British Journal of Multidisciplinary and Advanced Studies 6(1),1-10, 2024

Engineering and Technology

Print ISSN: 2517-276X

Online ISSN: 2517-2778

https://bjmas.org/index.php/bjmas/index

Published by the European Centre for Research Training and Development UK

NoSQL et Machine Learning : Cas d'usage et architectures typiques

Pierrot Mukendi Ngalamulume and Jean Nyembue Ilunga
(Tous à l'ISES KANANGA)

doi: https://doi.org/10.37745/bjmas.2022.04219 Published January 19, 2025

Citation : Ngalamulume P.M., and Ilunga J.N. (2025) NoSQL et Machine Learning : Cas d'usage et architectures typiques, *British Journal of Multidisciplinary and Advanced Studies,* 6(1),1-10

Résumé : *L'article explore l'interconnexion croissante entre les bases de données NoSQL et le Machine Learning (ML), soulignant leur rôle essentiel dans l'analyse et le traitement des données massives et hétérogènes générées par les entreprises modernes. Il détaille les cas d'usage, les architectures courantes et les avantages de cette association. Les bases NoSQL et le Machine Learning se complètent parfaitement pour relever les défis du Big Data. Leur combinaison permet de créer des architectures scalables, flexibles, et performantes, adaptées à des cas d'usage variés (recommandations, prédictions, détection de fraudes). Cette synergie ouvre des perspectives prometteuses pour l'intelligence artificielle et l'analyse de données en temps réel.*

Mots lés : NoSQL, Machine Learning, IoT

INTRODUCTION

L'évolution des besoins d'analyse des données est étroitement liée à l'explosion du big data, qui se réfère à la quantité, la variété, la vitesse et la véracité des données disponibles aujourd'hui. Les entreprises collectent désormais des pétaoctets de données provenant de diverses sources comme les médias sociaux, les objets connectés (IoT), les transactions commerciales, etc. Cela nécessite des outils de stockage et de traitement plus puissants, comme les clusters de calculs et les bases de données distribuées (Hadoop, Spark). Ces données ne sont plus uniquement structurées (tables, bases de données relationnelles), mais également non structurées (textes, images, vidéos), ce qui rend l'analyse plus complexe.

Pour gérer cette croissance des données, les technologies d'analyse ont évolué. Les solutions classiques comme les bases de données relationnelles sont complétées par des systèmes distribués (NoSQL, Cassandra, MongoDB). L'utilisation de technologies d'analyse en temps réel est de plus en plus courante pour répondre à la demande d'informations immédiates.

Aujourd'hui, l'analyse descriptive (expliquer ce qui s'est passé) ne suffit plus dans les entreprises. Elles exigent des analyses plus avancées telles que l'Analyse prédictive (Anticiper les tendances futures à l'aide de modèles statistiques et d'algorithmes de machine learning) et

l'Analyse prescriptive (Proposer des recommandations basées sur des prédictions, souvent automatisées par des systèmes d'intelligence artificielle).

Le lien entre NoSQL et le Machine Learning (ML) devient de plus en plus étroit à mesure que les entreprises cherchent à traiter des volumes massifs de données non structurées ou semi-structurées. Les bases NoSQL, par leur flexibilité et leur scalabilité, jouent un rôle crucial dans la collecte, le stockage et la gestion des données pour les applications de Machine Learning. Ainsi, une question reste de savoir : « Quels sont les cas d'usage ainsi que les architectures typiques associant le NoSQL au Machine Learning ? C'est autour de cette question que porte notre étude et nous allons dans la suite énumérer les cas spécifiques d'usage de NoSQL dans le Machine Learning puis les différentes architectures typiques y afférentes, ensuite nous allons donner une conclusion sur la thématique explorée.

Cas d'usage de NoSQL dans le Machine Learning

a. Personnalisation et recommandations

Les systèmes de recommandations (comme ceux utilisés par Netflix, Amazon ou Spotify) reposent sur des algorithmes de Machine Learning qui analysent de grandes quantités de données sur les utilisateurs et les contenus. Les bases NoSQL, telles que MongoDB ou Cassandra, permettent de stocker les données des utilisateurs et les interactions en temps réel. Les modèles de recommandation peuvent être entraînés avec ces données pour offrir des suggestions personnalisées[1].

- **Exemple** : Une base NoSQL comme MongoDB peut stocker des profils d'utilisateurs avec des informations non structurées (intérêts, historique de navigation), tandis qu'un modèle de Machine Learning, comme un filtre collaboratif, utilise ces données pour prédire les produits ou contenus susceptibles d'intéresser chaque utilisateur.[2][3]
- Exemple d'implémentation en Python d'un modèle de recommandations[4]

[1] **Grolinger, K., Hayes, M., & Higashino, W. (2013).** "Data Management in Cloud Environments: NoSQL and NewSQL Data Stores." *Journal of Cloud Computing: Advances, Systems and Applications.*
[2] **Chodorow, K. (2013).** *MongoDB: The Definitive Guide*. O'Reilly Media.
[3] **Koren, Y., Bell, R., & Volinsky, C. (2009).** "Matrix Factorization Techniques for Recommender Systems." *Computer*, 42(8), 30–37.
[4] **Aggarwal, C. C. (2016).** *Recommender Systems: The Textbook*. Springer

British Journal of Multidisciplinary and Advanced Studies 6(1),1-10, 2024
Engineering and Technology
Print ISSN: 2517-276X
Online ISSN: 2517-2778
https://bjmas.org/index.php/bjmas/index
Published by the European Centre for Research Training and Development UK

```python
import numpy as np

# Initialisation
num_users, num_items, k = 100, 50, 10
U = np.random.rand(num_users, k)
V = np.random.rand(num_items, k)
learning_rate, reg_lambda = 0.01, 0.1
R = np.random.randint(0, 5, (num_users, num_items))

# Entraînement
for epoch in range(100):
    for u in range(num_users):
        for i in range(num_items):
            if R[u, i] > 0:  # Seulement les notes observées
                error = R[u, i] - np.dot(U[u], V[i].T)
                U[u] += learning_rate * (error * V[i] - reg_lambda * U[u])
                V[i] += learning_rate * (error * U[u] - reg_lambda * V[i])

# Prédictions
R_pred = np.dot(U, V.T)
```

b. Analyse de sentiment et traitement du langage naturel (NLP)

Le traitement du langage naturel et l'analyse de sentiment nécessitent de grandes quantités de données textuelles provenant de réseaux sociaux, de critiques en ligne, ou d'autres sources. Les bases NoSQL sont idéales pour stocker et gérer ces flux de données.[5]

- **Exemple** : Un outil d'analyse de sentiment peut exploiter Couchbase pour ingérer des flux de données Twitter en temps réel, et utiliser des modèles de Machine Learning pour classer ces données comme positives, négatives ou neutres.[6]
- **Pipeline d'analyse de sentiment avec Python**

1. Extraction de données depuis Couchbase : utiliser le SDK Couchbase pour Python

[5] **Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013).** "New Avenues in Opinion Mining and Sentiment Analysis." *IEEE Intelligent Systems*, 28(2), 15–21.
[6] **Couchbase, Inc. (2020).** "Real-Time Analytics with Couchbase."

Print ISSN: 2517-276X
Online ISSN: 2517-2778
https://bjmas.org/index.php/bjmas/index
Published by the European Centre for Research Training and Development UK

```python
from couchbase.cluster import Cluster, ClusterOptions
from couchbase_core.cluster import PasswordAuthenticator

# Connexion au cluster Couchbase
auth = PasswordAuthenticator('username', 'password')
cluster = Cluster('couchbase://localhost', ClusterOptions(auth))

# Accéder au bucket
bucket = cluster.bucket('user_reviews')
collection = bucket.default_collection()

# Requête pour extraire les données
from couchbase.n1ql import N1QLQuery
query = N1QLQuery('SELECT id, comment FROM user_reviews WHERE comment IS NOT MISSING')
result = cluster.query(query)

# Extraire les commentaires
comments = [row['comment'] for row in result]
```

Prétraitement de données : nettoyage de textes

```python
import re
import string

def preprocess_text(text):
    # Convertir en minuscule
    text = text.lower()
    # Supprimer la ponctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Supprimer les chiffres
    text = re.sub(r'\d+', '', text)
    # Supprimer les espaces supplémentaires
    text = re.sub(r'\s+', ' ', text).strip()
    return text

comments_cleaned = [preprocess_text(comment) for comment in comments]
```

Analyse de sentiment : Utiliser un modèle comme **VADER** ou un modèle BERT préentraîné

```
from transformers import pipeline

# Charger un pipeline de classification pour les sentiments
sentiment_analyzer = pipeline('sentiment-analysis')

# Appliquer le modèle aux commentaires
sentiments = [sentiment_analyzer(comment) for comment in comments_cleaned]
```

Mise à jour des résultats dans Couchbase : jouter les scores de sentiment aux documents originaux dans Couchbase

```
for i, sentiment in enumerate(sentiments):
    score = sentiment[0]['score']
    label = sentiment[0]['label']
    doc_id = result[i]['id']

    # Mettre à jour le document
    collection.mutate_in(doc_id, [
        # Ajouter le score et le label au document JSON
        ('insert', 'sentiment_score', score),
        ('insert', 'sentiment_label', label)
    ])
```

c. Détection de fraudes

Dans le secteur financier ou du commerce électronique, la détection de fraudes repose sur l'analyse de grandes quantités de transactions en temps réel pour identifier les comportements suspects. Les bases NoSQL sont souvent utilisées pour gérer les données transactionnelles à grande échelle.[7]

- **Exemple** : Une solution utilisant Cassandra peut stocker les transactions à grande échelle, tandis que des modèles de Machine Learning sont appliqués pour détecter les anomalies ou fraudes potentielles en temps réel.[8]

[7] **Zheng, L., Chen, C., & Huang, C. (2018).** "Financial Fraud Detection Using Machine Learning Models." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[8] **Lakshman, A., & Malik, P. (2010).** "Cassandra: A Decentralized Structured Storage System." *ACM SIGOPS Operating Systems Review*, 44(2), 35–40.

Print ISSN: 2517-276X
Online ISSN: 2517-2778
https://bjmas.org/index.php/bjmas/index
Published by the European Centre for Research Training and Development UK

- **Modèle déployé en streaming** : Charger le modèle dans une API REST ou un microservice (via Flask, FastAPI, ou TensorFlow Serving). Avec Kafka :

✓ Les transactions sont ingérées via un topic Kafka.
✓ Les données sont prétraitées en temps réel.
✓ Les scores de fraude sont calculés via le modèle déployé.

```
from kafka import KafkaConsumer, KafkaProducer
import joblib
import json
import numpy as np

# Charger le modèle ML
model = joblib.load('fraud_detection_model.pkl')

# Consommateur Kafka
consumer = KafkaConsumer(
    'transactions',
    bootstrap_servers='localhost:9092',
    value_deserializer=lambda x: json.loads(x.decode('utf-8'))
)

# Producteur Kafka
producer = KafkaProducer(
    bootstrap_servers='localhost:9092',
    value_serializer=lambda x: json.dumps(x).encode('utf-8')
)
```

British Journal of Multidisciplinary and Advanced Studies 6(1),1-10, 2024
Engineering and Technology
Print ISSN: 2517-276X
Online ISSN: 2517-2778
https://bjmas.org/index.php/bjmas/index
Published by the European Centre for Research Training and Development UK

```
for message in consumer:
    transaction = message.value
    features = np.array([transaction['amount'], transaction['time_gap'], transaction['geo_
    # Prédiction
    fraud_probability = model.predict_proba(features)[0][1]
    # Ajouter un score de fraude
    transaction['fraud_probability'] = fraud_probability
    # Envoyer à un autre topic Kafka
    producer.send('fraud_scores', value=transaction)
```

Prévisions et maintenance prédictive

Dans l'industrie (automobile, manufacturing, énergie), les données des capteurs IoT sont souvent stockées dans des bases NoSQL en raison de leur nature volumineuse et hétérogène. Ces données sont ensuite utilisées pour former des modèles de Machine Learning qui prévoient les pannes ou optimisent les processus.[9]

- **Exemple** : Un modèle de Machine Learning peut utiliser des données stockées dans Cassandra provenant de milliers de capteurs pour anticiper une défaillance d'équipement dans une usine.[10][11]
- **Algorithme simple avec random Forest**

- **Données d'entrée** :

- X : caractéristiques extraites des séries temporelles (ex. : température moyenne, vibration maximale, etc.).
- Y: état de l'équipement (0 = bon, 1 = en panne).

- **Entraînement** :

- Diviser les données en ensemble d'entraînement et de test.

[9] **Chiang, L. H., Russell, E. L., & Braatz, R. D. (2000).** *Fault Detection and Diagnosis in Industrial Systems*. Springer.
[10] **Susto, G. A., Schirru, A., Pampuri, S., & McLoone, S. (2015).** "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach." *IEEE Transactions on Industrial Informatics*, 11(3), 812–820.
[11] **Lakshman, A., & Malik, P. (2010).** Op cit.

British Journal of Multidisciplinary and Advanced Studies 6(1),1-10, 2024
Engineering and Technology
Print ISSN: 2517-276X
Online ISSN: 2517-2778
https://bjmas.org/index.php/bjmas/index
Published by the European Centre for Research Training and Development UK

- Entraîner un modèle de Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Charger les données
X = data[['temperature_mean', 'vibration_std', 'pressure_max']]
y = data['is_faulty']

# Division entraînement/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

# Entraîner le modèle
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Évaluation
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Architectures typiques combinant NoSQL et Machine Learning[12][13]

a. Pipeline de données NoSQL pour le Machine Learning

Une architecture courante consiste à utiliser NoSQL pour gérer le pipeline de données d'entrée, où la base NoSQL joue le rôle de stockage avant l'entraînement et l'inférence des modèles Machine Learning.

- **Étape 1** : **Ingestion de données** : Les données brutes (log, capteurs, clics) sont ingérées dans la base NoSQL (par exemple, via MongoDB ou DynamoDB).
- **Étape 2** : **Prétraitement des données** : Les données sont nettoyées et prétraitées, souvent à l'aide de frameworks tels qu'Apache Spark, qui peuvent se connecter directement aux bases NoSQL.

[12] **Heudecker, N., & Luhn, H. P. (2019).** *Machine Learning Architectures in Big Data Systems*. Gartner
[13] **Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. M. (2014).** "Data Management in Cloud Environments: NoSQL and NewSQL Data Stores." *Journal of Cloud Computing: Advances, Systems and Applications*, 3(1), 1–17

- **Étape 3** : **Entraînement des modèles** : Les données prétraitées sont extraites de la base NoSQL pour entraîner des modèles via des frameworks Machine Learning tels que TensorFlow, PyTorch, ou scikit-learn.
- **Étape 4** : **Stockage des résultats** : Les modèles entraînés peuvent être stockés dans la base NoSQL ou dans un autre environnement pour servir les prédictions.

b. Architecture avec NoSQL et Spark pour Machine Learning distribué

Pour les applications nécessitant des capacités de traitement distribuées, Apache Spark est fréquemment utilisé en combinaison avec des bases NoSQL comme Cassandra ou HBase pour les tâches d'entraînement et d'inférence Machine Learning.

- **Étape 1** : Les données sont stockées dans une base NoSQL distribuée comme Cassandra.
- **Étape 2** : Spark se connecte directement à Cassandra pour charger les données et exécuter les algorithmes Machine Learning sur des clusters distribués.
- **Étape 3** : Les modèles de Machine Learning peuvent être sauvegardés ou déployés dans des bases NoSQL pour les requêtes en temps réel.

c. Architecture Lambda avec NoSQL pour Machine Learning en temps réel

L'architecture Lambda combine à la fois du traitement par lots et du traitement en temps réel. Une base NoSQL est utilisée pour la couche de persistance des données en temps réel.

- **Étape 1** : **Ingestion de données en temps réel** : Les événements ou les données des utilisateurs sont stockés dans une base NoSQL (par ex. DynamoDB ou MongoDB).
- **Étape 2** : **Traitement en temps réel** : Les données sont traitées en temps réel par un moteur de stream processing comme Apache Kafka ou Flink, qui exécute également des modèles de Machine Learning pour des prédictions immédiates.
- **Étape 3** : **Traitement batch** : En parallèle, des analyses plus lourdes peuvent être effectuées en lots avec des données agrégées via des frameworks comme Hadoop ou Spark.

d. NoSQL et le Edge Computing pour Machine Learning

Dans des applications où les données sont générées en périphérie (par exemple, dans des appareils IoT), les bases NoSQL peuvent être utilisées en tandem avec des modèles de Machine Learning déployés au plus proche des sources de données (edge computing).

- **Exemple** : Un système de maintenance prédictive où les données des capteurs sont stockées dans des bases de données NoSQL déployées à la périphérie (edge). Les modèles de Machine Learning peuvent être exécutés localement pour réduire la latence et la bande passante nécessaire pour l'envoi des données au cloud.

CONCLUSION

Les bases NoSQL, avec leur capacité à stocker des données non structurées et leur scalabilité, se combinent naturellement avec le Machine Learning, en particulier pour les applications qui nécessitent des analyses de données à grande échelle en temps réel. Que ce soit pour la personnalisation, l'analyse de sentiment ou la maintenance prédictive, les architectures qui intègrent NoSQL et Machine Learning sont variées et adaptées à différents cas d'usage, avec des possibilités de traitement distribué et de traitement en temps réel.

Bibliographie

1. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer
2. Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). "New Avenues in Opinion
3. Chiang, L. H., Russell, E. L., & Braatz, R. D. (2000). *Fault Detection and Diagnosis in Industrial Systems*. Springer.
4. Chodorow, K. (2013). *MongoDB: The Definitive Guide*. O'Reilly Media.
5. Couchbase, Inc. (2020). "Real-Time Analytics with Couchbase."
6. Grolinger, K., Hayes, M., & Higashino, W. (2013). "Data Management in Cloud Environments: NoSQL and NewSQL Data Stores." *Journal of Cloud Computing: Advances, Systems and Applications.*
7. Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. M. (2014). "Data Management in Cloud Environments: NoSQL and NewSQL Data Stores." *Journal of Cloud Computing: Advances, Systems and Applications*, 3(1), 1–17
8. Heudecker, N., & Luhn, H. P. (2019). *Machine Learning Architectures in Big Data Systems*. Gartner
9. Koren, Y., Bell, R., & Volinsky, C. (2009). "Matrix Factorization Techniques for Recommender Systems." *Computer*, 42(8), 30–37.
10. Lakshman, A., & Malik, P. (2010). "Cassandra: A Decentralized Structured Storage System." *ACM SIGOPS Operating Systems Review*, 44(2), 35–40.
11. Mining and Sentiment Analysis." *IEEE Intelligent Systems*, 28(2), 15–21.
12. Susto, G. A., Schirru, A., Pampuri, S., & McLoone, S. (2015). "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach." *IEEE Transactions on Industrial Informatics*, 11(3), 812–820.
13. Zheng, L., Chen, C., & Huang, C. (2018). "Financial Fraud Detection Using Machine Learning Models." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).*