UNIVERSITÉ DE MONTRÉAL

FORAGE DE DONNÉES POUR LA DÉTECTION D'UN ÉTAT DE BLOCAGE DE L'APPRENANT DANS LE CADRE DU SYSTÈME TUTORIEL INTELLIGENT QED-TUTRIX

JEAN-PHILIPPE CORBEIL
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

FORAGE DE DONNÉES POUR LA DÉTECTION D'UN ÉTAT DE BLOCAGE DE L'APPRENANT DANS LE CADRE DU SYSTÈME TUTORIEL INTELLIGENT QED-TUTRIX

présenté par : CORBEIL Jean-Philippe
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. ALOISE Daniel, Ph. D., président
M. GAGNON Michel, Ph. D., membre et directeur de recherche
M. RICHARD Philippe R., Ph. D., membre et codirecteur de recherche
M. DESMARAIS Michel C., Ph. D., membre

DÉDICACE

À la mémoire de Lise

REMERCIEMENTS

Je remercie mes directeurs de recherche, professeur Michel Gagnon et professeur Philippe R. Richard, pour leurs appuis exemplaires, leurs intuitions des plus intéressantes et leur dynamisme. Sans eux, ce mémoire n'aurait pas lieu d'être et je n'en ressors que d'autant plus grand, comme si je m'étais tenu sur les épaules de géants.

Je donne tous mes respects à l'équipe du laboratoire Turing : Nicolas Leduc, Michèle Tessier-Baillargeon, Ludovic Font, Sébastien Cyr et professeure Fabienne Venant. Ils sont des experts hors pair et des amis. Ils amènent une frénésie qui génère des idées qui révolutionneront l'éducation des mathématiques au Québec de même que l'avancement de l'intelligence artificielle appliquée à l'enseignement, j'en suis convaincu !

Je remercie aussi le FQRNT et le CRSH pour leurs subventions qui nous permettent de prendre le temps de bien se poser sur nos questions de recherche. Leur soutien financier démontre de leur grand intérêt pour nos travaux.

RÉSUMÉ

L'état de blocage est le moment où un apprenant, en pleine résolution de problème sur un système tutoriel intelligent, a besoin d'une intervention tutorielle pour poursuivre sa résolution. Dans ce mémoire, des modèles probabilistes seront développés pour détecter les états de blocage d'un apprenant qui résout un problème sur le système tutoriel intelligent en mathématiques QED-Tutrix. La méthodologie inclut deux expérimentations avec une version modifiée de QED-Tutrix pour recueillir des séquences d'actions associées à un état de blocage ou de non-blocage. Dans ces ensembles de données, des états de blocage ont été observés à partir des fréquences d'actions et des distributions de sous-séquences. Quatre modèles probabilistes ont été développés en tout : le modèle de processus de fréquence d'actions, le modèle bayésien en sous-séquences d'actions, le modèle du réseau de neurones convolutif et le modèle hybride. Ce dernier surpasse les autres avec un score F_1 de 80,4 % pour la classification des états de blocage sur l'ensemble d'entraînement et 77,3 % sur l'ensemble test.

L'application de cette recherche mène directement à l'amélioration de la machine à états de QED-Tutrix dans son interaction avec l'apprenant. Elle aboutit aussi sur une deuxième phase de travaux de recherche durant laquelle le développement d'interventions tutorielles ciblées est approché. Puisqu'il est possible d'identifier les moments de blocage de l'apprenant avec une bonne précision, il faut à présent concevoir des algorithmes pouvant comprendre le contexte du blocage et pouvant intervenir en conséquence. En ce qui concerne l'amélioration des performances des modèles, l'incorporation de l'historique des blocages dans les modèles probabilistes est à considérer en plus d'une considération du contexte mathématique.

ABSTRACT

A blocking state is a cognitive state in which a student cannot make any progress toward finding a solution to a problem. In this research, we present the development of probabilistic models to detect a blocking state while solving a Canadian high school-level problem in Euclidean geometry on an intelligent tutoring system. Our methodology includes an experimentation with a modified version of QED-Tutrix, an intelligent tutoring system, which was used to gather labelled datasets composed of sequences of mouse and keyboard actions. We observed blocking states in this dataset from subsequence distributions and frequency of states. Using a probabilistic framework, we developed four predicting models: an action-frequency model, a subsequence-detection model, a 1D convolutional neural network model and an hybrid model. The hybrid model outperforms the others with a F_1 score of 80.4 % on classification of blocking state on training set. It performs 77.3 % on test set.

The applications of this research lead to an upgrade of QED-Tutrix internal finite-state machine for its interactions with the learner. Also, this research opens a second research stage, in which targeted tutorial interventions in QED-Tutrix can be developed. This can be achieved with an algorithm that understands the context of intervention and that is able to help precisely the learner. In order to get better performances from the current models, the history of the previous blocking states needs to be incorporated. Moreover, the mathematical concepts used by the learner can be integrated.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE DES MATIÈRES vii

LISTE DES TABLEAUX x

LISTE DES FIGURES xi

LISTE DES ANNEXES xiii

LISTE DES SIGLES ET ABRÉVIATIONS xiv

CHAPITRE 1 INTRODUCTION 1
1.1 Cadre théorique 1
1.2 Problématique 3
1.3 Hypothèses 3
1.4 Objectifs de recherche 3
1.5 Plan du mémoire 4

CHAPITRE 2 REVUE DE LITTÉRATURE 5
2.1 QED-Tutrix 5
2.2 État wheel spinning 10
2.3 Prédiction de comportement dans les STI 13
2.4 Retour sur la revue de littérature 16

CHAPITRE 3 SYNTHÈSE DE L'ENSEMBLE DU TRAVAIL 18
3.1 Modifications appliquées à QED-Tutrix 18
3.1.1 Intégration des auditeurs logiciels dans QED-Tutrix 19
3.1.2 La chaîne de traitements pour l'acheminement des données 20
3.1.3 Désactivation des interventions tutorielles 22

3.2 Méthodologie de l'expérimentation 22
3.3 Définition de l'état de blocage 23
3.4 Prétraitement des données 24
3.5 Modèles 24
3.5.1 Modèle du processus des fréquences d'actions M_F 25
3.5.2 Modèle des sous-séquences M_S 27
3.5.3 Modèle du réseau de neurones convolutif 1D M_C 27
3.5.4 Modèle hybride M_H 27
3.6 Résultats 28
3.6.1 Comparaison avec le modèle précédent de l'apprenant de QED-Tutrix 32

CHAPITRE 4 DISCUSSION GÉNÉRALE 34
4.1 Bref retour sur les résultats 34
4.2 Comparatif avec la littérature 34
4.3 Analyse d'erreurs 36
4.4 Retour sur les objectifs et les hypothèses de recherche 36
4.5 Analyse des impacts sur l'implantation des modèles au sein de QED-Tutrix . 37
4.6 Impact sur l'espace de travail mathématique de QED-Tutrix 37

CHAPITRE 5 CONCLUSION 38
5.1 Synthèse des travaux 38
5.2 Limitations de la solution proposée 38
5.3 Améliorations futures 39

RÉFÉRENCES 40

ANNEXE A 42

LISTE DES TABLEAUX

Tableau 2.1 Un sous-ensemble des caractéristiques considérées par Xie et al. dans leur régression logistique. 14

Tableau 3.1 Tableau des actions monitorées de l'apprenant qui sont exclusivement des actions de base des séquences définies dans la section prétraitement des données. 19

Tableau 3.2 Tableau des actions monitorées de l'apprenant qui sont recueillies à titre de balises pour le prétraitement des données. 20

Tableau 3.3 Tableau des attributs de l'objet JSON enregistrant les actions dans la base de données NoSQL. 20

Tableau 3.4 Tableau des types de la clé valeur associée à l'objet JSON enregistrant les actions dans la base de données NoSQL. 21

Tableau 3.5 Performances moyennes des métriques en pourcentage pour tous les modèles pour l'entraînement et la validation (85/15) avec 10 échantillonnages sur la prédiction de H pour $\alpha_1 = 0,18$ et $\alpha_2 = 0,80$. . . . 29

Tableau 3.6 Les performances des métriques en pourcentage pour tous les modèles sur l'ensemble test pour la prédiction de H avec $\alpha_1 = 0,18$ et $\alpha_2 = 0,80$. 30

Tableau 3.7 L'entropie des sous-séquences d'actions pour les données de l'ensemble d'entraînement et celui de test pour une longueur de sous-séquence fixée à 12 actions. 31

Tableau 3.8 Les performances des métriques en pourcentage sur l'ensemble test avec les séquences coupées à une minute pour tous les modèles et le modèle de l'apprenant actuel de QED-Tutrix H sans validation croisée. Les hyperparamètres sont toujours $\alpha_1 = 0.184$ et $\alpha_2 = 0.796$. 33

Table A.1 Training set information. 50

Table A.2 Test set information. 51

Table A.3 Average performance metrics for all models on training/validation sets (85/15) with 10 samplings for prediction of H for $\alpha_1 = 0.184$ and $\alpha_2 = 0.796$. 57

Table A.4 Performance metrics for all models on test set with leave one out for prediction of H for $\alpha_1 = 0.184$ and $\alpha_2 = 0.796$. 58

Table A.5 Entropy for subsequences of actions for both training and validation sets. 59

LISTE DES FIGURES

Figure 2.1 L'interface de QED-Tutrix montrant le problème classique du rectangle. Le haut de l'interface énonce le problème avec un texte et une figure. Au centre, il y a le panneau Géogébra à gauche avec plusieurs autres onglets (Figure, Phrases, Schéma et Rédaction) et la boîte discursive du tuteur intelligent à droite. Dans le bas de l'interface, il y a la boîte de soumission permettant de compléter et soumettre l'inférence choisie après sélection à partir de l'onglet Phrase. (image de Leduc (2016)) 7

Figure 2.2 Exemple de fichier journal extrait de QED-Tutrix. 8

Figure 2.3 Schématisation de la sous-machine à états finis des états C du GMD. (image de Leduc (2016)) 9

Figure 2.4 Schématisation de la sous-machine à états finis des états N du GMD. (image de Leduc (2016)) 9

Figure 3.1 Schématisation de l'architecture web de recueillement des données. . . 21

Figure 3.2 Les distributions dans le temps des fréquences des actions des séquences en état bloqué. 26

Figure 3.3 Les distributions dans le temps des fréquences des actions des séquences en état non bloqué. 26

Figure 3.4 Diagramme en blocs séquentiel qui schématise la topologie du modèle M_C. 27

Figure 3.5 Fonctions de perte pour différentes configurations du nombre de filtres pour les deux couches CNN 1D. Pour la légende, le chiffre de gauche est le nombre de filtres dans la première couche convolutive et celui de droite, le deuxième. La fenêtre de la première fenêtre est fixée à 6 actions et celle de la deuxième est fixée à 3 actions. 28

Figure 3.6 Fonctions de perte pour différentes configurations de la largeur de fenêtre des filtres pour les deux couches CNN 1D. Pour la légende, le chiffre de gauche est la taille de la fenêtre de la première couche convolutive et celui de droite, la taille de fenêtre de la deuxième. Le nombre de filtres est maintenu fixe à 40 pour la première couche et 15 pour la deuxième. 29

Figure 3.7 Les scores F_1 pour les hyperparamètres α_1 et α_2, respectivement la proportion de la contribution du réseau de neurones convolutif et du modèle des sous-séquences. La proportion du modèle du processus des fréquences d'actions est déduite avec $1 - \alpha_1 - \alpha_2$. 30
Figure 3.8 Nombre de sous-séquences par longueur de sous-séquence maximale fixée pour les ensembles d'entraînement et de test. Cette courbe est comparable à la densité de probabilité par longueur de sous-séquence. 32
Figure A.1 QED-Tutrix (modified version) for this study. We can see the help button in red, bottom-right corner. 46
Figure A.2 Progressions of percentages of proof completion in relation with time in minutes. Each line corresponds to the resolution of one problem by one student in the training set. 50
Figure A.3 Distributions of action frequency in blocking state sequences. 52
Figure A.4 Distributions of action frequency in non-blocking state sequences. . . 53
Figure A.5 Scheme of Gaussian distributions of action frequencies given $H = 0$ and $H = 1$. We observe the probability of the observed frequency $f_a(S_i)$ as surfaces on the tails of both distribution. 54
Figure A.6 Performances across the hyperparameters α_1 and α_2, respectively the proportion of convolutional network model and subsequences model. The proportion of the action frequency model is deduced with $1-\alpha_1-\alpha_2$. 57

LISTE DES ANNEXES

ANNEXE A 42
ARTICLE 1 : A DATA MINING APPROACH TO DETECT BLOCKING STATES IN INTELLIGENT TUTORING SYSTEM 42

LISTE DES SIGLES ET ABRÉVIATIONS

STI	Système Tutoriel Intelligent
QEDX	QED-Tutrix
IU	Interface utilisateur
ETM	Espace de Travail Mathématique
HPDIC	Graphe inférenciel (Hypothèse, Propriété, Définition, Inférence et Conclusion)
MIA	ModélIsation de l'Apprenant
EDOI	Évaluation des Démonstrations et Ordonnancement des Inférences
GMD	Générateur de Message Discursif
CAT	Cognitive Algebra Tutor
BKT	Bayesian Knowledge Tracer (traceur de connaissances bayésien)
KST	Knowledge Space Theory
LP-ITS	Linear Programming Intelligent Tutorial System
MDP	Processus de décision markovien
VIF	Variance Inflation Factor
PC-BKT	Personalized, Clustered, Bayesian Knowledge Tracing
H	État bloqué
M_F	Modèle du processus des fréquences d'actions
M_S	Modèle bayésien des sous-séquences d'actions
M_C	Modèle du réseau de neuronnes convolutif en une dimension
M_H	Modèle hybride
F_1	Mesure de performance d'un modèle basé sur la mesure de précision et de rappel
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
E	L'action *Enter* sur QED-Tutrix
C	L'action *Click* sur QED-Tutrix
T	L'action *Change tabulation* sur QED-Tutrix
SC	L'action *Scroll Chatbox* sur QED-Tutrix
FI	L'action *Select filter* sur QED-Tutrix
RF	L'action *Reset filters* sur QED-Tutrix
CS	L'action *Chose sentence* sur QED-Tutrix
S	L'action *Submit* sur QED-Tutrix

EX	L'action *Exit* sur QED-Tutrix
JAVA	Language de programmation java
CNN	Convolutional Neural Network
1D	Une dimension
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
S_i	Séquence d'actions sur QED-Tutrix
$\mathcal{N}$	La distribution gaussienne
α	Hyperparamètres du modèle M_H
Λ	L'ensemble de tous les sous-séquences d'actions possibles
$H(\cdot)$	Mesure d'entropie

CHAPITRE 1 INTRODUCTION

Les systèmes tutoriels intelligents (STI) forment un pôle important de la recherche en éducation et en intelligence artificielle. Étant prometteurs d'une révolution en éducation, ils se basent sur les vastes théories de l'apprentissage humain et de la didactique. Ses systèmes sont des outils pour l'apprenant. Ils ont pour objectifs de donner toute l'aide nécessaire à celui-ci lors d'apprentissages variés tels que les mathématiques, les sciences naturelles, le français, la géographie, etc. Ce support informatique s'avère très utile d'une autre part pour les enseignants qui ont un temps restreint à accorder à l'ensemble d'une classe. Les STI permettent d'enregistrer un historique des apprentissages, de reconnaître les forces de l'apprenant, de cibler les faiblesses de l'étudiant, d'avoir une large librairie de problèmes, d'offrir une vue d'ensemble sur un groupe d'apprenants et plusieurs autres atouts. Ainsi, il est primordial que ces systèmes soient conçus avec une représentation précise de l'interaction qu'ils ont avec l'apprenant. Cette représentation doit permettre au STI d'apporter son aide à l'apprenant quand ce dernier la requiert. Dans le cadre de ce mémoire, le STI d'intérêt est QED-Tutrix, un système tutoriel en géométrie euclidienne.

1.1 Cadre théorique

Dans le logiciel QED-Tutrix, un utilisateur qui résout un problème mathématique est appelé *apprenant*. Un apprenant est dit en *état de blocage* quand il n'est plus en mesure de progresser dans la preuve qu'il était en train de construire. C'est lorsque l'apprenant est dans l'état de blocage qu'il requiert une aide contextualisée de la part du système tutoriel intelligent. L'état de blocage a été démontré à l'aide des actions de l'apprenant sur l'interface utilisateur. Il existe aussi un état *wheel-spinning* dans la littérature qui est différent de l'état de blocage.

L'*interface utilisateur* (IU) est la fenêtre de l'application informatique contenant l'ensemble des éléments graphiques et informatiques permettant l'interaction entre l'utilisateur et l'ordinateur avec la souris et le clavier. Une *action* sur l'IU est un état prédéfini de la souris et du clavier actionné par l'utilisateur. Une *séquence d'actions* est ainsi une suite ordonnée et finie d'actions. Cette séquence peut être coupée en plusieurs *sous-séquences d'actions*. Donc, l'apprenant génère une séquence d'actions particulière lorsqu'il résout un problème sur le système tutoriel QED-Tutrix.

QED-Tutrix est un système tutoriel intelligent en géométrie euclidienne bâti sur de forts fondements didactiques, conçu au laboratoire Turing. Il se concentre sur la résolution par preuve

en géométrie euclidienne. QED-Tutrix est particulièrement connu pour son mode exploratoire libre basé sur les graphes inférentiels, graphe HPDIC (**H**ypothèses, **P**ropriétés, **D**éfinitions, **I**nférences, **C**onclusion). Ces graphes permettent une habile gestion de l'ensemble de toutes les solutions possibles pour résoudre un problème. Les interactions entre le tuteur et l'apprenant dans QED-Tutrix ont été concentrées autour d'une discussion, dans une *fenêtre de discussion*. Le tuteur intervient textuellement dans cette fenêtre alors que l'apprenant passe par la *soumission d'inférence* en sélectionnant ce dernier dans la banque d'inférences pré-enregistrées. L'interaction entre QED-Tutrix et l'apprenant est gérée par le logiciel par une *machine à états finis*. Actuellement, QED-Tutrix est configuré pour interagir avec l'étudiant chaque minute pour donner des indices contextualisés ou pour donner une rétroaction, positive ou négative, à l'apprenant lors d'une soumission d'inférence. Cette interaction est de ce fait limitée à des échanges entre l'apprenant et QED-Tutrix sur l'interface utilisateur. Des représentations de ces échanges sont construites pour combler cette limitation.

Un *modèle* est une représentation, souvent mathématique, d'une situation ou d'un phénomène. Cette représentation permet de décrire ou même prédire le futur de ce phénomène à partir d'un point donné. Un modèle peut être *explicite*, c'est-à-dire formalisé pour comprendre les facteurs importants du phénomène. Il peut aussi être *implicite*, un cas où le modèle apprend sa configuration optimale à partir d'un ensemble de données. Par exemple, les réseaux de neurones ou les régressions linéaires sont des modèles implicites qui apprennent une représentation à partir d'un jeu de données. Les modèles peuvent être déterministes ou probabilistes. Les représentations *déterministes* se définissent à partir d'une causalité parfaite entre les variables de la représentation, l'erreur présente dans les résultats n'étant que le fruit de l'imprécision des mesures. Les représentations *probabilistes* en revanche présupposent des variables aléatoires, au sens mathématique, avec un lien de causalité aussi aléatoire. Parfois, les modèles sont *mixtes* avec une partie aléatoire et une autre déterminée.

L'apprentissage machine est un domaine de l'intelligence artificielle qui construit des modèles implicites et probabilistes. Son but est d'instancier un algorithme qui peut exécuter une tâche informatique en apprenant sa représentation à partir de données réelles sur la tâche. L'apprentissage profond se concentre sur le développement des réseaux de neurones profonds, qui est une sous-branche de l'apprentissage machine. Les deux modèles importants de l'apprentissage profond sont le *réseau de neurones convolutif* (CNN) très utilisé en vision par ordinateur et le *réseau de neurones récurrents* (RNN) surtout utilisé pour les séries temporelles.

1.2 Problématique

Il est important, lors d'interactions avec un apprenant, que celles-ci soient faites au bon moment. Cette intervention auprès de l'apprenant évite de le déranger pendant qu'il pense à sa solution ou d'apporter une aide hors contexte. Or, étant limité à une intervention avec l'apprenant chaque minute, il est primordial d'améliorer le critère de QED-Tutrix responsable de cette interaction. Pour combler cette lacune, nous avons conçu à l'aide de l'apprentissage machine des modèles de l'interaction entre le STI et l'apprenant.

1.3 Hypothèses

Dans ce mémoire, trois hypothèses sont émises :

— Il existe une *représentation de l'état de blocage* de l'apprenant à partir de son interaction avec la souris et le clavier.
— Dans cette représentation, l'état d'un apprenant bloqué dans sa résolution est *distinguable* d'un apprenant non bloqué et est donc un état prévisible.
— Les apprenants sont *impliqués* dans leur résolution.

La première hypothèse est préalable à la seconde hypothèse et elle stipule l'existence d'une représentation à partir des actions de la souris et du clavier générées par l'apprenant sur l'interface utilisateur de QED-Tutrix. Étant donné que nous ne voulons pas dépendre de techniques de détection d'état psychique plus avancées, comme une caméra de suivi du regard, il est fondamental de prouver que cette simple représentation existe.

La deuxième hypothèse définit la représentation en insistant sur l'existence de différences entre l'état de blocage et l'état de non-blocage d'un apprenant. Cette supposition est la fondation d'une application d'apprentissage machine, c'est-à-dire l'existence d'une séparation concrète entre les classes que l'on désire prédire.

La dernière hypothèse concerne l'implication de l'apprenant lors de la résolution d'un problème sur QED-Tutrix. En effet, étant donné qu'un apprenant bloqué peut être similaire à un apprenant simplement distrait, il est important de supposer préalablement le sérieux de ce dernier puisque cette recherche ne s'intéresse pas à l'étudiant distrait.

1.4 Objectifs de recherche

À partir des hypothèses, deux objectifs de la recherche sont établis :

— Trouver une représentation de l'état de blocage

— Concevoir un algorithme capable de prédire l'état de blocage

Le premier objectif concerne l'identification de caractéristiques sous-jacentes à l'état de blocage. Cette représentation doit être liée à l'interaction de l'apprenant avec le STI.

Le deuxième objectif se veut d'utiliser la représentation de l'état pour prédire sa présence lorsque l'étudiant utilise le STI. Cet objectif aboutit à un modèle de prévision de l'état de blocage utilisable dans la machine à états finis de QED-Tutrix.

1.5 Plan du mémoire

Le mémoire débute sur une revue critique de la littérature autour de QED-Tutrix, de l'état *wheel-spinning* et de la prédiction de comportement dans les STI. Par la suite, la synthèse de l'ensemble du travail fait dans l'article, ci-joint en annexe, est exposée. Cet article est le cœur du travail fait dans le cadre de ce mémoire. Cette synthèse donne lieu subséquemment à une discussion générale sur les résultats des modèles de l'état de blocage. Finalement, la conclusion fait la synthèse finale, énonce les limitations des modèles et les améliorations futures à considérer.

CHAPITRE 2 REVUE DE LITTÉRATURE

2.1 QED-Tutrix

QED-Tutrix est le produit du travail du laboratoire Turing provenant de la collaboration à la fois par l'Université de Montréal et par Polytechnique Montréal. Il est basé sur de forts principes didactiques comme la théorie des espaces de travail mathématique (ETM) de Kuzniak et al. (2016) et la théorie de l'instrumentalisation de Rabardel (1995). Tel que mentionné par Richard et al. (2011), il s'inscrit dans l'esprit de l'intégration des systèmes tutoriels intelligents dans l'interaction de l'élève avec le milieu permettant une aide adaptative dans le cadre de l'apprentissage par opportunité et permettant aussi de contextualiser le contrat didactique.

La théorie des ETM permet de montrer formellement que l'utilisation complète de QED-Tutrix repose sur le concept de l'ETM idoine (Tessier-Baillargeon (2016)). L'ETM se base sur trois piliers : la genèse instrumentale, la genèse sémiotique et la genèse discursive. La genèse instrumentale advient lorsqu'un élève apprend à faire son travail mathématique à travers l'utilisation d'un instrument. Par exemple, il peut utiliser une équerre sur un rectangle pour apprendre la propriété qu'un quadrilatère avec seulement des angles droits est nécessairement un rectangle. La genèse discursive survient lorsque l'apprentissage se produit à travers le discours oral ou écrit. Ainsi, un apprenant qui résout un problème uniquement à l'oral utiliserait principalement sa genèse discursive. La genèse sémiotique arrive lorsque l'apprentissage mathématique utilise les symboles. À titre d'exemples, un élève qui apprend une propriété géométrique de façon purement algébrique serait en genèse sémiotique. Ces trois piliers forment les liens entre le plan cognitif de l'apprenant et le plan épistémologique, ce qui a trait davantage au concret. Ils sont aussi liés à tous les aspects du travail mathématique en général et pas uniquement à l'apprentissage mathématique, qui en demeure néanmoins une partie importante. Les genèses se retrouvent par ailleurs dans l'interface utilisateur de QED-Tutrix et dans son fonctionnement. Entre autres, QED-Tutrix tourne autour de trois fenêtres sous forme d'onglets : Figure, Phrases et Rédaction. L'onglet Figure se concentre sur la manipulation de la figure géométrique, ce qui lui confère à la fois un lien avec la genèse instrumentale et celle sémiotique. Les onglets Phrases et Rédaction, comme leurs noms l'indiquent, monopolisent la genèse discursive.

Nous avons démontré dans Leduc et al. (2016) que QED-Tutrix est un agent qui résout aussi le problème en même temps que l'apprenant dans l'espace de travail mathématique. Or, le système résout nécessairement le problème en aidant l'apprenant dans sa preuve. Le

système tutoriel a une genèse instrumentale, car il mesure l'état de l'élève et garde la trace de ses actions. Il peut discuter avec l'étudiant de son travail mathématique dans la fenêtre de discussion, ce qui fait appel à la genèse discursive. QED-Tutrix arrive aussi à dégager un sens à partir des actions de l'apprenant. Le système tente de saisir l'état actuel de l'apprenant. Cette signification reflète la genèse sémiotique en action. Donc, nous concluons que QED-Tutrix est dans un espace de travail mathématique centré sur une interaction avec l'apprenant qui lui-même s'applique à son travail mathématique. Cette application de l'espace de travail mathématique est faite à partir des quatre couches logicielles de QED-Tutrix.

QED-Tutrix repose sur un système en quatre couches logicielles (Leduc (2016)) : HPDIC, MIA, EDOI et GMD. Le HPDIC (Hypothèse, Propriété, Définition, Inférence et Conclusion) se base sur la notion de graphe inférentiel permettant la gestion de l'ensemble exhaustif des solutions aux problèmes dans QED-Tutrix. La modélisation de l'apprenant (MIA) contient une séquence d'interventions auprès de l'étudiant pour l'aider dans sa résolution. Il faut rajouter à cela que le MIA garde en mémoire la liste des énoncés entrés par l'élève et conserve la progression de l'étudiant. L'évaluation des démonstrations et ordonnancement des inférences (EDOI) a comme fonction d'identifier la solution de l'élève la plus avancée étant donné que certains problèmes mènent à un graphe HPDIC beaucoup trop lourd pour permettre une intervention tutoriel en temps réel. Ainsi, l'EDOI permet une intervention rapide en ordonnant les inférences dans différents contextes. Le GMD (générateur de message discursif) est la couche responsable de l'interaction avec l'étudiant à travers la boîte discursive à l'interface utilisateur (voir Figure 2.1).

Depuis l'interface utilisateur de QED-Tutrix à la figure 2.1, il possible de suivre un flux de travail régulier de l'apprenant. Premièrement, l'apprenant va lire l'énoncé et prendre connaissance de la figure à l'aide du haut de l'interface utilisateur. Par la suite, il a le choix entre quatre onglets : *Figure*, *Phrases*, *Schéma* et *Rédaction*. L'onglet *Figure* est la fenêtre de géométrie dynamique Géogébra qui permet à l'apprenant de manipuler la figure. Cet onglet n'influence actuellement pas les décisions tutorielles. L'onglet *Phrases* permet de sélectionner une inférence particulière à partir de l'ensemble complet des inférences. Pour simplifier la recherche, un système de 5 filtres aide l'apprenant à réduire cet ensemble selon les termes sélectionnés. Par exemple, un étudiant peut sélectionner, comme filtres, un triangle et un angle droit pour obtenir les inférences liées aux triangles rectangles. Les inférences sont déclinées en hypothèse, résultat et justification. Une hypothèse est une inférence présente dans l'énoncé du problème ou dans la figure de départ. L'apprenant doit explicitement soumettre les hypothèses de travail à QED-Tutrix. Un résultat est une inférence découlant de l'emploi d'une justification. Les justifications sont les propriétés ou définitions qui établissent un raisonnement logique. Elles nécessitent des inférences, des hypothèses ou des résultats déjà

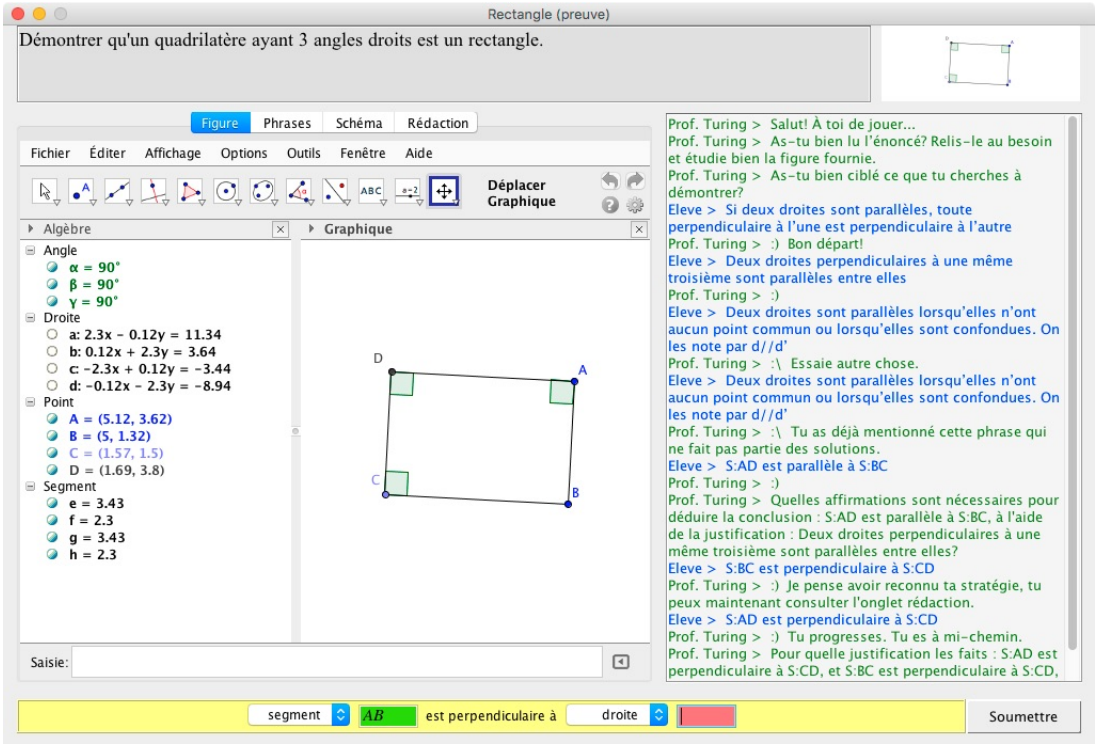

Figure 2.1 L'interface de QED-Tutrix montrant le problème classique du rectangle. Le haut de l'interface énonce le problème avec un texte et une figure. Au centre, il y a le panneau Géogébra à gauche avec plusieurs autres onglets (Figure, Phrases, Schéma et Rédaction) et la boîte discursive du tuteur intelligent à droite. Dans le bas de l'interface, il y a la boîte de soumission permettant de compléter et soumettre l'inférence choisie après sélection à partir de l'onglet Phrase. (image de Leduc (2016))

énoncés, comme assise et assurent un conséquent logique. À titre d'exemple, s'il existe un quadrilatère avec trois angles droits (hypothèses), il est possible de conclure que la forme est nécessairement un rectangle (résultat) étant donné la propriété inhérente de cette forme (justification). Une fois l'inférence choisie dans l'onglet *Phrases*, cette dernière est disposée dans le bas de l'interface utilisateur pour compléter les éléments précis de celle-ci. L'onglet *schéma* est actuellement inactif. L'onglet *Rédaction* s'active à partir du moment où l'apprenant a complété 50 % d'une solution, car une dictée trouée apparaît avec la preuve, ce qui donne un cadre rigoureux à la solution choisit. Cette rédaction encourage l'étudiant à suivre la solution qu'il a choisie. La partie droite de l'interface utilisateur contient la boîte de dialogue avec QED-Tutrix alimentée par le GMD. Un exemple d'interactions textuelles est présenté dans la figure 2.2. Un système de fichiers journaux dans le GMD extrait ces échanges textuels entre le tuteur et l'élève sous forme de fichier texte. Ainsi, les seules traces conservées par le logiciel se limitent aux interventions tutorielles et aux soumissions d'inférence de l'apprenant.

********** Fri, 08-Sep-2017 18 :10 :45 **********

18 :10 :53 : [SYSTEM] 18 :10 :53

18 :10 :53 : [TUTOR] Salut ! À toi de jouer...

18 :11 :08 : [STUDENT] Deux triangles ayant des bases et des hauteurs homologues congrues ont la même aire.

18 :11 :08 : [SYSTEM] 18 :11 :08

18 :11 :08 : [TUTOR] : As-tu bien réfléchi à ta stratégie ?

18 :11 :21 : [STUDENT] Si deux triangles ont une même aire et une base de même mesure, les hauteurs associées à ces bases sont elles aussi congrues

18 :11 :21 : [TUTOR] : As-tu bien lu l'énoncé ? Relis-le au besoin et étudie bien la figure fournie.

Figure 2.2 Exemple de fichier journal extrait de QED-Tutrix.

La machine à états finis du GMD, responsable directement de l'interaction avec l'apprenant, est complexe, car elle gère tout un protocole d'interaction avec l'utilisateur. Or, deux types nœuds dans cette machine à états, nommés C et N, sont représentables par des sous-machines à états finis similaires l'une et l'autre. Ce sont ces dernières qui dictent à QED-Tutrix quand émettre un message à l'apprenant. Ces sous-machines sont importantes, car elles appliquent le délai d'une minute qui indique à QED-Tutrix quand intervenir auprès de l'apprenant. Elles sont directement touchées par l'amélioration entreprise par les modèles de l'état de blocage.

La sous-machine à états C se charge des interventions liées à la mise en contexte. Celle N donne les indices par rapport au contenu d'un nœud du graphe HPDIC. La machine C est activée préalablement à celle N de sorte à faire une mise en contexte avant de donner un indice. Elles sont illustrées aux figures 2.3 et 2.4.

Dans les deux sous-machines, le nœud *In* vérifie si l'élève a déjà mentionné l'énoncé rattaché à ce contexte ou à cet indice. Si l'élève ne l'a pas mentionné (!*Act*), la machine attend une durée T_M au nœud *Wait*. Après un délai T_M, le nœud M est activé et affiche un message de l'ensemble des messages M. S'il y a toujours d'autres messages dans M ($M \neq \{\}$), le nœud *Wait* est réactivé. Autrement, la machine passe à une sous-machine N, pour donner un indice, si elle était dans une sous-machine contextuelle C. Dans le cas où elle était déjà dans

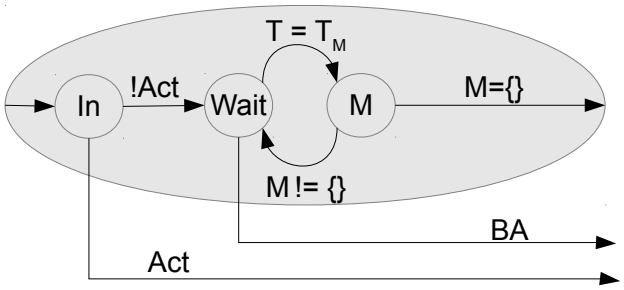

Figure 2.3 Schématisation de la sous-machine à états finis des états C du GMD. (image de Leduc (2016))

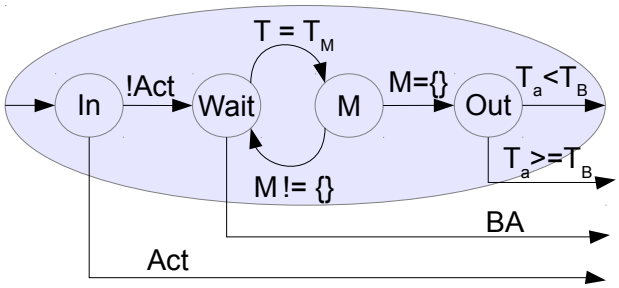

Figure 2.4 Schématisation de la sous-machine à états finis des états N du GMD. (image de Leduc (2016))

une sous-machine N, le nœud *Out* est activé et la machine à états vérifie s'il faut donner une partie de la réponse à l'élève selon la vérification $T_A >= T_B$, où T_A est le temps auquel l'élève a donné son dernier énoncé valide et T_B est le temps d'attente fixé pour donner une partie de la réponse. Si $T_A < T_B$, la machine passe à une autre sous-machine contextuelle C d'un autre énoncé qui pourrait aider à relancer l'apprenant dans sa résolution.

À titre d'exemple, un apprenant débute une résolution sur QED-Tutrix. Il a déjà soumis quelques hypothèses et quelques justifications. Le parcours des interventions de QED-Tutrix débute sur la mise en attente d'une sous-machine contextuelle C. Si l'apprenant n'a rien soumis suite au délai T_M, une mise en contexte d'une justification du graphe HPDIC, proche des inférences déjà soumises, est écrite à l'apprenant dans la fenêtre de discussion. Si aucune autre mise en contexte est disponible dans cette sous-machine à états, la machine passe à une sous-machine d'indice N, toujours pour la même justification. Après un autre délai T_M sans soumission de l'élève, un indice sur la justification est donné dans la fenêtre de discussion. Ce processus est répété tant que l'apprenant ne soumet pas de nouvelle inférence jusqu'à épuisement des indices sur cette justification ou si le délai T_B est dépassé. Si le délai T_B est épuisé, une partie de la justification est donnée directement à l'élève et la machine passe à une

nouvelle sous-machine contextuelle C pour une autre inférence. Lorsque l'apprenant soumet enfin une nouvelle inférence valide, la machine recalcule les priorités de passage des nœuds inférentiels du graphe HPDIC et repart sur une mise en contexte. Ce protocole d'intervention se fonde sur l'observation des interventions des enseignants auprès des élèves, qui utilisent une stratégie d'intervention en entonnoir pour relancer l'élève, donnant successivement une mise en contexte, des indices et une partie de la réponse. Bref, il est important de remarquer que ce processus repose sur une boucle d'attente entre les nœuds *Wait* et M avec un simple délai T_M. L'interaction avec l'utilisateur repose donc sur le délai T_M, qui est d'une minute actuellement.

QED-Tutrix est ainsi un système tutoriel intelligent dont la conception repose sur des fondements didactiques. Il est conçu en quatre couches : HPDIC, MIA, EDOI et GMD. La machine a états finis du GMD repose essentiellement sur l'utilisation d'un délai T_M et sur un historique restreint. Dans ce mémoire, nous proposons des améliorations des couches MIA et GMD par la réforme des critères de la machine à états finis dans ces derniers pour améliorer l'interaction entre le tuteur intelligent et l'apprenant. Pour ce faire, nous utilisons un système d'*auditeurs* (de *listeners* en anglais) qui vont chercher les actions de l'élève à l'interface utilisateur de QED-Tutrix et nous développerons des modèles probabilistes qui permettront de détecter l'état de blocage.

2.2 État wheel spinning

L'état wheel spinning est un état dans lequel l'apprenant n'arrive pas à maîtriser une compétence malgré les efforts qu'il dédie à la mise en pratique de celle-ci. Cet état provient du fait qu'une partie des apprenants sur les STI ne vont jamais pouvoir réussir à maîtriser une ou plusieurs compétences sur celui-ci. Cet état a été défini formellement à partir de données empiriques par Beck and Gong (2013). Ils ont défini arbitrairement la maîtrise d'une compétence comme étant la réussite consécutive de trois problèmes impliquant la compétence et un état wheel spinning comme étant l'échec de la maîtrise d'une compétence en 10 opportunités d'apprentissage. Ces opportunités d'apprentissage sont des résolutions d'exercices de mathématique.

Ils ont démontré ainsi qu'environ 60 % des étudiants dans des jeux de données provenant de Cognitive Algebra Tutor (CAT) et ASSISTments peuvent maîtriser une compétence. Ils démontrent de même qu'une régression logistique est capable de bien modéliser la situation.

Deux ans plus tard, Gong and Beck (2015) pousse le concept de l'état wheel spinning en améliorant l'analyse de leur régression logistique par l'ajout de nombreux facteurs. Les conditions

expérimentales demeurent les mêmes et ils montrent des performances de 76,6% en précision et de 53,1 % en rappel. Avec celui-ci, il démontre qu'il est possible de prédire l'état wheel spinning pour en informer l'enseignant. Pour ce détecteur, ils ont utilisé une grande variété de facteurs qui se séparent en trois catégories : les facteurs internes au tuteur, la sériosité de l'apprenant et les facteurs généraux.

Facteurs internes au tuteur ces facteurs modélisent la capacité de l'apprenant à comprendre la compétence qui est en jeu. Ces facteurs sont au nombre de 5 et s'énumèrent ainsi :

- Le nombre de réponses correctes (correct response count)
- Le nombre de bonnes réponses de suite (correct response in a row count)
- Le score z du temps de réponse moyen exponentiel (exponential mean response time z-score)
- Nombre de problèmes précédents avec demande d'indice (prior problem count with hint request)
- Nombre de problèmes précédents avec au moins 5 demandes d'indice (prior problem with at least 5 hint requests)

Sériosité ces facteurs modélisent le sérieux de l'apprenant dans son travail mathématique avec le STI. Ils sont calculés à travers l'ensemble des compétences. Il y a 8 caractéristiques :

- Le nombre de problèmes précédents répondus correctement et rapidement (prior problem count fast correct)
- Le nombre de problèmes précédents répondus correctement et normalement (prior problem count normal correct)
- Le nombre de problèmes précédents répondus correctement et lentement (prior problem count slow correct)
- Le nombre de problèmes précédents répondus incorrectement et rapidement (prior problem count fast incorrect)
- Le nombre de problèmes précédents répondus incorrectement et normalement (prior problem count normal incorrect)
- Le nombre de problèmes précédents répondus incorrectement et lentement (prior problem count slow in correct)
- Le nombre de problèmes précédents consécutifs avec une demande d'indice (prior problem count with hint request in a row)
- Le nombre de problèmes précédents consécutifs avec au moins 5 demandes d'indice (prior problem count with at least 5 hint requests in a row)

Facteurs généraux ces facteurs représentent vaguement l'état général de l'apprenant dans le STI.

— Le nombre de problèmes précédents (prior problem count)
— L'identifiant de la compétence en jeu (skill ID)

Force est d'admettre que l'état wheel spinning est défini dans cet article avec un ensemble de mesures lié au comportement de l'apprenant sur ces trois facteurs généraux : facteurs internes au tuteur, sériosité et les facteurs généraux. Les auteurs mesurent ici sur les anciens problèmes faits par l'élève le succès, la récurrence, le temps de réponse aux problèmes et les appels d'indices. Cette approche est possible, car les STI CAT et ASSISTment ont une banque de questions auxquelles il est rapide de répondre et l'interaction entre l'élève et le STI est plutôt réduite à une très courte réponse. Ajoutons à cela qu'ils ne peuvent agir qu'une fois un problème terminé, ce qui rend cette approche inadéquate à la réalisation des objectifs de recherche.

Matsuda et al. (2016) ont élaboré une approche basée sur un réseau de neurones particulier appelé un traceur de connaissances bayésien (Bayesian Knowledge Tracer) qu'ils ont appliqué au jeu de données «Cog Model Discovery Experiment Spring 2010». Leur recherche visait à démontrer qu'il était possible de détecter rapidement l'état wheel spinning, c'est-à-dire à l'intérieur de 10 opportunités de pratique.

Ce modèle procède en 4 couches. Initialement, il regarde la séquence des bonnes réponses à des problèmes et, dans une première couche, il calcule l'apprentissage de la compétence. La deuxième couche sert à calculer la pente d'apprentissage entre deux opportunités. À l'intérieur de la troisième couche, ils calculent la différence entre les pentes consécutives pour finalement utiliser une couche d'activation. Cette dernière indiquera si l'état actuel est un wheel spinning. Ce modèle obtient des performances faibles avec un score F_1 autour de 40 %, une performance variant peu sur la séquence de 10 problèmes. Pour la précision, ils ont obtenu 25 % environ constant et, pour le rappel, ils ont calculé en moyenne 80 %, un résultat variant de 89 % au premier problème jusqu'à 71 % aux 10^e problèmes. Ce modèle est plus intéressant topologiquement parlant. Or, il démontre de très faibles résultats. Force est d'admettre que l'entrée qui est une séquence des succès aux problèmes possède peu d'information sur l'état réel de l'élève.

L'état wheel spinning et l'état de blocage sont deux états proches sémantiquement. Par contre, l'état wheel spinning ne permet pas de savoir si l'étudiant nécessite une intervention tutorielle dans le présent. Il utilise plusieurs résolutions de problèmes pour diagnostiquer l'impossibilité de maîtriser une compétence. Or, le but de ce mémoire est de pouvoir intervenir en temps réel auprès de l'apprenant.

La ressemblance entre ces deux états complémentaires est que les deux représentations cherchent à modéliser les interactions déficientes entre le système et l'apprenant. L'état de blocage interprète les moments de faiblesse de l'élève pour solliciter l'aide du STI alors que l'état wheel spinning est l'impossibilité de produire une maîtrise de la compétence dans l'interaction de l'élève avec le STI. Il est possible d'utiliser l'état de blocage comme entrée au modèle de l'état wheel spinning. Cette approche raffinerait les représentations actuellement dans la littérature qui se base presque uniquement sur la réussite et le temps de réaction de l'élève aux problèmes mathématiques en plusieurs déclinaisons.

2.3 Prédiction de comportement dans les STI

Xie et al. (2017) ont démontré qu'il était possible de prédire le succès d'un apprenant à partir de ces stratégies et ces comportements. Pour ce faire, ils ont recueilli des données du STI ALEKS, un système tutoriel en mathématique adaptable aux connaissances de l'apprenant. Il est disponible en ligne et sa conception est basée sur la théorie de l'espace de connaissances (*Knowledge Space Theory*, KST) qui établit une carte de connaissances constituée de prérequis entre plusieurs états de connaissances. Cette théorie permet ainsi de planifier un itinéraire sur l'espace des connaissances pour l'étudiant.

Pour leur expérimentation, ils ont recueilli les données de 179 étudiants de 11 collèges à l'automne 2016. L'ensemble de données consistait d'information à propos des actions des apprenants et de leurs scores. Ils ont alors pu entraîner un modèle de régression logistique. Leurs modèles de régression utilisait 14 caractéristiques pour prédire le succès de l'étudiant et le sous-ensemble des facteurs les plus importants est présenté à la table 2.1.

Parmi ces facteurs, les quatre premiers facteurs s'avéraient de forts indicateurs d'échec pour l'apprenant, c'est-à-dire qu'un élève ne réussissant pas son problème démontre souvent : une erreur suivi d'une explication (WE), deux explications (EE), une erreur suivies d'une explication avec une autre erreur (WEW) et finalement les irrégularités de comportement mesurées par l'entropie (entropy). Finalement, un succès peut particulièrement bien être prédit avec les revues de sujets maîtrisés (PReview).

Leur régression logistique obtenait des résultats de 71 % en exactitude (*accuracy*). L'aire sous la courbe ROC était de 77 % ce qui donne un modèle assez fiable. De plus, les facteurs démontraient une faible colinéarité selon le score *Variance Inflation Factor* (VIF), ce qui vérifiait un faible impact des variables indépendantes entre elles-mêmes.

(Xie et al., 2017) offre une approche dont les résultats sont pertinents pour la détection d'état de blocage. Particulièrement, leur utilisation de la métrique d'entropie semble être un élément

Tableau 2.1 Un sous-ensemble des caractéristiques considérées par Xie et al. dans leur régression logistique.

WE	La probabilité de transition entre une erreur de l'apprenant et la demande d'un indice.
EE	La probabilité de transition entre deux demandes d'explication.
WEW	La probabilité de transition entre une erreur, une explication et une erreur encore.
PReview	Le pourcentage de sujets maîtrisés que l'apprenant a révisés.
Entropy	L'entropie produite par le comportement de l'apprenant dans son apprentissage (calculé à partir de l'entropie de Shannon).

clé dans la mesure du comportement de l'apprenant, d'autant plus qu'ils ont démontré que l'entropie était anticorrélée avec le succès de l'apprenant. En d'autres mots, un apprenant en besoin d'aide devrait exhiber un comportement chaotique et irrégulier à l'interface utilisateur comparé à un étudiant dans la bonne voie pour atteindre une solution.

Par contre, leur régression logistique basée sur 5 facteurs d'impact est plutôt une représentation limitée, voire peu précise et fragile, des patrons comportementaux de l'apprenant avec 71 % d'exactitude dans un contexte pratique. Dans le but d'obtenir un modèle de plus grande précision, une approche plus générale et probabiliste s'avère nécessaire.

Abu Naser (2012) a utilisé un réseau de neurones artificiels pour prédire les performances d'étudiants à partir de fichiers journaux de 2011 provenant du STI LP-ITS dans le domaine de la programmation linéaire. Pour entraîner son réseau, l'auteur a choisi un ensemble de données avec 25 étudiants et il a sélectionné 25 autres étudiants pour tester. L'ensemble d'entraînement avait ainsi un total de 1120 évènements et il y en avait 1024 pour le test.

La topologie de son réseau de neurones était une perceptron multicouche avec des activations sigmoïdes. Ce réseau avait une seule couche cachée de 5 neurones. Les entrées du réseau sont composées du : numéro de problème, niveau de difficulté, niveau d'expertise de l'étudiant, indicateur d'un essai antérieur du problème actuel, durée, niveau d'aide offert et le nombre d'erreurs. Le niveau de difficulté était préalablement attribué par un expert du domaine et le niveau d'expertise de l'étudiant était lié à son niveau d'accomplissement des problèmes du STI entre 1 (débutant) et 6 (expérimenté).

Il obtient comme résultat une exactitude de 92 % de prédiction. Il démontre en plus une convergence rapide des paramètres du réseau de neurones pour un nouveau problème. Cette convergence se produit environ après dix résolutions de ce problème par des apprenants.

Cette approche en réseau de neurones montre une meilleure exactitude que l'approche de la régression logistique de Xie et al. (2017). De plus, l'objectif de recherche de ce modèle est très similaire à celui que ce mémoire veut accomplir. Il faut cependant nuancer les résultats qu'il a obtenus, car la mesure d'exactitude (*accuracy*) n'est pas une bonne mesure de performance pour un modèle, particulièrement pour un jeu de données où les classes à prédire sont déséquilibrées. Il faut favoriser les mesures de *précision*, de *rappel* et de *score* F_1 sur la classe cible.

Barnes et Stamper (2008) ont développé un générateur d'indices dans un tuteur de preuve logique, NovaNET, à l'aide d'un modèle reposant sur le processus de décision markovien (MDP). NovaNET utilise la technique du chaînage avant dans sa conception, ce qui force l'étudiant dans une solution particulière et une résolution progressant du début vers la fin de la preuve.

Le processus de décision markovien repose sur l'hypothèse de transition markovienne et les chaînes de Markov dans lesquelles un agent prend une décision sur un espace d'états avec ensemble d'actions. Ainsi, pour transitionner dans l'espace d'états, l'agent choisit une action dans l'ensemble d'actions ce qui génère pour chaque action une matrice de transition d'états et, selon l'état, l'agent est récompensé ou pénalisé.

Les données d'entraînement ont été prises sur 4 sessions avec des étudiants universitaires en ingénierie dans le cadre d'un cours de mathématiques discrètes entre 2003 et 2006. Leur approche impliquait un apprentissage par renforcement ce qui conférait un aspect expérimental unique. Aucune ingénierie de facteurs n'a été effectuée dans cette recherche et aucune validation de modèle n'a pu être effectuée. Leur méthode prenait les anciennes données de résolution pour générer de nouveaux indices aux apprenants.

Ils démontrent un taux d'assistance hautement contextualisé autour de 80 %. Il obtient aussi une adaptation rapide du système à un nouvel utilisateur (problème de démarrage à froid ou *cold start*) avec un taux d'assistance à 50 % avec seulement 8 étudiants ayant complété le problème.

Dans Stamper et al. (2011), ils démontrent avec un groupe de contrôle une meilleure rétention des étudiants sur le logiciel lorsque le générateur d'indices est activé. De plus, il observe de meilleurs résultats.

Ce modèle basé sur MDP est pertinent dans la mesure où l'approche rejoint notre objectif de

recherche d'interagir avec l'apprenant au «bon» moment et qu'il est entraîné sur les actions de l'apprenant. Cependant, les bases comparatives sont limitées au qualitatif, car aucune métrique de performance n'a été calculée.

Dans le cadre de la littérature sur la prédiction de comportements dans les STI, l'observation d'une grande variété d'approches a été faite. Or, une grande majorité de celles-ci s'avère peu adaptée à la question de recherche par la conception de leur modèle. D'autres modèles n'ont pas été évalués adéquatement. Par contre, plusieurs éléments sont à retenir : l'entropie de la séquence d'actions est une mesure clé pour l'échec des étudiants, les réseaux de neurones ont une très bonne réponse pour la détection de comportements dans les STI et l'hypothèse de la représentation en actions pour offrir des indices est validée par le modèle MDP. Rajoutons à cela que l'état de blocage semble être une représentation adaptée à la prédiction de moments où l'apprenant à besoin d'aide comparativement à la prédiction pure de performance de ce dernier tel que dénotée dans plusieurs articles.

2.4 Retour sur la revue de littérature

QED-Tutrix est un système tutoriel intelligent en résolution de preuve de géométrie. Il est conçu sur de forts principes didactiques dont une approche exploratrice de la résolution de problèmes mathématiques. Il repose sur l'interaction de quatre couches logiciels : HPDIC, MIA, EDOI et GMD. L'objectif de la recherche de ce mémoire repose sur l'amélioration des critères de la machine à états finis de la couche GMD qui est constituée essentiellement du délai T_M. Cette amélioration passera par la détection de l'état de blocage qui se définit par les actions de l'apprenant sur l'interface utilisateur.

L'état wheel spinning est l'état de l'apprenant où il ne peut pas maîtriser une compétence. Beck et Gong ont arbitrairement conceptualisé la maîtrise d'une compétence comme la réussite consécutive de 3 problèmes en 10 opportunités de pratique. L'état wheel spinning est la non-maîtrise en 10 opportunités, une définition douteuse et qui marginalise carrément 40 % des apprenants sur le STI. Ils ont démontré avec 15 facteurs qu'ils pouvaient prédire cet état selon un modèle de régression logistique. Ces nombreux facteurs sont essentiellement des déclinaisons de temps de réponse, du nombre de bonnes réponses et de demandes d'indice. La performance de leur régression logistique était autour de 77 % en précision et 53 % en rappel. Matsuda et al. ont utilisé un BKT avec un réseau de neurones. Leur performance était faible autour de 40 % en score F_1 avec 80 % en rappel et 25 % en précision. L'état wheel spinning et l'état de blocage sont proches sémantiquement, mais ne ciblent pas le même résultat. L'un veut affirmer l'impossibilité de maîtrise une compétence et l'autre vise simplement le bon moment pour une intervention tutorielle. Or, il semble possible de définir l'état wheel spin-

ning à partir de l'état de blocage pour raffiner ces modèles peu performants et qui se fondent sur des définitions très arbitraires.

La prédiction de comportements de l'apprenant dans les STI se fait de façon très variée dans la littérature et avec une mesure de performance peu fiable. Cependant, il a été relevé que la représentation en actions et patrons d'actions est adéquate pour l'état de blocage. De plus, Xie et al. ont démontré que l'entropie d'une séquence d'actions était une mesure anticorrélée au succès dans un problème. La diversité des applications des modèles permet d'établir qu'il faut un modèle mieux adapté que la simple régression logistique qui repose sur des facteurs fixes, mais plus simples que BKT requérant une interprétation des données brutes en connaissances. L'apport d'un réseau de neurones est à considérer, car il démontre une bonne performance pour cette tâche. Donc, il faut un modèle implicite et probabiliste.

CHAPITRE 3 SYNTHÈSE DE L'ENSEMBLE DU TRAVAIL

La méthodologie de l'expérimentation, les modèles et les résultats étant présentés en détail dans l'article de l'annexe, une synthèse de l'ensemble du travail est faite. Comme plusieurs modifications ont été apportées à QED-Tutrix pour pouvoir réaliser la collecte de données, celles-ci sont présentées dans la première section. La méthodologie est ensuite abordée, spécifiant le choix des participants et les conditions expérimentales. Le prétraitement des données aide par la suite à saisir les entrées des modèles probabilistes. Ces derniers sont formalisés dans la partie suivante. Finalement, les résultats de performance sont détaillés.

3.1 Modifications appliquées à QED-Tutrix

Les modifications faites à QED-Tutrix permettent entre autres de recueillir à distance et massivement des données sur les actions faites par les élèves à l'écran en adoptant la technologie du web. Cette amélioration de QED-Tutrix s'inscrit comme première phase d'un plan de développement sur le web dans lequel une application web dans le fureteur est envisagée. Ainsi, les premiers changements augmentent la quantité de données extraites et les centralisent sur le serveur du laboratoire Turing.

Ces changements s'avèrent de plus une mise à jour du système de fichiers journaux. Le contenu de ces fichiers est inadéquat à l'élaboration de modèles pour l'état de blocage, car il ne conserve que l'ensemble des échanges textuels de la fenêtre de discussion de QED-tutrix. Or, l'état de blocage requiert l'enregistrement des actions de l'apprenant sur l'interface utilisateur. Ces actions sont définies dans le but d'obtenir le plus d'informations pertinentes à l'interface utilisateur avec le moins d'entrées possible dans la base de données. Une telle représentation des actions permet aussi de simplifier le traitement des données en amont en obtenant des données directement interprétables et utilisables à l'entrée des modèles prédictifs. Ce souci d'optimisation de l'espace mémoire passe par la définition d'actions précises. Ces structures de données des actions sont détaillées tout d'abord. Ensuite, l'intégration d'auditeurs (*listeners* en anglais) détectant ces actions dans les 4 couches logicielles du système est expliquée. Ces auditeurs recueillent les données sur les actions des apprenants sur l'interface utilisateur. Pour acheminer ces données jusqu'au serveur, une chaîne de traitements a été conçue entre l'application client de QED-Tutrix et le serveur du laboratoire Turing, qui est illustré et détaillé par la suite. Pour finir, le dernier ajustement apporté à QED-Tutrix est la désactivation des interventions tutoriels pour limiter les biais tutoriels.

3.1.1 Intégration des auditeurs logiciels dans QED-Tutrix

Pour obtenir les actions des apprenants à l'interface utilisateur de QED-Tutrix, des auditeurs logiciels ont été implantés à l'intérieur du code source du système écrit en java. Pour ce faire, l'ensemble des actions possibles a été défini et est présenté dans les tableaux 3.1 et 3.2. Ceux-ci sont recueillis dans la base de données NoSQL présentée dans la section suivante. Le tableau 3.1 contient les actions de l'apprenant, qui sont les éléments de base des séquences d'actions formulées dans la partie prétraitement des données. Le tableau 3.2 montre les actions supplémentaires au tableau précédent, mais qui agiront plutôt comme balises pour segmenter les sessions de résolution de problème des étudiants en une séquence d'actions de l'utilisateur.

Tableau 3.1 Tableau des actions monitorées de l'apprenant qui sont exclusivement des actions de base des séquences définies dans la section prétraitement des données.

Nom	Code assigné	Description
Enter	E	L'apprenant entre dans une des sections de l'interface utilisateur.
Click	C	L'apprenant appuie sur un des éléments de l'interface utilisateur.
Change tab	T	L'apprenant change de section entre Figure, Phrases et Rédaction.
Scroll chat box	SC	L'étudiant regarde l'historique de la conversation avec le système tutoriel.
Select filter	FI	L'élève sélectionne un filtre particulier pour l'aider à choisir une inférence.
Reset filters	RF	L'étudiant retire les filtres qu'il avait sélectionnés.
Choose sentence	CS	L'étudiant a choisi une inférence et il doit maintenant la compléter en entrant les noms des objets mathématiques qu'il cible dans la bas de l'interface.

Les tableaux 3.3 et 3.4 présentent la structure de données JSON des actions telles que transmises par l'interface java avec le protocole HTTP et enregistrées dans la base de données.

Suivant les observations faites expérimentalement, une séquence d'actions fréquentes se définit environ par les étapes suivantes (les parenthèses indiquent l'action en cours suivant son code défini ci-haut) :

1. L'apprenant regarde la figure pour comprendre la situation (E) et se dirige vers l'onglet

Tableau 3.2 Tableau des actions monitorées de l'apprenant qui sont recueillies à titre de balises pour le prétraitement des données.

Nom	Code assigné	Description
Submit	S	L'apprenant soumet l'inférence complétée à QED-Tutrix.
Exit	EX	L'apprenant quitte le logiciel.
Help flag	HF	L'étudiant appuie sur le bouton signalant le besoin d'aide.

Tableau 3.3 Tableau des attributs de l'objet JSON enregistrant les actions dans la base de données NoSQL.

Clé	Type de données
session code	Chaîne de caractères alphanumériques de 12 caractères.
action	Chaîne de caractères du nom des actions.
timestamp	Chaîne de caractères représentant la date et l'heure dans le format suivant : Fri Sep 08 18 :10 :53 EDT 2017.
value	Type de valeur dépendant de l'action (voir tableau 3.4).

Phrases (T).

2. Si l'onglet Rédaction est disponible, il cible une inférence en particulier (T) qu'il recherche par la suite de l'onglet Phrase (T).
3. L'étudiant choisit un certain nombre de filtres (FI) pour sélectionner une phrase qu'il utilisera comme inférence à soumettre (CS).
4. L'étudiant remplit l'inférence selon les informations spécifiques de la situation (SC) et soumet l'inférence (S).

S'il se trompe avec les filtres, il recommence sa sélection de filtres avec l'action RF. Si l'étudiant sent qu'il a besoin d'aide, il l'indique en appuyant le bouton «aide» (HF). Lorsqu'il aura fini sa session de résolution de problème mathématique, il quitte le logiciel (EX).

3.1.2 La chaîne de traitements pour l'acheminement des données

Une des modifications importantes consistait à intégrer le protocole HTTP pour sauvegarder l'ensemble des données des participants sur le serveur du laboratoire Turing. Ainsi, cette architecture est schématisée à la figure 3.1.

Tableau 3.4 Tableau des types de la clé valeur associée à l'objet JSON enregistrant les actions dans la base de données NoSQL.

Action	Valeur(s)
E	Chaîne de caractères du nom de classe java de l'objet du contexte.
C	Chaîne de caractères du nom de la classe java de l'objet du contexte.
T	Chaîne de caractères soit Figure, soit Phrases, soit Rédaction.
SC	Entier numérique de la position finale du déplacement dans l'historique de la conversation.
FI	Objet JSON avec les clés : *sélection* pour la chaîne de caractères du nom du filtre et *level* pour l'entier numérique de la position du filtre.
RF	Chaîne de caractères vide.
CS	Objet JSON avec les clés : *sentence* pour la chaîne de caractères de l'inférence et *code* pour la chaîne de caractères de l'identifiant de l'inférence dans le graphe HPDIC.
S	Chaîne de caractères contenant les éléments que l'étudiant a utilisés pour compléter l'inférence.
EX	Chaîne de caractères vide.
HF	Chaîne de caractères *HELP*.

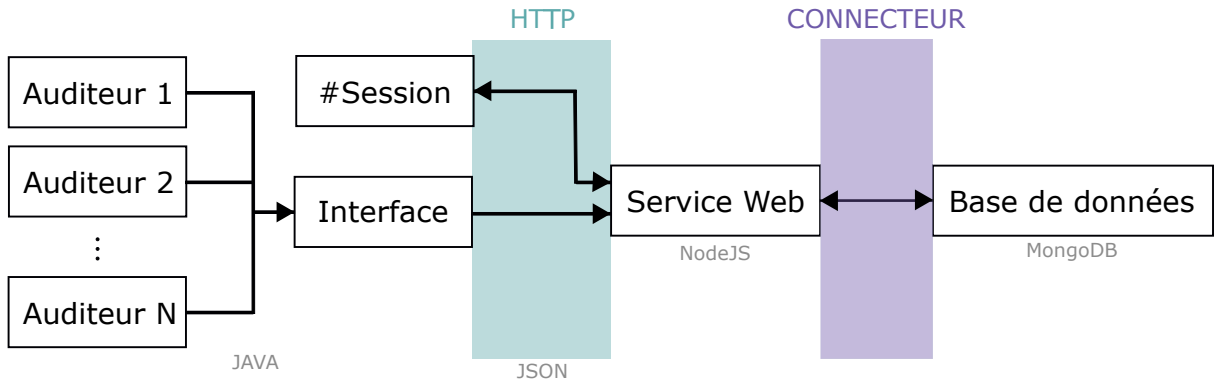

Figure 3.1 Schématisation de l'architecture web de recueillement des données.

Dans la figure 3.1, lorsqu'un problème est démarré, le module # Session à l'intérieur de QED-Tutrix envoie par HTTP le nom du problème et attend un identifiant alphanumérique

de 12 caractères uniques. Pour chaque action générée, le système d'auditeurs les transmet grâce à l'interface de communication HTTP qui envoie les données au serveur.

Au niveau du serveur, il y a le service web en NodeJS qui reçoit et traite les requêtes HTTP provenant du logiciel. Les actions générées par les auditeurs et les identifiants de sessions sont enregistrées en JSON à l'aide d'un connecteur dans une base de données MongoDB qui suit un schéma NoSQL. Les choix de conception associés à NodeJS et MongoDB incluent la facilité d'utilisation, la rapidité de développement, la force de ces systèmes et le grand appui de ces deux communautés.

3.1.3 Désactivation des interventions tutorielles

Considérant qu'une intervention tutorielle de QED-Tutrix crée un biais chez l'apprenant, celles-ci ont été désactivées. De ce fait, les seules interactions que QED-Tutrix avait avec les apprenants pendant les expérimentations étaient des rétroactions positives ou négatives face aux soumissions d'inférence. Autrement dit, QED-Tutrix indiquait uniquement à l'étudiant s'il se trompait sur les inférences qu'il soumettait. Ajoutons à cela que le système indiquait aussi : le message de départ usuel, la complétion d'une solution et la disponibilité de l'onglet Rédaction.

3.2 Méthodologie de l'expérimentation

L'expérimentation pour recueillir les données s'est faite en deux sessions. La première, faite avec 19 étudiants de deuxième année de l'Université de Montréal en enseignement des mathématiques, s'est déroulée en octobre 2017 pour obtenir des données d'entraînement. La deuxième, exécutée avec 4 étudiants diplômés en génie informatique et génie logiciel de l'école Polytechnique de Montréal, a eu lieu en mai 2018 pour obtenir des données de validation. Il y a évidemment une différence claire entre les deux groupes échantillons. Or, il était important de montrer l'application variée des modèles de l'état de blocage, particulièrement pour les valider sur deux groupes différents.

Les deux sessions se sont déroulées selon la même méthodologie :

1. Une pratique de 20 minutes sur la version originale de QED-Tutrix avec le problème du rectangle.
2. Séparation du groupe en trois sous-groupes qui exécuteront chacun des problèmes différents.
3. 3 périodes de 30 minutes pour résoudre le problème du triangle inscrit, le problème

du parallélogramme (aires) et le problème du parallélogramme (triangle).

4. Retour sur l'expérimentation.

La pratique avec le problème du rectangle permet à l'apprenant, nouveau sur QED-Tutrix, d'apprendre à utiliser le logiciel. De ce fait, les données ne seront pas biaisées par la découverte du logiciel par l'étudiant. Ensuite, les sous-groupes exécutent une rotation différente des problèmes, c'est-à-dire que, pendant les 3 périodes de 30 minutes, il y a des étudiants en résolution pour chacun des problèmes. Cet aspect méthodologique limite les effets de fatigue en moyennant la performance des apprenants pour chaque problème.

Les groupes sont aussi plus âgés que les groupes cibles de QED-Tutrix, c'est-à-dire les étudiants du secondaire. La raison s'avère être une simplification de la logistique des expérimentations. Les élèves du secondaire ont nécessairement plus de biais de première utilisation du logiciel, ce qui nécessite plus de temps de pratique avec le logiciel. De plus, ces élèves ont aussi moins d'endurance pour résoudre plusieurs problèmes de suite. Ainsi, des groupes plus âgés permettaient une plus grande concentration des effectifs en temps et en ressources. Puisque ce mémoire cible une preuve de concept de la prédiction de l'état de blocage, les groupes d'étudiants universitaires étaient jugés adéquats avec des résultats interprétables pour les élèves du secondaire.

3.3 Définition de l'état de blocage

L'état de blocage se définit comme étant le moment durant lequel l'apprenant bénéficie d'une intervention tutorielle. Il en résulte que deux situations découlent de cette définition. Dans la première situation, l'étudiant est bloqué dans sa résolution parce qu'il ne possède pas toutes les connaissances et/ou les stratégies nécessaires. L'intervention tutorielle est alors un complément de ces lacunes dans l'espace des connaissances et des stratégies. En ce qui concerne la deuxième situation, l'élève possède suffisamment de connaissances et de stratégies pour résoudre le problème, mais des connaissances ou stratégies erronées nuisent à la progression. Dans le cadre de ce mémoire, les deux situations ne sont pas distinguées.

Les moments mesurant exactement ces états à partir des actions ci-haut sont alors les actions du tableau 3.2 :

1. Lorsque l'apprenant soumet une inférence au système tutoriel, mais que cette inférence n'est pas un élément d'une solution admissible du graphe HPDIC (S)
2. Lorsque l'apprenant indique lui-même qu'il est bloqué avec le button Aide (HF)
3. Lorsqu'il quitte le logiciel sans avoir complété sa preuve (EX)

Ces moments sont des endroits clés où il est clair que l'apprenant est dans l'état de blocage. Les actions du tableau 3.1, qui sont présentes avant un de ces états de blocage, sont considérées alors comme contenant des patrons permettant de savoir si la séquence d'actions mène à cet état de blocage.

3.4 Prétraitement des données

Il est naturel de découper la session de résolution en séquences d'actions consécutives S_i mutuellement exclusives et balisées par les moments de mesure de l'état de blocage. Pour n séquences et n états de blocage, la session R est une suite de séquences d'actions :

$$R = [S_1, S_2, ..., S_n]\,. \tag{3.1}$$

Si le $i$ème état de blocage est noté par H_i pouvant prendre la valeur 0 ou 1, la session R peut être associée à une suite d'états

$$[H_1, H_2, ..., H_n]\,. \tag{3.2}$$

Il est alors supposé la relation de causalité $S_i \Rightarrow H_i$, de sorte qu'un modèle puisse établir cette relation. Étant donné la nature du problème, les modèles probabilistes sont utilisés.

3.5 Modèles

Les modèles probabilistes utiliseront comme variables indépendantes les séquences d'actions et seront des modèles implicites et mixtes qui seront entraînés sur l'ensemble de données recueillies pendant la première expérimentation. Ainsi, ces représentations apprendront les patrons récurrents générés par l'apprenant pour les deux valeurs possibles de l'état de blocage H et évalueront la probabilité d'être $H = 1$ ou $H = 0$. Les modèles se concentrent particulièrement sur les fréquences des actions et sur les fréquences des sous-séquences d'actions. Il en résulte le modèle du processus des fréquences d'actions M_F, le modèle bayésien des sous-séquences M_S et le modèle du réseau de neurones convolutif M_C.

Étant donné la nature de la question de recherche, deux types de réseau de neurones étaient de bons candidats pour obtenir de bonnes performances. Le réseau de neurones convolutif à une dimension et le réseau de neurones récurrents (deux architectures possibles : gated recurrent unit, GRU, ou long short-term memory, LSTM). Or, le réseau de neurones récurrents a été écarté vu sa plus grande complexité topologique. Rajoutons à cela que le réseau convo-

lutif en une dimension se compare directement avec le modèle bayésien en sous-séquence, ce qui permet une analyse plus profonde de l'utilisation de sous-séquences d'actions dans les représentations.

Les trois modèles sont détaillés dans l'article de l'annexe **??**. Donc, dans les sections qui suivent, seulement un résumé est exposé.

3.5.1 Modèle du processus des fréquences d'actions M_F

Le modèle du processus des fréquences d'actions se base sur une hypothèse fréquentielle des actions vues comme des processus stochastiques. Ces processus impliquent normalement une évolution temporelle des distributions des fréquences des actions dans le temps. Or, il existe un cas particulier de processus stochastique, dit processus stationnaire, qui résulte en un processus ne variant pas dans le temps.

En regardant les distributions des fréquences d'actions en état bloqué et non bloqué dans le temps aux figures 3.2 et 3.3, le processus s'avère être quasi gaussien et quasi-stationnaire dans le temps.

Dans ces figures, l'axe horizontal est l'action en position i, nommée Xi. Il est à noter que les fréquences d'actions varient autour d'une moyenne particulière dans le temps. Cette observation implique qu'il est possible d'utiliser une distribution gaussienne des fréquences pour modéliser la variation des fréquences. En calculant la moyenne et la variance des actions dans le temps, il est possible de représenter la distribution gaussienne des fréquences pour l'état de blocage et non bloqué.

Avec ces distributions, le calcul de la probabilité d'être en état de blocage H pour une séquence S_i particulière et une fréquence d'action a précise, comme démontré dans l'article, est :

$$P\left[H|S_i, a\right] \simeq 2 \cdot \int_{f_a(S_i)}^{\infty} Gaussian\left(\frac{|z - \mu_{a|H}|}{\sigma_{a|H}}\right) \frac{\mathrm{d}z}{\sigma_{a|H}} \tag{3.3}$$

où $\mu_{a|H}$ et $\sigma_{a|H}$ sont respectivement la moyenne et l'écart-type de la distribution gaussienne de l'action a pour un H donné. $f_a\left(S_i\right)$ est l'observation de la fréquence de cette action dans la séquence S_i. Le facteur 2 est la résultante de la symétrie de la distribution par l'utilisation de la valeur absolue telle que dans le test de Student. La différence avec le test de Student est qu'ici l'observation appartient nécessairement à l'une des deux distributions $H = 0$ ou $H = 1$ et donc les résultats de ce calcul pour $H = 0$ et $H = 1$ sont normalisés à 1.

Pour chaque action, la probabilité est au final combinée à l'aide de la combinaison linéaire

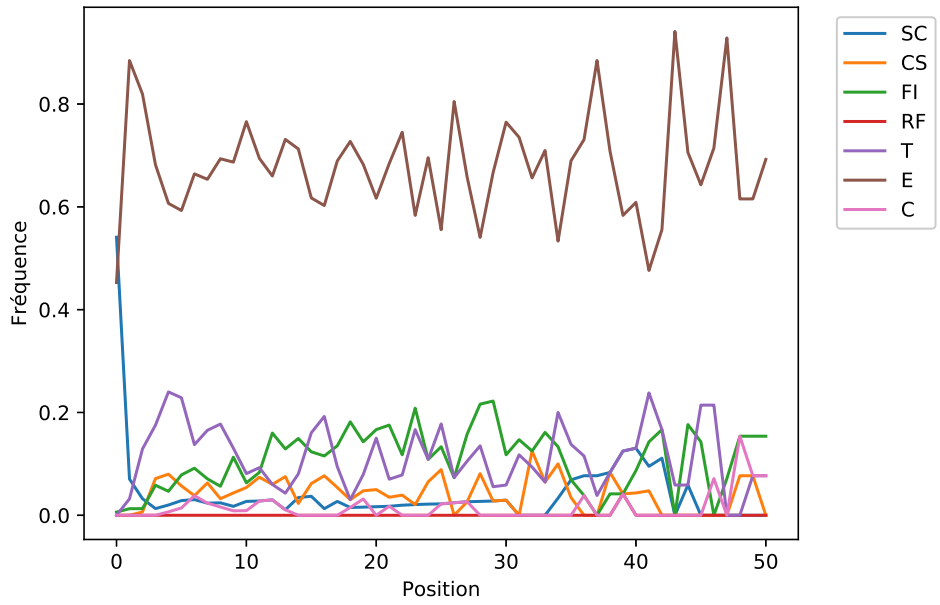

Figure 3.2 Les distributions dans le temps des fréquences des actions des séquences en état bloqué.

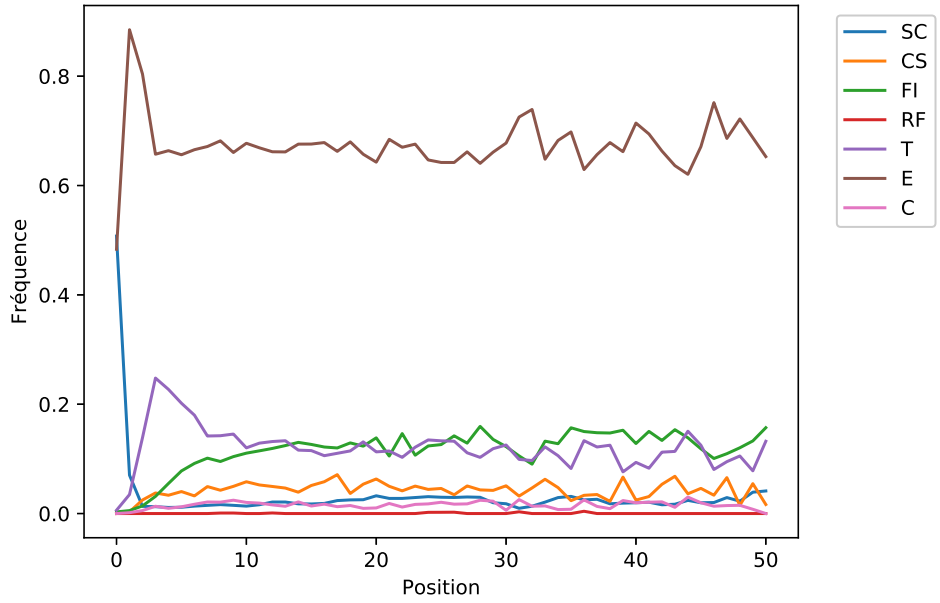

Figure 3.3 Les distributions dans le temps des fréquences des actions des séquences en état non bloqué.

sur les probabilités d'action :

$$P\left[H|S_i\right]_{M_F} = \sum_a P\left[H|S_i, a\right] P\left[a\right] \tag{3.4}$$

3.5.2 Modèle des sous-séquences M_S

Le modèle en sous-séquences bayésien se base sur la matrice de comptage des sous-séquences A qui a, pour lignes s, les séquences d'actions et, pour colonnes c, les sous-séquences. Donc, chaque élément de la matrice de comptage A peut s'écrire A_{sc}. La ligne A_{ic} est celle de la séquence S_i pour laquelle la probabilité d'être en blocage souhaite être connue. De plus, le comptage se fait sur l'ensemble des séquences $S(H)$ de l'état H. En considérant l'hypothèse bayésienne, le résultat est :

$$P\left[H|S_i\right]_{M_S} = \frac{\sum_{c,s\in S(H)} A_{ic}A_{sc}}{\sum_H \sum_{c,s\in S(H)} A_{ic}A_{sc}} \tag{3.5}$$

La preuve complète est présentée dans l'article de l'annexe.

3.5.3 Modèle du réseau de neurones convolutif 1D M_C

Le modèle M_C, le CNN 1D, a été conçu sur deux couches convolutives en une dimension et une couche d'activation sigmoïde. Deux couches de type *drop-out* ont été placées entre les 3 couches précédentes pour régulariser la réponse du réseau sensible au surapprentissage. Le taux de *drop-out* optimal est de 40 %. L'ensemble des hyperparamètres de ce modèle a été optimisé à l'aide des figures 3.5 et 3.6. La topologie résultante et optimale est illustrée par un diagramme en blocs à la figure 3.4.

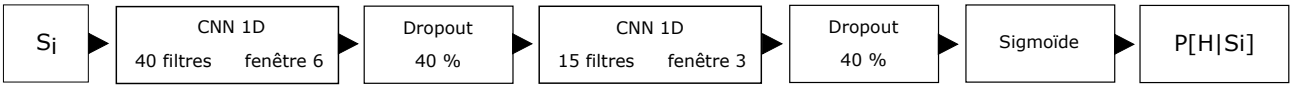

Figure 3.4 Diagramme en blocs séquentiel qui schématise la topologie du modèle M_C.

3.5.4 Modèle hybride M_H

Le modèle hybride est naturellement la combinaison linéaire des modèles précédents M sur les hyperparamètres α_M :

$$P\left[H|S_i\right] = \sum_M \alpha_M P\left[H|S_i\right]_M \tag{3.6}$$

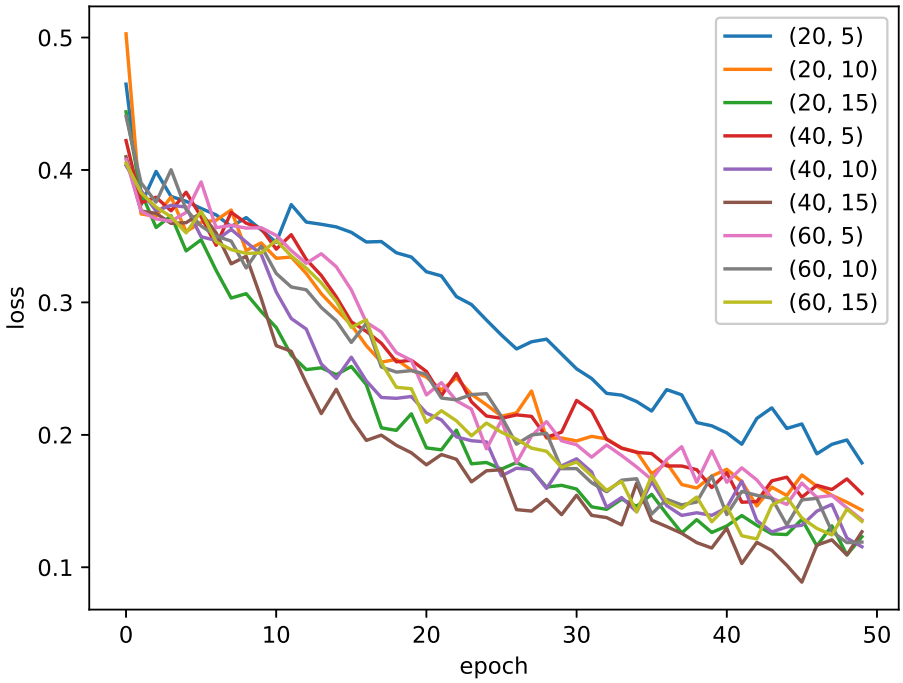

Figure 3.5 Fonctions de perte pour différentes configurations du nombre de filtres pour les deux couches CNN 1D. Pour la légende, le chiffre de gauche est le nombre de filtres dans la première couche convolutive et celui de droite, le deuxième. La fenêtre de la première fenêtre est fixée à 6 actions et celle de la deuxième est fixée à 3 actions.

avec $\sum_{M \in \Sigma} \alpha_M = 1$. L'espace des hyperparamètres α_M a été représenté pour optimiser la configuration du modèle hybride. Comme il y a trois modèles simples et une contrainte sur les hyperparamètres, il y a deux hyperparamètres libres à optimiser.

3.6 Résultats

La figure 3.7 permet de voir le score F_1 à travers l'ensemble des valeurs possibles pour α_1, la proportition du modèle M_C, et α_2, la proportion du modèle M_S. La proportion de la représentation M_F se déduit par la contrainte sur les α_M.

La présence de deux sommets de performance dans l'espace des hyperparamètres α montre une symétrie dans la performance des modèles M_F et M_S. Ils se produisent pour environ $\alpha_1 = 0,18$ et $\alpha_2 = 0,8$ l'autre en opposition à environ $\alpha_1 = 0,2$ et $\alpha_2 = 0,1$. Ainsi, leur proportion de contribution peut être inversée pour obtenir une performance finale similaire. Or, en ce qui concerne M_C, il est important de conserver sa contribution autour de $0,8$. Un

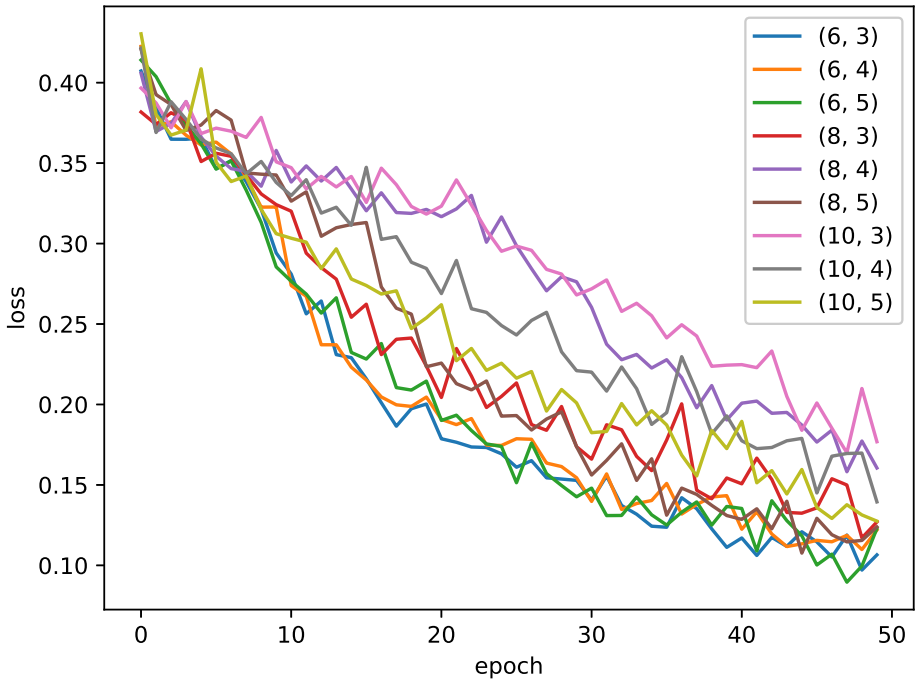

Figure 3.6 Fonctions de perte pour différentes configurations de la largeur de fenêtre des filtres pour les deux couches CNN 1D. Pour la légende, le chiffre de gauche est la taille de la fenêtre de la première couche convolutive et celui de droite, la taille de fenêtre de la deuxième. Le nombre de filtres est maintenu fixe à 40 pour la première couche et 15 pour la deuxième.

$\alpha_1 = 0,18$ et un $\alpha_2 = 0,80$ vont être considérés pour la suite, car empiriquement il offre une performance légèrement meilleure à 80,4 %. Il est à noter aussi qu'aucun des trois modèles simples n'a de performance proche de M_H, ils sont tous autour de 66 %. La fiche complète des performances de l'entraînement est affichée au tableau 3.5. La méthodologie de la validation est mentionnée dans l'article en annexe.

Tableau 3.5 Performances moyennes des métriques en pourcentage pour tous les modèles pour l'entraînement et la validation (85/15) avec 10 échantillonnages sur la prédiction de H pour $\alpha_1 = 0,18$ et $\alpha_2 = 0,80$.

Métriques	M_F	M_S	M_C	M_H
Précision	57,9	56,8	**99**	83,4
Rappel	**77,8**	**77,8**	50	**77,8**
Score F_1	66,3	65,5	66,4	**80,4**

Du tableau 3.5, les performances générales sur les métriques sont excellentes, ce qui démontre

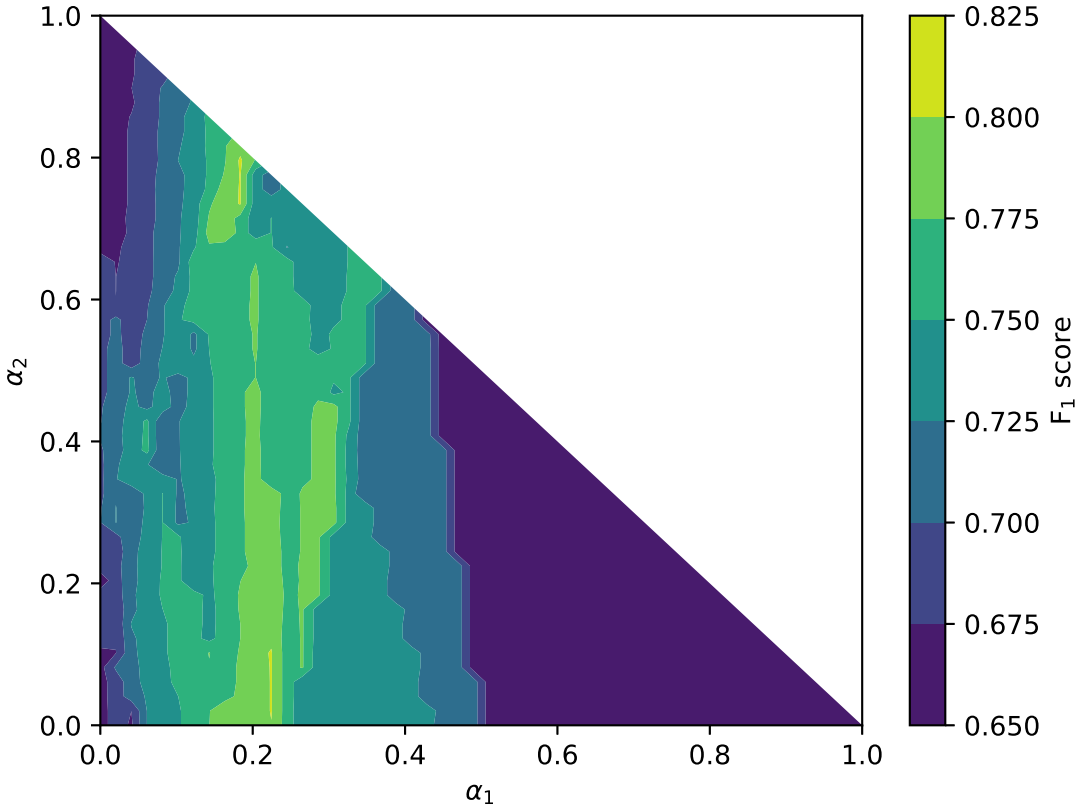

Figure 3.7 Les scores F_1 pour les hyperparamètres α_1 et α_2, respectivement la proportion de la contribution du réseau de neurones convolutif et du modèle des sous-séquences. La proportion du modèle du processus des fréquences d'actions est déduite avec $1 - \alpha_1 - \alpha_2$.

une bonne polyvalence du modèle, qui prend les meilleures performances de chacun des modèles simples. Le modèle M_C est particulièrement performant en précision, mais le rappel est plutôt faible, ce qui explique les résultats similaires en score F_1.

Les résultats de la phase de test sont montrés dans le tableau 3.6.

Tableau 3.6 Les performances des métriques en pourcentage pour tous les modèles sur l'ensemble test pour la prédiction de H avec $\alpha_1 = 0,18$ et $\alpha_2 = 0,80$.

Métriques	M_F	M_S	M_C	M_H
Précision	48,6	56,4	**92,4**	79,9
Rappel	62,5	**82,5**	65,0	75,0
Score F_1	54,6	67,0	76,3	**77,3**

Le modèle M_H perd environ 2 % sur l'ensemble de test. M_F est beaucoup moins robuste

avec une perte en score F_1 proche de 12 %. Étrangement, les modèles M_S et M_C ont une surperformance sur l'ensemble test, particulièrement M_C. Pour expliquer ce phénomène de légère surperformance en score F_1 sur l'ensemble test, il faut remarquer en premier que la surperformance se produit au niveau de la mesure de rappel des modèles M_S et M_C. L'ensemble d'entraînement et de test provenant de deux populations différentes, cette observation en est une conséquence.

Ainsi, les deux modèles étant bâtis sur la notion de sous-séquences, il faut regarder la mesure d'entropie des sous-séquences. L'entropie mesure la facilité de prédiction de l'ensemble de données, c'est-à-dire qu'un ensemble de données avec une faible entropie a plus de récurrences qu'un ensemble à haute entropie.

En calculant l'entropie des sous-séquences d'actions Λ des ensembles d'entraînement et de test avec l'équation 3.7, le tableau 3.7 est obtenu. Étant donné que ce calcul utilise une borne supérieure sur le nombre d'actions dans les sous-séquences, il faut regarder le nombre de sous-séquences apportées pour chaque action ajoutée pour assurer une approximation de l'entropie fiable. La figure 3.8 présente cette distribution. Selon cette dernière, une coupure à 12 actions assure une très bonne approximation, car la quantité d'états occupés par des longueurs supérieures à 12 est drastiquement plus faible, d'autant plus que la tendance va en diminuant.

$$H(\Lambda) = -\sum_{i \in \Lambda} p_i \ln(p_i) \tag{3.7}$$

Tableau 3.7 L'entropie des sous-séquences d'actions pour les données de l'ensemble d'entraînement et celui de test pour une longueur de sous-séquence fixée à 12 actions.

Ensemble de données	Entropie
Entraînement	3,99
Test	2,53

Il en résulte que les étudiants de la Polytechnique génèrent des séquences de plus faible entropie, parce que leurs comportements étaient plus réguliers. Ainsi, il est clair que les modèles utilisant les sous-séquences obtiennent de meilleurs résultats sur l'ensemble de test, car il y a un recoupement de l'ensemble des sous-séquences détectées sur l'ensemble d'entraînement avec celui de test. Ces sous-séquences particulières, faisant partie du recouvrement, permettent aux modèles de mieux prédire les blocages sur les apprenants de Polytechnique.

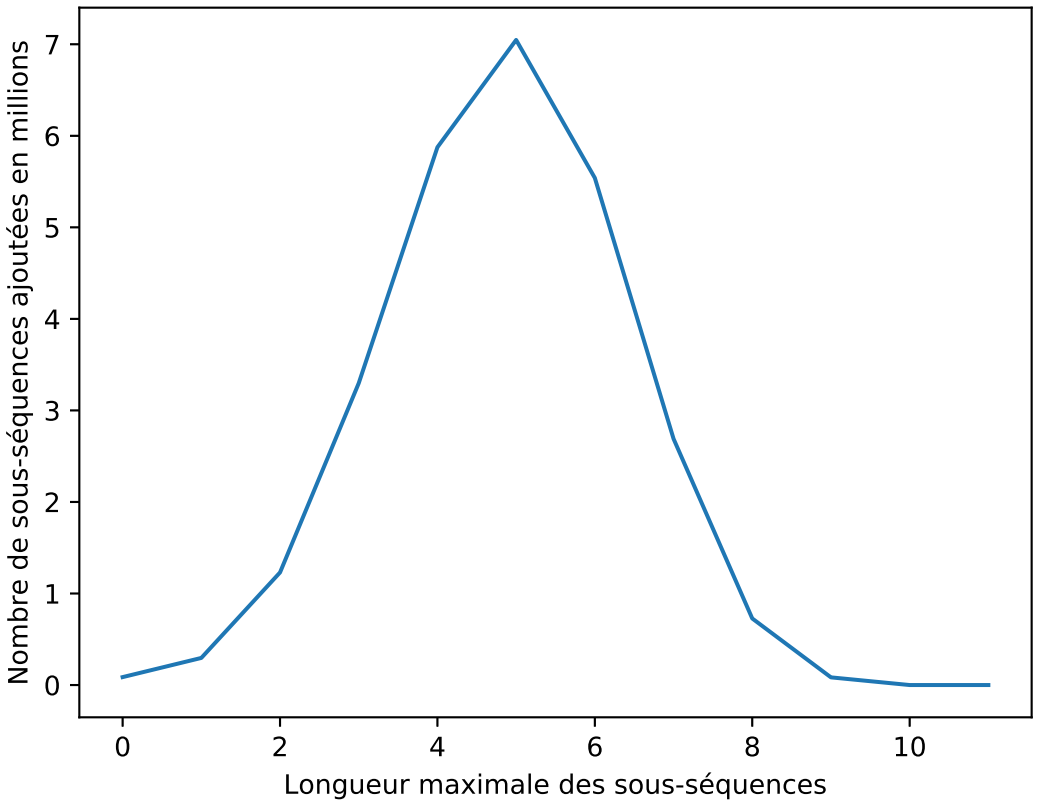

Figure 3.8 Nombre de sous-séquences par longueur de sous-séquence maximale fixée pour les ensembles d'entraînement et de test. Cette courbe est comparable à la densité de probabilité par longueur de sous-séquence.

3.6.1 Comparaison avec le modèle précédent de l'apprenant de QED-Tutrix

Dans la couche GMD de QED-Tutrix, le modèle d'intervention envoie un message à l'apprenant après une minute d'inactivité. Pour comparer ce modèle, noté M_{QEDX}, aux modèles de M_H, les séquences de l'ensemble de test ont été coupées à une minute en durée si elle était d'une durée plus longue. De ce fait, seules les actions se retrouvant dans la première minute sont conservées dans les séquences de l'ensemble de test. Le modèle M_{QEDX} prédit un état de blocage si l'apprenant est toujours en action à cette coupure. Si l'apprenant avait déjà soumis une inférence avant le délai d'une minute, M_{QEDX} prédit un non-blocage. Les prédictions de l'état de blocage pour l'ensemble des modèles ont été recueillies et les résultats sont présentés à la table 3.8.

Les performances des modèles M_F, M_S, M_C et M_H sont toutes nettement supérieures à celles

Tableau 3.8 Les performances des métriques en pourcentage sur l'ensemble test avec les séquences coupées à une minute pour tous les modèles et le modèle de l'apprenant actuel de QED-Tutrix H sans validation croisée. Les hyperparamètres sont toujours $\alpha_1 = 0.184$ et $\alpha_2 = 0.796$.

Métriques	M_F	M_S	M_C	M_H	M_{QEDX}
Précision	56,5	61,5	**91,7**	81,3	16,8
Rappel	72,2	**88.8**	61,1	72,2	50,0
Score F_1	63,4	72.7	73,3	**76,4**	25,1

du modèle M_{QEDX}. Ce modèle est même inadéquat à l'état cognitif de l'apprenant dans la perspective de l'état de blocage.

CHAPITRE 4 DISCUSSION GÉNÉRALE

4.1 Bref retour sur les résultats

Premièrement, pour le modèle des fréquences d'actions M_F, il semble que son impact soit faible dans le modèle hybride M_H, autour de 2 %, et qu'il est peu robuste. Ainsi, dans un modèle de production, l'élimination de ce modèle est à considérer, ce qui simplifie aussi l'algorithme.

Pour les modèles M_S et M_C, le premier est meilleur en rappel et le deuxième meilleur en précision. Il est important dans le cadre de cette recherche de pouvoir trouver le plus d'états de blocage que possible et de peu se tromper en les trouvant, car il faut aider tous les apprenants en besoin d'aide et s'assurer qu'ils soient bien en blocage pour éviter un dérangement. Un modèle robuste comme M_H répond bien à ce besoin. Cependant, il est possible, selon les hyperparamètres α, d'ajuster le compromis précision/rappel.

4.2 Comparatif avec la littérature

La comparaison avec la littérature doit être faite indirectement étant donné que l'état de blocage tel que nous le définissons n'y est pas abordé.

L'état wheel spinning n'ayant pas été construit pour le même objectif que l'état H, la comparaison est impossible à faire. Par contre, pour accomplir l'objectif de ce mémoire, la représentation trouvée avec l'état de blocage ne nécessite pas de facteurs sur le temps de réponse, le nombre d'indices demandés et le nombre de bonnes réponses, une approche qui requiert l'historique de l'apprenant sur plusieurs problèmes et la définition de plusieurs facteurs fixes. À l'inverse, l'état de blocage est défini à partir des comportements de l'apprenant. Utilisant les modèles prédictifs de l'état de blocage, il est possible d'intervenir auprès de l'étudiant à même le problème actuel, c'est-à-dire une approche en temps réel.

La proximité entre l'état de wheel-spinning et celui de blocage survient lorsqu'un apprenant est déclaré dans un état wheel spinning après plusieurs problèmes, il est presque garanti alors d'être majoritairement en état de blocage dans ses actions. La réciproque est d'autant plus vraie, un apprenant très peu bloqué n'est probablement pas en état wheel spinning. Cette logique permet de définir la représentation de l'état wheel spinning à partir de l'état de blocage.

En ce qui concerne la prédiction de comportements dans les STI, le résultat sur l'entropie des

jeux de données, dans le tableau A.5, peut être comparé avec le paramètre d'entropie mesuré par Xie et al. (2017). Il démontrait qu'une entropie élevée se corrélait avec un apprenant qui est plus souvent dans l'état de blocage. En regardant la proportion d'états de blocage sur le nombre total de séquences générées par les étudiants de l'ensemble d'entraînement et de l'ensemble test des tableaux A.1 et A.2, la proportion est presque deux fois plus petite pour les étudiants de Polytechnique, ce qui corrobore les observations de Xie et al. (2017). Avec son modèle de régression logistique, il obtenait 71 % d'exactitude. Puisqu'il y a une similarité dans la tâche de prédiction, le modèle M_H supplante ce modèle avec une métrique rigoureuse.

Abu Naser (2012) avait utilisé un perceptron pour prédire la performance d'étudiants sur des problèmes de programmation linéaire. Son modèle arrivait à une exactitude de 92 %. Cette mesure est généralement peu fiable lorsque les classes à prédire ne sont pas représentées en proportion égales. Avec le modèle convolutif M_C, les résultats en exactitude étaient élevés avec environ 96 %. Par souci de fiabilité, on a préféré le score F_1, qui était de 66 % en entraînement et 76 % en test, avec une excellente précision au-dessus de 90 %. De plus, la différence en topologie semble mieux adaptée étant donné la tâche d'entraînement sur des séquences d'actions. De ce fait, M_C a des avantages clairs sur la littérature, d'autant plus qu'il n'a aucune lacune sur la mesure de performance.

? ont utilisé un traceur de connaissances bayésien nommé PC-BKT, basé sur des profils d'apprenants, pour prédire les performances d'étudiants. Leur exactitude était de 82 %. Or, étant donné la complexité du modèle PC-BKT et la tâche de prédiction différente, les modèles M_H développés pour ce mémoire peuvent être considérés au moins d'un niveau comparable à la littérature avec 80 % en score F_1.

Pour faire suite à cette comparaison avec la littérature, les modèles développés sont compétitifs pour la tâche en jeu. Pour plusieurs auteurs de la littérature, la comparaison était difficile, mais elle s'appuie sur des bases fondamentales au problème et sur une méthodologie rigoureuse. L'approche utilisée dans ce mémoire est unique et pertinente au domaine des STI, qui souhaite avant tout aider les apprenants. Il a été possible de corroborer certaines observations de la littérature, comme la nature entropique de l'état de blocage et qu'une approche entre un modèle logistique et un modèle BKT peut bien prédire les moments de blocage. Cette approche gagne beaucoup a utiliser un réseau de neurones, dans le cas présent un réseau convolutif, pour obtenir de bonnes performances. En effet, M_C est particulièrement surperformant avec des ensembles de données à entropie plus faible.

Les modèles de ce mémoire ont l'avantage, en contraste avec ceux de la littérature, d'être bâtis sur une représentation qui est ultimement indépendante de l'apprenant et de l'historique complet, ce qui évite les problèmes de démarrage à froid.

4.3 Analyse d'erreurs

En regardant les états de blocage que l'algorithme M_H a mal classifiés, il y a un motif observable. Certaines actions sont associées par le modèle comme étant indicatrices de l'état H. Ainsi, les actions CS, choisir une inférence, et FI, choisir un filtre, sont des indicateurs de l'état non bloqué. Or, SC, qui est l'action de regarder dans l'historique de la conversation, et C, les clics, s'avèrent de forts indicateurs de l'état de blocage. Nous avons observé plusieurs séquences d'actions mal classifiées parce qu'elles avaient peu de CS et FI.

L'association des actions avec un état particulier faite par l'algorithme semble gêner la classification de certaines séquences dans l'état $H = 1$. Cependant, il faut nuancer cette analyse en surface, car les algorithmes M_S et M_C sont entraînés sur les sous-séquences d'actions. De ce fait, en augmentant le jeu de données et en s'assurant de son hétérogénéité avec la population échantillonnée, ce sont des erreurs qui devraient s'amenuiser.

4.4 Retour sur les objectifs et les hypothèses de recherche

Les objectifs de recherche étaient de trouver une représentation de l'état de blocage et de concevoir un algorithme capable de le prédire. En ce qui concerne la représentation, celle qui se base sur l'état de blocage comme étant un manque ou une erreur dans l'espace des connaissances mathématiques et des stratégies a été sélectionnée. En détectant ces manques et ces erreurs avec les rétroactions tutorielles et l'appui d'un bouton Aide, il a été possible d'enregistrer avec certitude des moments où l'apprenant a besoin d'une aide tutorielle. Les actions souris et clavier, à l'interface utilisateur, ont aussi été enregistrées et sont utilisées comme entrées pour les algorithmes de prédiction. Pour cet algorithme de prédiction de l'état, le modèle du processus de fréquence d'actions, le modèle de détection de sous-séquences, le modèle du réseau de neurones convolutif et le modèle hybride ont été développés et entraînés sur l'ensemble de données. Étant donné les résultats obtenus, les deux objectifs de recherche sont accomplis d'autant plus que les hypothèses de recherche ci rattachant sont aussi validées.

En ce qui concerne l'hypothèse sur l'implication des apprenants, celui-ci a davantage été observé qualitativement pendant les deux expérimentations et durant la période de retour sur l'expérimentation. Les apprenants étaient tous engagés en posant plusieurs questions, en s'intéressant au système tutoriel et en travaillant ardemment sur les problèmes donnés. Cette hypothèse est validée aussi dans le cadre de ce mémoire.

4.5 Analyse des impacts sur l'implantation des modèles au sein de QED-Tutrix

L'implantation des modèles présentés dans ce mémoire au sein de QED-Tutrix, qui est écrit en java, débute par la traduction des modèles. En effet, ces derniers étant écrits en R et python, il faut les traduire en java pour pouvoir les utiliser localement dans le système. Ensuite, il faut entraîner les modèles sur nos ensembles de données pour les rendre aptes à détecter un état de blocage.

Deuxièmement, le système d'auditeurs ajouté à QED-Tutrix doit être lié à la couche logicielle MIA. Celle-ci contient le modèle actuel de l'apprenant qui est basé sur l'historique des inférences soumises. Cette modification va permettre au GMD d'accéder directement aux séquences d'actions.

Finalement, le délai d'une minute servant de condition dans la machine à états du GMD doit être remplacé par la prédiction du modèle hybride. Comme les modèles sont entraînés avec des séquences complètes, il faudra limiter le nombre d'actions minimum qu'une séquence peut contenir. Ce minimum limite les faux positifs pouvant être déclenchés pour des séquences d'actions trop courtes et conserve de ce fait la performance de détection des blocages à un niveau proche de celles mesurées dans ce mémoire. Un minimum de 12 actions devrait assurer une bonne détection puisque, selon la figure 3.8, les modèles utilisent peu de sous-séquences ayant plus de 12 actions. La longueur moyenne des séquences de nos jeux de données est de 24 actions.

4.6 Impact sur l'espace de travail mathématique de QED-Tutrix

Avec ces modèles développés pour QED-Tutrix, on note des impacts notables sur les genèses de son ETM lorsque la mise à jour du GMD sera faite. Ayant une meilleure représentation de l'apprenant, le GMD augmentera la précision de la genèse sémiotique du système, la genèse instrumentale est elle aussi améliorée puisque les nouveaux modèles utilisent les séquences d'actions pour mesurer l'état de blocage. Indirectement, les échanges discursifs entre l'apprenant et le système s'avèreront meilleurs. De ce fait, les modèles de détection de l'état de blocage auront un impact fortement positif d'un point de vue didactique.

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

Pour conclure, ce mémoire avait pour objectif de trouver une représentation de l'état de blocage et de concevoir un algorithme capable de le prédire. Ces objectifs ont été accomplis et ont validé nos hypothèses de recherche : l'état de blocage peut être représenté avec des séquences d'actions du clavier et de la souris et cette représentation distingue les apprenants bloqués de ceux non bloqués. L'hypothèse d'implication de l'apprenant a été validée méthodologiquement durant les deux expérimentations.

Le modèle hybride M_H avait une performance de 77 % de prédiction avec un bon compromis précision/rappel. Ce modèle était construit à partir de la combinaison linaire des trois modèles : M_F, modèle des fréquences d'actions, M_S, modèle des sous-séquences, et M_C, modèle du réseau de neurones convolutif en une dimension. La proportion de contribution de ces trois modèles à M_H étaient optimalement de 2 %, 18 % et 80 % respectivement. Cela rend donc M_F négligeable. Nous avons aussi observé un deuxième maximum local sur l'espace des hyperparamètres α, ce qui implique une certaine équivalence entre M_F et M_S. Par contre, puisque M_F manque de robustesse face au jeu de données de test, seule la configuration du maximum est considérée. Du point de vue de l'implémentation, M_F est à négliger.

Le comparatif avec la littérature a permis de reconnaître une importance de cette représentation. Une méthodologie rigoureuse, la définition particulière de l'état de blocage et l'utilisation des séquences d'actions amènent un aspect unique à ce mémoire. Rajoutons à cela que les performances des modèles M_H sont au niveau de la littérature, proche de 80 %, avec une évaluation de performance utilisant des mesures plus fiables.

5.2 Limitations de la solution proposée

Plusieurs limitations des modèles sont à considérer, comme la dépendance de la représentation à un STI particulier et les biais des jeux de données. Comme les évènements utilisés sont directement liés à l'interface utilisateur, il y a une dépendance du modèle face à celui-ci, ce qui peut causer des variations d'une interface à une autre. Un changement d'interface utilisateur de QED-Tutrix requerrait un réentraînement complet des modèles et rendrait les jeux de données précédents désuets. Cette dépendance à l'interface utilisateur influence les résultats mêmes des performances des modèles, telle que vue entre l'ensemble d'entraînement et de test. D'ailleurs, le biais des jeux de données peut mettre en doute les résultats obtenus. Or, étant

donné l'analyse des entropies des jeux de données, ce doute peut être mitigé. Cependant, comme QED-Tutrix s'adresse à des adolescents, il est fort probable que les performances soient moins bonnes puisque cette population risque fortement d'être plus haute en entropie dans leurs séquences d'actions. Cela implique aussi qu'il est nécessaire d'obtenir un grand ensemble de données ciblé sur cette population pour mieux couvrir l'ensemble des patrons des sous-séquences propre à cette population.

5.3 Améliorations futures

Les améliorations à apporter aux modèles sont les suivantes : incorporer l'historique des états de blocage dans le calcul des probabilités et inclure explicitement les connaissances en jeu. L'historique des états de blocage, tel qu'il est utilisé dans la littérature, peut être ajouté aux modèles développés sur les actions. Pour ce faire, il faudrait élargir le nombre de variables aléatoires considéré dans le modèle pour inclure l'historique H_{i-1}. Cet ajout devrait améliorer les performances du modèle. Il ne devrait pas nuire à l'avantage du problème de démarrage à froid mentionné ci-haut, car le modèle d'actions demeurait la base même pour un nouvel apprenant.

Pour améliorer les performances davantage, il faudrait intégrer explicitement les connaissances en jeu supposant que ce sont les éléments manipulés par l'apprenant dans son espace de travail mathématique. De ce fait, avoir une idée claire de l'espace des connaissances de l'apprenant permettrait d'élargir les modèles pour inclure celui-ci, un ajout qui devrait être majeur. Il faut aussi mentionner que cette mise à jour permettrait de cibler l'intervention tutorielle, un objectif subséquent à cette recherche. Les données pour créer cette représentation sont déjà accessibles avec les ensembles de données recueillies, car elles sont contenues dans la clé *value* des données d'actions en format JSON.

D'un point de vue théorique, il pourrait être intéressant de définir l'état de blocage suivant la théorie des espaces de travail mathématique. Pour ce faire, les séquences d'actions des apprenants pourraient être associées à des probabilités pour chacune des genèses de l'ETM. De cette approche, il en découlerait des séquences de transition de l'ETM et, ce faisant, des patrons pourraient être observés.

RÉFÉRENCES

S. Abu Naser, “Predicting learners performance using artificial neural networks in linear programming intelligent tutoring system”, vol. 3, 03 2012.

T. Barnes et J. Stamper, “Toward automatic hint generation for logic proof tutoring using historical student data”, dans *Intelligent Tutoring Systems*, B. P. Woolf, E. Aïmeur, R. Nkambou, et S. Lajoie, éds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, pp. 373–382.

J. E. Beck et Y. Gong, “Wheel-spinning : Students who fail to master a skill”, dans *Artificial Intelligence in Education*, H. C. Lane, K. Yacef, J. Mostow, et P. Pavlik, éds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, pp. 431–440.

L. Font, P. R. Richard, et M. Gagnon, “Improving qed-tutrix by automating the generation of proofs”, *Proceedings 6th International Workshop on Theorem proving components for Educational software*, 2017.

A. Gabadinho, G. Ritschard, M. Studer, et N. S. Müller, “Mining sequence data in r with the traminer package : A users guide for version 1.2”, *Geneva : University of Geneva*, 2009.

A. Gabadinho, G. Ritschard, N. S. Mueller, et M. Studer, “Analyzing and visualizing state sequences in r with traminer”, *Journal of Statistical Software*, vol. 40, no. 4, pp. 1–37, 2011.

Y. Gong et J. E. Beck, “Towards detecting wheel-spinning : Future failure in mastery learning”, dans *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. ACM, 2015, pp. 67–74.

M. Hohenwarter et K. Fuchs, “Combination of dynamic geometry, algebra and calculus in the software system geogebra”, dans *Computer Algebra Systems and Dynamic Geometry Systems in Mathematics Teaching Conference*, 2004.

A. Kuzniak, D. Tanguay, et I. Elia, “Mathematical working spaces in schooling : an introduction”, *ZDM*, vol. 48, no. 6, pp. 721–737, Oct 2016. DOI : 10.1007/s11858-016-0812-x. En ligne : https://doi.org/10.1007/s11858-016-0812-x

N. Leduc, “Qed-tutrix : Système tutoriel intelligent pour l’accompagnement des élèves en situation de résolution de problèmes de démonstration en géométrie plane”, Thèse de doctorat, Ecole Polytechnique, Montreal (Canada), 2016.

N. Leduc, M. Tessier-Baillargeon, J.-P. Corbeil, P. R. Richard, et M. Gagnon, “Étude prospective d’un système tutoriel à l’aide du modèle des espaces de travail mathématique”, 2016.

N. Matsuda, S. Chandrasekaran, et J. C. Stamper, “How quickly can wheel spinning be detected ?” dans *EDM*, 2016, pp. 607–608.

P. Rabardel, *Les hommes et les technologies ; approche cognitive des instruments contemporains*. Armand Colin, 1995. En ligne : `https://hal.archives-ouvertes.fr/hal-01017462`

P. R. Richard, J. M. Fortuny, M. Gagnon, N. Leduc, E. Puertas, et M. Tessier-Baillargeon, “Didactic and theoretical-based perspectives in the experimental development of an intelligent tutorial system for the learning of geometry”, *ZDM*, vol. 43, no. 3, pp. 425–439, Jul 2011. DOI : `10.1007/s11858-011-0320-y`. En ligne : `https://doi.org/10.1007/s11858-011-0320-y`

J. C. Stamper, M. Eagle, T. Barnes, et M. Croy, “Experimental evaluation of automatic hint generation for a logic tutor”, dans *Artificial Intelligence in Education*, G. Biswas, S. Bull, J. Kay, et A. Mitrovic, éds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, pp. 345–352.

M. Tessier-Baillargeon, “Geogebratutor : développement d’un systéme tutoriel autonome pour l’accompagnement d’élèves en situation de résolution de problèmes de démonstration en géométrie plane et genèse d’un espace de travail géométrique idoine”, Thèse de doctorat, Université de Montréal, Montreal (Canada), 2016.

J. Xie, A. Essa, S. Mojarad, R. Baker, K. Shubeck, et X. Hu, “Student learning strategies and behaviors to predict success in an online adaptive mathematics tutoring system”, *Proceedings of the 10th International Conference on Educational Data Mining*, 01 2017.

ANNEXE A

ARTICLE 1 :

A DATA MINING APPROACH TO DETECT BLOCKING STATES IN INTELLIGENT TUTORING SYSTEM

Jean-Philippe Corbeil
Polytechnique Montréal
jean-philippe.corbeil@polymtl.ca

Michel Gagnon
Polytechnique Montréal
michel.gagnon@polymtl.ca

Philippe R. Richard
Université de Montréal
philippe.r.richard@umontreal.ca

Soumis à *Journal of Educational Data Mining*

A.1 Introduction

Intelligent tutoring systems (ITS) are important tools for teachers as well as students, because they can provide an adaptable framework for teaching and learning. An ITS usually provides a bank of problems and has some tutoring strategies that can be used to guide the student. Yet, one fundamental aspect of ITSs is the detection of moments where the learner is in need of help. In other words, the tutoring system needs to interact with the learner at "right" moments. That is, if the ITS interacts too quickly, it will disturb the learner, who will lose his chain of thoughts or be annoyed by the intervention. If the ITS intervenes too slowly, the learner will not get enough support from the system and this could lead him to give up the resolution process. One solution is to let the student asks for help on his own. Still, most often, the learner will inquire help out of context or game the ITS. In this paper, we present our solution that detects the moments when learners are in a blocking state. This solution is based on probabilistic models trained on sequential data gathered from an experimentation with QED-Tutrix (Leduc (2016); Tessier-Baillargeon (2016); Richard et al. (2011)), an intelligent tutoring system for Euclidean geometry.

QED-Tutrix is an intelligent tutorial system built to help students to produce a proof for a given problem (Leduc (2016); Tessier-Baillargeon (2016)) in Euclidean geometry. QED-

Tutrix aims to support students according to strong didactic principles (Richard et al. (2011)). It runs on top of Geogebra (Hohenwarter and Fuchs (2004)), a dynamic geometry software. The student is asked to submit inferences (hypothesis, results or justification), using a list of possible inferences. QED-Tutrix uses these submitted inferences to keep track of the student's progress in one of the many possible solutions. This exploratory solving process is in opposition with most ITSs, which usually constraint the learner to use forward or backward chaining in his resolution process. The solution management is done with the HPDIC graph (Leduc (2016); Font et al. (2017)), which contains all the solution paths from hypothesis nodes to the conclusion node. The student and the tutorial system can interact through a chat section and an inference submission section. The finite state machine actually managing interactions is tuned to give feedback on inferences and to interact every minute with the student, providing support on the most advanced solution. This interaction is therefore limited and cannot identify the student's cognitive state. This leads us to an important question that is addressed in the work presented in this paper: Can we detect a blocking state when a learner is in need of help, by considering only the actions on the interface of QED-Tutrix?

The objective of this research is to predict this cognitive state, which we represent by a boolean variable H ($H = true$ if the student is in a blocking state), according to a set of measurable features. These features are limited to a small set of keyboard and mouse actions. Furthermore, we want to define the necessary features from which we can detect a blocking state according to a probabilistic framework. The features are frequencies of actions and subsequences of actions which have been isolated from sequence analysis done with the library TraMineR from Gabadinho et al. (2009, 2011).

The organisation of this paper is as follows. The next section describes the state of the art in blocking state detection. Then we explain the overall approach we took to gather our train and test datasets in two experimentations and the preprocessing of the data. Section A.4 presents the models we developed to predict H using a probability framework and their performances on the dataset. Finally, we discuss the threats to validity of this research and future works.

A.2 Background

In a previous research, Beck and Gong (2013) showed that it is crucial to help students at the right moment to maintain their involvement and avoid negative behaviours. They defined Wheel spinning, related to our concept of blocking state, as a state where a student fails to master a skill.

More precisely, they used a logistic regression on data coming from the following ITSs for mathematics: the Cognitive Algebra Tutor and ASSISTments. They showed that around 60% of students mastered a skill in ten pratice opportunities. These are attempts in which the learner tries to solve mathematical problems requiring this peculiar skill. Within these 10 opportunities, the skill is mastered if there is 3 exercices in a row successfully solved. Otherwise, after ten attempts, the student is wheel spinning.

A second article from Gong and Beck (2015) developed a Wheel-Spinning detector that focused on three types of feature : the student in-tutor performance, the seriousness of the learner and other general factors. The in-tutor performance is defined as how the student understands the skill at work and is divided into 5 features. They are correct response counts, response times for problems involving this skill and hint requests. The seriousness of the learner is characterized by 8 features across all skills. Those features are based on response time of the learner and the number of hint requests. Finally, the general factors are features like: the total number of problems encountered and the skill identification. Most of these 15 features have been proved correlated positively or negatively to wheel spinning.

In their work, one dominant factor they found to predict wheel spinning is labelled as "correct response in a row count" for in-tutor performance. This metric is the number of time a student has succesfully answered to a problem and is negatively correlated to the wheel-spinning.

They also worked with the two same datasets gathered with the following ITSs for mathematics: CAT and ASSISTments. They had 146,479 problems done by 575 students for CAT and 220,539 problems done by 5,997 students for ASSISTments.

Gong et al. defined a metric of mastery of skill which corresponds to three consecutive problems answered correctly within a given amount of practice problems. This amount of problems is a threshold meaning that after this number of problems the student is considered in wheel-spinning if he did not answered correctly to three problems in a row. Gong et al. chose a limit threshold of 10 consecutive problems on CAT and 15 on ASSISTments attributed on the difficulty level of problems from the ITS.

Their model is based on the logistic regression, which performed better on ASSISTments. On this dataset, their model gets a precision of 76.6% and a recall of 53.1%. The precision is a metric defined as the number of wheel-spinning states succesfully predicted on the total number of predicted wheel-spinning states and the recall metric corresponds to the number of righfully predicted number of wheel-spinning states on the total wheel-spinning states.

Matsuda et al. (2016) built a recurrent neural net, a Bayesian Knowledge Tracer, which they applied on the dataset "Cog Model Discovery Experiment Spring 2010". This dataset

has 2883 sequences of problems solved by learners. They aimed to show how quickly we can detect wheel spinning on 10 consecutive problems solved by learners. Their Bayesian Knowledge Tracker took as inputs the sequence of binary states which are the successes and the fails at a particuliar skill. From these responses, they calculate the learning of the skill in the first layer of the neural net. Afterward, they compute the slope of learning showing rate of progression in learning inside a second layer. Then, they compute the difference between consecutive slope of learning in the last layer. Finally, they feed the differences to an activation layer which gives the probability of wheel spinning.

They showed with their Bayesian Knowledge Tracker that over 10 consecutive problems they can predict wheel spinning with good recall, but poor precision. With a 10-fold cross-validation, they obtained a F_1 score around 40%, with a precision around 25% and an average recall of 80%. On the prediction for each consecutive problems, they demonstrated a nearly constant precision and F_1 score, but a dropping recall from 89% on the first problem to 71% on the last problem.

There is two distinctions between our definition of blocking state and the Wheel-Spinning state from Beck and Gong. First, while we are representing action behaviours from the mathematical work of the student, Beck and Gong used the mastery of skills of the student. Since our goal is to detect a student in need of help within a given problem instead of on ten problems, we defined our blocking state directly on the actions of the student. Second, we are not trying to predict a state where the learner failed to master a skill. Rather, we are trying to predict moments where the learner is stuck and makes no progression toward finding a solution. Thus, we make no assumption about wether the blocking state is temporary or a dead-end to progression, the latter being closer to the wheel-spinning.

From a performance perspective, we observe that the Beck and Gong's logistic regression with a large set of features has good precision, but is weak on recall. In the case of Matsuda et al., their recurrent neural net looking only on response sequence is weak on precision, but really strong on recall.

Both models showed cold start issue, since we need many results from learner to detect wheel spinning.

A.3 Methodology

A.3.1 Data gathering

The training data was gathered from a group of 19 second-year students from Université de Montréal, in Mathematics Education program. Their age is around 20 years, both sexes were

well represented and their background in mathematics is a one-year study focused only in mathematics. They were asked to solve four Euclidean geometry problems in QED-Tutrix. The gathered data was fully anonymized.

The test data was gathered from a group of 4 graduated students in software engineering. Their age is around 24 years old. They were asked to solve the same four Euclidean geometry problems.

A.3.2 Experimental Set Up

The gathered data is about actions of the students on the QED-Tutrix interface. We used the version of QED-Tutrix shown in Figure A.1. We have a *help* button on the interface, which is an indicator of a bloking state where the student is conscious of his own cognitive state. Learners were instructed to click on this button when they felt that they could not progress anymore in their solution.

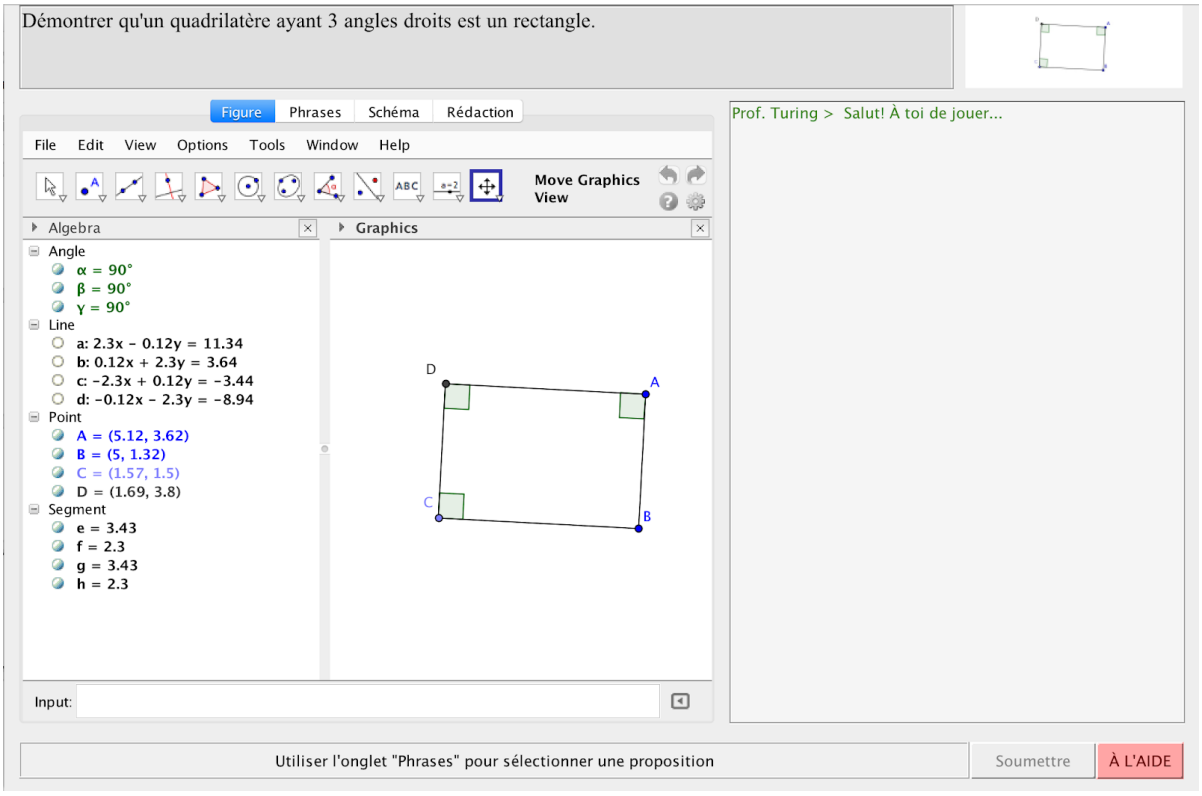

Figure A.1 QED-Tutrix (modified version) for this study. We can see the help button in red, bottom-right corner.

We decided to remove tutor interactions at every minutes to avoid tutorial bias which might affect the way that blocking states happens. We kept the positive and negative feedback

from the tutor at inferences submitted by the learner.

Furthermore, we added listeners in QED-Tutrix. These monitored mouse and keyboard actions that may occur in the resolution of a problem are the following ones:

- Enter (E) : the mouse enters a region of the interface.
- Click (C) : the user clicks on a region on the interface (panel, button, etc.).
- Change tab (T): the user switches to another panel (GeoGebra panel, Sentences panel or Writing panel Leduc (2016); Tessier-Baillargeon (2016)).
- Scroll chat box (SC): the user is scrolling to look at the history of interactions with QED-Tutrix.
- Select filter (FI): the user selects one of the filters to choose an inference (Sentences panel).
- Chose sentence (CS): user chooses an inference.

To these actions, we add special actions that are not part of the resolution, but are useful to indicate the end of a significant action sequence:

- Submit: the user completed an inference and clicks the Submit button to get QED-Tutrix's feedback.
- Exit: the user ended his resolution.
- Help flag: the user clicks the help button.

When the learner moves the mouse from panel to panel on QED-Tutrix, we get an E event (enter). When the learner clicks on any element, we have a C event (click). In QED-Tutrix, the student can change one panel in the centre of the interface with tabs, which corresponds to a T event (change tab). These tabs are GeoGebra, Sentence and Writing panels. In the Sentences panel, the learner has 5 filters to help him find the inference to submit to the tutor. These filters are mathematical object or propriety names that once selected filter the set of all inferences to only those containing every chosen filters. When the learner selects a filter, an FI event is gathered. When the student selects an inference he wants to complete, the event is CS (chose sentence). Yet, the student still has to fill in the blanks in the inferences to submit it (empty boxes). For instance, the learner has to type the four letters of the four corners of a rectangle if the inference is about a specific rectangle. When this is done, learner submit the inference (submit event). If the learner needs to revise his resolution with the chat box interactions, it is a SC event (scroll chat box). Finally, when the proof is completed or time is up, the closing of the application is marked by an Exit event.

A.3.3 Experimentation Details

The experimentation for the training set took place in Polytechnique Montréal on the 2nd of October 2017. First, students were given 20 minutes to get familiar with the software original version. They had to solve the classical rectangle problem. After that, using the modified version of QED-Tutrix, they were given 30 minutes to solve each problem. They were instructed to push the help button on the interface if they felt in need of help, but no help would be provided from this action. The group was partitioned into three subgroups. To limit fatigue effects, each subgroup had a different order of problems. Bias effects were mitigated by avoiding providing direct help to the students. So, we desactivated the modules that makes the tutor intervene at each minute and kept only the immediate feedback to learner's submissions. The same methodology was applied with the test set group on the 9th May 2018.

The difference of populations between the training and the test set is explained by our wish to show the generality of our models. Obviously, the learners from Mathematics Education are at ease with the mathematical content in QED-Tutrix and less with the software. On the contrary, the learners from software engineering are good with the software and less with the mathematical content. However, we argue that their actions on QED-Tutrix, when they are in a blocking state, should be similar. This is a consequence from QED-Tutrix workflow, which we will discussed in the next section.

A.3.4 Blocking State Definitions

Two blocking states are considered. The first one is a state where the student **can't progress in his proof**. This state is associated with a lack in the knowledge set or the usage schemes set of the student. The second type of blocking state that may occur when the student uses an **erroneous reasoning**. This state will be associated with a false element in the knowledge set or in the usage schemes set of the student. By definition, we consider that the learner is in either blocking states when a Help flag action occurs or when a negative feedback is given by QED-Tutrix after a *submit* action. We will consider that $H = true$ when one of these two events occurs, and $H = false$ for every other event that ends an action sequence (for example, when the proof is completed, or when an inference is submitted without negative feedback). In this setup, the two types of blocking state are supposed indiscernible, that is, the sequence of actions we are using do not provide enough information to distinguish these two states.

A.3.5 Data Preprocessing

Our definition of blocking state leads to a preprocessing of sequence of actions executed by the learner during the resolution of a problem. A resolution session is a sequence of actions that can be partitioned into mutuality exclusive subsequences, each one ending with one of the following situations, which correspond to a blocking state ($H = true$): Help Flag action, negative feedback from QEDX, or end of the session without completing the proof. A subsequence is considered as a non-blocking state ($H = false$) when its last action is the submission of an inference that received positive feedback. In other words, it happens when the system has validated the end process of a sequence of actions.

More formally, let us denote $S_1, S_2, \ldots S_i \ldots, S_n$ the n subsequences of actions of a student trying to solve a problem. We define one boolean variable H_i for each subsequence S_i. The value of H_i is $true$ if S_i is identified as a blocking state, according to the rules explained earlier.

Thus, each state H_i (either blocking or not) will be associated with the subsequence of previous actions that spans from the previous state H_{i-1} (or beginning of the proof) up to the state H_i. If the learner interrupts the resolution of the problem without completing the proof, it is considered as a blocking state. Our hypothesis is that a sequence of previous actions representative of the cognitive state of the learner.

We also suppose that there is a function f for the probability distribution of the blocking state for a given subsequence:

$$P\left[H = true|\, S_i\right] = f\left(S_i\right) \tag{A.1}$$

The class of all possible functions f represents instances of predictive models for blocking states.

Note that our models will not take into account the overall historic of a proof $[S_1, ..., S_n]$ and $[H_1, ..., H_n]$. We only take the current context of actions S_i and its result H_i to define our model. This is a considerable advantage for cold start issues observed in wheel spinning literature. Instead of using the historic of H_i for all i up to the current state, we solely take the sequence S_i. This input is convenient and is assumed to contain patterns for any learners.

A.3.6 Dataset

From the experimentation, we have collected actions from learners for each proof. Table A.1 presents basic information about the dataset.

Table A.1 Training set information.

Total number of actions	39777
Total number of resolutions	72
Total number of sequences	1436
Total number of blocking states ($H = true$)	148

The dataset used to train our models contains 72 proofs from our 19 participants. Using the preprocessing, we extracted a total of 1436 sequences from the 72 proofs. Among these sequences, there are 148 that could be labelled as blocking states. We clearly see that the blocking state is our minority class, nearly 1:10.

The 72 progression trajectories of the learners in our training set session are shown in Figure A.2. We can observe that most of our learners tend to progress quickly within the first 10 minutes. In most progression trajectories, we observe a lot of blocking states where no or few progress is made toward the completion.

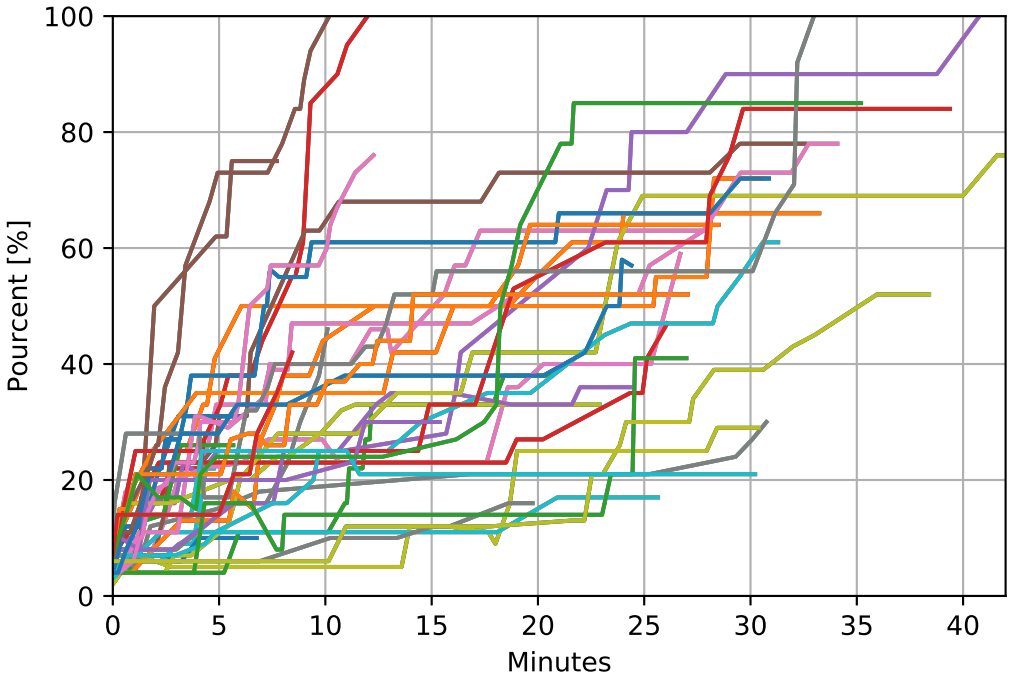

Figure A.2 Progressions of percentages of proof completion in relation with time in minutes. Each line corresponds to the resolution of one problem by one student in the training set.

The test set details are displayed in Table A.2. We note in this dataset that there is fewer blocking states observed. We dropped from about 10% in training set to near 6%, which

might be explained by the greater ease of use of the software from the test population.

Table A.2 Test set information.

Total number of actions	17796
Total number of resolutions	15
Total number of sequences	705
Total number of blocking states ($H = true$)	40

A.3.7 Model Validation

We have separated our training dataset into training/validation sets (85/15), being careful to maintain similar distributions of sequence length and H states. The training is done with all the training set. For validation, sequences were sampled such as to obtain equal numbers of non-blocking and blocking states from validation set. The sampling procedure is done 10 times and performance metrics are also evaluated 10 times (precision, recall and F_1-score). We take the average of these as our overall performance.

The test is done by computing the model with training data and predicting on validation set.

A.4 Models

We built four models. First, the action frequency distribution model uses distribution of actions in each one of the two possible H states ($H = true$ and $H = false$). These distributions are related to the number of occurrences of each action in a subsequence. The second model is the subsequence detection model. This model identifies recurrent patterns of actions. More specifically, we estimate the probability of a pattern (which is in fact a sequence of actions) by counting the number of occurrences of this pattern in all subsequences of our dataset. We compute the probabilities for both possible H states. The third model is a 1D convolutional neural network. This CNN is also looking for recurrent patterns in sequences through many filters within a given window. Finally, we instantiate one last model that combines all previous models.

A.4.1 Action Frequency Model M_F

The intuition for M_F comes from the observation of Figures A.3 and A.4.

These figures show, for the ith position (represented by Xi in the horizontal axes in the figures) in a sequence, the ratio of occurrences at this position for each possible action. The first figure

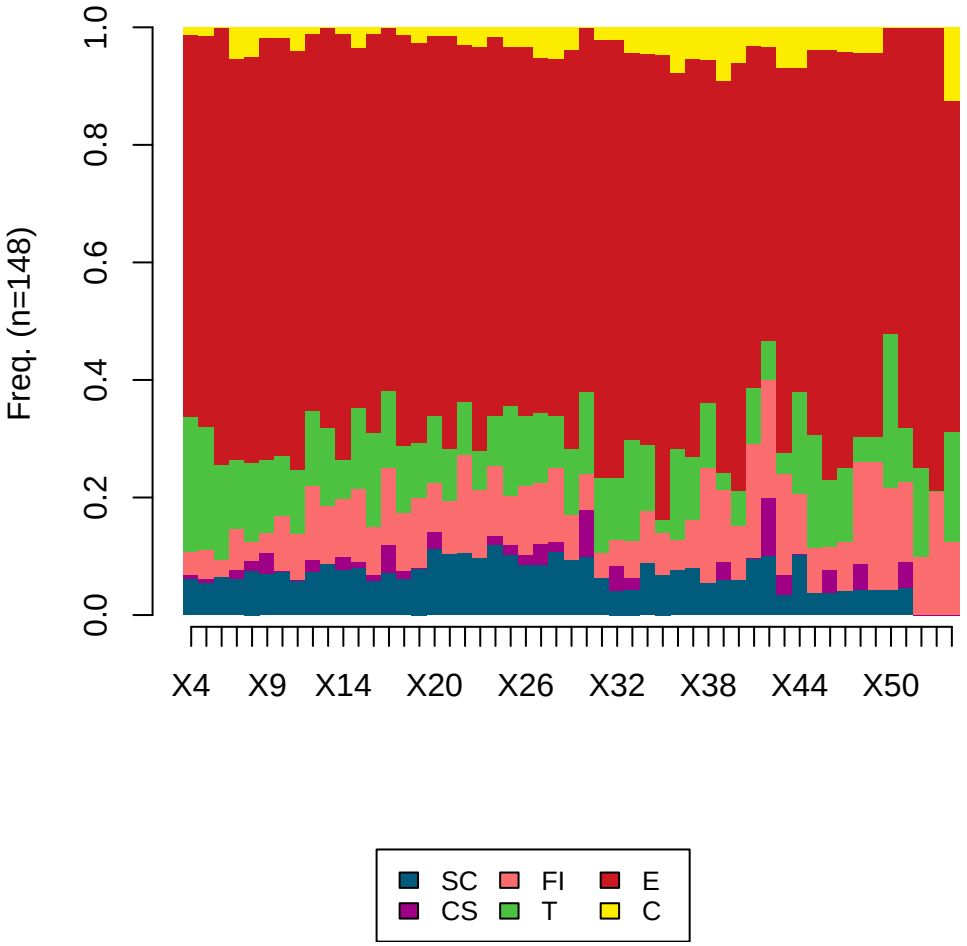

Figure A.3 Distributions of action frequency in blocking state sequences.

shows the distribution for blocking states, while the second one shows the distribution for non-blocking state. These distributions represent stochastic processes that are quasi-stationary across all positions in the sequence. We observe differences in the distributions. For instance, action SC (scroll chat box) is more frequent in blocking state sequences, whereas action CS (choose sentence) is more frequent and more evenly distributed in non-blocking sequence. Since all frequencies of action oscillate around an average value, we chose to represent the quasi-stationary stochastic process with a Gaussian function independent of the position in the sequence.

Thus, let A be the set of all actions $\{SC, CS, FI, T, E, C\}$ and $a \in A$. If we consider a

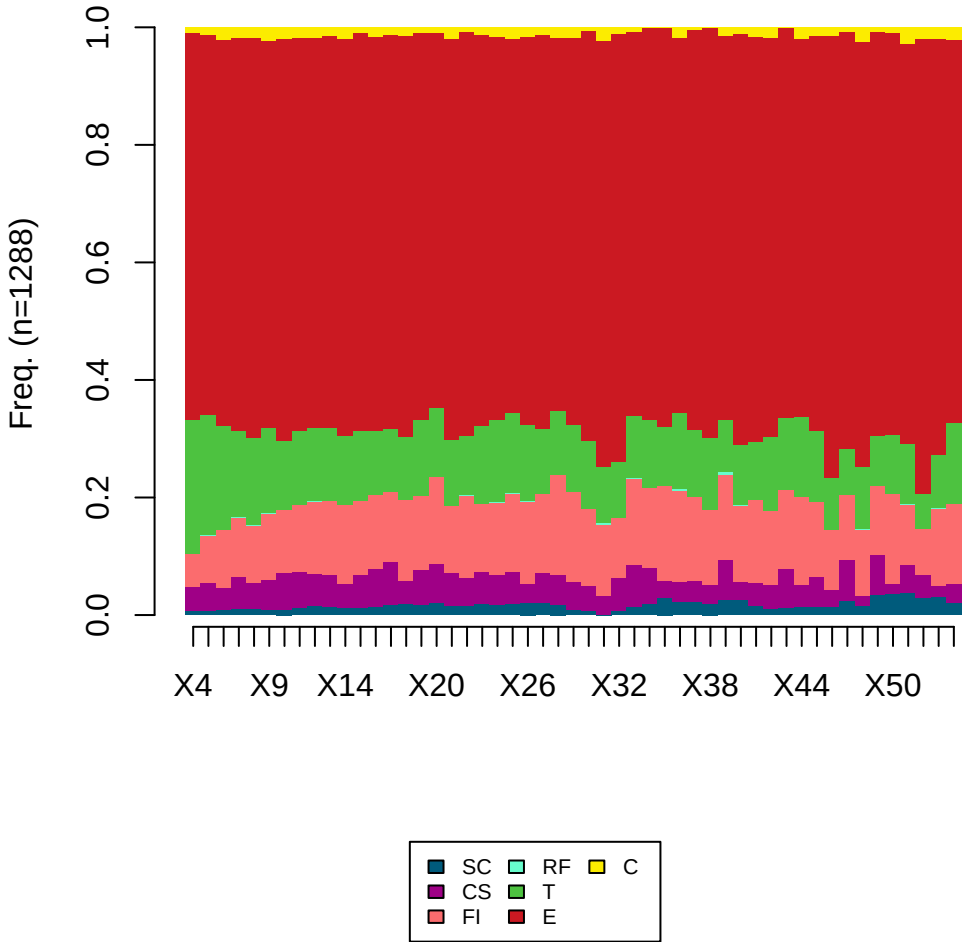

Figure A.4 Distributions of action frequency in non-blocking state sequences.

sequence S_i, we can compute the probability of a blocking state H for S_i by looking at the frequency of actions. For one specific sequence S_i, let $f_a(S_i)$ be the ratio of occurrences of action a in sequence S_i, that is, the number of occurrences of a in S_i over the total number of actions in S_i.

We need to estimate the distribution of f_a, the proportion of the action a, for both possible values of H ($true$ and $false$), to determine which one corresponds to our specific observation $f_a(S_i)$. The distributions of f_a for each H value is approximated with a Gaussian distribution. This frequency distribution will be fitted on our training set by taking the average of the proportions across the positions Xi (Figures A.3 and A.4) in the sequence and computing

the standard deviation.

So we can assimilate the probability $P[H|S_i, a]$ as the p-value of a null hypothesis test in which we know both distributions. The hypothesis is that $f_a(S_i)$ is from $\mathcal{N}_{a|H}$ (the distribution of f_a) for $H = 1$ or $H = 0$. The higher this probability, the more likely it is to come from this peculiar distribution. In Figure A.5, we have schematized the distributions $\mathcal{N}_{a|H}$ and the observation $f_a(S_i)$.

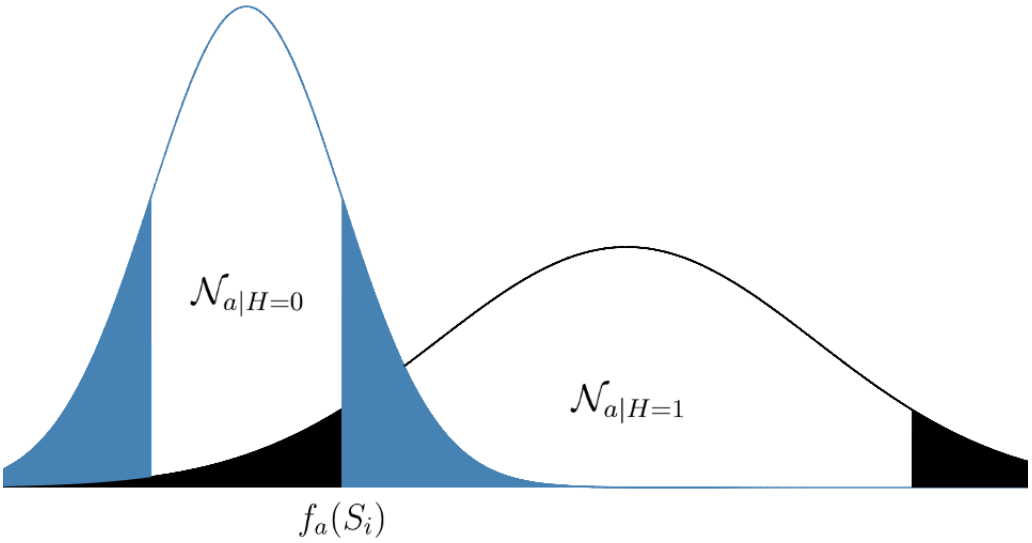

Figure A.5 Scheme of Gaussian distributions of action frequencies given $H = 0$ and $H = 1$. We observe the probability of the observed frequency $f_a(S_i)$ as surfaces on the tails of both distribution.

To compute it, considering the $Gaussian \sim \mathcal{N}(0, 1)$ and the scheme in Figure A.5, we have the same formula as for the double tail event null hypothesis test:

$$\begin{aligned} P[H|S_i, a] &\simeq 2 \cdot min\{P\left[f_a(S_i) \leq \mathcal{N}_{a|H}\right], P\left[f_a(S_i) \geq \mathcal{N}_{a|H}\right]\} \\ &\simeq 2 \cdot P\left[\frac{|f_a(S_i) - \mu_{a|H}|}{\sigma_{a|H}} \leq \mathcal{N}\right] \\ &\simeq 2 \cdot \int_{f_a(S_i)}^{\infty} Gaussian\left(\frac{|z - \mu_{a|H}|}{\sigma_{a|H}}\right) \frac{\mathrm{d}z}{\sigma_{a|H}} \end{aligned} \tag{A.2}$$

So, we can combine for all actions:

$$P[H|S_i]_{M_F} = \sum_{a \in A} \omega_a P[H|S_i, a] \tag{A.3}$$

where ω_a are weights favouring the most distinguishable actions and we have to normalize $\sum_H P\left[H|S_i\right]_{M_F} = 1$.

A.4.2 Subsequence Detection M_S

For Λ, the set of recurrent patterns, we defined the following conditional probability:

$$\begin{aligned} P\left[H|S_i\right]_{M_S} &= \frac{P\left[H, S_i\right]}{P\left[S_i\right]} \\ &= \frac{\sum_{\lambda\in\Lambda} P\left[H, S_i, \lambda\right]}{\sum_{\lambda\in\Lambda, H} P\left[H, S_i, \lambda\right]} \\ &= \frac{\sum_{\lambda\in\Lambda} P\left[S_i|\lambda\right] P\left[\lambda|H\right] P\left[H\right]}{\sum_{\lambda\in\Lambda, H} P\left[S_i|\lambda\right] P\left[\lambda|H\right] P\left[H\right]} \end{aligned} \tag{A.4}$$

Where we assumed the conditional independence between the random variable λ and H given S_i. We can substitute the probabilities with matrix arguments if we consider the matrix of counts such that subsequences are rows and patterns are columns (similar as the doc-term matrix in information processing), noted A_{sc}. The subsequence S_i is the row i in A_{ic}. With our methodology that splits the data equally for H, we get $P\left[H = true\right] = P\left[H = false\right]$. We then have

$$P\left[H|S_i\right]_{M_S} = \frac{\sum_{c,s\in S(H)} A_{ic}A_{sc}}{\sum_H \sum_{c,s\in S(H)} A_{ic}A_{sc}} \tag{A.5}$$

where $S\left(H\right)$ is the set of rows without i and with the specific blocking state H.

A.4.3 Convolutional Network Model M_C

The third model is a 1D convolutional neural network (CNN). The input of the network is our S_i and the output is a probability that the learner is in need of help $P\left[H|S_i\right]_{M_C}$.

The network topology is sequential and is as follow: 1D CNN (40 filters, 6 actions kernel size, relu), Dropout layer (40 %), 1D CNN (15 filters, 3 actions kernel size, relu), Flatten layer, Dropout layer (40 %) and dense layer (sigmoid activation). The model is built with Keras framework. The topology and the hyperparameters were tuned by hand on the training set.

A.4.4 Hybrid Model M_H

For Σ, the set of models $\{M_F, M_S, M_C\}$, we have the probability of being in a state H given a subsequence S_i:

$$P\left[H|S_i\right] = \sum_{M \in \Sigma} P\left[H|S_i, M\right] P\left[M\right] \tag{A.6}$$

However, we will work with the following modification, which emphasizes on the hyperparameters α_M.

$$P\left[H|S_i\right] = \sum_{M \in \Sigma} \alpha_M P\left[H|S_i, M\right] \tag{A.7}$$

where $\sum_{M \in \Sigma} \alpha_M = 1$. In the current notation, we can write:

$$P\left[H|S_i\right] = \sum_{M \in \Sigma} \alpha_M P\left[H|S_i\right]_M \tag{A.8}$$

We will use the training set and test set to optimize the α_M and since we have three models, we can set the last formula to two hyperparameters:

$$P\left[H|S_i\right] \quad = \quad \alpha_1 P\left[H|S_i\right]_{M_C} \quad + \quad \alpha_2 P\left[H|S_i\right]_{M_S} \quad + \quad \left(1 - \alpha_1 - \alpha_2\right) P\left[H|S_i\right]_{M_F} \tag{A.9}$$

A.5 Results

A.5.1 Models' Performances

In our evaluation, we will consider the F_1-score, because it includes both the precision and recall. This implies that we aim for models that are both good at identifying a sequence of actions in a blocking state and finding as much as possible these sequences. We need to maximize precision and recall, because we want the ITS to always correctly interact with the learner when he is in a blocking state, which requires high recall, and to never confuse the learner with inadequate assistance if he is not in a blocking state, which requires high precision.

In Figure A.6, we can see the F_1 score metric for all possible values α_1 and α_2. We clearly observe a performance peak at $\alpha_1 = 0.184$ and $\alpha_2 = 0.796$. The global maximum F_1 score is 80.4. The proportion of action frequency model is deduce with $1 - \alpha_1 - \alpha_2$, which gives

0.02. This low proportion of action frequency model seems to indicate that this model is irrelevant. However, from Figure A.6, we denote another local maximum of 80 % close to $\alpha_1 = 0.22$ and $\alpha_2 = 0.05$. This indicates similar performances from M_F and M_S.

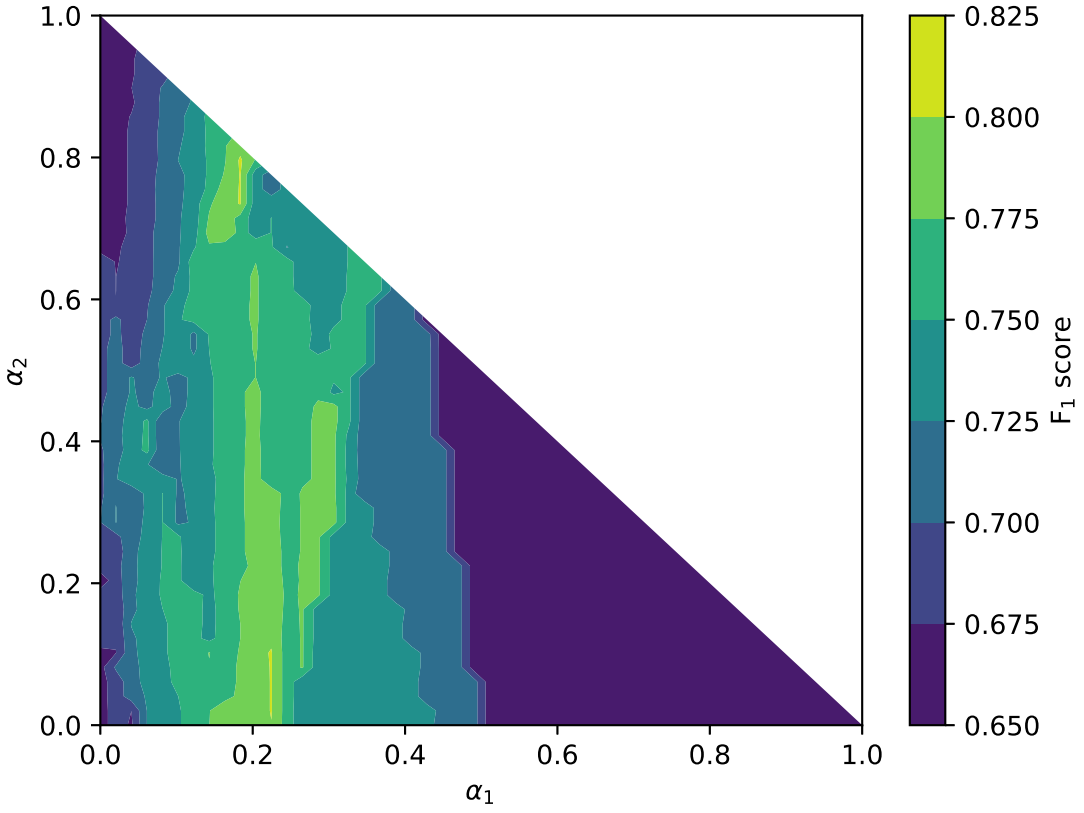

Figure A.6 Performances across the hyperparameters α_1 and α_2, respectively the proportion of convolutional network model and subsequences model. The proportion of the action frequency model is deduced with $1 - \alpha_1 - \alpha_2$.

Table A.3 shows the F_1 score for the linear combinations of probabilities from models M_C, M_S and M_F. We observe that an hybrid model outperforms all the others.

Table A.3 Average performance metrics for all models on training/validation sets (85/15) with 10 samplings for prediction of H for $\alpha_1 = 0.184$ and $\alpha_2 = 0.796$.

Metrics	M_F	M_S	M_C	M_H
Precision	57.9	56.8	99.0	83.4
Recall	77.8	77.8	50.0	77.8
F_1-*score*	66.3	65.5	66.4	80.4

The precision with M_C is high at 99%, which is probably caused by some overfitting. Yet, it struggles on recall with 50%. This mean that M_C can be sure that the learner is in need of help, but it is not the best model to find new sequences in this state. On the other hand, M_F and M_S are good to find new sequences in need of help with recalls of 77.8. Still, they are not precise, (57.9% and 56.8% respectively). The F_1-score for these three models are very close. Nevertheless, combining nearly two of the three models into M_H results on outstanding performances which combine the best of all models. This linear combination leads to a F_1-score of 80.4% with a good precision. Also, we can note that the precision and recall are close.

Table A.4 Performance metrics for all models on test set with leave one out for prediction of H for $\alpha_1 = 0.184$ and $\alpha_2 = 0.796$.

Metrics	M_F	M_S	M_C	M_H
Precision	48.6	56.4	92.4	79.9
Recall	62.5	82.5	65.0	75.0
F_1-*score*	54.6	67.0	76.3	77.3

From prediction on test set in Table A.4, with the best model M_H, we get slightly lower performances than on train set, with a F_1-score for blocking state of 77.3% and precision of 79.9%. This result is still very good since both populations have great differences. This indicates clearly that the behaviours of both groups are very close on QED-Tutrix's interface when they are in a blocking state. Yet, we note a drastically lower performance from M_F on the test set, which indicates underfitting. These results point out to an hybrid model composed with only M_S and M_H. M_C offering close performance to M_H, M_H stays a better option maximizing both precision and recall.

From M_S and M_C, we observe a better performance than on training set. We think that this better performance is also due to population differences. We argue that a subset of the learner's subsequences learned from our training set are more frequently used the software engineering graduated students. This claim seems reinforced by the higher recall of M_S and M_C, which is the capacity of the model to find less false negative. To prove this point, we have calculated the entropy $\mathcal{H}$ of subsequences of both training set and test set in Table A.5 with equation A.10. We observe a drastically lower entropy in our validation set, which represents a drop 1.5 in entropy.

$$\mathcal{H} = -\sum_{\lambda \in \Lambda} p_\lambda ln\left(p_\lambda\right) \tag{A.10}$$

Table A.5 Entropy for subsequences of actions for both training and validation sets.

Dataset	Entropy
Training	3.99
Validation	2.53

A.5.2 Misclassified blocking states

When looking at missed blocking states, we can clearly identify patterns leading to misclassification. First, the hybrid model seems to associate the chosen sentence action *CS* more to non-blocking states than blocking states, which causes most blocking state with *CS* action in the sequence to be harder to identify sucessfully. We also see the same association with the presence of *FI*, filters selection in the sentences panel. On the other hand, we notice that *SC* and *C* actions, scrolling in the historic of the chatbox and click, respectively, are strong indicators of blocking state. These observations are confirmed by looking at the differences in the distributions of action frequency in Figures A.3 and A.4. These have consequences on the type of sequence identified by our hybrid model. Most of the identified blocking states are simple sequences with E and T actions. Thus, these sequences are mostly learners roaming across the user interface. However, we have observed previously the action frequency model M_F's weaker performances. Thus, the datasets need to be from target population, that is, high school students. This population will most likely be more entropic than both our datasets. Therefore, in order to achieve better performance, M_H needs to be trained on a larger dataset.

A.6 Threats to validity

A.6.1 Internal threats

For the experimentation, internal threats were mitigated in the design of experimentation. They are the threats that compromise the validity of the relations between variables at study. Maturation threat, defined as changes in dependent variable during experimentation, was mitigated with problem rotation in the methodology. Further, we also removed tutorial biases by disabling interventions from QED-Tutrix. We kept only direct feedback to submitted inferences. Testing threat is present in our experimentation since a pretest with the rectangle problem was done. Yet, it was desirable to reduce blocking states caused by the user interface, since they are not of interest in this study. The instruments of measurement were the same from a software perspective. In addition, design contamination was not observed with a post-experimental interview session with participants.

A.6.2 External Threats

The generalization of this study is not straightforward, since the samples were from a specific population with only one ITS. Here, there is a clear bias in population to university students in mathematics teaching program from Montreal and graduated students from Polytechnique Montreal in software engineering. Yet, from a blocking state detection perspective, we still believe that this study will be close to a larger population, because usage schemes in QED-Tutrix were similar for both sampled populations. In order to gain this generalization, we need to train our models on more diversified data. From an ITS design perspective, the difficulty will be more about how to intervene with learners stuck in a blocking state.

There also might have Multiple-treatment interference which we tried to mitigate with problem rotations. Experimenter effects have been avoided with cautious interactions providing no direct help to students.

A.7 Future Works

We will at first work on gathering of a larger dataset with a large and diversified population of students from high school. Such dataset would enable better generalizations of this blocking state detector. Moreover, we will do more data mining and analyze the knowledge and specific usage schemes in the dataset of study. Such work would contribute to the advancement of learning theory and mathematical working space theory. It would also contribute further to QED-Tutrix development, in which we will try to integrate our algorithms into QED-Tutrix for real-time blocking state detection.

A.8 Conclusion

This work developed probabilistic models for the detection of blocking state, where the learner is in need of tutoring help. The action frequency model M_F is based on quasi-stationary Gaussian stochastic process of frequencies of action. The subsequence model is the model that detects recurrent patterns in action sequences. The convolutional neural network is a 1D CNN with two main CNN layers. The hybrid model, which combines all the others in a linear combination, outperformed the individual models on training set with performance of 83.4% on precision and a F_1-score of 80.4%. On test set, even if the test population had a very different background in mathematics from the training population, it also performs well with a F_1-score of 77.3%. This score is promising for a broader generalization of the hybrid model composed only of M_S and M_C in QED-Tutrix. Moreover, all models are historic- and

user-independent models, which means that they are robust to the cold start issue. The development of our models provides a new promising framework to support learners on ITS at the "right" moment.

A.9 Acknowledgements

We thank the FRQNT (Fonds de Recherche du Québec - Nature et Technologies) and CRSH (Conseil de Recherches en Sciences Humaines) for the grants making this research project possible.