



SORBONNE UNIVERSITÉ  
MASTER ANDROÏDE

---

## Estimer l'adversité aux recommandations

---

UE de projet M1

Saad MOUSSTAID – Ronan HOUÉE

2024

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>État de l’art</b>	<b>2</b>
2.1	Introduction . . . . .	2
2.2	Systèmes de Recommandation en IHM . . . . .	2
2.3	Modèles d’Acceptabilité des Technologies . . . . .	2
2.4	Interaction entre Recommandation et Prise de Décision Humaine . . . . .	3
2.5	Problématiques Spécifiques et Défis . . . . .	3
2.6	Conclusion . . . . .	3
<b>3</b>	<b>Contribution</b>	<b>4</b>
3.1	Conception de l’interface graphique . . . . .	4
3.1.1	Interface principale . . . . .	4
3.1.2	Interface secondaire (multi-armed bandit) . . . . .	6
3.2	Implémentation de l’interface graphique . . . . .	7
3.2.1	Utilisation de PyQt pour le Développement de l’Interface . . . . .	7
3.2.2	Gestion des Événements et Signaux . . . . .	7
3.2.3	Personnalisation de l’interface . . . . .	8
3.2.4	Développement Itératif . . . . .	8
3.3	Intégration des outils pour l’expérimentateur . . . . .	9
3.3.1	Gestion des fichiers de structure . . . . .	9
3.3.2	Gestion des blocs d’expérimentation . . . . .	9
3.3.3	Contrôle des paramètres des recommandations . . . . .	10
3.3.4	Log des événements utilisateur . . . . .	10
3.3.5	Eye tracking . . . . .	11
3.3.6	Questionnaire . . . . .	11
<b>4</b>	<b>Conclusion</b>	<b>14</b>
<b>A</b>	<b>Cahier des charges</b>	<b>16</b>
A.1	Contexte . . . . .	16
A.2	Objectifs . . . . .	17
A.3	Ressources . . . . .	17
A.4	Étapes de réalisation . . . . .	18
<b>B</b>	<b>Manuel utilisateur</b>	<b>19</b>
B.1	Utilisation générale . . . . .	19
B.2	Structure du code . . . . .	19

B.2.1	Recommandation . . . . .	19
B.2.2	Bandit . . . . .	20
B.2.3	Divers . . . . .	20
B.3	Fichiers utilisateur . . . . .	20
B.3.1	Règles . . . . .	20
B.3.2	Exemple . . . . .	21

# Chapitre 1

## Introduction

Dans le monde numérique actuel, la pertinence des recommandations personnalisées joue un rôle central dans l'expérience utilisateur. Cependant, le défi majeur de ces systèmes de recommandation réside dans leur capacité à équilibrer la fréquence et l'exactitude des suggestions faites à l'utilisateur. En effet, une recommandation inappropriée peut non seulement être source de frustration, mais également engendrer une adversité notable chez l'utilisateur, affectant ainsi son engagement futur avec le système.

Ce projet, dirigé par M. Julien Gori et M. Gilles Bailly, explore cette problématique à travers le prisme de l'interaction humain-machine, en combinant analyse des données et psychologie de l'utilisateur. L'objectif est de développer une interface graphique qui facilite la conduite d'expériences pour mesurer précisément cette adversité vis-à-vis des recommandations perçues comme non pertinentes. Cette interface représente un environnement expérimental contrôlé où divers paramètres de recommandation peuvent être manipulés pour observer directement les réactions des utilisateurs.

Pour aborder ce problème sous un nouvel angle, notre interface intègre des simulations inspirées des problématiques du bandit machot (Multi-Armed Bandit - MAB). Cette approche est bien établie pour étudier la prise de décision sous incertitude et nous permet d'examiner si les comportements observés lors d'interactions avec des MAB se transposent au contexte de nos expérimentations sur les recommandations.

L'intégration de ces éléments sont dans un logiciel dédié dont le code est accessible via notre dépôt Git : [github.com/Exteal/Adversity](https://github.com/Exteal/Adversity).

# Chapitre 2

## État de l'art

### 2.1 Introduction

Dans notre exploration des systèmes de recommandation, nous identifions un défi fondamental : l'équilibre entre la fréquence et l'exactitude des recommandations. Conceptuellement simple, ce défi implique un calcul d'espérance de l'acceptation des recommandations. Toutefois, sa complexité réelle émerge de la difficulté à estimer avec précision les variables nécessaires à ce calcul.

### 2.2 Systèmes de Recommandation en IHM

Les systèmes de recommandation en IHM ont été largement étudiés, notamment pour leur capacité à améliorer l'interaction utilisateur-machine en réduisant la complexité et en augmentant l'efficacité des commandes. Ces systèmes tentent souvent de prédire la commande la plus appropriée basée sur le contexte d'utilisation et les préférences historiques de l'utilisateur. Un exemple notable est le projet "CommunityCommands"[\[1\]](#) qui explore différentes méthodologies pour recommander des commandes logicielles aux utilisateurs en fonction de l'utilisation collective et des comportements précédents des utilisateurs.

### 2.3 Modèles d'Acceptabilité des Technologies

Pour évaluer l'acceptabilité des systèmes recommandés, le modèle d'acceptation technologique (TAM) est fréquemment utilisé. Ce modèle mesure l'utilité perçue et la facilité d'utilisation, qui influencent l'intention d'un utilisateur d'adopter une technologie. Dans le contexte des systèmes de recommandation basés sur la personnalité, des études [\[2\]](#) ont montré que bien que ces systèmes soient généralement bien acceptés, ils nécessitent une adaptation pour traiter spécifiquement des recommandations contextuellement adaptées et personnalisées.

## 2.4 Interaction entre Recommandation et Prise de Décision Humaine

La prise de décision humaine dans des scénarios de choix incertains peut être modélisée efficacement par des approches de type Multi-Armed Bandit (MAB). Ces modèles sont particulièrement pertinents pour notre projet car ils permettent de simuler des situations où un utilisateur doit choisir entre plusieurs options avec des récompenses potentielles et des risques. L'application de MAB à la recommandation de commandes pourrait aider à optimiser la balance entre la fréquence et la précision des suggestions faites aux utilisateurs. [3][4]

## 2.5 Problématiques Spécifiques et Défis

La principale difficulté dans l'application des systèmes de recommandation aux IHM réside dans l'évaluation précise de l'adversité des utilisateurs face aux recommandations incorrectes. Ce défi est poussé par le manque de recherches sur les modèles spécifiquement conçus pour évaluer cette adversité. Par ailleurs, la présentation des recommandations influence significativement leur acceptabilité. Des recommandations proactives pourraient être perçues comme intrusives si elles ne sont pas correctement contextualisées ou si elles interrompent le flux de travail de l'utilisateur.

## 2.6 Conclusion

Cet état de l'art révèle une richesse dans la littérature existante sur les systèmes de recommandation en IHM et l'acceptabilité des technologies, mais aussi un besoin clair de recherches plus approfondies sur l'intégration de l'adversité des utilisateurs dans la conception de ces systèmes. Notre projet contribue à cette littérature en proposant une méthode pour équilibrer la suggestion de commandes avec la minimisation de l'adversité, en s'appuyant sur des modèles de prise de décision humaine adaptés à des contextes incertains.

# Chapitre 3

## Contribution

### 3.1 Conception de l'interface graphique

#### 3.1.1 Interface principale

L'interface principale constitue le cœur de notre système expérimental, permettant aux chercheurs de manipuler les paramètres de recommandation et d'observer les réactions des utilisateurs en temps réel. Cette interface comprend **un terminal**, **un menu** de commandes et **une zone de recommandations**, offrant ainsi un cadre contrôlé pour évaluer l'adversité des utilisateurs face aux recommandations.

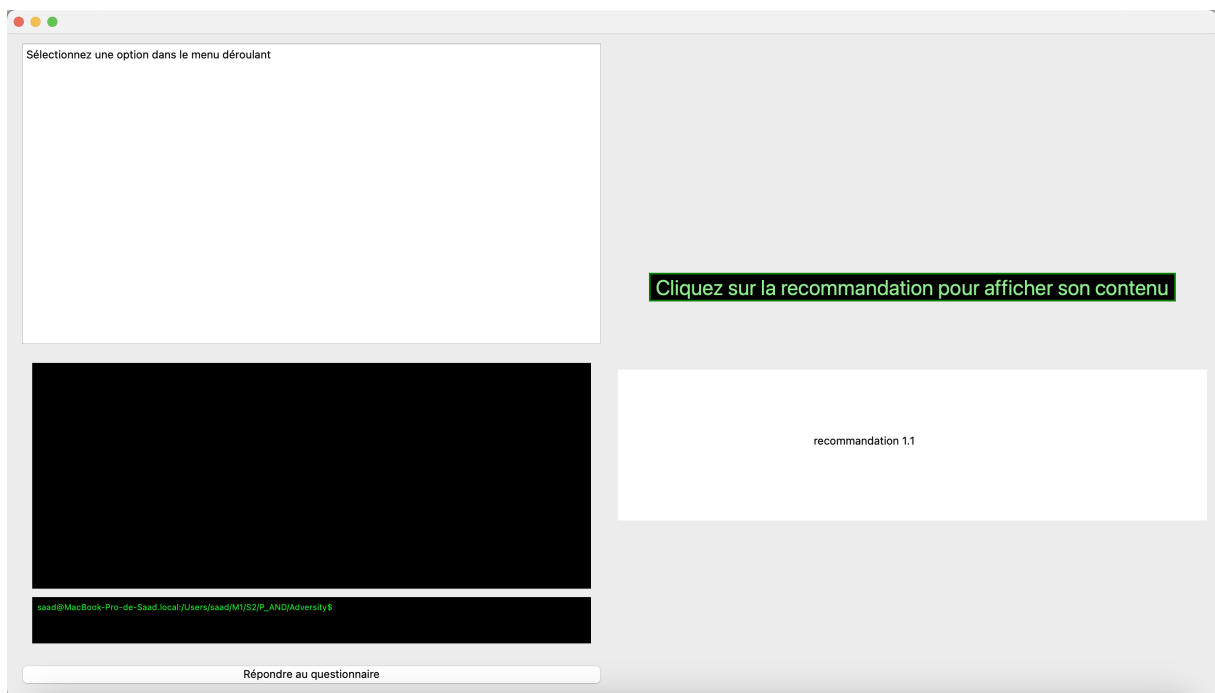


FIGURE 3.1 – Prototype de l'interface

## Dispositions des éléments à l'écran

La fenêtre contenant l'interface prend toute la place à l'écran et les composants de l'interface sont disposés de manière ergonomique pour optimiser l'expérience utilisateur et permettre l'intégration efficace de l'eye tracker. Le menu est placé naturellement en haut à gauche de la fenêtre, le terminal est centré légèrement à gauche, et la partie des recommandations se trouve à droite.

### Le Terminal

Le terminal constitue un espace interactif où l'utilisateur peut saisir des commandes et visualiser les résultats. Il supporte l'exécution de commandes et l'affichage de recommandations en temps réel.

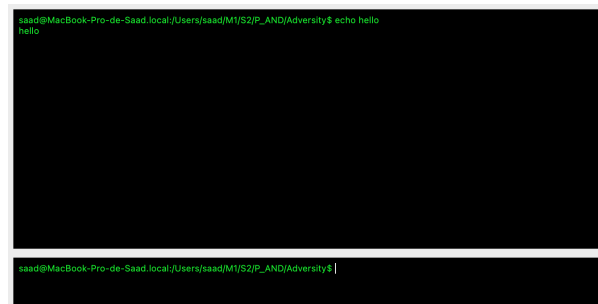


FIGURE 3.2 – Le terminal

### Le menu

Le menu, positionné dans le coin supérieur gauche de l'interface, offre à l'utilisateur la possibilité de sélectionner différentes actions qui seront automatiquement traduites en lignes de commande. Ces commandes générées seront ensuite affichées dans le champ d'entrée du terminal, prêtes à être exécutées.

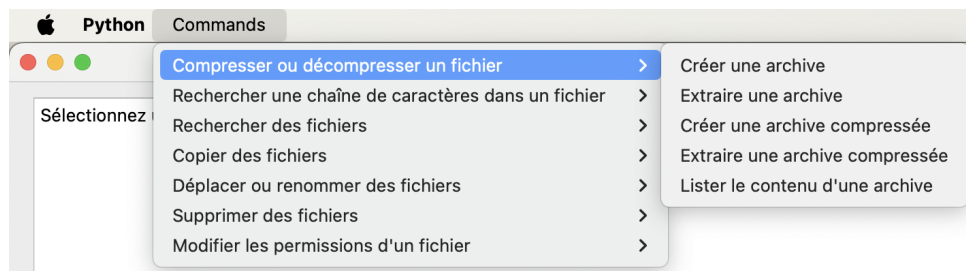


FIGURE 3.3 – Le menu

### Les recommandations

Les recommandations, apparaissant à droite de l'interface, sont le coeur du problème. Elles sont composées d'un titre, et du contenu.

Au moment du clic de l'utilisateur un décompte paramétrable pour chaque recommandation, se lance avant d'afficher le contenu de la recommandation. Cela permet d'influer sur



la pénalité associée à la recommandation, ce qui ajoute une autre dimension à l'analyse du comportement de l'utilisateur.

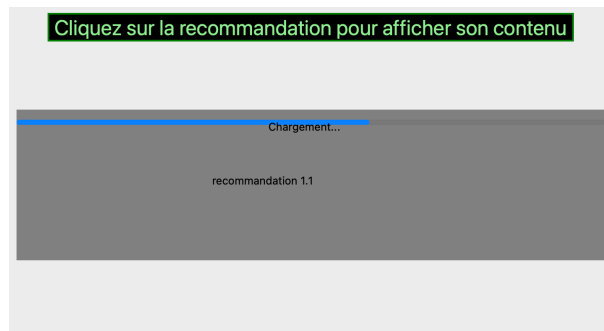


FIGURE 3.4 – Exemple d'une recommandation en chargement

### 3.1.2 Interface secondaire (multi-armed bandit)

Nous avons également conçu une interface avec des machines à sous afin de simuler des expériences de type multi-armed bandit, offrant ainsi aux chercheurs un outil pour évaluer et comparer différentes stratégies de recommandation en temps réel et leur impact sur le comportement utilisateur.

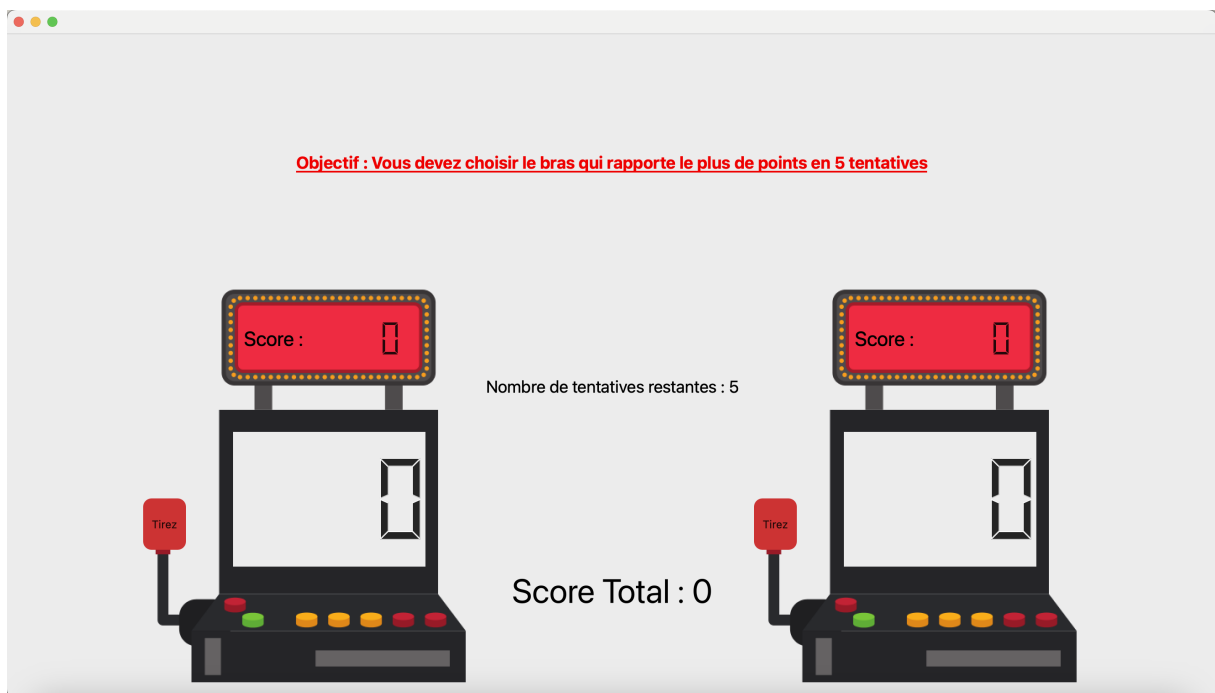


FIGURE 3.5 – L'interface Bandits

### Fonctionnalités

L'interface est conçue pour simuler des scénarios de décision complexes à l'aide de deux machines à sous distinctes, pouvant donner des récompenses lors de chaque tirage. Ces

récompenses sont supposées de deux distributions stationnaires mais inconnues. permettant aux chercheurs de mesurer l'efficacité des différentes stratégies en conditions contrôlées.

### **Conception visuelle des machines à sous**

L'aspect visuel des machines à sous est conçu pour être intuitif, facilitant ainsi la compréhension et l'interaction des utilisateurs avec le système expérimental.

### **Interactions utilisateur**

L'interface permet des interactions utilisateur directes et mesurables avec des retours immédiats sur les choix effectués. L'utilisateur peut choisir une des deux machines pour tirer sur son bras et générer une récompense sur l'écran de la machine. Cette récompense vient s'additionner en suite sur le panneau d'affichage en dessus indiquant le score obtenu avec la machine à l'instant  $t$ .

## **3.2 Implémentation de l'interface graphique**

### **3.2.1 Utilisation de PyQt pour le Développement de l'Interface**

L'implémentation de notre interface graphique repose sur PyQt5, un framework puissant qui facilite la création d'interfaces utilisateur graphiques. PyQt est particulièrement adapté pour des projets nécessitant une intégration profonde avec les fonctionnalités du système d'exploitation tout en fournissant une richesse fonctionnelle et une personnalisation étendue des composants UI.

#### **La fenêtre principale (QMainWindow)**

La fenêtre principale, développée avec QMainWindow, sert de conteneur pour tous les autres widgets de l'application. Elle est configurée pour maximiser l'espace disponible, améliorant ainsi l'expérience utilisateur en fournissant une interface claire et organisée.

#### **Les widgets**

Les widgets utilisés dans l'interface sont principalement des instances de QWidget, qui sont personnalisées pour réaliser des fonctionnalités spécifiques sont définis dans divers fichiers tels que *'terminal.py'* pour la zone de texte du terminal. Chaque widget a été conçu pour intégrer des fonctionnalités spécifiques qui facilitent la navigation et l'utilisation de l'interface par les utilisateurs.

### **3.2.2 Gestion des Événements et Signaux**

PyQt5 gère les interactions utilisateur à travers un système robuste de signaux et de slots, permettant aux widgets de communiquer entre eux de manière efficace. Par exemple, un signal peut être émis lorsqu'un utilisateur ajuste un slider, et un slot correspondant peut être déclenché pour traiter cette interaction.

### 3.2.3 Personnalisation de l'interface

La personnalisation de l'interface est effectuée en utilisant les feuilles de style de PyQt et Qt Designer, un outil puissant qui permet de concevoir et de modifier les interfaces graphiques de manière visuelle. Cette combinaison offre une flexibilité maximale dans la customisation de l'apparence et du comportement des widgets.

#### Utilisation de Qt Designer

Qt Designer est utilisé pour créer et personnaliser les interfaces de manière intuitive, en glissant et déposant les widgets dans la fenêtre de conception. Cela inclut la configuration des éléments suivants :

- **Machines à sous** : Le design des machines à sous est spécialement réalisé à l'aide de Qt Designer pour garantir que leur apparence est attrayante et fonctionnelle. Cela permet de simuler efficacement les scénarios de décision des expérimentations de type multi-armed bandit.
- **Dispositions et Alignements** : Les dispositions des widgets sont ajustées pour assurer une expérience utilisateur cohérente et ergonomique, optimisant l'utilisation de l'espace disponible et améliorant la navigation générale de l'interface.

#### Feuilles de Style PyQt

En complément à Qt Designer, les feuilles de style PyQt sont utilisées pour affiner l'apparence des widgets :

- **Couleurs et Polices** : Les couleurs et les polices sont sélectionnées pour assurer une bonne lisibilité et une esthétique agréable, ce qui est essentiel pour maintenir l'engagement des utilisateurs sur de longues sessions expérimentales.
- **Tailles des Éléments** : Les tailles des boutons, des textes et d'autres éléments de contrôle sont optimisées pour une interaction facile et intuitive.

Ces outils permettent de réaliser des interfaces qui ne sont pas seulement fonctionnelles mais également adaptées aux besoins spécifiques des utilisateurs et du contexte expérimental, rendant les sessions de tests à la fois efficaces et agréables.

### 3.2.4 Développement Itératif

Le processus de développement de l'interface suit une approche itérative, en commençant par des prototypes simples et en les raffinant progressivement en fonction des retours et des nécessités du projet. Cette méthode assure que l'interface répond bien aux exigences fonctionnelles et esthétiques du projet tout en étant adaptable aux changements potentiels dans les objectifs de recherche.

Cette approche structurée à l'implémentation de l'interface graphique garantit que notre système est non seulement fonctionnel mais aussi agréable à utiliser, augmentant ainsi l'engagement des participants et la qualité des données recueillies durant les expérimentations.

## 3.3 Intégration des outils pour l'expérimentateur

### 3.3.1 Gestion des fichiers de structure

#### Fichiers `.json`

Les fichiers de configuration JSON, gérés par ‘parameters.py’, sont utilisés pour définir les paramètres expérimentaux. Cette approche permet une flexibilité maximale et une personnalisation facile pour différents types de tests, facilitant l’adaptation de l’interface aux besoins spécifiques de chaque expérience.

#### Fichier `commands.json`

Ce fichier contient une liste de commandes UNIX/Linux avec des options détaillées et des descriptions. Il permet le chargement des données nécessaires à la barre de menu de l’interface. Chaque commande est illustrée par des exemples spécifiques, ce qui permet à notre système de recommandation de proposer des suggestions pertinentes et contextuelles. Les éléments principaux de ce fichier incluent :

- **Commande** : Nom de la commande, tel que `tar`, `grep`, ou `find`.
- **Description** : Explication du fonctionnement de la commande.
- **Options** : Différentes manières d’utiliser la commande, chacune accompagnée d’un exemple concret.

#### Fichier `recommandation_bandit.json`

Ce fichier structure les blocs d’expérimentation impliquant des scénarios de bandits à bras multiples et des recommandations directes. Chaque bloc est configuré pour tester différents aspects du comportement utilisateur et de la performance du système. Les éléments clés comprennent :

- **Nom du Bloc** : Identifiant du bloc, comme *premier block* ou *dernier block*.
- **Type de Bloc** : Définit si le bloc est destiné à des expériences de type *bandit* ou *recommandation*.
- **Contenu du Bloc** : Détails des tâches spécifiques, comme les bras de bandit à interagir, les récompenses, et les moments pour présenter les questionnaires.

### 3.3.2 Gestion des blocs d’expérimentation

Les blocs d’expérimentation sont configurés pour permettre aux expérimentateurs de structurer les sessions en fonction des objectifs de l’étude.

#### Blocs de Type Bandit

Ces blocs permettent aux participants d’interagir avec des configurations spécifiques du bandit manchot, testant la prise de décision sous incertitude. Les moments pour charger les questionnaires sont prédéfinis pour capturer les perceptions et réactions des utilisateurs après des interactions clés.

## Blocs de Type Recommandation

Dans ce type de bloc, des recommandations textuelles sont présentées avec des délais spécifiés, permettant de tester la réactivité des participants et leur évaluation des recommandations. Les paramètres comme le *timeout* des recommandations sont implémentés pour comprendre l'impact temporel et la pertinence perçue des suggestions faites par le système.

### 3.3.3 Contrôle des paramètres des recommandations

Les paramètres des recommandations peuvent être ajustés au préalable pour répondre aux besoins spécifiques de chaque session expérimentale. Ces ajustements incluent :

- **Fréquence des Recommandations** : Détermine à quelle fréquence les recommandations sont présentées à l'utilisateur. Un contrôle précis de la fréquence permet de tester l'impact de l'intensité des recommandations sur l'expérience utilisateur.
- **Précision des Recommandations** : Ajuste la pertinence des suggestions faites par le système. Cette précision peut être modulée pour observer comment les utilisateurs réagissent à des recommandations plus ou moins adaptées à leurs besoins.

### 3.3.4 Log des événements utilisateur

La journalisation des événements utilisateur permet de capturer les interactions des utilisateurs avec l'interface, y compris les mouvements de la souris, les clics, les frappes au clavier, et les interactions avec les widgets. Ces données sont essentielles pour permettre d'analyser l'efficacité des recommandations et comprendre en détail le comportement des utilisateurs.

#### Mécanismes de capture des événements

Les événements suivants sont systématiquement enregistrés pour fournir une compréhension complète des actions des utilisateurs au sein de l'interface :

- **Mouvements de la souris** : Chaque position du curseur est enregistrée à chaque instant, permettant de suivre les mouvements précis sur toute l'interface. Cela inclut les coordonnées (x, y) du curseur, fournissant des données sur la navigation de l'utilisateur.
- **Survols de widgets** : Les événements de survol sur des éléments spécifiques comme les recommandations ou les items de la barre de menu sont capturés. On peut alors identifier les zones de l'interface qui attirent le plus l'attention ou qui peuvent prêter à confusion.
- **Clics de souris et interactions avec les widgets** : Chaque clic est enregistré avec des informations sur le widget ciblé, permettant d'analyser quelle partie de l'interface est la plus interactive.
- **Frappes au clavier** : Les commandes saisies et les entrées dans les champs de texte sont enregistrées, y compris les interactions dans le terminal.

#### Structure du fichier de log

Les données sont consignées dans un fichier CSV structuré avec précision, utilisant les champs suivants pour une analyse détaillée :

- **Temps** : Le moment exact de l'événement.
- **Coordonnées du curseur** : Les positions x et y du curseur au moment de l'événement.
- **Événement** : Le type d'interaction (clic, frappe au clavier, mouvement du curseur, etc.).
- **Widget** : Le widget ciblé par l'action de l'utilisateur.
- **Répertoire terminal** : Le répertoire actuel dans le terminal lors de l'interaction.
- **Entrée terminal** : Les commandes ou textes saisis dans le terminal.

Ces champs sont définis dans un script Python utilisant 'csv.DictWriter', garantissant que toutes les interactions sont enregistrées de manière uniforme et complète pour les analyses futures.

## Analyse des données de log

Les données collectées permettent d'effectuer des analyses comportementales et temporelles détaillées. En examinant les mouvements de la souris et les interactions avec les widgets, il est possible de déduire les éléments de l'interface qui fonctionnent bien ou ceux qui nécessitent des ajustements. L'analyse de ces logs aide à optimiser l'interface pour améliorer l'expérience utilisateur, en ajustant les aspects tels que la clarté des informations présentées.

### 3.3.5 Eye tracking

Initialement, une partie d'eye-tracking était prévue conjointement avec l'analyse des entrées clavier/souris.

L'idée principale était de pouvoir mettre en lien les entrées clavier/souris avec le regard de l'utilisateur à chaque instant.

Cela aurait permis d'identifier des points d'intérêt, d'éventuelles saccades ou fixations, et ainsi, couplé au log, de mieux comprendre les raisons derrière les actions de l'utilisateur.

Nous avons pour cela trouvé une bibliothèque python simple à intégrer dans notre projet : [Gaze Tracking](#)

Une fois l'intégration terminée, nous avons cependant constaté que cette librairie manquait de précision pour détecter la direction du regard, la rendant inadaptée dans le cadre de ce projet.

La librairie [OpenFace](#) proposée par nos encadrants était trop lourde pour pouvoir simplement être ajoutée à notre code

Ces imprévus tout au long du projet nous ont forcé à reconsidérer cette option du eye-tracking, puis à la supprimer.

### 3.3.6 Questionnaire

À un moment précis choisi par l'expérimentateur durant la session expérimentale, un questionnaire est présenté aux participants pour évaluer leur expérience avec le système de

recommandation. Les questions sont conçues pour recueillir des évaluations quantitatives sur plusieurs aspects critiques :

1. **Précision de la recommandation** : Les participants sont invités à estimer le niveau de précision des recommandations reçues. Cette question vise à comprendre comment les utilisateurs perçoivent la pertinence des suggestions du système. Ils doivent fournir une estimation en pourcentage, par exemple, indiquant que la précision est entre 65% et 85%.
2. **Gain de temps lors de recommandations correctes** : Cette question cherche à quantifier le temps économisé lorsque le système fournit une recommandation correcte. Comprendre cet aspect aide à évaluer l'efficacité du système en termes de contribution à l'amélioration de la productivité de l'utilisateur.
3. **Perte de temps lors de recommandations incorrectes** : Inversement, il est tout aussi important de mesurer le temps perdu à cause de recommandations inappropriées. Cette mesure est essentielle pour évaluer les coûts potentiels d'erreurs dans le système de recommandation.

## Interactivité

Les sliders sont connectés à des fonctions lambda qui mettent à jour les labels correspondants lorsque les valeurs des sliders changent. Cela permet aux participants de voir immédiatement la plage de valeurs qu'ils ont sélectionnées.

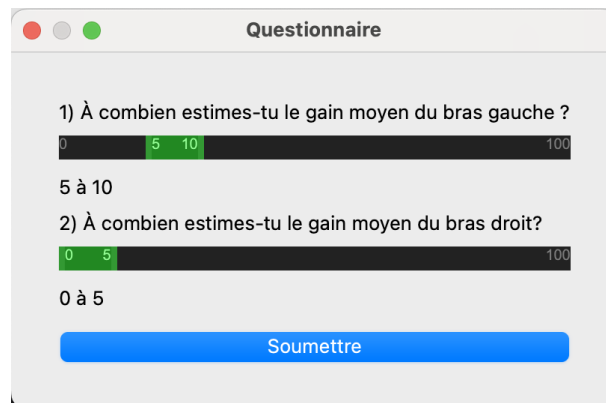


FIGURE 3.6 – Exemple de questionnaire dans l'interface bandit

## Précision de la recommandation

Un `QRangeSlider` permet aux utilisateurs de spécifier une plage de pourcentage pour la précision des recommandations (de 0 à 100 % par exemple). Un label affiche la valeur actuelle du slider, mise à jour dynamiquement lors du changement de valeur du slider.

## Collecte de données

Les participants répondent en spécifiant des intervalles pour refléter leur estimation, ce qui permet une évaluation plus nuancée que de simples réponses absolues.

Après soumission du questionnaire, les réponses des utilisateurs sont collectées dans des fichiers CSV.

Les données collectées pourront ensuite être analysées pour identifier des tendances, des points d'amélioration pour le système de recommandation, ou pour valider les hypothèses concernant l'efficacité des recommandations dans différents contextes d'utilisation.



# Chapitre 4

## Conclusion

L'objectif du projet était de mettre à disposition des chercheurs une interface facilement modulable pour leurs expériences. L'interface proposée, paramétrable à l'aide de fichiers .json, est réutilisable pour divers types d'expérience, et fournit les données adéquates permettant l'analyse du comportement de l'utilisateur. Une zone d'ombre est l'absence de l'analyse du regard, qui serait une piste d'amélioration évidente.

# Bibliographie

- [1] Justin MATEJKA, Wei LI, Tovi GROSSMAN et George FITZMAURICE. « Community-Commands : command recommendations for software applications ». In : *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 2009, p. 193-202 (page 2).
- [2] Rong HU et Pearl PU. « Acceptance issues of personality-based recommender systems ». In : *Proceedings of the third ACM conference on Recommender systems*. 2009, p. 221-224 (page 2).
- [3] Dalin GUO et AJ YU. « Human learning and decision-making in the bandit task : Three wrongs make a right ». In : *Conference on Cognitive Computational Neuroscience*. 2019 (page 3).
- [4] Shunan ZHANG et Angela J YU. « Forgetful bayes and myopic planning : Human learning and decision-making in a bandit setting ». In : *Advances in Neural Information Processing Systems*. T. 26. 2013 (page 3).

# Annexe A

## Cahier des charges

### A.1 Contexte

Dans un monde numérique où les recommandations personnalisées sont devenues monnaie courante, il est important de comprendre comment les utilisateurs réagissent face à ces suggestions, surtout lorsque celles-ci sont perçues comme inappropriées. L'adversité de l'utilisateur, c'est-à-dire sa réaction négative face à des recommandations non pertinentes, est un aspect souvent négligé mais essentiel à prendre en compte pour améliorer les systèmes de recommandation.

Le projet se situe à l'intersection de l'interaction humain-machine, de l'analyse des données et de la psychologie de l'utilisateur. Il vise à développer une interface graphique permettant de mener des expériences visant à évaluer l'adversité des utilisateurs face à des recommandations.

L'interface joue un rôle crucial dans cette évaluation en fournissant un environnement contrôlé où les chercheurs peuvent manipuler différents paramètres de recommandation et observer les réactions des utilisateurs en temps réel. Cette approche permettra de mieux comprendre comment les utilisateurs interagissent avec les recommandations et comment ces interactions peuvent être améliorées pour offrir une expérience utilisateur plus satisfaisante.

## A.2 Objectifs

- **Développer une Interface Graphique Intuitive :**

Concevoir une interface avec trois zones distinctes :

- \* un terminal
- \* un menu de commandes
- \* une zone de recommandations.

Cette interface doit permettre une interaction aisée avec l'utilisateur.

- **Implémenter un Système de Chargement de Données :**

Permettre le chargement facile de fichiers structurés contenant des séquences de recommandations et des menus à partir de sources externes. Cela facilitera la modification et la réutilisation des configurations expérimentales.

- **Gérer les Paramètres de Recommandation :**

Intégrer des fonctionnalités permettant de manipuler les paramètres des recommandations, tels que la fiabilité de la recommandation et les délais d'affichage, afin de pouvoir tester différents scénarios et observer leurs impacts sur l'adversité de l'utilisateur.

- **Logger les interactions utilisateur :**

Enregistrer les données liées aux interactions utilisateur, y compris les mouvements de la souris, les clics, les entrées clavier, et les choix de commandes dans le menu, afin de recueillir des données précieuses pour l'analyse des expériences.

- **Intégration de l'Eye Tracking :**

Intégrer une composante d'eye tracking à l'interface pour suivre le regard de l'utilisateur et déterminer quelle zone de l'interface est observée à tout moment. Cela permettra de corréler les réactions des utilisateurs avec les recommandations affichées.

## A.3 Ressources

- **Langage de programmation :**

Utiliser Python pour le développement de l'interface graphique en raison de sa popularité, de sa simplicité et de la disponibilité de bibliothèques telles que PyQt.

- **Bibliothèques :**

Utiliser PyQt pour la conception de l'interface graphique et OpenFace pour l'eye tracking. Ces bibliothèques offrent des fonctionnalités robustes et sont bien documentées.

- **Matériel :**

Assurer la disponibilité d'une webcam pour l'eye tracking et d'un ordinateur capable d'exécuter les tâches de développement et d'expérimentation.

## A.4 Étapes de réalisation

1. **Conception de l'interface :**  
Définir les spécifications détaillées de l'interface utilisateur, y compris la disposition des zones et les fonctionnalités interactives.
2. **Implémentation de l'interface :**
  - Développer les différents composants de l'interface en utilisant les widgets fournis par PyQt.
  - Intégrer les éléments d'interface utilisateur tels que les zones de texte, les boutons, les menus déroulants, etc., conformément aux spécifications de conception.
3. **Chargement des fichiers de structure :**
  - Écrire le code permettant de charger les fichiers décrivant la séquence de recommandations et le menu de commandes dans l'interface.
  - Utiliser de préférence un format “.json” pour les fichiers à charger.
4. **Contrôle des paramètres des recommandations :**  
Ajouter des fonctionnalités pour ajuster les paramètres des recommandations et les manipuler pendant les expériences.
5. **Contrôle des actions du menu :**  
Ajouter une fonctionnalité pour introduire un exemple d'une ligne de commande choisie dans le menu, dans le champ d'entrée du terminal avec des zones à remplir.
6. **Implémentation de la Fonctionnalité de Log :**
  - Mettre en place un système de log pour enregistrer les événements utilisateur tels que les clics, les mouvements de souris, les entrées clavier, ainsi que les actions effectuées dans l'interface.
  - Organiser les données de manière structurée et les stocker dans des fichiers log (csv) pour une analyse ultérieure.
7. **Intégration de l'eye tracking :**
  - Intégrer la bibliothèque OpenFace dans l'interface pour capturer et analyser les mouvements oculaires de l'utilisateur.
  - Configurer les paramètres de l'eye tracker pour le calibrage avec l'utilisateur.
  - Mettre en place un mécanisme pour enregistrer les données pertinentes, telles que les saccades, fixations et les coordonnées du regard à l'écran.
8. **Tests et débogage :**  
Tester l'interface pour s'assurer que toutes les fonctionnalités fonctionnent correctement et effectuer des ajustements si nécessaire.
9. **Documentation et rapports :**  
Documenter le code de l'interface, fournir des instructions d'utilisation et rédiger des rapports détaillant les résultats des expériences menées avec l'interface.
10. **Révision et amélioration :**
  - Réviser et améliorer l'interface en fonction des retours d'expérience et des résultats des tests.
  - Réfléchir à des améliorations pour optimiser la fiabilité des données recueillies.

# Annexe B

## Manuel utilisateur

### B.1 Utilisation générale

Lors du lancement de l'application, un sélecteur de dossier apparaît pour permettre de choisir le dossier contenant les [fichiers utilisateurs compatibles](#).

Pour éviter d'apparition du sélecteur à chaque ouverture de l'application, un dossier par défaut peut être précisé dans la variable `user_files_directory` du fichier `Utils.py`.

On mettra dans cette variable le chemin vers le dossier à utiliser à l'ouverture de l'application, sous forme de chaîne de caractères python, ou bien la valeur `None` pour procéder à la sélection du dossier manuellement.

Une fois les fichiers utilisateur localisés, l'application se chargera de déterminer lesquelles sont correctement formatés, et, pour chacun d'entre eux, affichera fenêtre contenant une liste des noms des participants. Chaque nom est cliquable, et cela lancera les différents blocs liés à ce participant.

### B.2 Structure du code

Le code, écrit en langage Python, est séparé en divers fichiers. Le fichier `main.py` comprend la boucle principale du programme, dont le sélecteur de dossier et la création des différentes fenêtres pour chaque bloc du participant sélectionné.

#### B.2.1 Recommandation

Pour les blocks de type `Recommandation`, on s'intéresse au fichier `interface.py`

La classe `InterfaceRecommandation` est la définition de l'interface visuelle pour les blocs de recommandations.

Elle contient aussi les différents évènements à log durant un bloc de ce type.

Elle utilise principalement deux widgets :

- le menu, placé dans le fichier `menu.py`
- le terminal, placé dans le fichier `terminal.py`, une réplique fonctionnelle d'un terminal Linux

## B.2.2 Bandit

Pour les blocks de type Bandit, on s'intéresse au fichier `interfaceBandit.py`

La classe `InterfaceBandit` définit quant à elle l'interface visuelle pour les blocs de type Bandit.

Elle utilise pour cela deux `BanditWidget`, la classe qui régit la partie fonctionnalité interne de cette interface.

Chaque `BanditWidget` correspond à un des bras que l'utilisateur doit choisir. Lorsqu'il en lance un, le `BanditWidget` met à jour son propre score en ajoutant la nouvelle valeur, et l'interface s'actualise ensuite en recalculant la somme des deux scores.

## B.2.3 Divers

Le fichier `questionnaire.py` contient les deux boîtes de dialogues des questionnaires affichés lors des blocs `Recommandation` et des blocs `Bandits`.

Les fichiers `DoubleSlider.py`, `EndPageWidget.py`, et `WidgetSlotMachine.py` contiennent différents widgets PyQt utilisés par l'interface, respectivement les sliders des questionnaires, la page affichée à la fin d'une expérience et les machines à sous du MAB.

Le fichier `Styles.py` définit différents styles utilisés par l'application, de manière à pouvoir les changer facilement.

# B.3 Fichiers utilisateur

## B.3.1 Règles

Ces fichiers, d'extension `.json`, ont pour objectif de rendre facilement paramétrable l'application, et ainsi réutilisable dans différents contextes. Ils suivent toujours la même structure :

- une clef **participant** ayant pour valeur une chaîne de caractères représentant le nom du participant
- une clef **blocks** ayant pour valeur une liste d'objets structurés de type bloc.

Les objets de type bloc ont eux aussi une structure bien définie :

- une clef **block\_name** ayant pour valeur une chaîne de caractères représentant le nom du bloc
- une clef **block\_type** ayant pour valeur une chaîne de caractères représentant le type de bloc. Celle-ci peut actuellement prendre 2 valeurs : `"recommandation"` ou `"bandit"`

- une clef **block\_content**, contenant toutes les informations nécessaires à l'exécution du bloc.

Pour les blocs de type bandit, sa valeur sera un objet json à deux clefs :

- **chargements\_questionnaires**, une liste des numéros de tirages du MAB après lesquels afficher le questionnaire
- **bras**, deux objets json structurés, contenant trois clefs : **side**, **rewards**, et **a priori**

Pour les blocs de type recommandation, sa valeur sera une liste d'objets json représentant chacun une recommandation à afficher ; ceux-ci possèdent 3 clefs :

- **header**, ce qui sera affiché avant de cliquer sur la recommandation
- **body**, le contenu de la recommandation, affiché après le clic
- **timeout\_seconds**, le temps entre le clic et l'affichage du contenu.

Voici un exemple de fichier correctement formaté, contenant à la fois un bloc de type recommandation et bandit.

### B.3.2 Exemple

```
{
  "participant" : "bandito",
  "blocks" : [
    {
      "block_name" : "premier block",
      "block_type" : "bandit",
      "block_content" :
        {
          "bras" : [
            {
              "side" : "gauche",
              "rewards" : [2, 5, 30],
              "a priori" : "environ 20 points"
            },
            {
              "side" : "droit",
              "rewards" : [-10, 60, 156],
              "a priori" : "environ 100 points"
            }
          ],
          "chargements_questionnaire" : [2, 6, 10]
        }
    ],
  },
}
```



```

{
  "block_name" : "dernier block",
  "block_type" : "recommandation",
  "block_content" : [
    {
      "header" : "recommandation une",
      "body" : "Salut.",
      "timeout_seconds" : 2
    },

    {
      "header" : "recommandation deux",
      "body" : "Bonne chance",
      "timeout_seconds" : 4
    }
  ]
}

```