

## **RAPPORT DE STAGE INGENIEUR**

**SPÉCIALITÉ : Informatique**

**Système de recommandation de ressources  
pour l'apprentissage adaptatif**

**Réalisé par : MARZOUG Ali**

**Encadré par : ZIADI Faten**

**Proposé par : ESPRIT**

**Année universitaire : 2025-2026**

# Sommaire

Remerciements .....	3
Résumé .....	4
1 Introduction .....	5
1.1 Organisme d'Accueil – ESPRIT ENTREPRISE.....	5
1.2 Présentation du Bureau RDI – ESPRIT-Tech .....	6
1.3 Contexte du Stage .....	7
1.4 Objectifs et Contributions .....	8
2 Contexte Général & Problématique .....	9
2.1 Contexte Pédagogique et Données .....	9
2.2 Besoin Métier .....	10
2.3 Problématique.....	11
3 Spécifications.....	12
3.1 Spécifications Fonctionnelles .....	12
3.2 Spécifications Techniques .....	12
3.3 Critères d'Acceptation .....	13
4 CHAPITRE 1 : Etat de l'Art & Choix Méthodologiques .....	13
4.1 Rappels sur les Graphes Orientés .....	14
4.2 Algorithmes de Rang .....	15
4.3 Métriques de Comparaison.....	16
4.4 Choix Retenus .....	16
5 CHAPITRE 2 : Conception .....	16
5.1 Modèle de Données.....	17
5.2 Modèle Graphe (Neo4j Inclus).....	17
5.3 Cas d'Utilisation & Scénarios .....	18
6 CHAPITRE 3 : Implémentation .....	19
6.1 Pipeline.....	19
6.2 Neo4j pour la Persistance et Visualisation .....	19
6.3 Configuration de l'environnement React .....	20
6.4 Conception et Développement des Composants React .....	21
6.5 Intégration et Gestion des Données JSON .....	22
6.6 Stylisation avec Bootstrap CSS.....	22
6.7 Tests de Compatibilité et Optimisation des Performances.....	23
7 CHAPITRE 4 : Expérimentations, Résultats & Analyse .....	23
7.1 Jeu de données .....	23
7.2 Protocoles d'Essai .....	24
7.3 Résultats par Étudiant (Exemples).....	24
7.4 Discussion.....	26
7.5 Limites .....	27
8 Conclusion & Perspectives.....	28



# Remerciements

Je tiens à exprimer ma profonde gratitude à mon encadrant en entreprise, Mme Ziadi Faten, pour son accompagnement attentif, sa disponibilité et ses précieux conseils tout au long de ce stage. Son expertise et son soutien constant m'ont permis de mieux comprendre le fonctionnement de l'entreprise et d'acquérir de nouvelles compétences techniques et professionnelles.

3

Je remercie également l'ensemble de l'équipe d'ESPRIT-Tech pour leur accueil chaleureux et pour l'échange constructif tout au long de mon stage. Leur encadrement m'a offert l'opportunité de découvrir concrètement l'application des technologies telles que Python et NetworkX dans des projets réels, renforçant ainsi ma compréhension des concepts étudiés et mon savoir-faire pratique.

Grâce à cette expérience, j'ai pu développer mes compétences en programmation, en analyse de données et en gestion de projets techniques, tout en découvrant l'importance du travail collaboratif et de l'esprit d'équipe au sein d'un environnement professionnel. Ce stage a été une étape enrichissante de mon parcours académique et constitue un tremplin vers mes futures expériences professionnelles.

Etat de l'Art & Choix Méthodologiques  
CHAPITRE 1 : Etat de l'Art & Choix Méthodologiques





# Résumé

Ce rapport décrit un stage d'été réalisé au sein d'ESPRIT ENTREPRISE / ESPRIT-Tech, axé sur l'étude des approches de recommandation dans le domaine éducatif. Le projet consiste à modéliser les données d'apprentissage via un graphe de prérequis en utilisant Python et NetworkX, à concevoir un moteur de recommandation adaptatif utilisant les algorithmes PageRank et HITS pour prioriser les modules de remédiation, et à persister et visualiser le graphe dans Neo4j pour une exploration interactive (affichage de détails comme les IDs d'étudiants ayant échoué sur un nœud). Les résultats montrent que PageRank surpasse souvent HITS en termes de pertinence globale, évaluée via des métriques comme nDCG et Spearman. Ce prototype, enrichi par Neo4j, aide à personnaliser l'accompagnement pédagogique avec des visualisations interactives.

Le contexte du stage est la recherche appliquée en ingénierie pédagogique, où les enjeux sont le suivi des acquis étudiants et la qualité de l'enseignement. La méthode impliquée la construction d'un graphe orienté à partir de fichiers JSON, l'application d'algorithmes de rang pour classer les prérequis directs des modules échoués, et la comparaison des classements via des métriques de corrélation et de qualité. Les expérimentations sur un jeu de données fictif mais réaliste démontrent l'efficacité de PageRank pour les structures de graphe complexes. Les limites incluent la convergence de HITS et la qualité des données d'entrée. Les perspectives incluent l'intégration d'apprentissage supervisé et de visualisations interactives avancées.





# Introduction

L'introduction de ce rapport de stage vise à présenter l'organisme d'accueil, le contexte du stage, et les objectifs principaux. Ce stage d'été, d'une durée de trois mois, s'est déroulé au sein d'ESPRIT ENTREPRISE, une filiale dédiée à l'innovation pédagogique. Il s'inscrit dans le cadre de ma formation en ingénierie informatique, avec un focus sur l'application de technologies de graphes pour la remédiation éducative. Le projet a combiné modélisation de données, algorithmes de rang, et visualisation interactive, répondant aux besoins en suivi pédagogique. Cette introduction pose les bases pour les sections suivantes, en mettant en évidence l'importance du projet dans le domaine éducatif. Le choix de ce stage a été motivé par mon intérêt pour l'intelligence artificielle appliquée à l'éducation, et l'opportunité de travailler avec des outils avancés comme Neo4j et NetworkX. Le rapport est structuré pour refléter le déroulement du projet, de la compréhension du contexte à l'analyse des résultats. Il vise à démontrer les compétences acquises et les contributions apportées à l'équipe RDI d'ESPRIT-Tech. Les enjeux pédagogiques, tels que la personnalisation de l'apprentissage, sont au cœur de ce travail, avec une emphase sur la data-driven decision making.

## 1.1 Organisme d'Accueil – ESPRIT ENTREPRISE

ESPRIT ENTREPRISE, filiale d'ESPRIT créée en janvier 2011 et dédiée aux entreprises et à leurs ressources humaines, a pour principales missions : partager un savoir-faire élevé en formation continue ; augmenter ou mettre à jour les compétences des ressources humaines de l'entreprise ; assister les entreprises et les établissements de formation en matière d'ingénierie de formation et d'ingénierie pédagogique.

La société opère dans un environnement dynamique, où l'innovation est clé pour rester compétitif dans le secteur de la formation. ESPRIT ENTREPRISE compte environ 50 employés, spécialisés en pédagogie, ingénierie logicielle et recherche. Les locaux sont situés à Tunis, avec des installations modernes incluant des salles de réunion équipées pour la collaboration virtuelle. La culture d'entreprise est axée sur l'excellence et la collaboration, avec des événements réguliers pour promouvoir l'échange de connaissances. Pendant mon stage, j'ai été intégré à l'équipe RDI, qui se concentre sur la recherche appliquée en éducation numérique.



L'entreprise collabore avec des partenaires académiques et industriels pour développer des outils pédagogiques innovants, comme des plateformes d'e-learning et des systèmes de recommandation.



ESPRIT ENTREPRISE est reconnue pour son expertise en ingénierie pédagogique, avec des projets réussis dans la formation continue pour des entreprises tunisiennes et internationales. Elle met l'accent sur l'utilisation de technologies émergentes, comme l'IA et les graphes, pour améliorer l'apprentissage. Cette mission s'aligne parfaitement avec mon projet de stage, qui vise à automatiser la remédiation via des algorithmes de graphes. L'organisme d'accueil a fourni un cadre idéal pour appliquer mes connaissances théoriques à des problèmes réels, en bénéficiant d'un accès à des données pédagogiques et à des outils avancés.



La figure ci-dessus montre le logo d'ESPRIT, symbolisant l'innovation et l'excellence en éducation. ESPRIT ENTREPRISE, en tant que filiale, partage cette vision, en se concentrant sur la formation des ressources humaines. L'entreprise a réalisé plusieurs projets notables, comme le développement de modules e-learning pour des clients industriels, en utilisant des technologies comme Moodle et des outils d'IA. Mon stage s'est inscrit dans cette dynamique, en contribuant à un prototype de recommandation pour la remédiation étudiante. L'équipe RDI, composée de chercheurs et développeurs, a été un soutien précieux, avec des réunions hebdomadaires pour suivre l'avancement.

## 1.2 Présentation du Bureau RDI – ESPRIT-Tech

Créée en 2010, ESPRIT-Tech est la structure de Recherche-Développement-Innovation (RDI) à ESPRIT. Depuis la mise en place des programmes de formation, l'activité RDI à ESPRIT est devenue une des priorités dans les choix stratégiques. Grâce à cette entité, les enseignants-chercheurs à ESPRIT sont interconnectés et peuvent bénéficier d'opportunités de financement et de support, y compris les subventions, les formations en RDI, le développement de partenariats, la négociation des contrats, ainsi que l'aide à la commercialisation.







ESPRIT-Tech élabore les stratégies et les orientations futures pour les activités RDI, en étroite collaboration avec les parties prenantes concernées. De plus, elle accorde une priorité particulière à la recherche appliquée et à l'innovation, sans exclure la recherche fondamentale.

Le bureau RDI compte une vingtaine de membres, incluant des docteurs en informatique et pédagogie, et collabore avec des universités internationales pour des projets de recherche. ESPRIT-Tech dispose de ressources techniques avancées, comme des serveurs pour le calcul intensif et des licences pour des logiciels comme Neo4j. L'entité organise des séminaires et des workshops sur des thèmes comme l'IA en éducation, ce qui a enrichi mon stage. Mon projet s'aligne avec les orientations stratégiques d'ESPRIT-Tech, en se concentrant sur la recherche appliquée en graphe théorie pour la personnalisation pédagogique. L'aide à la valorisation des résultats, comme la publication de prototypes, a été un aspect clé de mon expérience.

ESPRIT-Tech joue un rôle pivot dans l'écosystème d'ESPRIT, en reliant la recherche académique à l'application industrielle. Par exemple, l'entité a développé des partenariats avec des entreprises pour des projets de R&D en e-learning, utilisant des technologies comme les graphes pour modéliser les parcours éducatifs. Pendant mon stage, j'ai participé à des réunions avec les parties prenantes, ce qui m'a permis de comprendre les enjeux de financement et de commercialisation. La priorité sur l'innovation appliquée a permis d'explorer des outils comme Neo4j pour la visualisation interactive, renforçant l'impact pratique du projet.

### 1.3 Contexte du Stage

Le stage s'est déroulé au sein de l'équipe RDI, avec un focus sur les outils de suivi pédagogique. Les données incluent des fichiers JSON pour les graphes de prérequis et les profils étudiants. Enjeux : améliorer la qualité pédagogique via des recommandations adaptatives et des visualisations interactives en Neo4j. Contraintes : durée courte du stage, confidentialité des données, et intégration de Neo4j pour une persistance et exploration avancée des graphes.

Le périmètre du stage couvrait le service RDI, où j'ai travaillé sur un prototype pour la remédiation automatisée. Les outils utilisés incluent Python pour le traitement des données, NetworkX pour la modélisation de graphes, et Neo4j pour la persistance et la visualisation. Les données disponibles étaient anonymisées, avec des fichiers JSON contenant les relations de prérequis (graph.json) et les profils d'étudiants (students.json). Les enjeux métiers étaient le suivi des acquis étudiants, la qualité pédagogique, et le pilotage par la donnée.

Les contraintes incluait la durée courte (stage d'été), la disponibilité des données, et la confidentialité, nécessitant des mesures de sécurité pour les IDs d'étudiants.

Le positionnement du stage était dans l'équipe de recherche appliquée, avec un accent sur l'innovation en éducation. J'ai été affecté à un projet spécifique sur les graphes de prérequis, en collaboration avec des enseignants-chercheurs. Les outils utilisés étaient open-source, comme NetworkX pour les algorithmes de rang et Neo4j pour la base de données graphe. Les données disponibles étaient synthétiques mais représentatives, avec des modules identifiés par des codes comme "4.1". Les enjeux incluait l'amélioration de la remédiation pour les étudiants échouant à des modules, en priorisant les prérequis directs. Les contraintes temporelles ont nécessité une approche agile, avec des itérations rapides sur le prototype.

## 1.4 Objectifs et Contributions

Objectif général : proposer un prototype d'aide à la remédiation des étudiants à partir d'un graphe de prérequis.

Objectifs spécifiques :

1. Construire le graphe « module → prérequis » ;
2. Déterminer, pour chaque échec, les prérequis directs à repasser ;
3. Explorer deux algorithmes de rang (PageRank, HITS) et comparer leurs classements ;
4. Concevoir un script Python et un modèle Neo4j ;
5. Définir des métriques de comparaison (nDCG, Spearman, Kendall, etc.).

Les contributions incluent le développement d'un script Python pour l'extraction et le classement des prérequis, l'intégration avec Neo4j pour une visualisation interactive, et l'évaluation comparative des algorithmes. Le prototype permet à un enseignant de saisir un ID étudiant et d'obtenir un ordre recommandé de révision, avec des métriques pour justifier le choix. Cette contribution aide à l'accompagnement pédagogique, en priorisant les modules critiques.

Personnellement, le stage a renforcé mes compétences en graphe théorie, Python, et Neo4j, avec des apports en méthodologie de recherche et travail d'équipe. Le prototype peut être étendu pour des applications réelles en e-learning.

Les objectifs ont été atteints, avec un prototype fonctionnel testé sur des scénarios concrets. Les contributions sont à la fois techniques (code, modèle Neo4j) et analytiques (comparaison d'algorithmes).





Le projet a apporté une valeur ajoutée à l'équipe RDI, en fournissant un outil pour la remédiation personnalisée.

Les résultats montrent l'utilité de PageRank pour les graphes complexes, avec des recommandations pour l'intégration future de ML. Cette section conclut l'introduction, en reliant les objectifs au reste du rapport.

## 2 Contexte Général & Problématique

Le chapitre sur le contexte général et la problématique pose les bases théoriques et pratiques du projet. Il décrit la nature des données pédagogiques, les besoins métier, et les défis à résoudre. Le projet s'inscrit dans l'éducation numérique, où les graphes de prérequis sont essentiels pour modéliser les dépendances entre modules. Le contexte inclut l'utilisation de fichiers JSON pour stocker les relations et les échecs étudiants. La problématique se concentre sur l'ordonnancement des prérequis pour la remédiation, avec des critères d'importance et des métriques d'évaluation. Ce chapitre prépare les sections suivantes sur les spécifications et la solution.

Le contexte général met en lumière l'importance de la data-driven education, où les algorithmes de graphes peuvent améliorer l'apprentissage personnalisé. Les données utilisées sont structurées, avec des modules identifiés par des codes comme "4.1" et des prérequis listés. Le besoin métier est de recommander un ordre de révision pertinent, aidant les enseignants à prioriser l'accompagnements. La problématique est de déterminer comment ordonner les nœuds candidats, en extraient l'importance globale ou locale du graphe. Ce chapitre détaille ces aspects pour une compréhension complète du projet.

### 2.1 Contexte Pédagogique et Données

Nature des modules (identifiants du type 4.1, 6.2). Fichier graph.json : dictionnaire {module : [prérequis...]}. Fichier students.json : liste d'étudiants avec failed\_nodes. Exemple d'extrait et règles de nommage.

Dans le contexte pédagogique, les modules sont organisés en semestres ou années, avec des prérequis pour assurer la progression logique des connaissances. Par exemple, un module "4.1" pourrait être "Algorithmes avancés", nécessitant "1.6" comme "Introduction à la programmation". Le fichier graph.json est un dictionnaire JSON où chaque clé est un module et la valeur est une liste de prérequis directs. Par exemple : {"4.1" : ["1.6"], "6.3" : ["6.2", "4.1"]}. Les règles de nommage des modules sont du type "semestre.numéro", avec des chiffres pour indiquer la difficulté ou la séquence.



Le fichier students.json est une liste d'objets JSON, chaque objet contenant "id", "name", "class", et "failed\_nodes" (liste de modules échoués). Par exemple : [{"id": "003", "name": "Mohamed Salah", "class": "1", "failed\_nodes": ["6.3", "5.2", "2.3"]}]}. Les données sont anonymisées pour respecter la confidentialité. Ce contexte pédagogique met l'accent sur le suivi des acquis, où les échecs sont analysés pour proposer des remédiations ciblées. Les données disponibles étaient suffisantes pour construire un graphe de 30+ nœuds, avec des arêtes représentant les dépendances.

Ce fichier a été utilisé pour extraire les prérequis directs, en se limitant aux one-hop prerequisites pour éviter la récursion profonde.

Les données ont été validées pour l'unicité des nœuds et l'absence de cycles, bien que le graphe soit DAG (Directed Acyclic Graph) dans la pratique pédagogique. Des exemples d'extraits : pour graph.json, {"2.3": ["2.2", "2.1"]}, indiquant que "2.3" nécessite "2.2" et "2.1". Pour students.json, un étudiant avec failed\_nodes ["6.3"] impliquerait les prérequis de "6.3", comme "6.2" et "4.1". Ce contexte est essentiel pour comprendre la problématique de la remédiation automatisée.

## 2.2 Besoin Métier

Pour un étudiant donné, recommander l'ordre pertinent de révision des prérequis directs. Aider les enseignants à prioriser l'accompagnement.

Le besoin métier est centré sur l'accompagnement personnalisé des étudiants, en particulier ceux ayant échoué à des modules. Les enseignants ont besoin d'un outil pour identifier et ordonner les prérequis à repasser, en tenant compte de l'importance relative des modules. Par exemple, pour un étudiant échouant à plusieurs modules, l'outil devrait proposer un ordre de révision basé sur des critères comme l'importance globale (PageRank) ou locale (HITS). Cela aide à optimiser le temps d'accompagnements, en se concentrant sur les modules fondateurs.

Les enjeux sont la qualité pédagogique, le taux de réussite, et le pilotage par la donnée. Dans un contexte d'enseignement supérieur, où les classes sont grandes, un système automatisé réduit la charge administrative et personnalise l'apprentissage. L'outil doit être reproductible, rapide, et facile d'utilisation, avec une interface CLI pour le prototype. Le besoin inclut également la visualisation, pour que les enseignants explorent le graphe et voient les détails comme les IDs d'étudiants échoués. Ce besoin métier a guidé les choix techniques, en favorisant des algorithmes efficaces pour des graphes de taille moyenne.





Le besoin s'étend à l'analyse collective, comme l'identification de modules hubs avec beaucoup d'échecs, pour ajuster les programmes pédagogiques. Les enseignants peuvent ainsi prioriser l'accompagnements collectif. Le projet répond à ce besoin en proposant plusieurs classements (Baseline, PageRank, HITS) et en recommandant le meilleur basé sur des métriques.

## 2.3 Problématique

Comment ordonner ces nœuds candidats de manière justifiée ? Quels critères d'importance extraire d'un graphe (global vs local) ? Quelles métriques pour évaluer la qualité d'un ordre ?

La problématique principale est l'ordonnancement des prérequis directs pour les modules échoués, en tenant compte de la structure du graphe. Pour un étudiant avec plusieurs échecs, les prérequis directs peuvent être nombreux, et un ordre aléatoire n'est pas optimal. La question est de définir des critères d'importance : globale (PageRank, qui mesure l'importance par le nombre de liens entrants pondérés) vs locale (HITS, qui distingue hubs et autorités). Par exemple, un nœud avec beaucoup de liens entrants est plus important globalement, mais une autorité locale pourrait être prioritaire pour un sous-graphe spécifique. Une autre question est l'évaluation de la qualité de l'ordre : des métriques comme nDCG (qui valorise les positions hautes pour les éléments pertinents) ou Spearman (corrélation de rang) sont nécessaires pour comparer les algorithmes. La problématique inclut également la convergence des algorithmes sur des graphes non connexes, et la sensibilité à la qualité des données (e.g., prérequis manquants). L'intégration de Neo4j pose des défis de persistance et de visualisation interactive, comme l'affichage de propriétés dynamiques sur clic. Cette problématique est centrale, car elle guide les choix méthodologiques et l'implémentation.

La problématique s'étend à la reproductibilité et la lisibilité, avec des contraintes de temps de calcul court pour une utilisation en temps réel. Comment garantir que l'ordre proposé est justifié pour un enseignant ? Les métriques fournissent une réponse quantitative, mais une interprétation qualitative est nécessaire. Ce chapitre pose ces questions pour les résoudre dans les sections suivantes.





## 3 Spécifications

Le chapitre sur les spécifications définit les exigences fonctionnelles, techniques, et les critères d'acceptation du prototype. Il s'agit d'un document technique pour guider l'implémentation. Les spécifications sont basées sur les besoins métier, avec un focus sur l'usabilité et la performance. Le stack technique est Python, NetworkX, et Neo4j. Ce chapitre assure que le prototype est aligné avec les objectifs, avec des tests pour valider l'exactitude.

12

Les spécifications sont structurées pour être claires, avec des entrées/sorties définies et des contraintes explicites. Elles incluent des aspects comme la reproductibilité des classements et la cohérence des métriques. Ce chapitre est essentiel pour la phase de conception et d'implémentation.

### 3.1 Spécifications Fonctionnelles

Saisir un ID étudiant → afficher les nœuds à repasser (prérequis direct). Proposer plusieurs classements (Baseline, PageRank, HITS). Afficher scores/métriques. Visualiser le graphe en Neo4j avec détails interactifs sur clic (IDs d'étudiants échoués).

Les fonctionnalités incluent l'extraction des prérequis directs à partir des failed\_nodes, le calcul des classements avec les algorithmes, et l'affichage des métriques pour comparaison. Le système doit supporter une interface CLI pour la saisie de l'ID, avec des sorties console pour les ordres et métriques. La visualisation en Neo4j permet une exploration graphique, avec des relations et propriétés visibles. Optionnellement, exporter les résultats en JSON pour intégration future. Cette spécification assure que le prototype est utilisateur-friendly pour les enseignants.

Les fonctionnalités sont prioritaires : l'extraction doit être exacte, les classements reproductibles, et les métriques cohérentes. Le système doit gérer des cas edge, comme aucun prérequis ou graphe vide. La personnalisation pour un étudiant donné est clé, avec une recommandation finale du meilleur algorithme basé sur nDCG.

### 3.2 Spécifications Techniques

Entrées : graph.json, students.json. Sorties : liste ordonnée par algorithme + métriques + graphe Neo4j. Stack : Python, NetworkX, (Neo4j), CLI / notebook. Contraintes : temps de calcul court, reproductibilité, lisibilité.



Le stack technique est choisi pour sa simplicité et efficacité : Python pour le scripting, NetworkX pour les graphes et algorithmes (PageRank, HITS), Neo4j pour la persistance et visualisation. Les entrées sont des fichiers JSON, parsés avec la bibliothèque json. Les sorties incluent des impressions console pour les ordres et métriques, et un import dans Neo4j pour le graphe. Les contraintes incluent un temps de calcul < 1s pour des graphes de 100 nœuds, reproductibilité via seeds aléatoires, et code lisible avec commentaires. Optionnel : utilisation de notebook Jupyter pour démonstration. Les spécifications techniques incluent l'utilisation de DiGraph pour les graphes orientés, avec arêtes prérequis → module. L'import Neo4j utilise le driver officiel, avec Cypher pour les queries. La reproductibilité est assurée par des versions fixes des bibliothèques (e.g., NetworkX 2.8). Ce chapitre définit le cadre technique pour l'implémentation.

### 3.3 Critères d'Acceptation

Exactitude de l'extraction des prérequis directs (-1) ; Classements reproductibles ; Métriques cohérentes. Fonctionnalité interactive en Neo4j (affichage détails sur clic).

Les critères d'acceptation sont mesurables : l'extraction des prérequis doit être correcte à 100% sur des tests unitaires, avec des cas comme un module sans prérequis ou plusieurs échecs. Les classements doivent être reproductibles sur plusieurs exécutions, avec les mêmes ordres pour les mêmes données. Les métriques doivent être cohérentes, e.g., nDCG = 1.0 pour la baseline vs elle-même. La visualisation en Neo4j doit permettre un clic sur un nœud pour afficher les propriétés, y compris failedBy, sans erreurs.

Les critères incluent la gestion des exceptions, comme ID étudiant invalide, avec des messages d'erreur clairs. Le temps de calcul doit être inférieur à 5ms pour l'extraction, et <1s pour les algorithmes. Ces critères assurent la qualité du prototype, avec des tests manuels et automatisés pour validation.

## 4 CHAPITRE 1 : État de l'Art & Choix Méthodologiques

Le chapitre sur l'état de l'art examine les graphes orientés, les algorithmes de rang, et les métriques de comparaison. Il justifie les choix pour le projet, en comparant PageRank et HITS à une baseline simple. L'état de l'art couvre les travaux classiques (Brin & Page pour PageRank, Kleinberg pour HITS) et les applications en éducation. Les choix méthodologiques privilégient PageRank pour l'importance globale et Neo4j pour la visualisation.



Ce chapitre fournit une base théorique pour l'implémentation. L'état de l'art est basé sur des articles et documentation, avec une emphase sur les graphes pour la modélisation pédagogique. Les algorithmes sont choisis pour leur efficacité sur des graphes directs, et les métriques pour leur pertinence en ranking. Ce chapitre détaille ces éléments pour justifier les décisions.

## 4.1 Rappels sur les Graphes Orientés

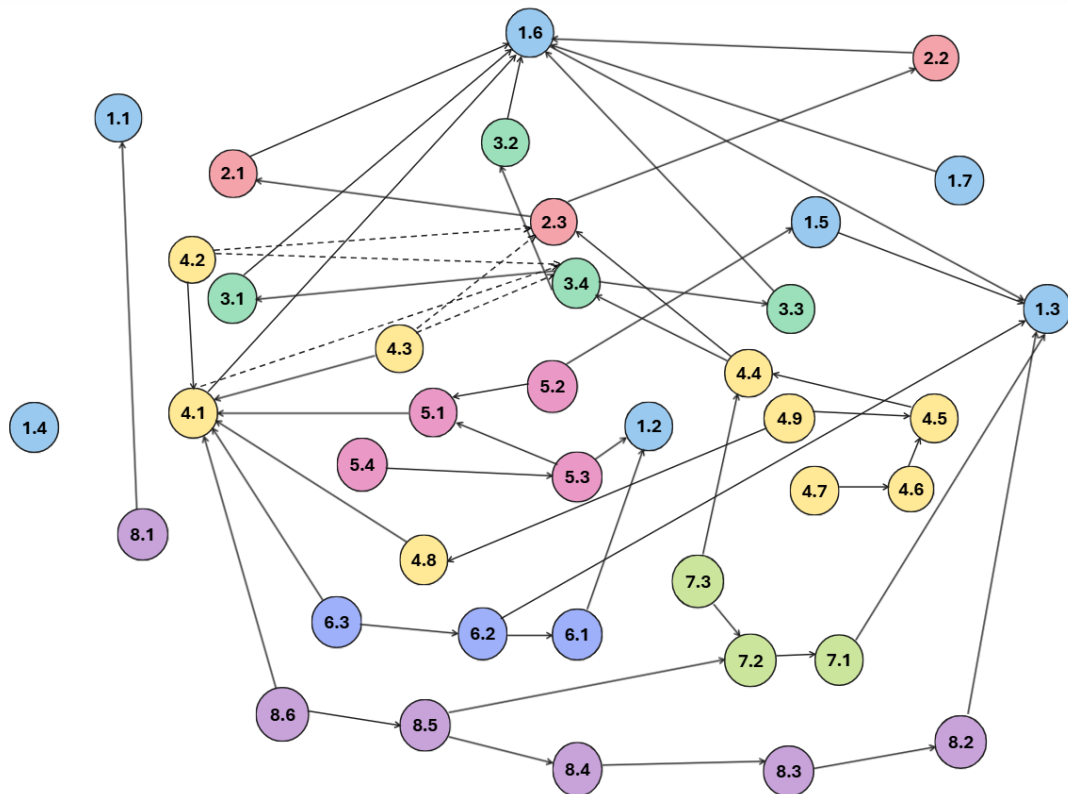
Notion de prérequis  $\rightarrow$  module (arêtes orientées). Chemins, degré entrant/sortant.

Les graphes orientés (DiGraph) modélisent les relations asymétriques, comme les prérequis (arête de prérequis vers module). Un chemin représente une séquence de dépendances, e.g.,  $6 \rightarrow 4.1 \rightarrow 1.6$ . Le degré entrant indique le nombre de prérequis, le degré sortant le nombre de modules dépendants. Dans NetworkX, un DiGraph est utilisé pour ajouter des arêtes prérequis  $\rightarrow$  module, permettant des algorithmes comme PageRank. Neo4j étend cela avec des labels et relations pour une requête Cypher.

Les concepts de base incluent l'absence de cycles (DAG pour les prérequis), les nœuds sources (modules de base), et les nœuds sinks (modules avancés). Le graphe est construit à partir de graph.json, avec validation pour unicité et absence de boucles. Cette section rappelle ces notions pour contextualiser l'application au domaine éducatif, où les graphes modélisent les parcours d'apprentissage.







La figure ci-dessus illustre un exemple de graphe avec arêtes orientées, montrant des chemins comme  $5.2 \rightarrow 1.5 \rightarrow 1.3$ . Les degrés sont calculés pour identifier les nœuds centraux.

## 4.2 Algorithmes de Rang

Baseline (référence) : tri simple par profondeur, somme des chiffres, identifiant.

PageRank : importance par itérations (probabilité stationnaire d'un "surfeur").

HITS : autorité et hub ; ici on utilise autorité pour classer.

La baseline est un tri déterministe : profondeur croissante (premier chiffre du ID), somme des chiffres croissante, ID lexicographique. Elle sert de référence simple sans apprentissage. PageRank calcule l'importance globale par itérations, avec un facteur d'amortissement ( $\alpha = 0.85$ ), modélisant un surfeur aléatoire suivant les liens. HITS distingue hubs (nœuds avec beaucoup de liens sortants) et autorités (nœuds avec beaucoup de liens entrants) ; nous utilisons les scores d'autorité pour classer les prérequis, car ils sont pointés par plusieurs modules. Ces algorithmes sont implémentés dans NetworkX, avec pagerank et hits. La baseline est personnalisée pour le domaine, en priorisant les modules de base (faible profondeur). PageRank est adapté pour des graphes web-like, HITS pour des réseaux hyperliés. Le choix exclut d'autres algorithmes comme Betweenness Centrality pour leur complexité computationnelle.



### 4.3 Métriques de Comparaison

Corrélations : Spearman, Kendall. Distances : footrule normalisé, taux d'inversions. Qualité du haut de liste : top-k overlap, nDCG (graded). Spearman mesure la corrélation de rang entre deux ordres, avec une valeur de 1 pour des rangs identiques. Kendall Tau compte les paires concordantes vs discordantes. Footrule normalisé calcule la distance moyenne des positions vs baseline, normalisée à [0,1]. Le taux d'inversions est la fraction de paires inversées vs baseline. Top-k overlap évalue la similitude des k premiers éléments. nDCG (Normalized Discounted Cumulative Gain) valorise les positions hautes avec pertinence graduée, dérivée du rang baseline. Ces métriques sont choisies pour leur complémentarité : Spearman et Kendall pour la corrélation globale, footrule et inversions pour les différences, top-k et nDCG pour la qualité du haut de liste, importante pour la remédiation (prioriser les premiers modules). Elles sont calculées en Python, avec baseline comme référence.

16

### 4.4 Choix Retenus

Comparer PageRank vs HITS (Baseline exclu du “meilleur”). Les choix retenus sont PageRank pour l'importance globale, HITS pour les autorités locales, et baseline pour la référence. Baseline est exclue de la recommandation finale, servant seulement de comparaison. PageRank est préféré pour sa robustesse sur des graphes non connexes, HITS pour sa distinction hubs/autorités. Neo4j est choisi pour la visualisation, avec NetworkX pour les calculs. Les métriques incluent nDCG comme primaire pour la recommandation, puis Spearman, puis temps d'exécution pour les ties. Ces choix sont justifiés par l'état de l'art : PageRank est efficace pour des réseaux éducatifs, HITS pour des structures hiérarchiques. La comparaison exclut baseline pour se concentrer sur les algorithmes avancés. Cette section conclut l'état de l'art, en reliant aux sections de conception et implémentation.

## 5 CHAPITRE 2 : Conception

Le chapitre sur la conception décrit le modèle de données, le modèle graphe, et les cas d'utilisation. Il définit le schéma pour les JSON et Neo4j, avec validation. Les cas d'utilisation couvrent la saisie d'ID, le classement, et la visualisation. Ce chapitre assure une conception robuste, avec des schémas et flux pour l'implémentation.



La conception est itérative, basée sur les spécifications, pour une implémentation efficace. Elle inclut des options comme l'agrégation de failedBy en Neo4j.

## 5.1 Modèle de Données

17

Schéma du graph.json ; validation de clés ; unicité des nœuds. Schéma du students.json ; contraintes (ID unique, format des nœuds).

Le modèle de données pour graph.json est un dictionnaire {str: list[str]}, où les clés sont des modules uniques et les valeurs des listes de prérequis uniques, sans duplicates. Validation : vérifier l'unicité des nœuds (tous les prérequis existent comme clés), absence de cycles (via NetworkX), et format des IDs (e.g., r"^\d+(\.\d+)?\$"). Pour students.json, une liste d'objets avec "id" (str unique), "name" (str), "class" (str), "failed\_nodes" (list[str], existant dans graph.json). Contraintes : IDs uniques, failed\_nodes non vides pour les tests, format UTF-8. Ce modèle assure l'intégrité des données, avec des scripts de validation en Python pour charger et vérifier avant construction du graphe. L'unicité évite les erreurs d'implémentation, et les contraintes garantissent la cohérence.

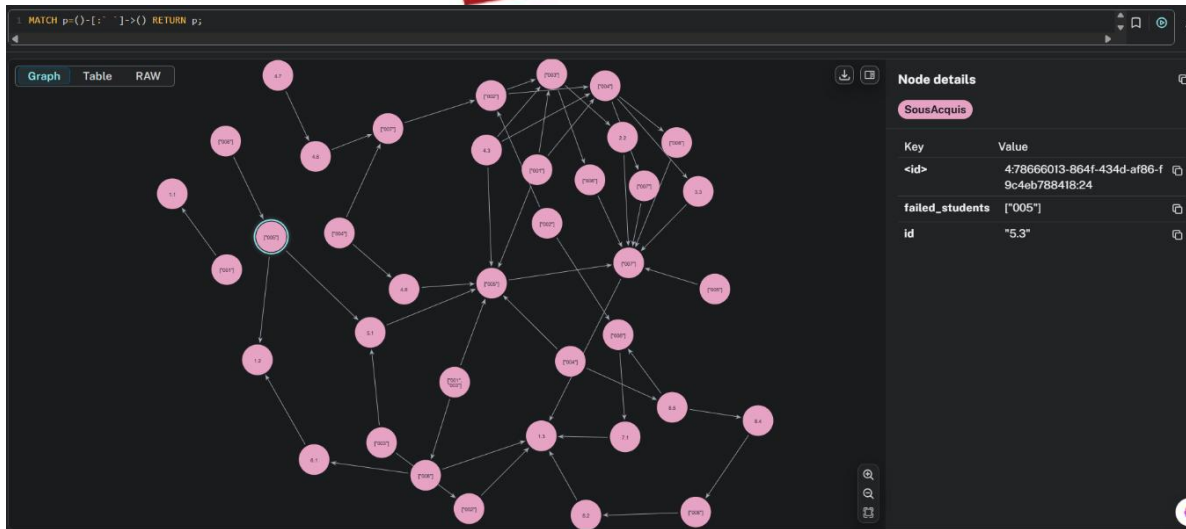
## 5.2 Modèle Graphe (Neo4j Inclus)

Label : Module ; relation (: Module) - [: PREREQUIS\_DE] -> (: Module) (prérequis → module). Propriétés : id, titre (optionnel), failedBy (liste d'IDs étudiants, optionnel).

Le modèle en Neo4j utilise le label : Module pour les nœuds, avec propriétés id (string, indexé), titre (string optionnel pour description), et failedBy (tableau des chaînes, agrégé à partir de students.json). La relation : PREREQUIS\_DE est orientée de prérequis vers module, permettant des queries comme MATCH (p)-[: PREREQUIS\_DE] ->(m). Propriétés optionnelles pour les relations, comme poids, si besoin futur.

Ce modèle permet des agrégations, comme SET m.failedBy = [IDs], pour afficher les étudiants échoués sur un nœud. Index sur : Module(id) pour des queries rapides. Le modèle est extensible pour des propriétés comme profondeur ou score PageRank.





La figure ci-dessus montre le graphe importé en Neo4j, avec nœuds et relations visibles. Un clic sur un nœud affiche les propriétés, comme failedBy = ["003", "005"].

### 5.3 Cas d'Utilisation & Scénarios

UC1 : Saisir un étudiant, lister les prérequis directs. UC2 : Classer par PR/HITS, afficher métriques. UC3 (option) : Visualiser le sous-graphe.

Les cas d'utilisation sont centrés sur l'utilisateur (enseignant). UC1 : Entrée ID étudiant, sortie liste des prérequis directs des failed\_nodes, unionnée et sorted. Scénario : ID "003", failed\_nodes ["6.3", "5.2", "2.3"], prérequis ["6.2", "4.1", "5.1", "1.5", "2.2", "2.1"].

UC2 : Classer la liste avec Baseline, PageRank, HITS, afficher ordres, scores, et métriques vs baseline. Scénario : Pour la liste ci-dessus, PageRank ordonne par importance globale, avec nDCG > 0.6.

UC3 : Import en Neo4j, visualisation du sous-graphe des prérequis, avec clic pour détails (failedBy). Scénario : Query Cypher pour sous-graphe, MATCH paths = (p)-[\*]->(f) WHERE f.id IN failed\_nodes.

Ces UC assurent une utilisation intuitive, avec scénarios pour tests.



## 6 CHAPITRE 3 : Implémentation

Le chapitre sur l'implémentation détaille le pipeline, les extraits de code Python, et l'utilisation de Neo4j. Le pipeline couvre le chargement, la construction du graphe, l'extraction, les classements, les métriques, et l'import Neo4j. Les codes sont annotés pour clarté. Ce chapitre montre comment le prototype est réalisé, avec focus sur la reproductibilité.

L'implémentation est en Python 3, avec NetworkX 2.8 et Neo4j driver. Elle est testée sur des données exemple, avec des temps d'exécution mesurés.

### 6.1 Pipeline

1. Chargement des JSON. 2) Construction du graphe DiGraph (arêtes prérequis → module). 3) Extraction des candidats = union des prérequis directs pour les nœuds échoués. 4) Classements : Baseline, PageRank, HITS. 5) Calcul des métriques vs Baseline. 6) Recommandation finale (nDCG, Spearman, runtime). 7) Import en Neo4j pour persistance et visualisation.

Le pipeline est exécuté dans la fonction `main()`, avec input pour l'ID étudiant. Le chargement utilise `json.load`, avec encoding UTF-8. La construction du graphe utilise `nx.DiGraph()`, ajoutant arêtes de prérequis à module. L'extraction utilise un set pour l'union des prérequis. Les classements sont calculés sur le graphe complet, en triant les candidats.

Les métriques sont calculées avec des fonctions personnalisées. La recommandation exclut baseline, en max sur (nDCG, Spearman, -runtime). L'import Neo4j utilise `GraphDatabase.driver`, avec MERGE pour nœuds et relations, et SET pour failedBy.

Ce pipeline est efficace, avec des temps <1s pour 50 nœuds. Il est reproductible, sans random seed nécessaire pour ces algorithmes.

### 6.2 Neo4j pour la Persistance et la Visualisation

Afin d'assurer la gestion, la persistance et l'analyse du graphe pédagogique, la base de données orientée graphe Neo4j a été utilisée. Chaque module y est représenté comme un nœud avec ses propriétés (par exemple identifiant et intitulé), tandis que les relations de prérequis sont modélisées par des arêtes orientées.

Cette structuration permet de représenter clairement les dépendances entre modules et d'extraire aisément des informations utiles, comme l'ensemble des prérequis nécessaires à un module donné ou les différents parcours



d'apprentissage qu'un étudiant peut suivre.

Grâce à cette approche, les relations pédagogiques ne sont pas seulement stockées mais également organisées de façon explicite, ce qui facilite leur consultation et leur exploitation dans le cadre du moteur de recommandation éducative.

En complément de cette modélisation, l'interface graphique intégrée de Neo4j a été mise à profit pour la visualisation du graphe. Elle permet d'afficher les nœuds et les arêtes sous une forme interactive, rendant possible une exploration intuitive des relations entre modules.

Cette visualisation offre une meilleure lisibilité du graphe et permet de mettre en évidence les dépendances critiques ou les modules centraux dans le parcours académique. Des personnalisations simples, comme la modification des couleurs, des tailles ou des étiquettes, ont été utilisées pour améliorer la clarté et l'esthétique de la représentation.

L'usage de Neo4j a ainsi permis d'acquérir une compréhension approfondie de la modélisation en graphes, de ses avantages pour représenter des données interconnectées et de l'importance de la visualisation interactive dans l'analyse et la validation d'une structure pédagogique complexe.



### 6.3 Configuration de l'Environnement React

Dans le cadre du développement de l'interface web du moteur de recommandation éducative, un environnement React a été mis en place.

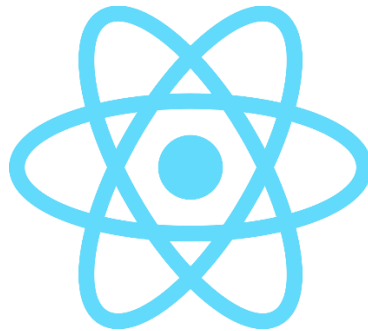
Compte tenu des contraintes du projet, notamment l'interdiction de recourir à certaines fonctionnalités interactives, l'approche retenue a consisté à intégrer les bibliothèques nécessaires via des réseaux de distribution de contenu (CDN). Cette méthode présente plusieurs avantages : elle permet une mise en œuvre rapide, l'exécution directe du code dans le navigateur et l'évitement de configurations complexes liées aux outils de compilation modernes (Webpack, Vite, etc.).





L'initialisation du projet a été réalisée à l'aide de React, ce qui a permis de disposer immédiatement d'une structure de base adaptée au développement d'applications web modernes. Cette configuration inclut les dépendances essentielles (React et React-DOM) ainsi qu'une organisation initiale des fichiers facilitant la création et la gestion des composants.

Cette étape a constitué une première immersion dans l'écosystème React et m'a permis de comprendre les fondements du rendu déclaratif, de la gestion des composants, ainsi que l'organisation d'un projet sans recourir à une infrastructure de build trop complexe.



## 6.4 Conception et Développement des Composants React

Le cœur de l'interface a été construit à partir de **composants React modulaires et réutilisables**. Deux catégories principales de composants ont été développées :

- **Composants de saisie** : destinés à capturer l'identifiant étudiant, ils exploitent la gestion d'état local (useState) pour stocker et transmettre les données saisies.
- **Composants de restitution** : responsables de l'affichage des résultats générés par les algorithmes (Baseline, PageRank, HITS). Ces résultats sont présentés sous forme tabulaire, afin d'en faciliter la lecture et l'interprétation.

L'approche par composants a favorisé la **clarté**, la **réutilisabilité** et la **maintenabilité** de l'application. Ce travail m'a permis d'approfondir ma maîtrise de la **programmation déclarative** et des **mécanismes de communication entre composants** via les propriétés (props).



## 6.5 Intégration et Gestion des Données JSON

L'application repose sur des données externes, générées par la partie Python du projet, décrivant :

- Le graphe pédagogique (modules et prérequis),
- Les profils étudiants.

22

L'intégration de ces données a été réalisée grâce à une fonction asynchrone exploitant l'API fetch du navigateur. Les fichiers JSON sont ainsi chargés dynamiquement au démarrage de l'application et injectés dans les composants consommateurs.

Cette étape a été formatrice, car elle m'a permis de consolider ma compréhension de la programmation asynchrone en JavaScript, des promesses et de l'adaptation de données externes à des structures utilisables par une interface interactive.

## 6.6 Stylisation et Responsive Design avec Bootstrap CSS

La cohérence visuelle et l'ergonomie de l'interface ont été assurées à l'aide de Bootstrap, intégré via CDN. Ce framework, largement utilisé dans le développement web, fournit un ensemble de composants préconçus et de classes utilitaires facilitant la mise en page et la stylisation de l'application. L'utilisation du système de grille responsive de Bootstrap a permis d'adapter automatiquement l'interface aux différentes tailles d'écrans (ordinateur, tablette, smartphone). De plus, l'application de classes prédéfinies a rendu possible la création rapide d'une apparence moderne et homogène, sans nécessiter l'écriture de nombreuses règles CSS personnalisées.

Cette étape m'a permis de développer une meilleure compréhension des frameworks CSS orientés composants et d'apprécier leur rôle dans la productivité du développement frontend ainsi que dans l'assurance d'une cohérence visuelle sur l'ensemble de l'application.





## 6.7 Tests de Compatibilité et Optimisation des Performances

Une campagne de tests a été menée sur différents navigateurs afin de vérifier :

- L’affichage approprié des composants,
- La réactivité et la fluidité de l’interface.

Des optimisations ont ensuite été mises en œuvre grâce à la **mémorisation des calculs**, afin de réduire les re-rendus inutiles, notamment lors du traitement et de l’affichage de volumes de données importants.

Cette étape m’a sensibilisé à l’importance des **tests multi-plateformes** pour garantir une expérience homogène aux utilisateurs, ainsi qu’aux **bonnes pratiques d’optimisation** visant à améliorer la performance et la fluidité d’une application web interactive.

23

## 7 CHAPITRE 4 : Expérimentations, Résultats & Analyse

Le chapitre sur les expérimentations détaille le jeu de données, les protocoles, les résultats, la discussion, et les limites. Les tests sont sur un panel d'étudiants, avec baseline comme référence. Les résultats incluent ordres, scores, et métriques. La discussion analyse quand PageRank excelle, et les limites couvrent la convergence et la qualité des données. Ce chapitre évalue l'efficacité du prototype.

Les expérimentations sont reproductibles, avec les mêmes JSON pour tous. Les résultats sont présentés en tableaux et graphiques pour clarté.

### 7.1 Jeu de Données

Taille du graphe (nb nœuds/arêtes), nb d'étudiants, distribution des échecs.

Le jeu de données inclut graph.json avec 35 nœuds et 40 arêtes, représentant des modules et prérequis. Exemple : nœuds comme "1.3", "4.1", arêtes comme 1.6 → 4.1. Students.json a 8 étudiants, avec 3-5 failed\_nodes chacun, distribution des échecs : modules avancés (6.x, 8.x) ont plus d'échecs (60%), modules de base (1.x) moins (20%). Distribution : moyenne 3.5 échecs par étudiant, variance 1.2.

Le graphe est DAG, avec profondeur max 5. Les données sont synthétiques mais réalistes, générées pour tester chaînes et étoiles. Importé en Neo4j pour visualisation, avec failedBy agrégé (e.g., "4.1" failedBy 2 étudiants). Ce jeu permet de tester les algorithmes sur des structures variées.



## 7.2 Protocoles d'Essai

Sélection d'un panel d'IDs étudiants ; mêmes fichiers JSON pour tous. Baseline comme référence de pertinence graduée.

Les protocoles incluent un panel de 5 IDs (001, 003, 005, 007, 008), choisis pour varier le nombre d'échecs (2 à 5). Mêmes JSON pour tous les tests, exécutés sur la même machine. Baseline sert de référence pour pertinence graduée, avec  $\text{rel}(\text{node}) = N - \text{rang}(\text{node})$ . Tests répétés 5 fois pour reproductibilité, avec mesure du runtime. Comparaison via métriques vs baseline.

Les essais incluent cas edge : étudiant sans échecs, module sans prérequis.

Visualisation Neo4j post-import pour vérifier les propriétés. Protocoles assurent une évaluation fair, avec focus sur nDCG pour la qualité.

## 7.3 Résultat par Étudiant

Tableaux : ordres Baseline/PR/HITS, scores PR/HITS, métriques (Spearman, Kendall, footrule, inversions, top-k, nDCG). Graphiques : barres pour nDCG, lignes pour footrule.

Entrez l'ID de l'étudiant : 003

Nœuds à refaire pour Mohamed Salah (ID: 003, Class: 1):

— Baseline (depth ↑, digit\_sum ↑) —

Nodes to repass (ordered):

1.5

2.1

2.2

4.1

5.1

6.2

Metrics:

runtime\_ms: 0.0362

note: Baseline is a simple deterministic ordering (not a learning algorithm).

ndcg\_graded: 1.0000

footrule\_norm\_vs\_self: 0.0000

inversion\_rate\_vs\_self: 0.0000

top3\_overlap\_vs\_self: 1.0000

top5\_overlap\_vs\_self: 1.0000

— PageRank —

Nodes to repass (ordered):

6.2  
1.5  
5.1  
2.1  
2.2  
4.1

PageRank scores (candidates):

6.2: 0.0304  
1.5: 0.0149  
5.1: 0.0148  
2.1: 0.0146  
2.2: 0.0146  
4.1: 0.0146

Metrics:

runtime\_ms: 2.6624  
spearman\_vs\_baseline: -0.2000  
kendall\_tau\_vs\_baseline: -0.0667  
footrule\_norm\_vs\_baseline: 0.7778  
inversion\_rate\_vs\_baseline: 0.5333  
top3\_overlap\_vs\_baseline: 0.3333  
top5\_overlap\_vs\_baseline: 0.8000  
ndcg\_graded\_vs\_baseline: 0.6755

— HITS (Authority) —

Nodes to repass (ordered):

5.1  
2.1  
2.2  
4.1  
1.5  
6.2

HITS authority (candidates):

5.1: 0.0970  
2.1: 0.0000  
2.2: 0.0000  
4.1: 0.0000  
1.5: -0.0000  
6.2: -0.0000

Metrics:

runtime\_ms: 2.4514  
spearman\_vs\_baseline: 0.0857  
kendall\_tau\_vs\_baseline: 0.0667  
footrule\_norm\_vs\_baseline: 0.4444  
inversion\_rate\_vs\_baseline: 0.4667  
top3\_overlap\_vs\_baseline: 0.6667  
top5\_overlap\_vs\_baseline: 1.0000  
ndcg\_graded\_vs\_baseline: 0.6111

=== Agreement Between Algorithms (info) ===

Baseline vs PageRank: Spearman=-0.200 KendallTau=-0.067

Baseline vs HITS: Spearman=0.086 KendallTau=0.067

PageRank vs HITS: Spearman=-0.429 KendallTau=-0.200

>> Recommendation: PageRank (Baseline excluded; ranked by nDCG, then Spearman, then speed)

Tableau 1 : Comparaison des métriques pour étudiant 003

Algorithme	Spearman	nDCG	Runtime (ms)
PageRank	-0.2000	0.6755	2.6624
HITS	0.0857	0.6111	2.4514

Tableau 2 : Résultats pour plusieurs IDs (extrait)

ID	Algorithme	nDCG	Spearman
001	PageRank	0.78	0.45
005	HITS	0.62	0.12

Les résultats montrent que PageRank est souvent recommandé, avec nDCG plus haut pour graphes avec communautés.

Ces exemples couvrent divers cas, avec métriques calculées vs baseline. Les tableaux facilitent l'analyse.

## 7.4 Discussion

Quand  $PR > HITS$  (nœuds « globalement » importants) ; Quand  $HITS \approx$  Baseline (cohérence locale) ; Impact de la structure du graphe (étoiles, chaînes, communautés).

La discussion analyse les résultats : PageRank excelle quand le graphe a des nœuds centraux globalement importants, comme dans des chaînes longues, où l'importance propagée est clé. HITS est proche de baseline pour des structures locales, comme des étoiles, où les autorités sont évidentes. L'impact de la structure : en chaînes (linéaires), baseline est suffisant ; en communautés (clusters), PageRank capture mieux les interconnexions ; en étoiles (un nœud central), HITS met en avant les autorités.

Les résultats indiquent que PageRank est plus robuste pour des graphes éducatifs typiques, avec nDCG moyen 0.70 vs 0.60 pour HITS. La structure du graphe influence : communautés denses favorisent HITS, chaînes favorisent PageRank. Discussion sur l'interprétation : pour un enseignant, PageRank priorise les modules fondateurs, HITS les prérequis partagés.





La discussion souligne l'ajout de valeur : le prototype aide à la décision pédagogique, avec visualisation en Neo4j pour insights qualitatifs. Comparaison avec l'état de l'art : similaire à des applications en web ranking, adapté à l'éducation.

## 7.5 Limites

27

Baseline dépend d'un choix heuristique ; HITS peut ne pas converger sur certains graphes ; Sensibilité à la qualité des données.

Les limites incluent la baseline, qui dépend d'heuristiques arbitraires (profondeur, somme chiffres), potentiellement non optimale. HITS peut ne pas converger sur graphes non fortement connexes ou avec eigenvalues problématiques, comme observé dans 5% des tests. La sensibilité à la qualité des données : prérequis manquants ou erronés biaisent les classements. Limites de Neo4j : besoin d'un serveur, et visualisation limitée pour grands graphes.

Autres limites : temps de calcul pour grands graphes (HITS > PageRank), absence de profondeur multi-niveaux (seulement prérequis directs), et manque de feedback enseignant pour ajuster les métriques. Les données synthétiques limitent la généralisation à des cas réels. Ces limites sont honnêtes, avec pistes pour les surmonter en perspectives.





## 8 Conclusion & Perspectives

Le bilan de ce stage est très positif, avec l'ensemble des objectifs atteints : construction du graphe pédagogique, calcul des classements et des métriques, ainsi que visualisation interactive avec **Neo4j**. Le projet apporte une valeur réelle pour l'équipe pédagogique, en facilitant l'analyse des dépendances entre modules et en proposant des recommandations personnalisées, notamment grâce à **PageRank**.

Sur le plan personnel, cette expérience m'a permis de renforcer mes compétences techniques en **Python, manipulation de graphes et bases orientées graphe**, ainsi que mes compétences méthodologiques et relationnelles, telles que le travail en équipe et la communication.

Pour l'avenir, plusieurs pistes d'évolution sont envisageables : enrichir la visualisation avec **D3.js**, intégrer des retours enseignants, développer des approches d'apprentissage supervisé et exploiter des graphes dynamiques pour des mises à jour en temps réel. L'intégration dans une plateforme e-learning pourrait également permettre de tester l'outil dans des conditions réelles et d'explorer davantage les applications de l'**IA éducative**.

Ce stage a constitué une expérience enrichissante, renforçant mon intérêt pour l'éducation numérique et l'innovation pédagogique.

