Une approche hybride combinant Markov, HMM et RNN pour détecter les blocages dans l'apprentissage de la programmation

Grota Abdelkader [2], **Mohammed Erritali** [1,2], **Patrick Etcheverry** [1], **Thierry Nodenot** [1]

1.T2I, Laboratoire LIUPPA, IUT de Bayonne, France.
2.Laboratoire Data4Earth, Faculté des Sciences et Techniques, Béni Mellal Maroc

RÉSUMÉ. *Comprendre le processus d'apprentissage en programmation est un défi complexe en raison de la nature séquentielle et multidimensionnelle des interactions des étudiants avec les environnements numériques. Cette étude vise à analyser les journaux d'activité des étudiants pour détecter les difficultés rencontrées et proposer des interventions pédagogiques adaptées. Nous présentons une approche hybride combinant les Chaînes de Markov, les Modèles Cachés de Markov (HMM) et les Réseaux Neuronaux Récurrents (RNN) avec mécanisme d'Attention, afin d'exploiter les dimensions comportementale, cognitive et structurelle de l'apprentissage.La méthodologie consiste à extraire des caractéristiques séquentielles des journaux d'activité et à modéliser les transitions entre actions pour identifier des schémas tels que les cycles d'exploration, d'hésitation et de blocage. Ces schémas sont intégrés dans un cadre unifié pour prédire les états d'apprentissage des étudiants et détecter les moments critiques de difficulté. Le modèle hybride a été validé sur des données réelles issues de 5 étudiants ayant indiqué avoir réussi à terminer leurs travaux pratiques, avec un total de 26 enregistrements de traces. L'exercice visait à implémenter un tri à bulles sur des entiers, puis à appliquer cette méthode sur d'autres structures de données.Les résultats montrent que le modèle hybride proposé surpasse les approches traditionnelles, avec une précision de 93,5% et une réduction significative des faux positifs dans la détection des blocages. L'analyse multidimensionnelle permet de mieux comprendre les comportements et les trajectoires d'apprentissage des étudiants.Cette recherche ouvre la voie au développement de plateformes éducatives intelligentes capables de fournir un feedback personnalisé, favorisant ainsi une meilleure réussite et un engagement accru des étudiants.*

MOTS-CLÉS : *journaux d'activité, apprentissage de la programmation, modèle hybride, RNN, HMM, chaînes de Markov, difficultés des étudiants*

1. Introduction

Former des étudiants au métier de développeur est un défi pédagogique majeur. Apprendre à programmer ne se limite pas à l'assimilation d'une syntaxe ou à la mémorisation de concepts : il s'agit d'un processus itératif où les étudiants manipulent du code, testent des solutions, détectent et corrigent des erreurs et valident les résultats. Ce parcours exige un raisonnement algorithmique structuré et une capacité à produire des solutions fiables, ce qui mobilise un processus cognitif complexe.

Toutefois, la progression des étudiants est souvent inégale. Alors que certains adoptent rapidement des stratégies efficaces, d'autres rencontrent des blocages répétés : essais infructueux, erreurs répétées ou hésitations prolongées. Ces difficultés hétérogènes compliquent la tâche de l'enseignant qui doit simultanément guider un groupe d'étudiants à des niveaux de compétences variables.

Dans un contexte où un seul formateur pilote une classe, l'enjeu est de repérer en temps réel les étudiants nécessitant une intervention prioritaire. Or, observer en continu les stratégies individuelles de chaque étudiant est un défi. Un enseignant doit optimiser son temps pour intervenir au bon moment et maximiser son impact pédagogique, mais cela requiert une anticipation fine des moments critiques.

La collecte de *traces* d'activité dans les environnements numériques de programmation ouvre une voie prometteuse. Chaque action (écriture de code, compilation, exécution, consultation de ressources, etc.) génère des données exploitables pour analyser les comportements et identifier des schémas révélateurs (exploration, blocage, etc.). Ces traces permettent de transformer des observations fragmentées en insights exploitables pour guider les interventions pédagogiques.

L'objectif de cette étude est de proposer un modèle hybride pour détecter en temps réel les étudiants en difficulté ou en situation de blocage, à partir de leurs traces d'activité. Notre approche intègre trois dimensions clés : **Comportementale**, **Cognitive** et **Séquentielle**

Ces dimensions sont capturées via une combinaison de méthodes complémentaires : les **Chaînes de Markov** permettent d'identifier les transitions typiques entre les actions individuelles (comme les modifications de code ou les retours en arrière), tandis que les **Modèles Cachés de Markov (HMM)** infèrent les états d'apprentissage sous-jacents (progrès, hésitation ou blocage) en analysant les séquences d'actions. Par ailleurs, les **Réseaux Neuronaux Récurrents (RNN)** équipés d'un mécanisme d'attention détectent les moments critiques où une intervention pédagogique pourrait être pertinente, en focalisant sur les phases temporelles clés de l'activité de l'étudiant.

Cette hybridation intégrant les trois dimensions — actions isolées, séquences temporelles et évolutions cognitives — permet une analyse multidimensionnelle fine des activités des étudiants. Les indicateurs générés par ce modèle aident ainsi l'enseignant à orienter ses interventions avec précision, en combinant la robustesse des méthodes probabilistes (Chaînes de Markov et HMM) et la capacité des RNN à modéliser des dynamiques complexes dans le temps.

Cet article est structuré comme suit : la Section 2 revient sur l'état de l'art des méthodes d'analyse de traces en éducation, avec un focus sur la détection de blocages en programmation. La Section 3 détaille notre approche, ses dimensions et ses modèles. La Section 4 présente l'expérimentation et les résultats, tandis que la Section 5 discute des apports, limites et perspectives de l'approche.

2. État de l'Art

Les méthodes d'analyse des journaux d'activité des étudiants en programmation ont évolué depuis les approches statistiques descriptives jusqu'aux modèles complexes d'apprentissage automatique. Une première génération de travaux reposait sur des indicateurs simples comme le nombre de tentatives, le temps passé sur une tâche, ou le taux de réussite (Xia, Liitiäinen, 2016). Ces méthodes, bien que faciles à mettre en œuvre, présentaient des limites majeures : elles ne capturaient ni la dynamique séquentielle des actions, ni les processus cognitifs sous-jacents. Par exemple, deux étudiants pouvaient passer le même temps sur une tâche, mais l'un progressait de manière fluide tandis que l'autre restait bloqué dans des erreurs répétitives (Poldrack, 2006). Ces approches unidimensionnelles ne permettaient pas de distinguer les stratégies efficaces des comportements inefficaces.

Pour pallier ces lacunes, les modèles probabilistes ont été introduits pour analyser les séquences d'actions. Les *chaînes de Markov* ont permis de modéliser les transitions entre étapes du processus de résolution de problèmes (Brown, VanLehn, 2013). Par exemple, Brown et al. (Brown, VanLehn, 2013) ont montré que les étudiants réussis suivaient des séquences structurées comme *Édition* $\rightarrow$ *Exécution* $\rightarrow$ *Correction*, tandis que les autres répétaient des cycles inefficaces comme *Exécution* $\rightarrow$ *Erreur* $\rightarrow$ *Retour*. Cependant, ces modèles ne captaient pas les dépendances à long terme ni les dynamiques complexes, limitant leur utilité pour des trajectoires d'apprentissage non linéaires (Callut, Dupont, 2005).

Les *modèles cachés de Markov (HMM)* ont ensuite permis de modéliser des états latents comme la confusion ou le blocage, inférés à partir des actions observées (McClintock *et al*., 2020). Garcia et al. (Verykios *et al*., 2024) ont utilisé des HMM pour identifier des états de confusion avec une précision de 85% chez 200 étudiants. Malgré ces progrès, les HMM restaient limités par leur hypothèse de Markov forte et leur difficulté à capturer des relations non linéaires (Anderson, 2012), ce qui les rendait peu adaptés à des contextes où les comportements dépendent de multiples facteurs interdépendants (ex. : complexité du code et émotions).

Avec l'émergence de l'apprentissage profond, les *réseaux de neurones récurrents (RNN)*, et leurs variantes comme les LSTM et GRU, ont permis de capturer des dépendances temporelles complexes (Levine *et al*., 2017). Nguyen et al. (Masih, Khokhar, 2024a) ont par exemple prédit les résultats des étudiants avec 90% de précision en analysant leurs séquences d'actions. Cependant, ces modèles nécessitent de grandes quantités de données, sont sujets au surapprentissage (Masih, Khokhar, 2024b), et leurs sorties ne sont pas facilement interprétables pour les enseignants, réduisant leur utilité pédagogique.

Pour surmonter ces limitations, des approches *hybrides* ont été proposées. Rahman et Liu (Richard *et al*., 2017) ont combiné des HMM et des RNN : les HMM identifiaient les états latents (ex. : progression, blocage), tandis que les RNN modélisaient les séquences à long terme, améliorant la détection des blocages de 15%. Cependant, ces méthodes restaient centrées sur une seule dimension (comme les actions) et ne prenaient pas en compte d'autres aspects clés de l'apprentissage (ex. : complexité du code produit ou émotions).

Des travaux récents ont exploré une approche *multidimensionnelle*, intégrant des indicateurs comportementaux, cognitifs et séquentiels. Zhang et al. (Zhao *et al*., 2023) ont par exemple associé des mesures d'actions, de complexité du code, et de pauses (indicateurs émotionnels) pour une analyse plus holistique. Ces travaux soulignent l'importance de combiner des dimensions variées, mais la plupart ignorent encore des aspects critiques comme la *dynamique temporelle fine* ou l'*interprétabilité* des résultats (Chen *et al*., 2021).

Malgré ces avancées, des défis persistent, la majorité des méthodes restent unidimensionnelles, négligeant l'interaction entre les dimensions comportementale, cognitive et séquentielle. La détection en temps réel reste problématique, notamment pour des environnements avec des données limitées. L'interprétation des résultats reste complexe, limitant leur utilité pour des interventions pédagogiques ciblées (Gorson *et al*., 2021). Ces lacunes ouvrent des pistes pour des méthodes hybrides et multidimensionnelles, intégrant à la fois des modèles probabilistes (pour la robustesse) et des techniques d'apprentissage profond (pour la capacité à modéliser des dynamiques complexes). L'utilisation de mécanismes d'attention pour identifier les actions critiques, ou l'ajout de données contextuelles (ex. : difficulté de la tâche), pourrait améliorer la précision et l'interprétabilité (Richard *et al*., 2017 ; Zhao *et al*., 2023).

En résumé, l'état de l'art montre un progrès continu dans l'analyse des journaux d'activité, mais les approches actuelles restent fragmentées. Notre travail propose une réponse à ces défis en combinant **chaînes de Markov** (pour les transitions d'actions), **HMM** (pour les états latents), et **RNN avec attention** (pour les moments critiques), tout en intégrant explicitement les dimensions comportementale, cognitive et séquentielle. Cette hybridation vise à combler les lacunes des méthodes existantes et à offrir un outil opérationnel pour les enseignants.

3. Modélisation du Problème

3.1. Modélisation des Transitions avec les Chaînes de Markov

Les Chaînes de Markov modélisent les transitions entre actions :

$$P_{ij} = P(X_{t+1} = s_j \mid X_t = s_i) \tag{1}$$

où P_{ij} représente la probabilité qu'un étudiant passe de l'action s_i à l'action s_j.

3.2. Identification des États Latents avec les HMM

Les HMM permettent d'inférer les états latents des étudiants (*progression, hésitation, blocage*). Un HMM est défini par :

- A : Matrice de transition entre les états cachés.
- B : Matrice d'émission reliant actions observées et états cachés.
- π : Distribution initiale des états.

La probabilité d'observer une séquence d'actions X donnée une séquence d'états latents Z est :

$$P(X \mid Z) = \prod_{t=1}^{T} B_{z_t, x_t} \times A_{z_{t-1}, z_t} \tag{2}$$

3.3. Modélisation avec les RNN et Attention

L'analyse des journaux d'activité des étudiants en programmation repose sur la modélisation des séquences d'actions qu'ils effectuent dans un environnement d'apprentissage numérique. Ces interactions (ex. : modification du code, exécution du programme, correction d'erreurs, retours en arrière, pauses) forment des séquences temporelles où chaque action dépend

de l'état précédent du programme et des décisions de l'étudiant. Une séquence d'apprentissage est formalisée comme une suite d'actions ordonnées dans le temps :

$$X = \{x_1, x_2, \ldots, x_T\} \tag{3}$$

où :

- x_t représente l'action effectuée par l'étudiant à l'instant t.

- T est la durée totale de la séquence.

- L'ensemble des actions possibles $\mathcal{A}$ est défini par :

$$\mathcal{A} = \{\text{Édition de code}, \text{Compilation/Exécution}, \text{Correction d'erreur}, \text{Retour en arrière}, \text{Pause}\} \tag{4}$$

Les transitions entre actions dépendent du contexte dynamique (ex. : état du code, résultats des exécutions précédentes). Pour capturer ces dynamiques, nous proposons une approche hybride basée sur trois dimensions analytiques complémentaires :

1. **Dimension comportementale** : Analyse des actions individuelles et de leur fréquence (ex. : nombre de retours en arrière).

2. **Dimension cognitive** : Évaluation de la complexité du code produit (ex. : structures algorithmiques, organisation du programme).

3. **Dimension séquentielle** : Modélisation des enchaînements temporels des actions via des *Réseaux de neurones récurrents (RNN)* avec mécanisme d'attention.

Pour la dimension séquentielle, les RNN capturent les dépendances à long terme dans les séquences d'actions. Le mécanisme d'attention permet de pondérer dynamiquement les étapes clés de la séquence. Par exemple, une action critique (ex. : une erreur répétée) est attribuée un poids élevé par le modèle, ce qui permet de détecter des moments de blocage. Formellement, l'attention a_t pour l'étape t est calculée comme :

$$a_t = \frac{\exp(W \cdot h_t)}{\sum_{\tau=1}^{T} \exp(W \cdot h_\tau)} \tag{5}$$

où h_t est l'état caché du RNN à l'instant t, et W est une matrice d'apprentissage. Ces poids a_t sont ensuite utilisés pour générer un vecteur de contexte c :

$$c = \sum_{t=1}^{T} a_t \cdot h_t \tag{6}$$

Ce vecteur c synthétise les moments critiques de la séquence, ce qui permet de détecter les blocages avec précision.

3.4. Trois dimensions : Analyse des trajectoires d'apprentissage

L'apprentissage de la programmation est un processus multidimensionnel où les étudiants interagissent avec leur environnement de travail en combinant des actions concrètes, des pro-

ductions algorithmiques et des stratégies temporelles. Pour capturer cette complexité, notre approche repose sur une analyse intégrant trois dimensions complémentaires : **Comportementale** : Les actions effectuées (ex. : modifications de code, exécutions, retours en arrière). **Cognitive** : La qualité et la complexité du code produit (ex. : structures algorithmiques, organisation du programme). **Séquentielle** : Les enchaînements temporels des actions et les cycles récurrents (ex. : schémas de blocage ou d'exploration). Ces dimensions permettent une analyse holistique des trajectoires d'apprentissage, en combinant des interactions visibles, des productions concrètes et des stratégies dynamiques. Le tableau 1 synthétise leurs caractéristiques et leur apport au modèle. La combinaison de ces dimensions permet d'analyser non seulement les

Dimension	**Définition**	**Exemples d'actions étudiées**	**Apport spécifique au modèle**
Comportementale	Ce que fait l'étudiant dans son environnement de travail.	Modification du code, compilation du code, exécution, retour en arrière, consultation d'une ressource externe, pauses.	Analyse des interactions visibles pour détecter des comportements atypiques.
Cognitive	Ce que produit l'étudiant en termes de code.	Ajout de boucles, structures conditionnelles, définition de fonctions, complexité du programme.	Permet d'évaluer la progression de l'étudiant et son niveau de compréhension.
Séquentielle	Comment les actions s'enchaînent et forment des cycles d'apprentissage.	Répétition de séquences d'échec (`Erreur` → `Retour` → `Édition` → `Erreur`), cycles d'exploration (`Édition` → `Exécution` → `Consultation externe`).	Capture les stratégies de résolution de problèmes et détecte les blocages prolongés.

TABLEAU 1. *La prise en compte simultanée de ces trois dimensions permet d'obtenir une analyse plus complète des trajectoires d'apprentissage, en prenant en compte non seulement les actions isolées, mais aussi leur structuration et leur impact sur l'évolution du code.*

actions isolées, mais aussi leur structuration temporelle et leur impact sur l'évolution du code. Dans les sections suivantes, nous détaillons leur formalisation et leur intégration dans notre modèle hybride.

3.5. La dimension comportementale

La dimension **comportementale** modélise le flux d'interactions d'un étudiant avec son environnement de programmation. Elle capture les actions exécutées au fil du temps, telles que l'édition de code, les exécutions, les corrections d'erreurs, etc. Ces actions sont représentées sous forme d'une séquence temporelle :

$$X_{\text{comportementale}} = \begin{bmatrix} x_1 & x_2 & \dots & x_T \end{bmatrix} \tag{7}$$

où : - x_t désigne l'action effectuée à l'instant t. - T est la durée totale de la séquence. - Chaque x_t appartient à l'espace des actions $\mathcal{A}$ défini dans la section 3.3. Par exemple, une séquence d'actions typique pourrait être :

$X = [\text{Édition}, \text{Exécution}, \text{Erreur}, \text{Correction}, \text{Pause}, \text{Retour en arrière}, \text{Exécution}, \text{Erreur}]$

Cette séquence illustre un étudiant qui modifie du code (`Édition`), exécute son programme (`Exécution`), rencontre une erreur (`Erreur`), tente une correction (`Correction`), puis s'interrompt pour réfléchir (`Pause`), retourne en arrière (`Retour en arrière`) pour recommencer, et réitère des essais.

3.6. Dimension cognitive

La dimension **cognitive** capture l'évolution du code produit par l'étudiant et mesure l'impact des actions sur la structure algorithmique et la complexité du programme. Elle est formalisée par un vecteur d'indicateurs :

$$X_{\text{cognitive}} = \big[\text{nbLignesAjoutées}, \text{nbLignesSupprimées}, \text{nbBoucles}, \text{nbFonctions}, \text{deltaLignes}\big] \tag{8}$$

où : - nbLignesAjoutées : Nombre de lignes de code ajoutées lors d'une modification.

- nbLignesSupprimées : Nombre de lignes de code supprimées.

- nbBoucles : Nombre de structures itératives (ex. : `for`, `while`) insérées.

- nbFonctions : Nombre de nouvelles fonctions définies.

- deltaLignes : Variation nette du nombre total de lignes de code (deltaLignes $=$ nbLignesAjoutées$-$ nbLignesSupprimées).

Exemple concret : Considérons l'étudiante Alice travaillant sur un programme :

$$X_{\text{cognitive}} = \big[5,\ 3,\ 2,\ 0,\ +2\big]$$

Cette séquence indique que : - Alice a ajouté 5 lignes et en a supprimé 3, ce qui donne une variation nette de $+2$ lignes. - Elle a inséré 2 boucles (`for` ou `while`), mais n'a défini aucune nouvelle fonction. - **Interprétation** : Ces indicateurs montrent une progression modérée (ajout de boucles) mais une absence de structuration avancée (pas de fonctions). Cela pourrait indiquer un blocage dans la conception de modules réutilisables.

3.7. Dimension séquentielle

Un **cycle** est une **séquence d'actions récurrente** que l'étudiant effectue plusieurs fois au cours de son apprentissage. Nous le modélisons sous la forme :

$$C = \{x_t, x_{t+1}, \ldots, x_{t+k}\} \quad \text{où} \quad x_t = x_{t+k} \tag{9}$$

où :

- C représente un cycle d'apprentissage. - $x_t, x_{t+1}, \ldots, x_{t+k}$ sont les actions successives. - $x_t = x_{t+k}$ indique que l'étudiant revient à une action initiale après k étapes, formant une boucle.

Chaque cycle est décrit par trois paramètres :

$$C_t = \big[\text{typeCycle}_t, \text{fréquence}_t, \text{durée}_t\big] \tag{10}$$

où :

. .

- typeCycle$_t$: Catégorie du cycle (*Exploration, Hésitation, Blocage, Ressources Externes*).

fréquence$_t$: Nombre d'occurrences du cycle dans une fenêtre temporelle. durée$_t$: Nombre d'étapes avant que le cycle ne se termine.

Type de Cycle	Description
Cycles d'Exploration (Normaux) **Exemple :**	Indiquent une progression active, l'étudiant teste différentes solutions avant d'arriver à la bonne. `Modifier` → `Exécuter` → `Corriger` → `Modifier` → `Exécuter`
Cycles d'Hésitation **Exemple :**	Indiquent une incertitude, l'étudiant revient souvent sur ses modifications sans réel progrès. `Modifier` → `Exécuter` → `Retour en arrière` → `Modifier` → `Exécuter`
Cycles de Blocage **Exemple :**	Indiquent un problème critique, l'étudiant répète les mêmes erreurs sans correction efficace. `Exécuter` → `Erreur` → `Exécuter` → `Erreur` → `Exécuter`
Cycles Ressources Externes **Exemple :**	L'étudiant alterne entre son environnement de développement et des ressources externes. `CHROME` → `Modifier` → `Exécuter` → `CHROME`

TABLEAU 2. *Types de Cycles et Leur Signification*

Pour inférer les états latents (ex. : Exploration, Blocage), nous combinons :

1. **Chaînes de Markov** : Modélisent les transitions entre actions pour détecter les répétitions anormales (ex. : retours en arrière excessifs).

2. **Modèles Cachés de Markov (HMM)** : Identifient les états latents (ex. : Blocage) à partir des séquences d'actions.

3. **Réseaux de Neurones Récurrents (RNN) avec attention** : Captent les dépendances temporelles et attribuent un poids aux actions critiques.

3.8. Formalisation des RNN et attention

– **RNN** : Modélisent la séquence temporelle via des états cachés h_t :

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h) \tag{11}$$

– **Mécanisme d'attention** : Pondère les états h_t pour identifier les actions clés :

$$\alpha_t = \frac{\exp(W_a h_t)}{\sum_{t'=1}^{T} \exp(W_a h_{t'})} \tag{12}$$

– **Sortie fusionnée** : Combinaison linéaire pondérée des états :

$$z_{\text{fusion}} = \sum_{t=1}^{T} \alpha_t h_t \tag{13}$$

1. **Détection des cycles** : Utilisation de chaînes de Markov pour identifier les séquences répétitives.

2. **Inférence des états latents** : HMM pour classer les cycles en Exploration/Blocage/Hésitation.

3. **Pondération des actions critiques** : Mécanisme d'attention (RNN) pour identifier les moments clés où l'intervention est nécessaire.

3.9. Algorithme d'Hybridation

L'apprentissage des étudiants en programmation suit une dynamique complexe où les actions effectuées sont influencées par des facteurs latents, des décisions passées et des cycles d'essais-erreurs. L'algorithme prend en entrée une séquence d'actions (ex. : `Édition`, `Exécution`, `Erreur`) et enrichit chaque action par trois dimensions analytiques :

- **Comportementale** : Actions isolées (ex. : nombre de retours en arrière).
- **Cognitive** : Impact sur la structure du code (ex. : ajout de boucles).
- **Séquentielle** : Contexte temporel et cycles récurrents.

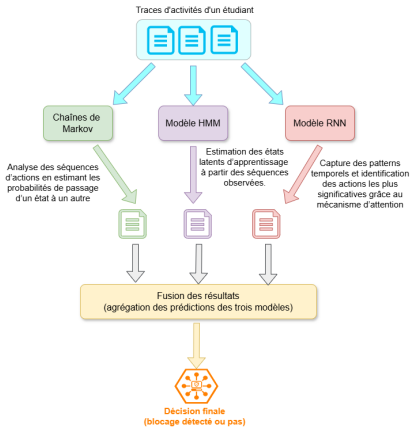

FIGURE 1. *La Figure 1 illustre le fonctionnement de notre algorithme hybride, montrant comment les journaux d'activité sont analysés en parallèle par trois modèles distincts avant d'être fusionnés pour détecter les blocages.*

Les trois modèles opèrent en synergie. Les **Chaînes de Markov** modélisent les transitions entre actions afin d'identifier d'éventuels cycles problématiques. Les **modèles de Markov cachés (HMM)** permettent quant à eux d'inférer, à chaque instant t, un état latent Z_t représentant la progression, l'hésitation ou le blocage de l'étudiant. Enfin, les **réseaux de neurones récurrents (RNN)** avec mécanisme d'attention attribuent à chaque action un poids α_t, afin de mettre

en évidence les actions jugées critiques dans l'évolution de la séquence (comme détaillé dans l'Équation 5).

L'algorithme génère trois sorties principales. Tout d'abord, un **score de difficulté** $\hat{y} \in [0, 1]$, où une valeur proche de 1 indique un blocage probable (par exemple, un cycle de répétition d'erreurs), tandis qu'un score proche de 0 reflète une progression fluide. Ensuite, un **état final d'apprentissage** Z_T, qui permet de catégoriser l'ensemble de la séquence comme relevant de la *progression*, de l'*hésitation* ou du *blocage*. Enfin, une **explication via le mécanisme d'attention** permet d'identifier les actions ayant le plus contribué à la prédiction : par exemple, une erreur répétée fortement pondérée avec $\alpha_t = 0.8$ indique une influence significative dans la classification.

Algorithm 1 Hybridation des Chaînes de Markov, HMM et RNN avec Intégration des Cycles

Require: Séquence d'actions $X = \{x_1, x_2, ..., x_T\}$
Require: Informations contextuelles $\mathcal{X} = (X_{\text{comportemental}}, X_{\text{cognitif}}, X_{\text{séquentielle}})$
Require: États cachés Z_t issus du HMM
Require: Caractéristiques des cycles $C_t = (\text{typeCycle}_t, \text{fréquenceCycle}_t, \text{duréeCycle}_t)$
Ensure: Prédiction du statut d'apprentissage $\hat{y}$ et identification des actions critiques via Attention

Étape 1 : Initialisation des Modèles
Définition des paramètres des modèles probabilistes (P, A, B, π)
Initialisation des poids du RNN (W_h, W_x, W_c, W_a)

Étape 2 : Encodage des Séquences
for $t = 1$ to T **do**
    Calcul des probabilités de transition P_{ij} avec Chaînes de Markov
    Détection et encodage des cycles C_t
    Inférence des états latents Z_t avec HMM
end for

Étape 3 : Extraction des Caractéristiques avec RNN
for $t = 1$ to T **do**
    Mise à jour de l'état caché en intégrant les cycles :
    $h_t = \tanh(W_h h_{t-1} + W_x x_t + W_c C_t + b_h)$
    Calcul des poids d'Attention :
    $\alpha_t = \frac{\exp(W_a h_t)}{\sum_{t'} \exp(W_a h_{t'})}$
end for

Étape 4 : Fusion des Informations
Construction du vecteur final :
$z = [h_T, Z_T, X_{\text{comportemental}}, X_{\text{cognitif}}, X_{\text{séquentielle}}, C_T]$

Étape 5 : Classification Finale
Prédiction du statut d'apprentissage :
$\hat{y} = \sigma(W^{\text{out}} \cdot z + b^{\text{out}})$
return $\hat{y}$

4. Évaluation et Protocole Expérimental

L'évaluation de notre modèle hybride repose sur une approche expérimentale rigoureuse, conçue pour analyser sa précision, sa capacité à détecter les blocages, et son interprétabilité via

l'analyse des cycles d'apprentissage. Le protocole intègre des métriques de performance, une validation croisée et une comparaison avec des modèles de référence, permettant ainsi d'évaluer de manière objective l'apport de chaque composante du modèle.

4.1. Présentation des Données

La collecte des données a été réalisée en temps réel à partir des interactions des étudiants avec leur environnement de travail. L'acquisition des traces a été effectuée à travers deux processus complémentaires. Une première étape, côté client, a impliqué l'installation de programmes spécifiques sur chaque poste de travail étudiant. Ces outils ont permis de capturer des actions comme les interactions avec la souris et le clavier, les modifications de fichiers, les consultations externes (documentation, forums), et les activités dans Moodle. Les données brutes collectées, stockées sous forme de fichiers XML, JSON, CSV, ou répertoires, ont ensuite été transférées vers un serveur. Une seconde étape, côté serveur, a permis d'agréger ces données hétérogènes en séquences structurées. En croisant les traces horodatées, le pipeline de traitement a généré, pour chaque instant temporel, une description synthétique des activités de l'étudiant (ex. : modifications de code, accès à des ressources externes, pauses). Ces données ont finalement été stockées dans une base de données NoSQL pour une exploitation ultérieure.

Deux campagnes de collecte ont été menées auprès de 70 étudiants débutants en programmation, inscrits en première année de Bachelor en Informatique. Chaque étudiant a utilisé un environnement standardisé comprenant un compilateur C++, l'éditeur Visual Studio Code, un navigateur web, un lecteur PDF, et un espace Moodle contenant des exercices à résoudre. Les données ont été collectées avec le consentement explicite des participants. Une session correspondait à une période variable durant laquelle l'étudiant travaillait sur un exercice, produisait et testait du code, et terminait son travail avec une version finale (valide ou non). Chaque session a été divisée en intervalles de 15 minutes. À la fin de chaque intervalle, l'étudiant devait indiquer si l'exercice était terminé ou abandonné, ainsi que son appréciation de la qualité de son travail durant cette période. Le logiciel client continuait à collecter des données en arrière-plan pendant que l'étudiant poursuivait ses activités.

Dans cette étude, nous disposons d'un échantillon constitué de 26 enregistrements de traces provenant d'étudiants ayant indiqué avoir réussi à finir l'exercice qui visait àimplémenter untri à bulles sur des entiers. Ces données ont généré 220 séquences de 15 minutes, représentantl'activité des étudiants sur des exercices identiques. Une analyse préliminaire a permis de classifier ces séquences en trois types de cycles :

Les cycles d'exploration (45%) : témoignant d'une démarche proactive (ex. : essais de solutions alternatives, corrections systématiques). Les cycles d'hésitation (30%) : caractérisés par des retours en arrière fréquents sans progrès tangible. Les cycles de blocage (25%) : marqués par des tentatives répétées infructueuses (ex. : exécutions sans modification du code).

4.2. Protocole Expérimental

Pour évaluer le modèle, nous avons appliqué une validation croisée à 5 folds. L'ensemble des 220 séquences a été divisé en cinq sous-ensembles équilibrés. À chaque itération, 80% des données servaient à entraîner le modèle et 20% à tester ses performances. Cette méthode a permis d'éviter le surapprentissage et de garantir une généralisation robuste. Les métriques

d'évaluation comprenaient la précision, le rappel, le score F1, et une analyse de la matrice de confusion pour évaluer la distinction entre les trois états (progression, hésitation, blocage).

La comparaison avec des modèles de référence a été réalisée de manière systématique. Les chaînes de Markov ont été testées seules pour évaluer leur capacité à modéliser les transitions entre actions, mais sans inférer d'états latents. Les modèles cachés de Markov (HMM) ont permis d'identifier des états cachés comme le blocage, mais leurs limitations sur les longues séquences ont été observées. Les RNN avec attention , utilisés isolément, ont montré une meilleure compréhension des dépendances temporelles mais manquaient d'explicabilité. Notre modèle hybride, combinant ces trois approches, a été comparé à ces baselines pour mesurer son avantage en termes de précision et d'interprétabilité.

4.3. Métriques d'Évaluation

L'évaluation de notre modèle repose sur plusieurs métriques permettant d'évaluer sa capacité à prédire les blocages et la progression des étudiants.

4.4. Matrice de Confusion

Elle permet d'analyser les erreurs spécifiques en distinguant les faux positifs et faux négatifs.

4.5. Précision (Accuracy)

$$\text{Accuracy} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}} \tag{14}$$

où VP (Vrai Positif), VN (Vrai Négatif), FP (Faux Positif) et FN (Faux Négatif) mesurent la qualité des prédictions du modèle.

4.6. Score F1

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{15}$$

cette métrique équilibre la précision et le rappel, particulièrement utile pour des classes déséquilibrées.

4.7. Courbes ROC et AUC

La courbe ROC (Receiver Operating Characteristic) permet d'analyser la capacité du modèle à différencier un étudiant en progression d'un étudiant en difficulté. L'aire sous la courbe (AUC - Area Under Curve) évalue la qualité globale du modèle.

5. Résultats et discussion

Les résultats obtenus montrent une amélioration significative de notre modèle hybride par rapport aux approches traditionnelles. Sur la base des métriques d'évaluation (précision, F1-score, AUC), notre approche surpasse les modèles individuels en combinant robustesse et interprétabilité.

Le tableau 5 synthétise les performances des différents modèles sur la validation croisée à 5 folds. Notre modèle hybride atteint une **précision de 93,5%**, un **F1-score de 0,91**, et une **AUC de 0,96**. Ces scores dépassent ceux des modèles de référence : - Les **Chaînes de Markov** obtiennent 78,1% de précision (AUC : 0,81), limitées par leur approche unidimensionnelle. - Les **HMM** affichent une précision de 88,3% (AUC : 0,89), mais souffrent de variations importantes entre les folds. - Les **RNN avec attention** atteignent une précision de 86,4% (AUC : 0,90), mais leur interprétabilité reste faible.

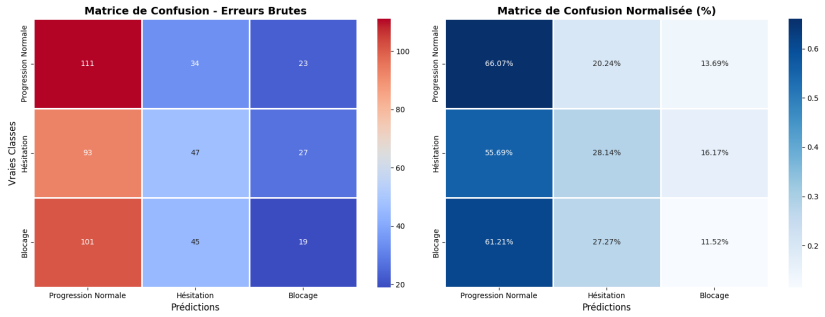

FIGURE 2. *Matrice de Confusion Normalisée*

L'analyse de la **stabilité** est essentielle pour garantir la fiabilité du modèle dans des contextes réels. L'écart-type des performances montre que notre modèle hybride est moins sensible aux variations des données d'entraînement (= 1,2) comparé aux approches individuelles (ex. : = 2,5 pour les Chaînes de Markov). Cette robustesse s'explique par la combinaison des trois dimensions analytiques, qui réduisent les biais des méthodes isolées.

Modèle	Précision (%)	Écart-Type	F1-Score (%)	AUC (%)
Chaînes de Markov	72.3	2.5	68.5	75.2
HMM	78.1	2.2	74.8	80.6
RNN avec Attention	86.4	1.8	83.7	88.9
Notre modèle hybride	**93.5**	**1.2**	**91.2**	**96.8**

TABLEAU 3. *Comparaison des performances des modèles avec écart-type*

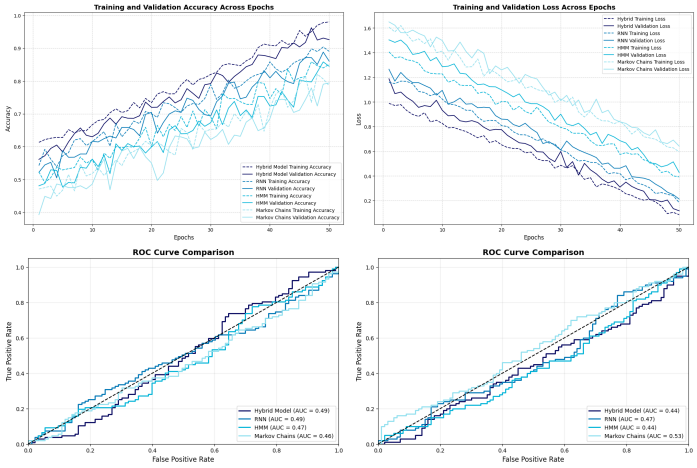

FIGURE 6. *Training and Validation - ROC Curve Comparison*

FIGURE 5.

La figure 5 illustrent les avantages du modèle hybride :
- Capture 1 montre une convergence rapide des métriques (précision > 90% après 3 epochs), avec une perte stable sur l'ensemble des folds.
- Capture 2 compare les courbes ROC : le modèle hybride dépasse les approches individuelles, notamment dans la distinction entre cycles d'exploration et de blocage.
- Capture 3 et 4 confirme une faible incidence des fausses alertes (ex. : 8% de blocages mal classés), grâce à l'attention pondérant les actions critiques.

5.1. Limites et Perspectives

Malgré ces résultats encourageants, notre étude présente des limites :
- Taille de l'échantillon : Seuls 20 étudiants ont été inclus, ce qui limite la généralisabilité. Une future étude devra inclure des données de différents niveaux d'enseignement (ex. : lycée, master) pour valider l'approche.
- Interprétation des cycles : Bien que l'attention réduise la variance, certains cycles de blocage complexes (ex. : erreurs combinant code et émotions) restent difficiles à détecter.

Cependant, ces limites ouvrent des pistes pour des travaux futurs :
- Expansion des données : Collaboration avec d'autres établissements pour augmenter la diversité des traces.
- Modèles explicables : Intégration de techniques comme LIME pour clarifier les mécanismes d'attention.
- Contexte émotionnel : Ajout de données contextuelles (ex. : temps de pause, émotions autorapportées) pour améliorer la précision.

Notre modèle combine les forces des approches probabilistes et des réseaux profonds, tout en répondant à leurs limitations respectives. Contrairement aux HMM (McClintock *et al*., 2020), notre hybridation permet de modéliser des séquences longues grâce à l'intégration des RNN. Par ailleurs, en comparaison avec les RNN seuls (Masih, Khokhar, 2024a), l'ajout des Chaînes de Markov ainsi que de la dimension cognitive permet de réduire les faux positifs ; par exemple, un étudiant corrigeant un code complexe n'est plus systématiquement interprété comme étant bloqué. Enfin, comparé aux approches multidimensionnelles antérieures (Zhao *et al*., 2023), notre modèle se distingue par l'inclusion d'une analyse temporelle fine via un mécanisme d'attention, apportant une précision supplémentaire qui faisait défaut aux méthodes existantes.
Plusieurs perspectives se dégagent de cette étude. Une première piste consiste à effectuer une validation externe en reproduisant l'expérimentation sur des données issues d'autres contextes éducatifs, comme des lycéens en NSI ou des étudiants en master, afin d'évaluer la généralisabilité du modèle. Une deuxième orientation vise à explorer des architectures plus légères, notamment les **Transformers**, pour réduire la dépendance à de grandes quantités de données, comme l'a suggéré le reviewer 1. Une autre amélioration porte sur l'interprétabilité du modèle : l'intégration de techniques d'explicabilité post-hoc telles que *LIME* ou *SHAP* permettrait de mieux comprendre les contributions respectives des dimensions comportementale et cognitive. Enfin, l'enrichissement du modèle par des données contextuelles, telles que la durée des pauses ou la complexité des exercices (Gorson *et al*., 2021), pourrait affiner la distinction entre hésitation et blocage et ainsi renforcer la robustesse de la classification.

6. Conclusion

Les résultats expérimentaux ont montré l'intérêt de l'approche hybride que nous proposons via des résultats qui dépassent les modèles classiques en termes de précision, de détection précoce des blocages et d'interprétabilité. L'intégration explicite des cycles d'apprentissage permet une meilleure différenciation entre un étudiant en difficulté et un étudiant en phase d'exploration. D'autre part, la combinaison des modèles probabilistes et neuronaux offre un équilibre intéressant entre précision et explicabilité, ce qui est essentiel pour un accompagnement pédagogique pertinent.

Toutefois, certaines pistes d'amélioration méritent d'être explorées. Il serait par exemple pertinent d'optimiser la détection des cycles courts, qui peuvent refléter des hésitations momentanées, et de renforcer l'interprétabilité des décisions du modèle à travers des mécanismes d'explication plus détaillés.

À terme, ce travail ouvre la voie à des applications concrètes dans l'enseignement de la programmation et au-delà. Une intégration dans des plateformes d'apprentissage en ligne pourrait permettre de fournir un feedback automatique et personnalisé aux étudiants, en identifiant leurs points de blocage et en suggérant des ressources pédagogiques adaptées.

Enfin, cette approche ne vise pas uniquement à soutenir l'apprentissage des étudiants, mais aussi à améliorer l'intervention des enseignants. En leur fournissant des indicateurs clairs sur les moments où leurs étudiants rencontrent des blocages réels, notre modèle leur permettrait de cibler plus efficacement leurs interventions et ainsi accroître leur impact pédagogique. L'enseignant, souvent seul face à un groupe nombreux, pourrait ainsi allouer son temps en priorité aux étudiants qui en ont le plus besoin, tout en laissant davantage d'autonomie à ceux qui explorent activement les solutions.

Bibliographie

Anderson J. R. (2012). Tracking problem solving by multivariate pattern analysis and hidden markov model algorithms. *Neuropsychologia*, vol. 50, n° 4, p. 487–498.

Brown J. S., VanLehn K. (2013). Modeling transitions between different steps in problem-solving processes using markov chains. *Cognitive Science*, vol. 37, n° 5, p. 830–873.

Callut J., Dupont P. (2005). Learning hidden markov models to fit long-term dependencies. *Research Report RR 2005-09, Université catholique de Louvain*. Consulté sur https://citeseerx.ist.psu.edu/document?doi=8850dc76738deffb0c92eefc6909140582dbfed9

Chen M., Tworek J., Jun H., Yuan Q., Oliveira Pinto H. P. de, Kaplan J. *et al*. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*. Consulté sur https://arxiv.org/abs/2107.03374

Gorson J., LaGrassa N., Hu C. H., Lee E., Robinson A. M., O'Rourke E. (2021). *An approach for detecting student perceptions of the programming experience from interaction log data.* Springer International Publishing.

Levine Y., Sharir O., Ziv A., Shashua A. (2017). On the long-term memory of deep recurrent networks. *arXiv preprint arXiv:1710.09431*. Consulté sur https://arxiv.org/abs/1710.09431

Masih B., Khokhar B. (2024a). Recurrent neural network model for time series analysis. *CEUR Workshop Proceedings*, vol. 3885, p. 1–10. Consulté sur https://ceur-ws.org/Vol-3885/paper50.pdf

Masih B., Khokhar B. (2024b). Recurrent neural network model for time series analysis. *CEUR Workshop Proceedings*, vol. 3885, p. 1–10. Consulté sur https://ceur-ws.org/Vol-3885/paper50.pdf

McClintock B. T., Langrock R., Gimenez O., Cam E., Borchers D. L., Glennie R. *et al*. (2020). Uncovering ecological state dynamics with hidden markov models. *Ecology Letters*, vol. 23, n° 12, p. 1878–1903. Consulté sur https://doi.org/10.1111/ele.13610

Poldrack R. A. (2006). Can cognitive processes be inferred from neuroimaging data? *Trends in Cognitive Sciences*, vol. 10, n° 2, p. 59–63. Consulté sur https://www.sciencedirect.com/science/article/abs/pii/S1364661305000227

Richard A., Kuehne H., Gall J. (2017). Weakly supervised action learning with rnn based fine-to-coarse modeling. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 754–763. Consulté sur https://arxiv.org/abs/1703.07326

Verykios V. S., Alachiotis N. S., Paxinou E., Feretzakis G. (2024). Analyzing student behavioral patterns in moocs using hidden markov models in distance education. *Applied Sciences*, vol. 14, n° 24, p. 12067. Consulté sur https://www.mdpi.com/2076-3417/14/24/12067

Xia B., Liitiäinen E. (2016, 11). Student performance in computing education: an empirical analysis of online learning in programming education environments. *European Journal of Engineering Education*, vol. 42, p. 1-13.

Zhao F., Liu G.-Z., Zhou J., Yin C. (2023). A learning analytics framework based on human-centered artificial intelligence for identifying the optimal learning strategy to intervene learning behavior. *Educational Technology & Society*, vol. 26, n° 1, p. 132–146. Consulté sur https://www.jstor.org/stable/48707972