Sorbonne Université
Master ANDROIDE

Assister l'utilisateur à utiliser les bonnes commandes

UE de projet M1

Alexandre Xia – Christian Zhuang – Nassim Ahmed-Ali
encadré par Gilles BAILLY et Julien GORI

2023

Table des matières

Chapitre 1

Introduction

Dans la plupart de nos interfaces, on retrouve de nombreuses commandes, mais elles sont en générales inconnues et/ou non utilisées par les utilisateurs qui vont donc utiliser des commandes sous-optimales. L'objectif de notre projet est de créer un assistant logiciel qui peut détecter les commandes qu'un utilisateur utilise afin de lui suggérer les bonnes commandes dans le cas où ceux-ci existent. Par exemple, une commande "retirer les yeux rouges" (si cela existe dans le logiciel) sur une photo, plutôt que de les retirer manuellement (pinceau + choix de couleur).

Pour ce faire, il nous est demandé de créer un modèle simplifié dans lequel on devra inclure un assistant entraîné à partir d'une base de donné avec un classifieur (on pourra utiliser directement ceux fournis par SKlearn). On aura donc, un player qui aura pour rôle de nous générer une base de données en imitant ce que ferait l'utilisateur avec les commandes du logiciel. Ensuite, un logger pourra enregistrer les commandes utilisées pour en faire un CSV d'exemples et enfin on pourra entraîner un classifieur qui pourra donc prédire les commandes, ce qui permet de savoir si l'utilisateur utilise ou non les bonnes commandes.

Pour le projet, nous sommes encadré par Gilles BAILLY et Julien GORI

Le lien du dépôt git est ici : NOTRE GITHUB

Chapitre 2

État de l'art

2.1 4 domains of interface performance improvement

En ce qui concerne le concept d'aide pour les commandes, il existe différentes approches permettant d'améliorer les performances de l'utilisateur avec les interfaces. Ces approches se basent sur 4 points :

- L'amélioration intramodale (intramodal improvement) concerne la vitesse et l'ampleur des performances d'une méthode spécifique du logiciel. Par exemple, aider l'utilisateur à prendre en main un clavier spécifique (ShapeWriter [1])

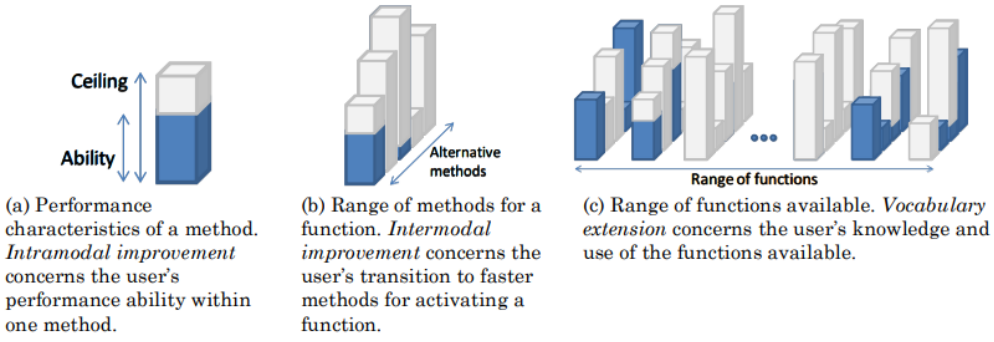
- L'amélioration intermodale (intermodal improvement) pour le passage vers des méthodes d'accès plus efficace pour une fonction spécifique avec un plafond ("ceiling") de performances plus élevé. Par exemple, Le passage d'une commande sélectionnée à la souris vers l'équivalent en raccourci clavier.

- L'extension vocabulaire (vocabulary extension) par rapport aux connaissances de l'utilisateur vis-à-vis des outils du logiciel. On voudrait que l'usager découvre et apprend un certain nombre de commandes d'un logiciel qui lui serait utile. Par exemple, l'utilisateur qui utilise habituellement "copier-coller" pourrait utiliser la commande "dupliquer" qui est une alternative (plus rapide dans ce cas).

Il faut noter que l'amélioration intermodale concerne une fonction spécifique et une alternative pour l'utiliser alors que l'extension vocabulaire concerne les commandes que l'utilisateur connaît.

- La planification des tâches (tasks mapping) qui concerne les stratégies employées par l'utilisateur pour faire une tâche (plus difficile). Pour dessiner 2 rectangles, il y aurait l'utilisateur qui en dessinerait 2 et un autre qui dupliquerait un rectangle qu'il aurait dessiné.

Ces notions sont toutes précisées dans l'article Supporting Novice to Expert Transitions in User Interfaces [1]



(a) Performance characteristics of a method. *Intramodal improvement* concerns the user's performance ability within one method.

(b) Range of methods for a function. *Intermodal improvement* concerns the user's transition to faster methods for activating a function.

(c) Range of functions available. *Vocabulary extension* concerns the user's knowledge and use of the functions available.

FIGURE 2.1 – Représentation des approches amélioration intramodale (a), amélioration intermodale (b) et de l'extension vocabulaire (c) [1]

2.2 Exemples d'interfaces pour l'extension vocabulaire (vocabulary extension) et lien avec notre projet

On s'intéresse de notre point de vue, plutôt à la notion d'extension vocabulaire. En effet, dans notre cas, l'approche qui consiste à permettre à l'utilisateur de découvrir et d'utiliser les bonnes commande. Donc d'élargir les connaissances de l'usager par rapport à un logiciel. Un problème lié à cette catégorie est que l'utilisateur doit être connaître l'existence de la commande avant de pouvoir l'utiliser. Il existe quelques méthodes qui ont été proposées pour permettre à l'utilisateur de les découvrir.

2.2.1 Exemple proche de notre cas

Une méthode étudiée est des exemples alternatifs pour les boutons cachés lorsqu'on consulte des mails sur un Iphone ont été étudiés. Par défaut il faut glisser horizontalement sur le mail en question pour faire apparaître les commandes. L'alternative qui a été étudiée propose de suggérer leur existence (comme montré dans la figure suivante). C'est une méthode qui est un peu en lien avec notre assistant. En effet, on souhaite que l'utilisateur découvre des commandes dont il ne connaît pas l'existence (comme les commandes cachées) et qu'en plus, il les utilise dans la bonne situation.

FIGURE 2.2 – Autres versions pour découvrir des boutons cachés [2]

Ce sont des alternatives un peu plus directes qui rendent ces boutons connus pour les nouveaux utilisateurs et ceux qui sont plus habitués. Dans notre cas, on essaye d'afficher la bonne commande en question à l'usager pour l'inciter à utiliser une commande selon la situation plutôt que de juste montrer l'existence de ces fonctions qui nécessiteront à l'usager de découvrir lui-même leur utilité.

2.2.2 Autre exemple en rapport avec la notion

Une autre étude propose également l'utilisation d'une carte de chaleur des commandes recommandées, cela permet à la fois d'inciter l'utilisateur de découvrir des commandes qu'ils n'ont jamais usées mais aussi d'y accéder sans entrave. Cette méthode est moins en rapport avec notre travail qui a pour idée de recommander des méthodes en fonction des commandes utilisées par la communauté ou l'utilisateur qui sont visibles directement depuis l'interface. C'est dans ce cas plutôt du feed-forward alors que nous essayons d'utiliser du feed-back pour notre assistant.

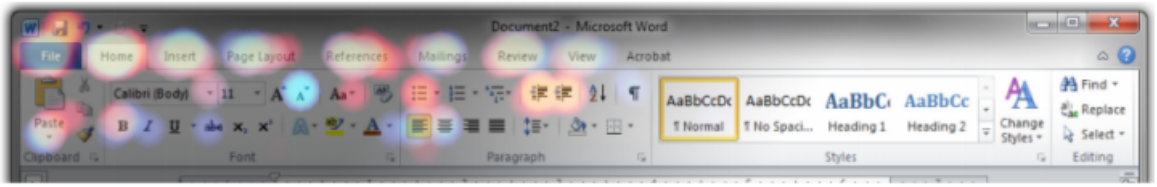


FIGURE 2.3 – Carte de chaleur pour la recommandation de boutons [1]

Dans la litérature, nous n'avons pas réussi à avoir beaucoup d'informations sur l'utilisation direct du Machine Learning pour aider un utilisateur à découvrir des commandes. Il existe toutefois des interfaces adaptatifs qui se basent sur le reinforcement learning pour changer le design d'une interface par rapport à l'utilisateur mais cela n'est pas directement en lien avec notre cas d'élargissement des connaissances de l'utilisateur.

Malgré cela, l'idée dans notre cas sera surtout de pouvoir comprendre si notre approche est fonctionnelle pour permettre à l'utilisateur de à la fois découvrir et apprendre des commandes dans la bonne situation et s'il l'acceptera d'apprendre de cette manière.

Chapitre 3

Contribution

Dans cette partie nous allons présenter les différents points traités avec nos encadrants pour mener à bien notre projet

3.1 Les composantes essentielles et difficultés

3.1.1 Les points à implémenter

Pour mener à bien notre projet, plusieurs éléments doivent êtres accomplis afin que notre système fonctionne. Une base de données sera nécessaire pour entraîner un modèle sur la tâche. Pour faire cela, on utilise un "Player" dont le rôle à pour but "d'imiter" l'utilisateur en utilisant les commandes du logiciel pour générer cette base. Ensuite, un "Logger" enregistre ces données et les mets sous un certain format qui sera utilisable par le modèle. Une fois formaté, on utilise un modèle qui prend en entrée ces données pour prédire des commandes (frontière de décision) ce qui permet de suggérer la bonne commande si l'utilisateur emploie des commandes moins efficace. Tout ceci se passe hors-ligne.

En ligne, le "Logger" va prendre les commandes utilisées par l'usager et envoyer cela au classifieur qui renvoit une décision en sortie. À partir de cette décision, l'assistant doit suggérer une commande plus optimale s'il y en a une.

3.1.2 Difficultés

Dans cette section, nous abordons quelques difficultés rencontrées lors de la réalisation des différentes parties de notre projet et les stratégies que nous avons mis en place pour les résoudre.

Difficulté de choix du modèle et représentation des données

Une difficulté principale est la représentation des données et d'un modèle associé. Dans le cas du projet, il faut donc bien représenter ce qu'est un exemple. Nous avons choisi une représentations avec des états qui seront défini plus loin dans le rapport en fonction du logiciel. Pour le modèle associé, Scikit-learn fourni des modèles déjà fait sur lesquels on peut entrainer avec la base de données qu'on a généré.

Choix des états à tester

Le nombre de requêtes que l'assistant peut soumettre au modèle est limité afin de garantir des recommandations en temps réel. Par conséquent, il est crucial de trouver les requêtes les plus pertinentes à poser au modèle. Différentes options sont envisageables, mais elles ne sont pas nécessairement bonnes ou mauvaises. Par exemple, augmenter le nombre d'états testés, choisir des états non continus ou sélectionner des états aléatoires. Dans notre logiciel de test, nous avons décidé d'effectuer des requêtes sur les 7 derniers états. Cette approche nous permet de fournir des recommandations basées sur les actions et les contextes récents de l'utilisateur, ce qui est susceptible d'améliorer la pertinence des suggestions.

Efficacité de l'apprentissage des commandes

Lors de la conception de l'assistant, il est essentiel de déterminer quelles informations afficher, car l'espace occupé par l'assistant doit être aussi restreint que possible afin de ne pas perturber le flux de travail de l'utilisateur. De plus, il est nécessaire de trouver un moyen de quantifier l'efficacité d'une interface. À cet égard, plusieurs options s'offrent à nous, comme étudier le temps nécessaire à l'utilisateur pour accomplir une tâche, le nombre de commandes utilisés, le nombre de fonctions pertinentes employé ou encore analyser la rétention des informations apprises. Dans le cadre de notre logiciel de test, nous avons choisi d'étudier le nombre de commandes (clavier) que l'utilisateur adopte, ce qui nous permettra d'évaluer l'efficacité de l'interface en termes d'adoption des méthodes proposés.

3.2 Logiciel de test

Pour tester le modèle et l'assistant, nous avons entrepris de développer une application dédiée. Notre approche de développement a été itérative, ce qui signifie que nous avons d'abord travaillé avec un modèle initial que nous avons testé sur un logiciel de dessin très basique, puis une seconde version sur une version plus adapté pour mener des expériences utilisateurs.

3.2.1 Premier logiciel : Dessin basique

Dans cette première version, nous avons un ensemble restreint d'action possible. L'utilisateur peut :

— Dessiner une forme d'une couleur.

— Changer la couleur de la forme déjà dessinée.

Nous avons deux formes possible, **rectangle** ou **ellipse**, et deux couleurs possible, **rouge** et **bleu**. Ce qui nous fais un total de 5 commandes :

— Dessiner un rectangle bleu.

— Dessiner un rectangle rouge.

— Dessiner une ellipse bleu.

— Dessiner une ellipse rouge.

— Changer la couleurs de la forme dessinée.

Grâce à cet exemple, nous avons pu effectuer rapidement des tests pour vérifier le bon fonctionnement de notre modèle et de notre assistant. L'objectif était de s'assurer que les recommandations fournies par l'assistant étaient similaires à celles de l'exemple donné.

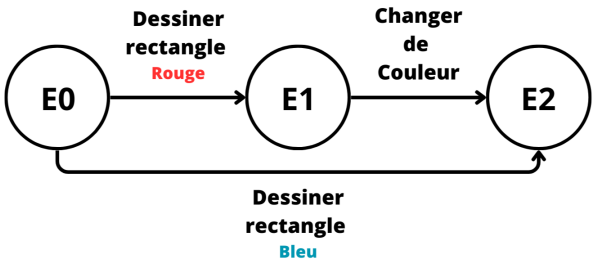


FIGURE 3.1 – Exemple de recommandation du logiciel 1

Cette étape nous a permis de valider la cohérence et l'efficacité de notre approche, en nous assurant que les résultats obtenus étaient conformes à nos attentes ainsi qu'identifier les aspects à améliorer et les ajustements à apporter.

3.2.2 Second logiciel : Déplacement de forme

L'objectif de cette version finale de l'application est de permettre à l'utilisateur de déplacer un objet généré vers des emplacements prédéfinis. Ces emplacements sont représentés par quatre points fixes vers lesquels l'utilisateur devra déplacer l'objet, sachant que l'un de ces points sera l'objectif à atteindre, indiqué par un indicateur visuel (A.1.2).

L'utilisateur a la possibilité de déplacer et de faire pivoter la forme pour l'ajuster à l'objectif. Chacune de ces actions possède une variante permettant de couvrir une distance plus importante ou de pivoter de plusieurs degrés. Voici donc la liste des commandes disponibles :

— Déplacer (droit, gauche, haut, bas)
— Pivoter (droit ou gauche)

3.3 Player et Logger

Les deux composants, à savoir le player et le logger, jouent un rôle essentiel dans la création d'une base de données pour l'entraînement du modèle. Le player est chargé de lancer des commandes de manière aléatoire dans le logiciel et d'enregistrer les effets de ces commandes, c'est-à-dire les états initial et final correspondants. Cela nous permet d'obtenir une variété de scénarios d'utilisation du logiciel, reflétant différentes séquences de commandes et de transitions entre les états. Éventuellement, comme amélioration, on pourrait enregistrer les commandes que l'utilisateur utilise réellement lorsqu'il interagit avec le logiciel, fournissant ainsi des données plus pertinentes et spécifiques à l'utilisateur réel. Cette approche nous permettrait de disposer d'une base de données plus diversifiée et riche, qui servira à entraîner le modèle pour des recommandations plus précises et adaptées à chaque utilisateur.

3.4 Modèle et données

Dans ce projet, Le modèle/classifieur est une composante essentielle comme dit précédemment, mais c'est également en partie le coeur de l'assistant. En effet, c'est ce qui nous permet de proposer une commande de manière automatique parmi les différentes commandes existantes du logiciel.

3.4.1 Fonctionnement du modèle de classification

Pour son fonctionnement, il faut définir des descripteurs pour un exemple d'apprentissage. Une façon de modéliser les descripteurs (features) du modèle dans notre cas est de représenter un exemple de donnée avec l'état de départ et l'état après une ou plusieurs commandes. Un label représente une commande.

Pour le premier logiciel (plus simple), l'état de départ correspond à la forme (s'il y en a une) et la couleur et de même pour l'état de fin on a donc un vecteur x de taille 4 (avec des valeurs catégorielles). Cependant, pour pouvoir tester sur plusieurs modèles il fallait passer en One-hot (on a des chaînes de caractères qu'il faut transformer en valeurs numériques) et on se retrouve avec un vecteur $x \in \{0,1\}^{12}$. Nous avons essayé quelques classifieurs (Réseau de Neurones, Perceptron, Arbre de décision, Kdtree) et l'arbre de décision semble nous donner les meilleurs résultats.

Pour l'autre logiciel où l'on doit déplacer une figure à partir des flèches directionnelles 3.2.2 les états correspondent aux vecteurs (abscisse, ordonnée, angle), on a donc un exemple $x \in \mathbb{R}^{3*2}$. Nous avons décidé pour simplifier les données qui seront données en entrée au classifieur en utilisant à la place, une différence entre les deux états, on se retrouve donc avec un vecteur d'exemple $x \in \mathbb{R}^3$ à la place avec un descripteur de taille 3. Cela simplifie beaucoup plus l'apprentissage et permet à plus de modèles de classifications d'êtres fonctionnelles. Pour le modèle, les descripteurs étant assez simplifiés, nous pouvions utiliser la plupart des classifieurs mais pour rester consistent au premier, nous avons décidé de rester sur l'arbre de décision.

3.4.2 Étude du temps de prédiction en fonction de la taille de l'historique

Une étude qu'on a mené en plus, est le calcul du temps qu'il faut pour avoir en réponse d'une commande en fonction de la taille de l'historique. Lorsque l'utilisateur lance une commande en s_n, l'application transite vers un nouveau état de l'application, s_{n+1}. Sachant que la taille maximale de l'historique est de h (une valeur choisie arbitrairement), on vérifie une transition plus optimale entre les h précédents états de l'application et l'état s_{n+1}.

Par exemple, lorsque l'historique du logiciel contient 3 états avec les transitions entre ces états, on transition vers un état 4 (un état futur) et en faisant des paires avec les états passés, on cherche à prédire une commande qui sera potentiellement suggérée, si nous ne trouvons pas de commande, alors il n'y a pas de méthode plus efficace (on fait donc h prédictions avec le classifieur dans le pire cas avec h = 3). De plus, si on trouve une prédiction de 1 vers 4 alors une prédiction de 2 vers 4 serait une commande moins efficace. Cette façon de procéder permet de ne pas avoir à refaire certaines prédictions et aussi éviter de faire des prédictions qui sont potentiellement moins efficaces.

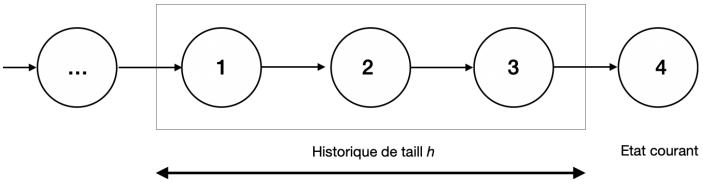


FIGURE 3.2 – Schéma exemple des états avec la taille de l'historique

Le temps est donc pris entre ce dernier état s_{n+1} avec les h états précédents. On a fait deux petites expériences :

— Une première avec une séquence commune générée aléatoirement

— Une autre avec une séquence qui cherche à réaliser la tâche demandée

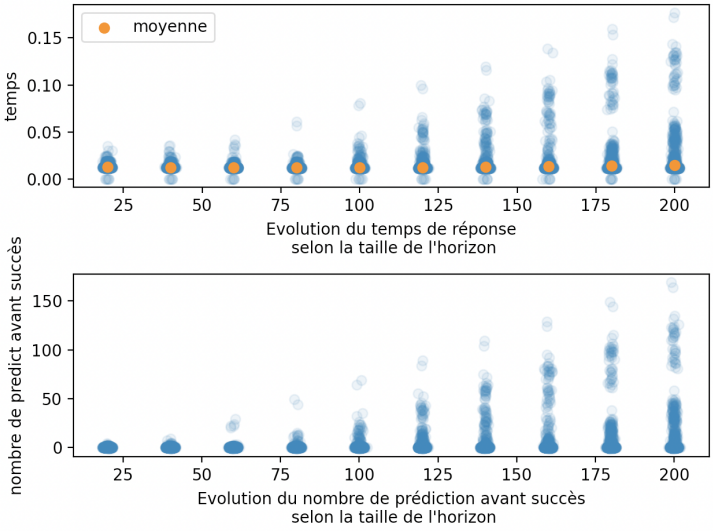


FIGURE 3.3 – Évolution du temps (en secondes) du nombre de nombres de prédictions avant succès en fonction de la taille d'historique pour une séquence aléatoire

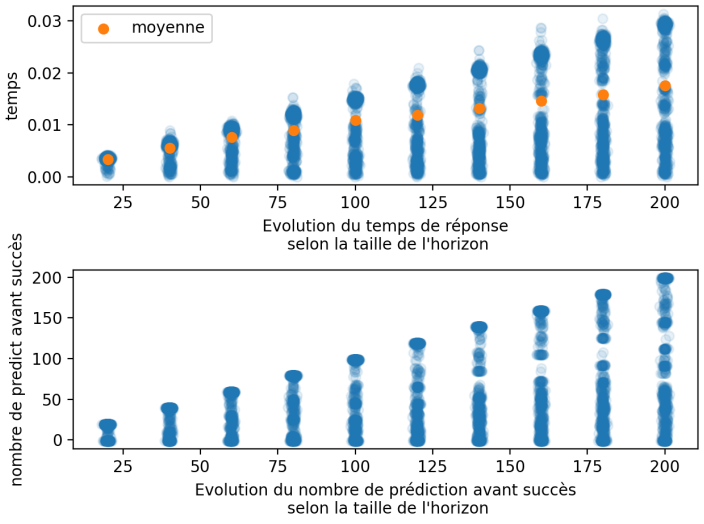


FIGURE 3.4 – Évolution du temps (en secondes) et du nombre de nombres de prédictions avant succès en fonction de la taille d'historique avec réalisation de la tâche

À partir des figures, on observe une évolution linéaire pour le temps et du nombre de prédictions avant succès en fonction de la taille de l'historique. Le cas de la séquence aléatoire ne donne pas une bonne estimation même on observe quelques valeurs plus grandes lorsque l'historique grandi.

3.5 Assistant

L'assistant joue un rôle central dans notre logiciel car il joue un rôle "d'interface interactive" avec l'utilisateur. L'objectif final de l'assistant est d'assister l'utilisateur dans la découverte de nouvelles commandes au moment le plus approprié. Lors de sa conception, nous avons soigneusement examiné plusieurs problématiques auxquelles l'assistant devait répondre, ce qui nous a conduits à prendre des décisions et à faire des compromis que nous allons détailler.

3.5.1 Critère de l'assistant

Lors de notre analyse, nous avons identifié plusieurs variables clés qui peuvent influencer nos choix concernant l'assistant :

Espace : L'assistant doit être conçu de manière à occuper un espace minimal à l'écran, afin de ne pas gêner la vue ou l'utilisation d'autres éléments de l'interface.

Quantité d'information : L'assistant doit être en mesure d'afficher de manière claire et visible les raccourcis clavier suggérés, de sorte que l'utilisateur puisse les consulter facilement et rapidement sans confusion.

Intégration : L'assistant doit être facilement s'intégrer dans la charge de travail (workflow) habituelle de l'utilisateur, de manière à ne pas perturber ou ralentir ses activités. Il doit se fondre harmonieusement dans l'interface existante, sans créer de friction ou d'obstacles supplémentaires.

En prenant en compte ces variables, nous chercherons à concevoir un assistant qui réponde aux besoins spécifiques des utilisateurs tout en minimisant les perturbations potentielles dans leur flux de travail quotidien.

Nous avons pris la décision d'explorer trois interfaces distinctes pour l'assistant, dans le but de mener des expériences afin d'évaluer l'efficacité de chacune d'entre elles. Notre objectif est d'analyser et de comparer ces interfaces afin de déterminer celle qui offre la meilleure performance et l'expérience la plus optimale pour les utilisateurs.

3.5.2 Assistant en utilisant un graphe

L'objectif de cette interface est de représenter l'état de l'application à l'aide de nœuds, et les transitions d'un état à un autre sont représentées par des arêtes étiquetées avec le nom et le raccourci de la commande.

A- Graphe et noeud : Pour maintenir un schéma clair et lisible, il n'est pas possible d'afficher toutes les transitions d'état possibles. Nous avons donc décidé de nous concentrer sur deux approches :

Afficher les recommandations à partir de l'état le plus récent. Les utilisateurs sont plus susceptibles de se souvenir de ce qu'ils ont fait et ainsi de comprendre l'utilité du raccourci utilisé.
Afficher les recommandations les plus pertinentes. Les utilisateurs pourront connaître la commande qui aurait pu leur éviter le plus d'états, ce qui sera plus utile pour eux.

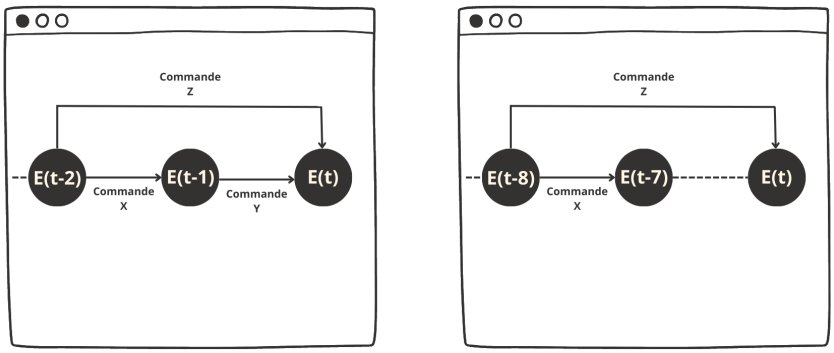


FIGURE 3.5 – Exemple d'affichage des recommandations à partir de l'état le plus récent et des recommandations les plus pertinentes

B - Affichage de l'état : Représenter un état par un nœud ne fournit pas suffisamment d'informations à l'utilisateur sur cet état. Une solution proposée consiste à permettre aux utilisateurs de cliquer sur les nœuds du graphe pour afficher plus d'informations. Cependant, cela réduit l'intégration de l'assistant, car l'utilisateur devra interagir avec l'interface afin d'ajouter plus de sens à la représentation des états. L'utilisateur peut cliquer sur un état pour obtenir un aperçu de cet état. Cet aperçu peut être une capture d'écran de l'application à l'état correspondant ou, si le logiciel le permet, une superposition de l'ancien état directement dans l'interface de l'application.

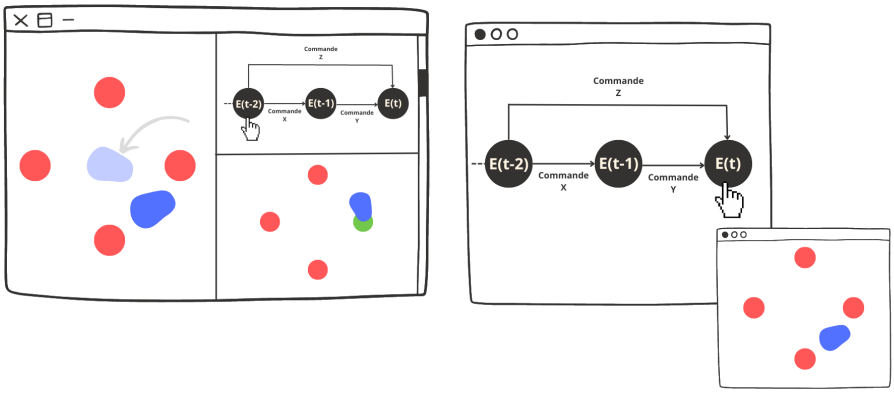


FIGURE 3.6 – Exemples d'affichage des états

C - Historique : L'affichage sous forme de graphe permet de conserver un historique de tous les états précédents. L'utilisateur peut ainsi consulter l'ensemble des commandes recommandées précédemment. Cependant, cette approche réduit considérablement l'intégration de l'assistant, car il nécessite une interaction avec la fenêtre de l'application.

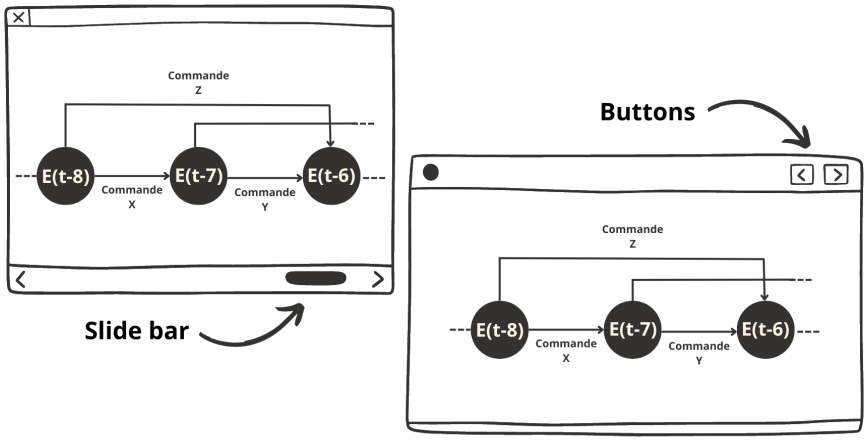


FIGURE 3.7 – Exemples d'affichage d'un historique d'états

Grâce à cette interface, nous sommes en mesure d'afficher un grand nombre d'états, cependant, cela requiert un espace assez conséquent pour représenter les données. De plus, les interactions avec le graphe peuvent potentiellement distraire l'utilisateur.

3.5.3 Assistant en utilisant du texte

L'idée de cette interface est de fournir une fenêtre affichant une liste de raccourcis que l'utilisateur aurait pu utiliser. Cette liste s'agrandit au fur et à mesure que l'assistant suggère des raccourcis. Après un certain nombre de commandes non utilisées, les recommandations gagnent en priorité et sont affichées de manière plus visible à l'utilisateur. Cette interface offre plusieurs niveaux d'affichage en fonction de la priorité de la commande, ce qui permet de s'adapter facilement à l'espace disponible en réduisant le nombre maximal de commandes affichées.

Pour cette interface, nous avons réduit l'espace d'affichage ce qui permet de ne pas perturber excessivement l'utilisateur dans ses tâches habituelles, tout en présentant une quantité significative d'informations affichés.

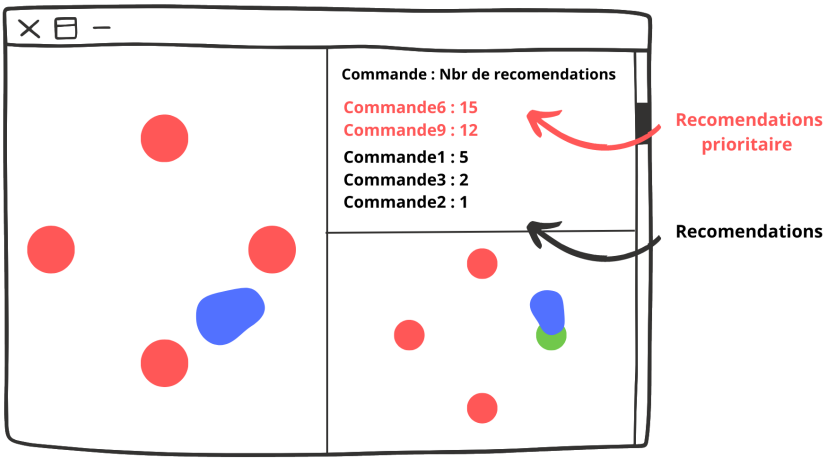


FIGURE 3.8 – Exemples d'assistant avec texte

3.5.4 Assistant en utilisant une alerte

L'approche ici est très similaire à celle du texte, mais au lieu d'avoir une liste constante de raccourcis, nous allons interrompre temporairement l'utilisateur pour lui recommander une commande pertinente. Dans ce cas, nous utiliserons une alerte dédiée à la commande, ce qui nous permettra d'afficher plus d'informations à son sujet. Par exemple, nous pourrions inclure une animation expliquant ce que fait la commande.

Dans le cas de cette interface, il n'est pas nécessaire de conserver un espace constant pour l'utilisateur. En effet, l'utilisateur ne voit que son application habituelle jusqu'à ce qu'une alerte survienne. Après avoir déclenché l'alerte, il nous est possible d'afficher différentes informations concernant la commande recommandée.

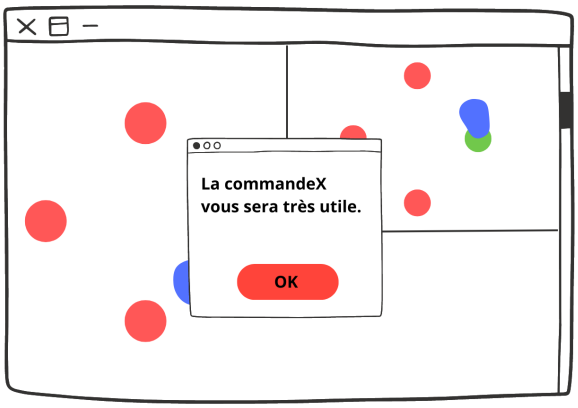


FIGURE 3.9 – Exemples d'assistant avec alerte

Chapitre 4

Conclusion

Pour conclure, nous avons donc pour notre travail conçu un assistant qui aide l'utilisateur à utiliser les bonnes commandes. Les différents points énoncés dans le sujet ont été fait. Nous avons généré une base de données à partir d'une simulation d'utilisateur, nous avons entraîné un modèle (classifieur) à prédire des commandes à suggérer et enfin l'assistant peut utiliser cette prédiction pour faire la bonne suggestion de commande.

Dans notre cas de flèches directionnelles et de dessin de figures, les données sont relativement simples et la plupart des modèles de classifications sont assez efficaces pour bien prédire les commandes et les données générées par simulation permettent d'avoir une base d'entraînement assez facilement accessible.

Cependant, il faut noter que si nous passons à des données plus compliquées, tous les modèles ne fonctionneraient probablement pas, une image en entrée par exemple, pourrait nécessiter des réseaux de convolution. Transformer la donnée en image est une piste qu'il est possible d'explorer avec donc une entrée qui est une matrice de pixels.

Une autre piste qu'il faudrait explorer est l'emploie d'un logiciel avec plus de commandes que la version simplifiée qui nous permet surtout de voir si le concept d'assistant autonome était possible. Il faut donc pouvoir expérimenter plus pour comprendre à quel point ce système est robuste.

Bibliographie

[1] Andy Cockburn - Carl Gutwin - Joey Scarr - Sylvain Malacria. « Supporting Novice to Expert Transitions in User Interfaces ». In : *ACM Comput. Surv.* (2014). URL : https://inria.hal.science/hal-02874746/document (pages 2-4).

[2] Nicole Ke Cheng Pong. « Understanding and Increasing Users' Interaction Vocabulary ». In : *29ème conférence francophone sur l'Interaction Homme-Machine, AFIHM* (2017). URL : https://hal.science/hal-01577901/file/RD-pong.pdf (page 3).

Annexe A

Cahier des charges et Manuel Utilisateur

A.1 Cahier des charges

A.1.1 Introduction

Ce projet a pour objectif de créer un assistant qui incite l'utilisateur à découvrir l'usage de certaines commandes qui sont potentiellement meilleures que d'autres. L'assistant devra dans un premier temps analyser les commandes utilisées par l'usager et en fonction des commandes détectées, il proposera une meilleure commande s'il en existe une. Par exemple, l'utilisateur pourrait déplacer un objet (une forme) 2 fois vers la droite avec la touche directionnelle droite mais il existe dans le logiciel une touche raccourci qui permet de faire ces 2 actions en une, l'assistant devra détecter cette possibilité (si l'utilisateur ne l'a pas utilisé pour cet action) et proposer à l'usager le raccourci en question.

Le travail à réaliser sera représenté dans un premier temps par le schéma suivant (cela permettra de faire la base du projet) :

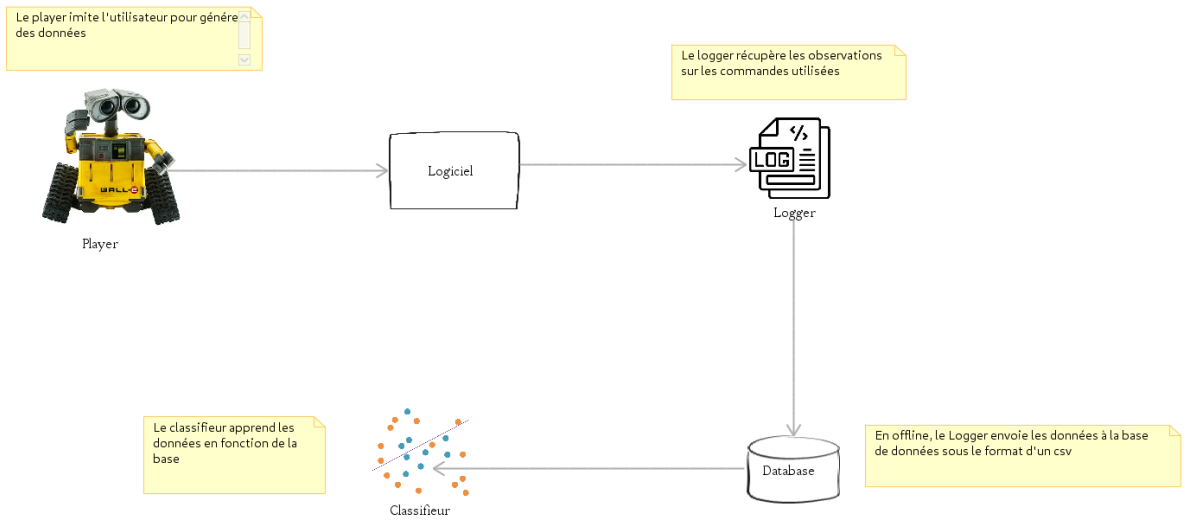


FIGURE A.1 – Génération des données et apprentissage du modèle

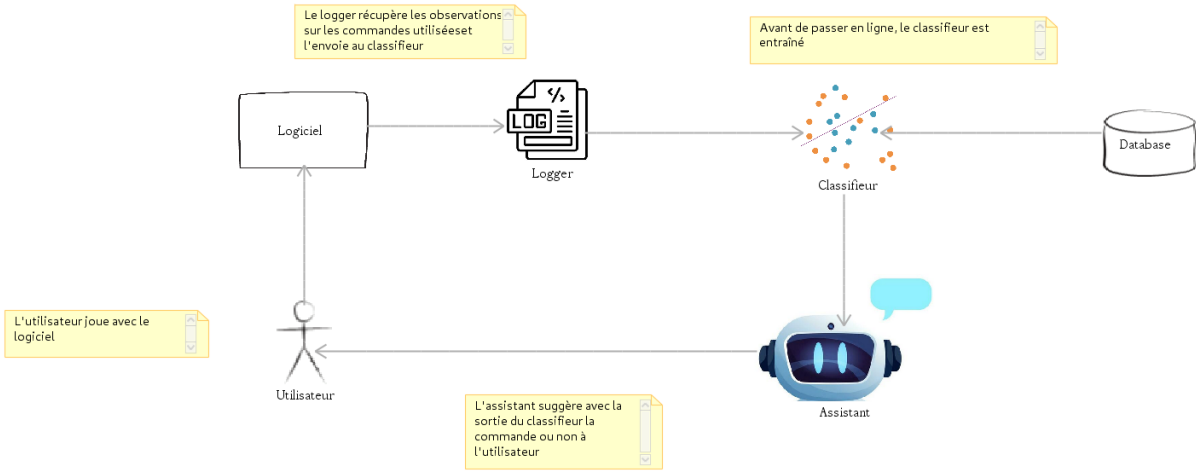


FIGURE A.2 – Passage en ligne

Par rapport aux approches énoncées dans le sujet :

- La database et le modèle correspondent au point de vue machine learning. (1)
- Le logger concerne la partie : détecter les résultats données (les états) du player et/ou de l'utilisateur. (2)
- L'assistant et la sortie du modèle selon l'approche de suggestion de commande. (3)

A.1.2 Les fonctionnalités

Logiciel

Pour le logiciel utilisé, il s'agira d'un logiciel de dessin qui sera développé (qui l'est plus ou moins déjà) et auquel on ajoutera des options afin de pouvoir mieux connaître les commandes sur lesquelles il serait possible de faire l'assistant.

- Logiciel 1

Les fonctionnalités de la première version du logiciel simplifié sont les suivantes :

- Dessiner des formes (rectangles et ellipses)
  - Possibilité de dessiner directement des rectangles/ellipses avec la couleur souhaité
- Changer la couleur de la forme actuelle
  - Rouge ou bleue

Les fonctionnalités sur lesquelles l'assistant peut être utilisé :

- Ajouter une figure
- Commandes de changements de couleurs versus forme déjà colorée

- Logiciel 2

Les fonctionnalités de la deuxième version du logiciel simplifié sont les suivantes :

— Déplacer un objet qui sera généré dans un emplacement fixe (qui sera plus tard aléatoire) avec 4 points fixes vers lesquels l'utilisateur devra déplacer l'objet (un des quatres points sera l'objectif qu'on peut indiquer par une image)

Les fonctionnalités sur lesquelles l'assistant peut être utilisé :

— Toutes les fonctions liés au déplacement
  - les 4 flèches directionnelles
  - ctrl + une des flèches pour faire un saut
  - commandes de rotations (lettres du clavier par exemple)
  - ctrl + une commande de rotation pour une plus grande rotation

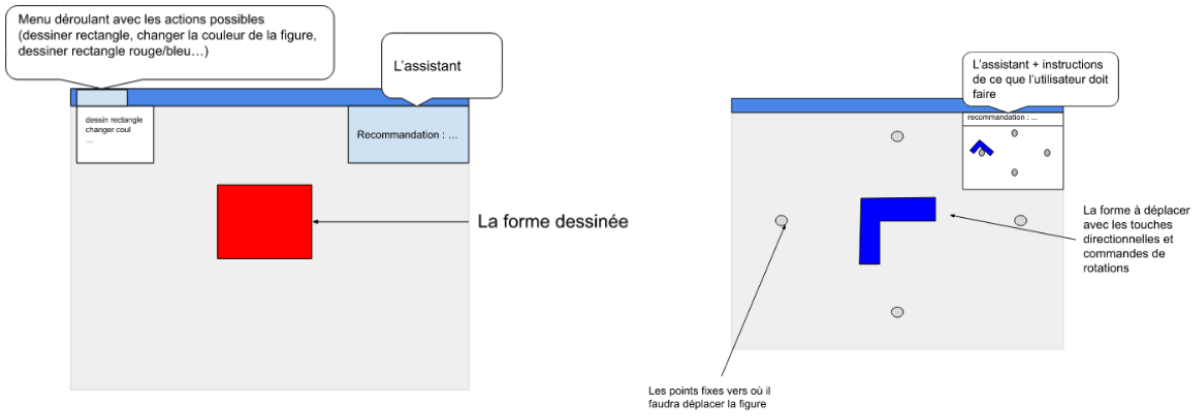


FIGURE A.3 – Les 2 logiciels (premier à gauche et second à droite)

Player

Pour le player qui sera un imitateur de l'utilisateur, il faudra récupérer les commandes dont il sera possible de générer des données d'apprentissage pour le modèle.

Les fonctionnalités du player sont :

— Imiter le rôle de l'utilisateur afin de générer des données qui serviront à l'apprentissage.
  - exemple simplifié 1, dessin automatique de formes et possibilité de changements de couleur
  - exemple simplifié 2, déplacement d'une figure aléatoirement (il n'y a pas besoin que le player atteigne forcément l'un des 4 points d'intérêts).

Les fonctionnalités optionnelles :

— Séparer complètement le Player du logiciel et donc pouvoir avoir un Player général qui pourrait être adapté à d'autres logiciels. On peut utiliser l'api donnée par les logiciels.

Logger

Le logger aura pour objectif de récupérer les commandes utilisées ainsi que les états avant et après l'utilisation de ces commandes afin de générer les données qui seront utilisées pour le modèle niveau machine learning.

Les fonctionnalités du logger sont :

— Prendre les états simplifiés ainsi que la liste des commandes utilisées par l'utilisateur/player entre les états
  - Exemple simplifié 1, la description de la figure (rectangle, couleur)
  - Exemple simplifié 2, les positions (x,y,angle) d'état de départ et d'état de fin de la figure en question.

Les fonctionnalités optionnelles :

— Séparer complètement le logger du logiciel. Un des moyens serait d'observer seulement les touches du clavier et récupérer l'image du logiciel.

Database

La base de donnée sera composée d'une matrice de taille N * P, avec N le nombre d'exemples (de lignes) et P le nombre de descripteurs (colonnes) avec la dernière colonne qui correspond à un raccourci/commande (label) dont le nombre sera décidé arbitrairement.

La forme pour la database est :

— Pour l'exemple 1, la description de la figure avec pour label, la commande utilisée pour passer vers l'état final correspondant.
— Pour l'exemple 2, les coordonnées et angle (x,y,angle) de la figure de l'état de départ et de fin ainsi que la ou les commande(s) utilisée(s) pour passer de l'un à l'autre.

Les fonctionnalités optionnelles :

— On pourrait potentiellement si cela fonctionne bien, passer par une image au lieu de descripteurs simples (les descripteurs deviendraient donc des pixels) ou juste augmenter le nombre de descripteurs pour essayer de généraliser le tout.

Modèle

Le modèle de machine learning, on a beaucoup de choix de modèle grâce à la bibliothèque ScikitLearn (https ://scikit-learn.org/stable/). Le réseau de neurones semble être le modèle le plus performant au niveau de l'apprentissage supervisé. On pourra donc directement en choisir un parmi ceux existants.

Assistant

L'assistant qui sera en contact avec l'utilisateur et qui utilisera la sortie produite par le modèle.

L'assistant pourra :

— Comparer les commandes utilisées par l'utilisateur avec la sortie donnée par le modèle pour proposer un raccourci si nécessaire dans une boîte de texte.

Les fonctionnalités complémentaires pourraient êtres :

— Afficher de manière plus visuelle les possibilités entre les états pour que l'utilisateur puisse comparer ses commandes avec d'autres plus efficaces.

A.2 Manuel Utilisateur

Certaines librairies ont été utilisé pour notre implémentation :

— Numpy, matplotlib, pandas (librairies mathématiques)

— scikit-learn

— pyQT

— networkx

— pyvis

— csv

Ensuite il existe plusieurs versions du projet :

— Projet V4 correspond à la dernière version du projet avec des points d'ouverture qu'on a commencé à traiter et une interface utilisateur (expériences utilisateurs).

— Projet V3 est la version sans l'interface utilisateur.

— PlotTime correspond correspond à la version V3 avec la possibilité d'étudier les temps en fonction de la taille de l'historique (l'hyperparamètre size dans la partie AllAssistant dans MainWindow).

— projet est la version fonctionnelle du projet sans les explorations des bonus (points d'ouverture).

— Projet-Androide-Logiciel1 correspond au premier logiciel simplifié.

Pour pouvoir lancer le code, il faut se lancer un terminal et exécuter la commande suivante dans le bon répertoire (ProjetV4, ProjetV3, PlotTime, projet ou Dessin pour Projet-Androide-Logiciel1) :

python MainWindow.py (ou python3).