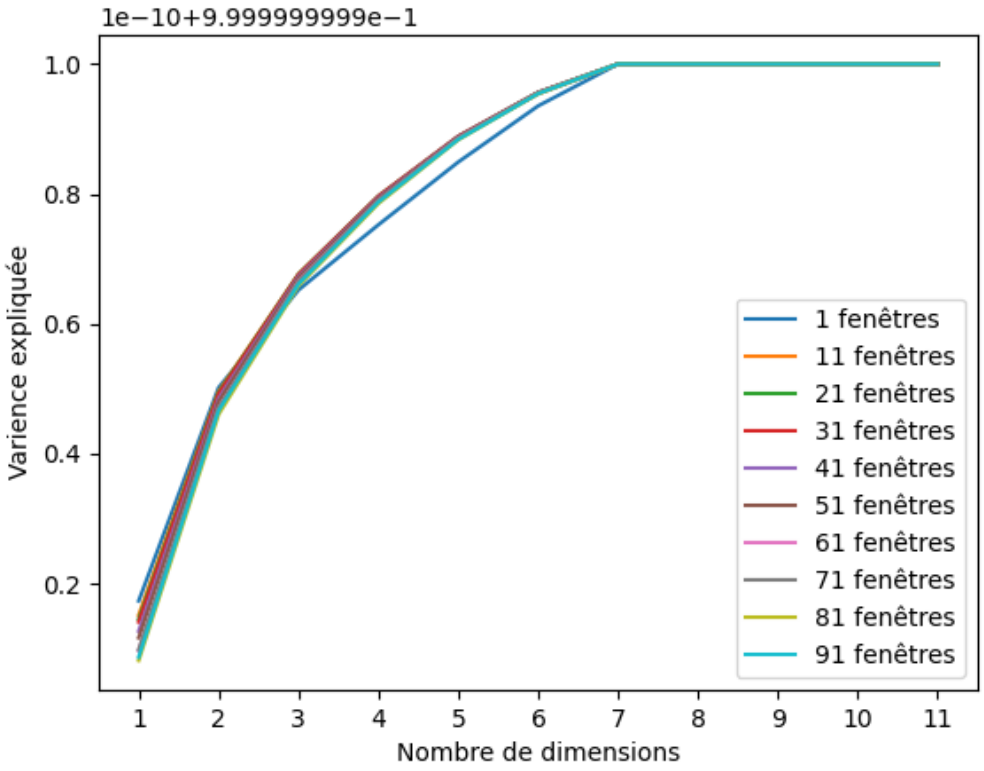

Figure 19: Variances expliqué en fonction du nombre de fenêtres temporelle

La représentation de la variance expliquée en fonction du nombre de dimensions nous confirme qu'il existe peu voir pas de différence entre le nombre de fenêtres choisies. Afin de représenter graphiquement les données, nous sommes obligés de les projeter en 2 dimensions, ce qui expliquerait 50% de la variance. Nous effectuerons donc notre t-ACP sur 21 fenêtres et deux dimensions pour la représentation. Il est important de noter que si nous avions pour objectif d'effectuer des tests statistiques en aval de cette projection vectorielle, il aurait été plus pertinent de choisir une projection en 5 dimensions, ce qui aurait expliqué 90% de la variance. Ainsi, nous aurions pu capturer la quasi-totalité de la variance pour la moitié des dimensions.

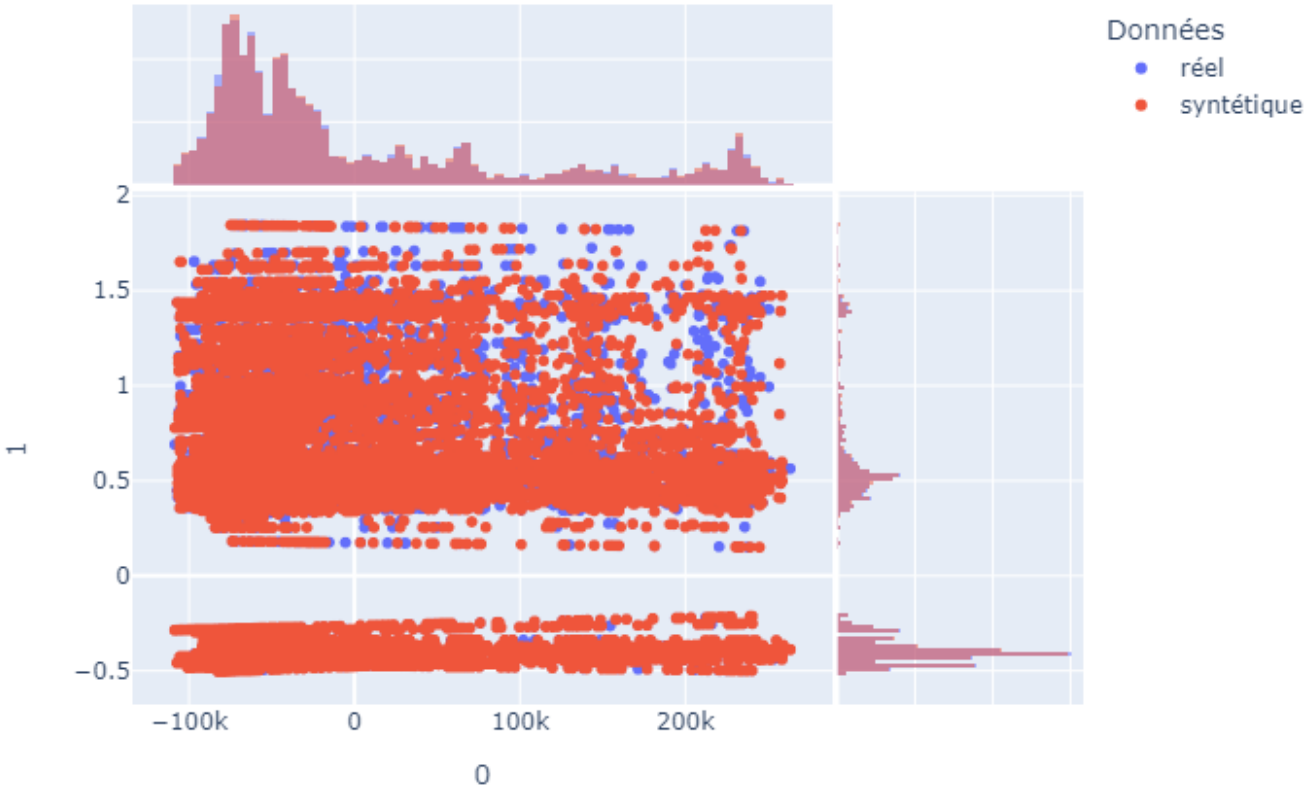

Figure 20: ACP temporel

La projection de nos données ne reflète aucune divergence dans les répartitions. Cela correspond au résultat attendu, car nous avons effectué cette t-ACP sur le classeur *False*. En effet, ce test démontre que 50% de l'information spatiale est similaire.

3 Résumé

Le récapitulatif des scores est donné dans le tableau ci-dessous. Il est clair par presque toutes les métriques que CTGAN est plus performant que PAR et False.

Si CTGAN, PAR et False ont tous les trois des scores de confidentialité très mauvais lors des tests de χ^2, le CTGAN est beaucoup plus performant sur les tables de contingence que le PAR (respectivement 0/le meilleur possible et 1/le pire possible). Cependant, il est important de noter la meilleure performance de PAR(0.0) par rapport à CTGAN(0.11) et False(0.02) en ce qui concerne le V de Cramer. Cette performance est sûrement explicable par la manière dont nous avons décidé d'implémenter cette mesure.

La différence de performance entre CTGAN, PAR et False est de nouveau retrouvée dans la fidélité des données, avec une entropie croisée globale significativement supérieure pour CTGAN (pour CTGAN:30.5, PAR:18.2, False:12.69).

Ne pouvant pas évaluer l'utilité de nos données générées, il nous est impossible de conclure sur le classificateur le plus performant.

Confidentialité				
Évaluation		Ct-GAN	PAR	False synthétique
DCR	All			0,498
	Object			0,499
	Actor			0,502
	Verbe			0,5
χ^2 p_value	Actor réel vs syn	trop similaire	trop similaire	trop similaire
	Object réel vs syn	0,895	0,577	0,999
	Verb réel vs syn	trop similaire	trop similaire	trop similaire
χ^2 sur contingence	Actor vs Verb vs Object	1.0	1.0	1.0
	Actor vs Verb	0.002396	0.560720	0
	Actor vs Object	1.0	0.002284	0
	Verb vs Object	0.0	0.1742569	0
Coefficient de contingence		0.9933	0.9624	0.9358
V de Cramer		0,1189	0	0,0196

Utilité				
Évaluation		Ct-GAN	PAR	False synthétique
F1 score	Gaussian NB			0,585
	Decission tree classifier			0,497
	KNeighborsClassifier			0,497
	LogisiticRegression			0,544
MAPE	Gaussian NB			1,50E+15
	Decission tree classifier			1,10E+15
	KNeighborsClassifier			1,34E+15
	LogisiticRegression			1,34E+15
pMSE	Gaussian NB			0,029
	Decission tree classifier			0,028
	KNeighborsClassifier			0,029
	LogisiticRegression			0,028

Fidélité				
Évaluation		Ct-GAN	PAR	False synthétique
TVC	All	0.11	0.25	0.69
	Object	0.62	0.46	0.98
	Actor	0.0	0.0	0.91
	Verbe	0.79	0.48	0.99
MSE	All	290	290	4
	Object	21785143	20018232	1115
	Actor	8815	8815	57
	Verbe	2197576717	2025350140	7928
KL	All	0.0000476	0.000033	0.0002
	Object	0.0069	0.0056	0.0001
	Actor	0.0005	0.0015	0.00004
	Verbe	0.31	0.25	0.000026
Entropie croisée	All	30.5	18.2	12.69
	Object	2.46	3.05	2.04
	Actor	34.4	34.4	6,79
	Verbe	0,42	0,88	0,27

Table 15 – Résultats

4 Conclusion

Malgré les difficultés rencontrées pour générer des données synthétiques fidèles, utiles et anonymes, nous sommes parvenus à des résultats convenables qui nous ont permis de tester efficacement nos méthodes d'évaluation. La pluralité des méthodes de génération explorées nous a finalement permis de tester de manière plus robuste nos méthodes d'évaluation et de nous faire une idée plus claire de leurs avantages et inconvénients respectifs. Nous nous sommes par exemple rendu compte que certaines méthodes d'évaluation censées refléter la qualité de la dimension temporelle ne suffisaient pas à discriminer les méthodes de génération possédant une dimension temporelle de celles n'en possédant pas. Cela peut signifier que ces méthodes d'évaluation ne sont pas très puissantes. Une autre explication pourrait être que le CTGAN, étant très performant, comprend implicitement la dimension temporelle et obtient donc des résultats similaires à ceux des autres méthodes dans ces tests.

Durant la réalisation de notre projet, nous n'avons pas utilisé certaines de nos méthodes d'évaluation car elles ne s'appliquaient pas au type de données très spécifique que nous avions. Cependant, elles pourront s'avérer utiles dans le futur pour d'autres jeux de données de types différents. En plus de cela, nous avons approfondi a posteriori d'autres méthodes que nous avons ajoutées dans la bibliographie. Nous n'avons pas pu couvrir toutes les méthodes existantes et nous nous sommes donc concentrés sur les plus pertinentes et/ou adaptées à nos données.

Au final, nos méthodes d'évaluation permettent de saisir efficacement les forces et faiblesses des générations de données synthétiques et pourraient être réemployées pour d'autres ensembles de données ultérieurement. Elles permettent d'évaluer correctement les trois métriques que sont la confidentialité, l'utilité et la fidélité des données. Cependant, en vue d'une utilisation plus large, il serait intéressant de simplifier la lecture des résultats à l'aide d'une interface qui permettrait de donner plus ou moins de poids à chaque critère. Il serait aussi bien de normaliser les scores avec des pourcentages par exemple pour permettre à l'utilisateur de lire plus aisément les résultats des évaluations.

References

Goodfellow, Ian et al. (2020). "Generative adversarial networks". In: *Communications of the ACM* 63.11, pp. 139–144.

Jamin, Antoine and Anne Humeau-Heurtier (2020). "(Multiscale) Cross-Entropy Methods: A Review". In: *Entropy* 22.1. ISSN: 1099-4300. DOI: 10.3390/e22010045. URL: https://www.mdpi.com/1099-4300/22/1/45.

Kullback, Solomon and Richard A Leibler (1951). "On the divergence of probability measures". In: *Annals of Mathematical Statistics* 22.1, pp. 79–86.

Rosenbaum, Paul R and Donald B Rubin (1983). "Propensity score methods for bias reduction in the comparison of a treatment to a non-randomized control group". In: *Journal of the American Statistical Association* 79.387, pp. 516–524.

Xu, Lei et al. (2019). "Modeling tabular data using conditional gan". In: *Advances in neural information processing systems* 32.