Université Tunis-El Manar
Faculté des Sciences de Tunis
Département Informatique

Projet IGL3 / IDS3

Adem Medyouni
IDS3

Analyse Numérique Matricielle

Sujet : Systèmes de recherche documentaire

Enseignant responsable :
M. Atef Ben Essid

Date de début : 23 Avril 2025
Date de remise : 07 Mai 2025

Table des matières

Introduction

Dans ce projet, nous explorons l'application des méthodes d'analyse matricielle, et plus particulièrement la Décomposition en Valeurs Singulières (SVD), à la gestion et à l'interrogation de bases de données documentaires. Nous abordons successivement la modélisation vectorielle des documents, la recherche par requête, et les améliorations apportées par les techniques de réduction de dimension.

Chapitre 1

Modélisation Mathématique

1.1 Matrice termes-documents

La base documentaire est représentée par une matrice binaire $D \in \mathbb{R}^{N_t \times N_d}$, où chaque colonne correspond à un document et chaque ligne à un terme.

$$d_{ij} = \begin{cases} 1 & \text{si le terme } T_i \text{ apparaît dans le document } D_j \\ 0 & \text{sinon} \end{cases}$$

```
---- EXEMPLE 1 ----
Matrice D (termes-documents) :
[[1 0 0 0]
 [0 1 0 0]
 [0 0 0 1]
 [1 0 0 1]
 [0 0 1 0]
 [0 0 1 0]
 [0 1 0 0]]

Vecteur de requête q1 : [1 0 0 1 0 0 0]
```

1.2 Score de pertinence

Pour une requête $q \in \mathbb{R}^{N_t}$, on calcule le score de pertinence du document D_j :

$$s_j(q) = \frac{\langle q, d_j \rangle}{\|q\|_2 \cdot \|d_j\|_2}$$

```python
def construire_vecteur_requete(termes, requete):
    """
    Construit un vecteur de requête q où q[i] = 1 si le terme i est dans la requête
    """
    q = np.zeros(len(termes), dtype=int)
    for i, terme in enumerate(termes):
        if terme in requete:
            q[i] = 1
    return q
```

```python
def calculer_scores_standard(D, q):
    """
    Calcule les scores de pertinence sans SVD
    s_j(q) = <q, d_j> / (||q||_2 * ||d_j||_2)
    """
    norme_q = np.linalg.norm(q)
    if norme_q == 0:
        return np.zeros(D.shape[1])

    scores = []
    for j in range(D.shape[1]):
        d_j = D[:, j]
        norme_d_j = np.linalg.norm(d_j)
        if norme_d_j == 0:
            scores.append(0)
        else:
            produit_scalaire = np.dot(q, d_j)
            score = produit_scalaire / (norme_q * norme_d_j)
            scores.append(score)
    return np.array(scores)
```

```
Vecteur de requête q1 : [1 0 0 1 0 0 0]

--- Méthode standard (sans SVD) ---

Classement des documents (score > 0) :
D1 : 1.00
D4 : 0.50
```

Chapitre 2

Réduction de Dimension

2.1 Décomposition en Valeurs Singulières

Donnée une matrice D, on peut écrire :

$$D = U\Sigma V^T$$

où :
— U et V sont orthogonales
— Σ est diagonale avec les valeurs singulières

```python
def decomposition_SVD(D):
    """
    Calcule directement la décomposition SVD d'une matrice D
    Retourne U, S, V^T tels que D = U * S * V^T
    """
    return np.linalg.svd(D, full_matrices=False)
```

2.2 Approximation de rang k

$$D_k = U_k\Sigma_k V_k^T$$

Cette approximation réduit le bruit, la redondance et améliore la qualité des résultats de recherche.

```python
def approximation_rang_k(U, S, Vt, k):
    """
    Calcule l'approximation de rang k d'une matrice à partir de sa SVD
    """
    k = min(k, U.shape[1], S.shape[0], Vt.shape[0])
    U_k = U[:, :k]
    S_k = S[:k, :k] if S.ndim == 2 else np.diag(S[:k])
    Vt_k = Vt[:k, :]
    return U_k, S_k, Vt_k
```

Chapitre 3

Implémentation

L'implémentation a été réalisée en Python, avec trois variantes :
— Modèle vectoriel simple (sans SVD)
— Modèle SVD complet
— Modèle SVD avec bidiagonalisation + QR

3.1 Fonctions principales

— `calcul_score()`
— `svd_score()`
— `bidiag_qr_score()`

```python
def calculer_scores_standard(D, q):
    """
    Calcule les scores de pertinence sans SVD
    s_j(q) = <q, d_j> / (||q||_2 * ||d_j||_2)
    """
    norme_q = np.linalg.norm(q)
    if norme_q == 0:
        return np.zeros(D.shape[1])

    scores = []
    for j in range(D.shape[1]):
        d_j = D[:, j]
        norme_d_j = np.linalg.norm(d_j)
        if norme_d_j == 0:
            scores.append(0)
        else:
            produit_scalaire = np.dot(q, d_j)
            score = produit_scalaire / (norme_q * norme_d_j)
            scores.append(score)
    return np.array(scores)
```

```python
def decomposition_SVD(D):
    """
    Calcule directement la décomposition SVD d'une matrice D
    Retourne U, S, V^T tels que D = U * S * V^T
    """
    return np.linalg.svd(D, full_matrices=False)
```

```python
def approximation_rang_k(U, S, Vt, k):
    """
    Calcule l'approximation de rang k d'une matrice à partir de sa SVD
    """
    k = min(k, U.shape[1], S.shape[0], Vt.shape[0])
    U_k = U[:, :k]
    S_k = S[:k, :k] if S.ndim == 2 else np.diag(S[:k])
    Vt_k = Vt[:k, :]
    return U_k, S_k, Vt_k
```

```python
def calculer_scores_svd(U_k, S_k, Vt_k, q):
    """
    Calcule les scores de pertinence avec SVD
    s_j(q) = <U_k^T q, S_k V_k^T e_j> / (||U_k^T q||_2 * ||S_k V_k^T e_j||_2)
    """
    q_proj = np.dot(U_k.T, q)
    norme_q_proj = np.linalg.norm(q_proj)
    if norme_q_proj == 0:
        return np.zeros(Vt_k.shape[1])

    scores = []
    for j in range(Vt_k.shape[1]):
        e_j = np.zeros(Vt_k.shape[1])
        e_j[j] = 1
        d_j_proj = np.dot(S_k, np.dot(Vt_k, e_j))
        norme_d_j_proj = np.linalg.norm(d_j_proj)
        if norme_d_j_proj == 0:
            scores.append(0)
        else:
            produit_scalaire = np.dot(q_proj, d_j_proj)
            score = produit_scalaire / (norme_q_proj * norme_d_j_proj)
            scores.append(score)
    return np.array(scores)
```

```python
def methode_qr_bidiagonale(B, max_iterations=100, tol=1e-8):
    """
    Méthode QR pour extraire les valeurs singulières d'une matrice bidiagonale
    """
    B_k = B.copy()
    n = B.shape[0]

    for k in range(max_iterations):
        if np.max(np.abs(np.tril(B_k, -1))) < tol:
            break
        Q_k, R_k = qr_factorisation(B_k.T)
        Q_tilde_k, B_k_plus_1 = qr_factorisation(R_k.T)
        B_k = B_k_plus_1.T

    valeurs_singulieres = np.abs(np.diag(B_k))
    indices = np.argsort(valeurs_singulieres)[::-1]
    valeurs_singulieres = valeurs_singulieres[indices]
    return valeurs_singulieres
```

```python
def decomposition_SVD_bidiag_QR(D):
    """
    Calcule la décomposition SVD en utilisant la bidiagonalisation suivie de QR
    """
    U, B, V = bidiagonalisation(D)
    valeurs_singulieres = methode_qr_bidiagonale(B)

    r = min(U.shape[1], V.shape[1], len(valeurs_singulieres))
    S = np.zeros((r, r))
    np.fill_diagonal(S, valeurs_singulieres[:r])

    return U, S, V.T
```

```python
def calculer_scores_bidiag_qr(U_k, S_k, Vt_k, q):
    """
    Calcule les scores de pertinence pour la méthode bidiag+QR
    """
    return calculer_scores_svd(U_k, S_k, Vt_k, q)
```

```
--- Méthode standard (sans SVD) ---

Classement des documents (score > 0) :
D1 : 1.00
D4 : 0.50
--- Méthode SVD directe ---

k = 1, Erreur de reconstruction: 1.414214

Classement des documents (score > 0) :
D1 : 1.00
D4 : 1.00

k = 2, Erreur de reconstruction: 1.414214

Classement des documents (score > 0) :
D1 : 1.00
D4 : 1.00

k = 3, Erreur de reconstruction: 1.000000

Classement des documents (score > 0) :
D1 : 1.00
D4 : 1.00

k = 4, Erreur de reconstruction: 0.000000

Classement des documents (score > 0) :
D1 : 1.00
D4 : 0.50
```

```
--- Méthode Bidiagonalisation + QR ---

k = 1, Erreur de reconstruction: 1.660011

Classement des documents (score > 0) :
D1 : 1.00
D2 : 1.00
D3 : 1.00
D4 : 1.00

k = 2, Erreur de reconstruction: 1.414214

Classement des documents (score > 0) :
D1 : 0.90
D4 : 0.49
D2 : 0.36
D3 : 0.36

k = 3, Erreur de reconstruction: 1.414214

Classement des documents (score > 0) :
D1 : 0.97
D4 : 0.43
D3 : 0.21
D2 : 0.21

k = 4, Erreur de reconstruction: 1.414214

Classement des documents (score > 0) :
D1 : 0.99
D3 : 0.20
D2 : 0.20
D4 : 0.15
```

Chapitre 4

Résultats Expérimentaux

4.1 Exemples étudiés

— Exemple 1 : Titres de livres de mathématiques

```python
# Partie 3: Tests et analyses
def exemple1():
    #Test pour l'exemple 1 avec la requête q1 = {matrices, algèbre}

    print("\n==== EXEMPLE 1 ====")

    termes = ["algebre", "analyse", "determinants", "matrices", "probabilites", "statistiques", "suites"]
    documents = [
        ["algebre", "lineaire", "matrices"],
        ["analyse", "reelle", "suites"],
        ["probabilites", "statistiques"],
        ["matrices", "determinants"]
    ]

    D = construire_matrice_td(termes, documents)
    print("Matrice D (termes-documents) :")
    print(D)

    requete = ["matrices", "algebre"]
    q = construire_vecteur_requete(termes, requete)
    print("\nVecteur de requête q1 :", q)

    # Méthode standard
    print("\n--- Méthode standard (sans SVD) ---")
    scores_standard = calculer_scores_standard(D, q)
    afficher_classement(scores_standard)

    # Méthode SVD directe
    print("\n--- Méthode SVD directe ---")
    for k in range(1, min(D.shape[0], D.shape[1]) + 1):
        U, S, Vt = decomposition_SVD(D)
        U_k, S_k, Vt_k = approximation_rang_k(U, S, Vt, k)
        scores_svd = calculer_scores_svd(U_k, S_k, Vt_k, q)
        D_k = np.dot(U_k, np.dot(S_k, Vt_k))
        err = erreur_reconstruction(D, D_k)
        print(f"\nk = {k}, Erreur de reconstruction: {err:.6f}")
        afficher_classement(scores_svd)

    # Méthode Bidiagonalisation + QR
    print("\n--- Méthode Bidiagonalisation + QR ---")
    for k in range(1, min(D.shape[0], D.shape[1]) + 1):
        U, S, Vt = decomposition_SVD_bidiag_QR(D)
        U_k, S_k, Vt_k = approximation_rang_k(U, S, Vt, k)
        scores_bidiag_qr = calculer_scores_bidiag_qr(U_k, S_k, Vt_k, q)
        D_k = np.dot(U_k, np.dot(S_k, Vt_k))
        err = erreur_reconstruction(D, D_k)
        print(f"\nk = {k}, Erreur de reconstruction: {err:.6f}")
        afficher_classement(scores_bidiag_qr)
```

— Exemple 3 : Documents en économie et géologie

```python
def exemple3():
    """
    Test pour l'exemple 3 avec les requêtes:
    - q2 = (dépression, croissance)
    - q3 = (bassin, fiscalité)
    """
    print("\n===== EXEMPLE 3 =====")

    termes = ["bassin", "chomage", "commerce", "couches", "croissance", "depression",
              "emploi", "erosion", "exportation", "faille", "fiscalite", "forage",
              "geologie", "gisement", "impots", "inflation", "investissement",
              "monnaie", "pib", "plaque", "revenu", "roche", "salaires", "seisme",
              "stratigraphie", "tectonique", "tremblement", "volcan"]

    documents = [
        ["croissance", "pib", "investissement"],
        ["inflation", "monnaie", "depression"],
        ["commerce", "exportation", "croissance"],
        ["emploi", "chomage", "salaires"],
        ["impots", "fiscalite", "revenu"],
        ["geologie", "faille", "tremblement"],
        ["volcan", "seisme", "plaque", "tectonique"],
        ["depression", "bassin", "erosion"],
        ["stratigraphie", "couches", "roche"],
        ["gisement", "forage", "bassin"]
    ]

    D = construire_matrice_td(termes, documents)
    print("Matrice termes-documents pour l'exemple 3 (dimensions):", D.shape)

    requetes = [
        ["depression", "croissance"],
        ["bassin", "fiscalite"]
    ]

    for i, requete in enumerate(requetes, 2):
        print(f"\n===== Requête q{i} = {requete} =====")
        q = construire_vecteur_requete(termes, requete)

        # Méthode standard
        print("\n--- Méthode standard (sans SVD) ---")
        scores_standard = calculer_scores_standard(D, q)
        afficher_classement(scores_standard, seuil=0.5)

        # Méthode SVD directe
        print("\n--- Méthode SVD directe ---")
        erreurs = []
        for k in range(1, min(D.shape) + 1):
            U, S, Vt = decomposition_SVD(D)
            U_k, S_k, Vt_k = approximation_rang_k(U, S, Vt, k)
            D_k = np.dot(U_k, np.dot(S_k, Vt_k))
            err = erreur_reconstruction(D, D_k)
            erreurs.append(err)
            if k == 3:
                scores_svd = calculer_scores_svd(U_k, S_k, Vt_k, q)
                print(f"\nk = {k}, Erreur de reconstruction: {err:.6f}")
                afficher_classement(scores_svd, seuil=0.5)

        # Méthode Bidiagonalisation + QR
        print("\n--- Méthode Bidiagonalisation + QR ---")
        U, S, Vt = decomposition_SVD_bidiag_QR(D)
        for k in [3]:
            U_k, S_k, Vt_k = approximation_rang_k(U, S, Vt, k)
            scores_bidiag_qr = calculer_scores_bidiag_qr(U_k, S_k, Vt_k, q)
            D_k = np.dot(U_k, np.dot(S_k, Vt_k))
            err = erreur_reconstruction(D, D_k)
            print(f"\nk = {k}, Erreur de reconstruction: {err:.6f}")
            afficher_classement(scores_bidiag_qr, seuil=0.5)

        # Plot reconstruction error
        plt.figure(figsize=(10, 6))
        plt.plot(range(1, len(erreurs) + 1), erreurs, 'b-o')
        plt.title(f"Erreur de reconstruction ||D - D_k||_2 pour la requête q{i}")
        plt.xlabel("Rang k")
        plt.ylabel("Erreur")
        plt.grid(True)
        plt.show()
        plt.close()
```

— test avec Documents.txt :

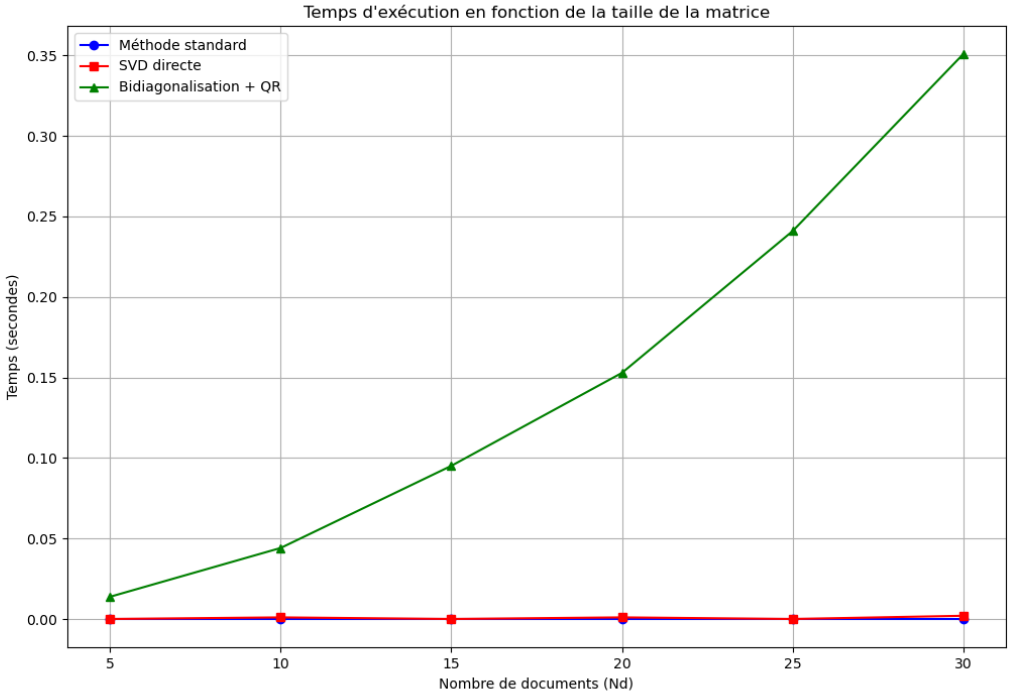
```python
def traiter_fichier_documents(chemin_fichier):
    """
    Traite le fichier documents.txt pour extraire la matrice termes-documents
    """
    try:
        with open(chemin_fichier, 'r', encoding='utf-8') as file:
            contenu = file.read()

        documents_bruts = [line.strip() for line in contenu.split('\n') if line.strip()]
        termes_set = set()
        documents = []

        for doc in documents_bruts:
            mots = [mot.lower().strip('.,;:!?()[]{}\"\'') for mot in doc.split()]
            mots = [mot for mot in mots if mot]
            termes_set.update(mots)
            documents.append(mots)

        termes = sorted(list(termes_set))
        print(f"Nombre de documents: {len(documents)}")
        print(f"Nombre de termes: {len(termes)}")

        D = construire_matrice_td(termes, documents)
        return termes, documents, D

    except FileNotFoundError:
        print(f"Erreur: Le fichier {chemin_fichier} n'a pas été trouvé.")
        return [], [], np.array([])
```

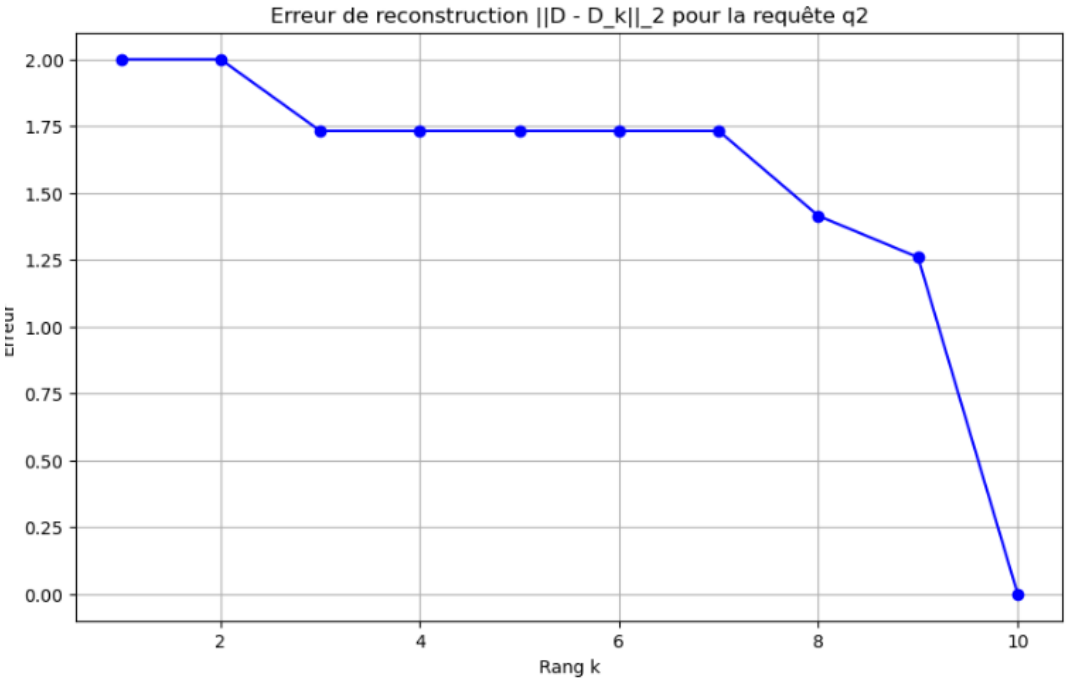
```python
def test_fichier_documents(chemin_fichier):
    """
    Test avec le fichier documents.txt
    """
    print("\n==== TEST AVEC LE FICHIER DOCUMENTS.TXT ====")

    termes, documents, D = traiter_fichier_documents(chemin_fichier)

    if len(termes) == 0:
        print("Utilisation d'un exemple simplifié...")
        termes = ["algebre", "analyse", "matrices", "statistiques"]
        documents = [
            ["algebre", "matrices"],
            ["analyse", "suites"],
            ["statistiques"],
            ["matrices"]
        ]
        D = construire_matrice_td(termes, documents)

    mots_requete = random.sample(termes, min(3, len(termes)))
    print(f"\nRequête aléatoire: {mots_requete}")

    q = construire_vecteur_requete(termes, mots_requete)

    # Méthode standard
    print("\n--- Méthode standard (sans SVD) ---")
    scores_standard = calculer_scores_standard(D, q)
    top_docs = np.argsort(scores_standard)[::-1][:5]
    print("\nTop 5 documents:")
    for i, doc_id in enumerate(top_docs, 1):
        print(f"{i}. Document {doc_id+1}: Score = {scores_standard[doc_id]:.4f}")
        print(f"   Contenu: {' '.join(documents[doc_id][:10])}...")

    # Méthode SVD directe
    print("\n--- Méthode SVD directe (k=5) ---")
    k = min(5, min(D.shape))
    U, S, Vt = decomposition_SVD(D)
    U_k, S_k, Vt_k = approximation_rang_k(U, S, Vt, k)
    scores_svd = calculer_scores_svd(U_k, S_k, Vt_k, q)
    top_docs_svd = np.argsort(scores_svd)[::-1][:5]

    print("\nTop 5 documents (SVD):")
    for i, doc_id in enumerate(top_docs_svd, 1):
        print(f"{i}. Document {doc_id+1}: Score = {scores_svd[doc_id]:.4f}")
        print(f"   Contenu: {' '.join(documents[doc_id][:10])}...")

    print("\n--- Comparaison des méthodes ---")
    print(f"Documents communs entre les deux méthodes: {len(set(top_docs) & set(top_docs_svd))}")
```

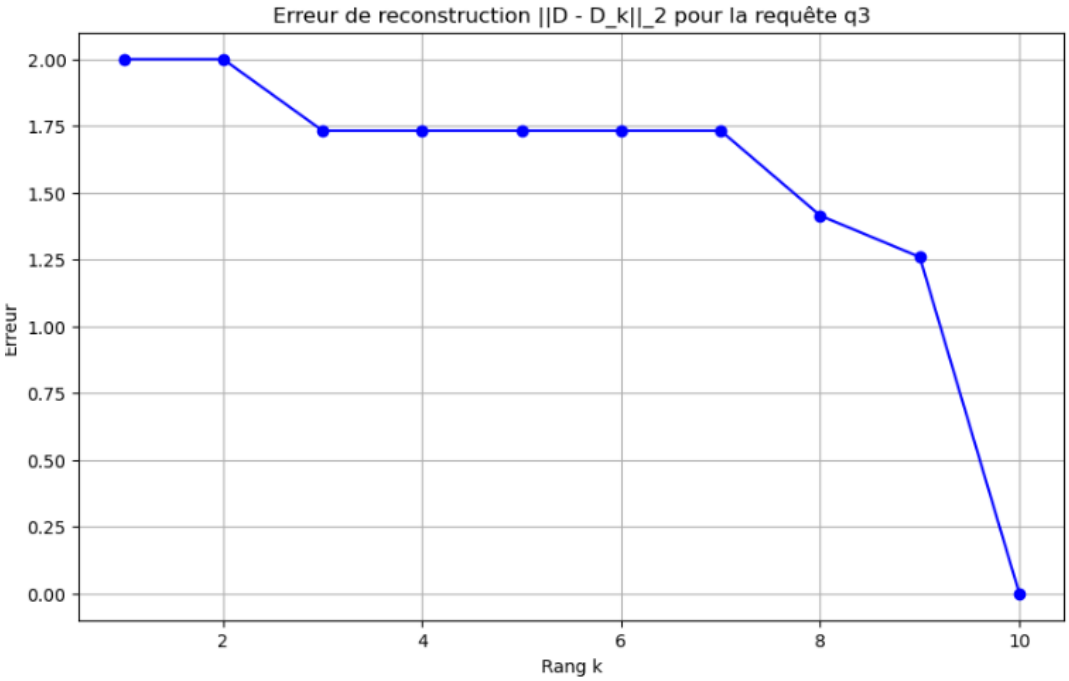
```
==== TEST AVEC LE FICHIER DOCUMENTS.TXT ====
Nombre de documents: 1000
Nombre de termes: 1075

Requête aléatoire: ["750", "70", "919"]

--- Méthode standard (sans SVD) ---

Top 5 documents:
1. Document 750: Score = 0.2357
   Contenu: 750 geo tectonique sédiment érosion bassin...
2. Document 70: Score = 0.2357
   Contenu: 70 image seuillage pixel gradient matrice...
3. Document 919: Score = 0.2357
   Contenu: 919 info programmation compilation java structure...
4. Document 328: Score = 0.0000
   Contenu: 328 info java base compilation structure...
5. Document 340: Score = 0.0000
   Contenu: 340 son onde bruit amplitude harmonique...

--- Méthode SVD directe (k=5) ---

Top 5 documents (SVD):
1. Document 118: Score = 0.7154
   Contenu: 118 geo stratification plaque bassin volcan...
2. Document 574: Score = 0.7154
   Contenu: 574 geo volcan tectonique bassin érosion...
3. Document 637: Score = 0.7154
   Contenu: 637 geo tectonique bassin sédiment volcan...
4. Document 136: Score = 0.7154
   Contenu: 136 geo stratification sédiment séisme volcan...
5. Document 300: Score = 0.7154
   Contenu: 300 geo plaque tectonique séisme sédiment...

--- Comparaison des méthodes ---
Documents communs entre les deux méthodes: 0

==== TESTS TERMINÉS ====
```

4.2 Classement des documents

Des graphiques sont utilisés pour comparer les scores selon les différentes approches et analyser les effets de la polysémie et synonymie.




Erreur de reconstruction ||D - D_k||_2 pour la requête q2
Erreur
Rang k
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2
4
6
8
10


Erreur de reconstruction ||D - D_k||_2 pour la requête q3
Erreur
Rang k
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2
4
6
8
10

Chapitre 5

Conclusion

Ce projet a mis en lumière la puissance de l'analyse matricielle pour l'amélioration des moteurs de recherche documentaire. La réduction de dimension par SVD, bien que coûteuse en calcul, offre une meilleure robustesse aux redondances et ambiguïtés linguistiques.