Sorbonne Université
Master ANDROIDE

DISCO: Contextually Improving Command Discoverability

Stage de Master 1

Réalisé par :

Alexandre XIA

Encadré par :
Gilles BAILLY, ISIR, Sorbonne Université
Julien GORI, ISIR, Sorbonne Université

Référent :
Thibaut Lust, LIP6, Sorbonne Université

3 juillet 2023

Table des matières

Chapitre 1

Introduction

Les applications telles que PowerPoint ou Adobe Photoshop qu'on utilise tous les jours fournissent de nombreuses commandes selon nos besoins. Parmis ces commandes, certaines sont plus efficaces en regroupant d'autres commandes plus simples, ce qui permet d'accélérer une tâche. Par exemple, Adobe Photoshop propose la commande "retirer les yeux rouges" qui permet à l'usager de cliquer à partir de l'outil sur les zones correspondantes et de les repreindre en noir. Un certain nombre d'utilisateurs ne connaisse pas l'existence de cette commande et choisisse l'outil "pinceau" et changer sa couleur en noir avant de repeindre manuellement les zones rouges. Un autre exemple est l'alignement d'éléments sur Microsoft Word, l'usager qui ne connait pas la commande d'alignement procèdera par déplacer chaque objet un à un vers la position qui le convient alors qu'il aurait pu juste sélectionner les éléments et appeler la fonction d'alignement. Le problème étant que cela prenne plus de temps à exécuter

Plusieurs méthodes pour recommander des commandes ont été proposées mais une partie suggère que l'utilisateur est déjà familier avec les commandes optimales. Les outils permettant à l'utilisateur de découvrir de nouvelles commandes utilisent en majorité des recommandations selon ce que l'utilisateur fait.

Dans la continuité du projet de Master I, nous avons durant le stage essayer d'aboutir à une preuve de concept pour la création d'un modèle de recommandation de commandes. Ce modèle se baserait plus sur l'intention de l'utilisateur en utilisant l'état de l'application (une image ou un vecteur de descripteurs avant et après modification pour l'utilisateur) plutôt que la façon dont il s'y prend (commandes utilisées) pour atteindre son objectif. Cette état serait donnée à un classifieur (pas forcément appris) qui pourra suggérer potentiellement la bonne commande.

Pour cela, on cherche à tester des commandes pertinentes et pour lesquelles on a des effets assez différenciables. Par exemple, "ctrl+a" sélectionne tout le document, mais "ctrl+shift+fin" a le même effet si on se trouve au début du document ce qui rendrait difficile de savoir s'il faudrait suggérer "ctrl+a" qui serait la bonne commande si l'utilisateur sélectionne tout le document avec "shift" et les flèches.

Ce projet est fait en collaboration avec mes camarades : Christian ZHUANG et Nassim AHMED-ALI. 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])

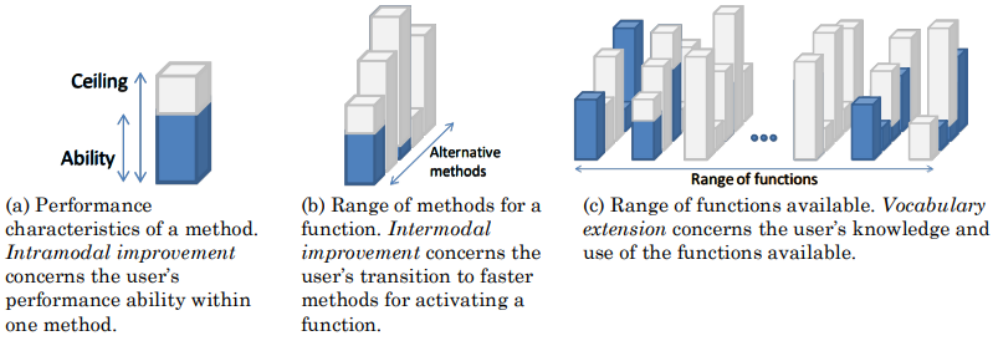
- L'amélioration intermodale (intermodal improvement) pour le passage vers des méthodes d'accès plus efficace pour une fonction spécifique avec un plafond ("ceiling") de performances plus élevé. Par exemple, Le passage d'une commande sélectionnée à la souris vers l'équivalent en raccourci clavier.

- 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, l'utilisateur n'a probablement pas conscience de l'existence des commandes et notre objectif est de pouvoir lui donner la possibilité de les découvrir.

2.2.1 Exemple avec des similarités avec 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 qui s'approche beaucoup plus de notre cas

L'article "Sequence Prediction Applied to BIM Log Data, an Approach to Develop a Command Recommender System for BIM Software Application" propose l'implémentation d'un outil de recommandation de commande pour les utilisateurs de logiciels de modélisation des données du bâtiments [quotation to add].

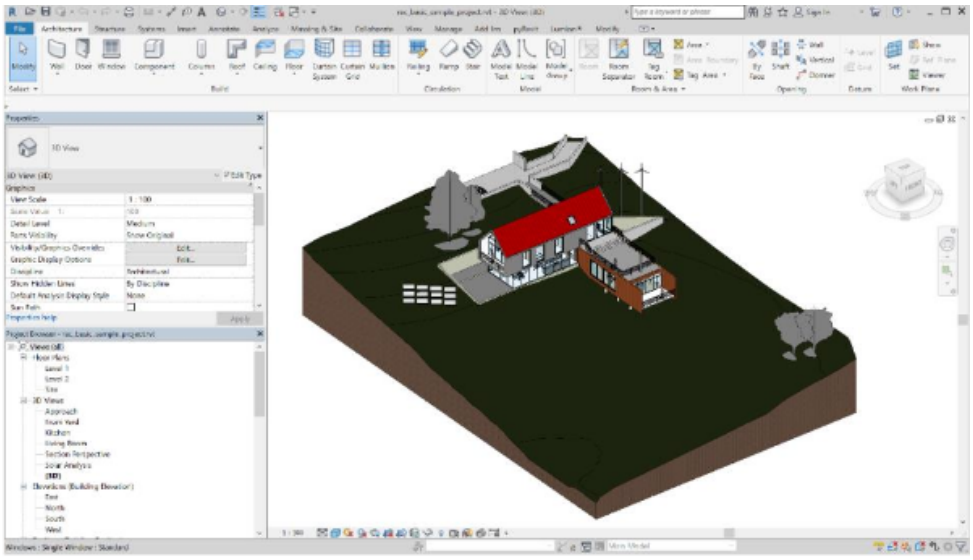


FIGURE 2.3 – AutoDesk Revit (image de l'article) [1]

L'approche consiste à utiliser les journaux ("logs") d'un logiciel tel que Autodesk Revit. Les données de ces journaux seront traitées puis employées pour développer un modèle de "Machine learning" pour suggérer des commandes à l'utilisateur afin d'améliorer leur rythme de travail. L'article emploie l'étude de l'apprentissage supervisé pour prédire une suite de commande qui serait utile à l'utilisateur. Notre cas serait de recommander la commande dans le cas où l'usager n'utiliserait pas la commande optimale ce qui est en partie similaire à l'étude proposer par l'article.

Cependant, la principale différence est que dans notre cas on utilise en entrée l'état du logiciel (image avant et après l'utilisation d'une commande) alors que l'article étudie l'historique des commandes.

Chapitre 3

Contribution

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

3.1 Photoshop

3.2 Programme de présentation

3.3 Éditeur de texte

3.3.1 Implémentation de l'éditeur de texte

Pour l'implémentation de l'éditeur de texte, PyQt nous permet d'avoir déjà la plupart des commandes de sélection de texte implémentées.

3.3.2 Base de données

- Player (grâce à pyqt) clavier car manipulation de texte.

- Base de données séparée en 2 (partie selection avec tout ce qui concerne le curseur et partie Bag of word)

- Modèles : Classifieur codé en dur pour la sélection et un classifieur entraîné à partir d'un vecteur bag of word. Remarque : Il faut gérer tous les cas possibles et ignorer les cas où le document ne change pas

3.3.3 Génération des données et difficultés liées au modèle

- Les différents essais de classifieurs : Classifieur unique vs 2 classifieurs entraînés vs le classifieur codé en dur + classifieur entraîné

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

A.1 Cahier des charges (MODIFIER LES SCHEMAS)

A.1.1 Introduction (Schéma à modifier

Ce projet a pour objectif de créer un assistant qui incite l'utilisateur à découvrir certaines commandes qui sont potentiellement meilleures que d'autres. Dans la suite du projet de Master I, certains points restent identiques : On aura un assistant qui analysera l'état du logiciel (image ou vecteur représentant cela) et proposera par un classifieur, une meilleure commande s'il en existe une (qui combinerait d'autres commandes basiques par exemple).

Pour mener à bien la preuve de concept, on implémentera cette assistant dans 3 logiciels différents : Un éditeur de texte (type word), Un programme de présentation (Google slide, Powerpoint) et enfin un logiciel de traitement d'images/dessins (photoshop). En prenant un exemple dans un éditeur de texte, l'utilisateur pourrait faire supprimer un mot en appuyant plusieurs fois sur "backspace" au lieu de juste appuyer sur "CTRL + backspace" qui supprime le mot directement et qui serait plus efficace au niveau du temps et effort.

Par rapport à d'autres travaux, il faut ici qu'on utilise l'état de l'application (le contexte en d'autres termes) pour comprendre l'intention de l'utilisateur plutôt que de suggérer une commande en fonction de ceux qui ont été utilisées par l'usager.

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

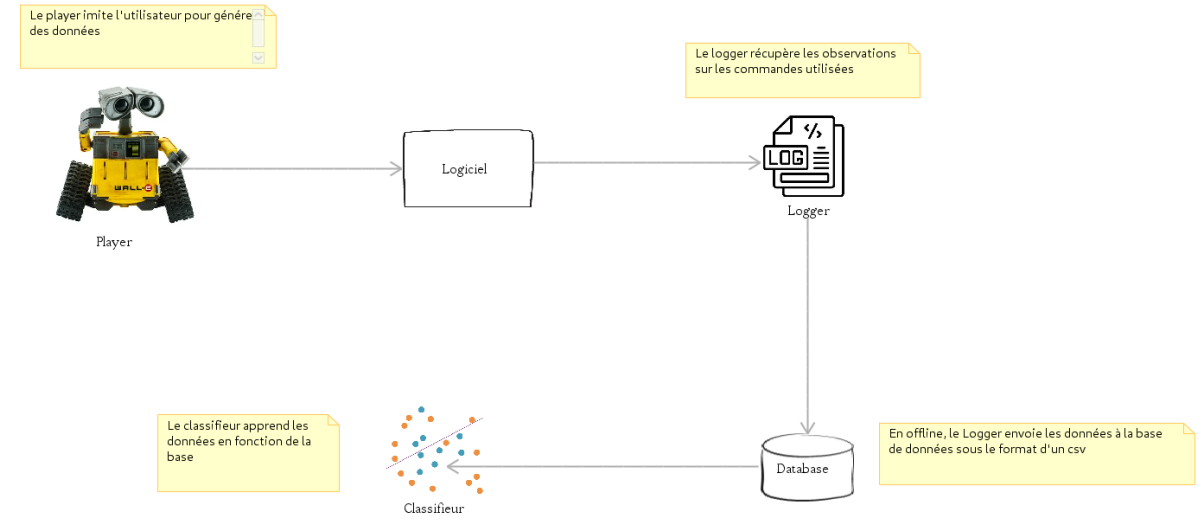


FIGURE A.1 – PLACEHOLDER 1

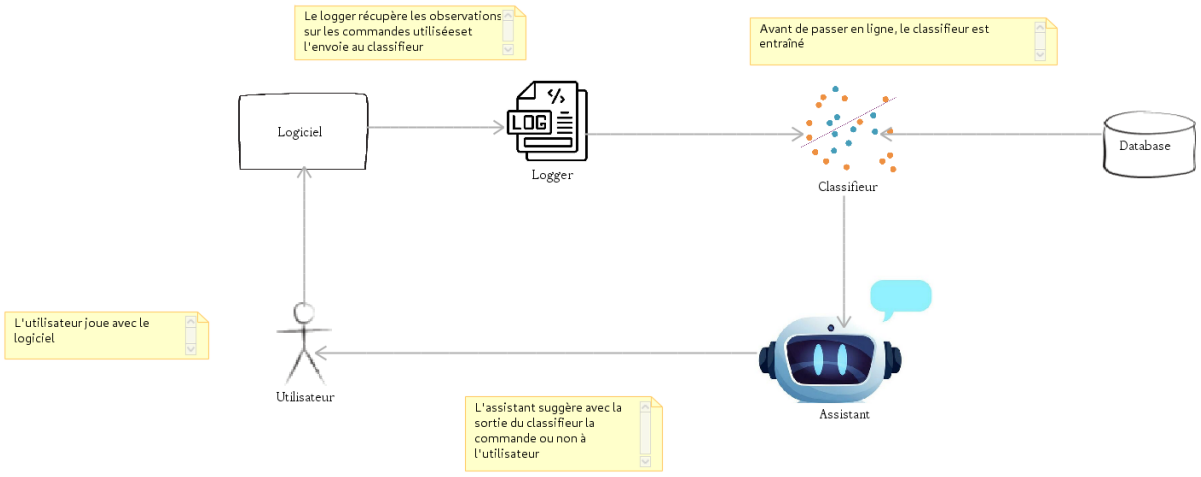


FIGURE A.2 – PLACEHOLDER 2

Le travail consiste à :

généraliser à différents contextes (Éditeurs de forme, éditeur photo, éditeur de texte).

- Réaliser un composant qui capture l'ensemble des commandes (pertinentes) de l'application
- Réaliser un modèle qui, à partir de deux états de l'application retourne la commande la plus appropriée.
- Réaliser un assistant qui repose sur les composants précédents pour présenter les raccourcis les plus pertinents aux bons moments.
- Réaliser des évaluations techniques pour comprendre les performances du modèle
- Réaliser des études utilisateurs pour évaluer la pertinence de l'approche.
- Réaliser des démonstrateurs pour étudier dans quelle mesure cette approche peut être

A.1.2 Logiciel

Les 3 Logiciels

Pour ce qui concerne les logiciels, l'éditeur de texte et le programme de présentation seront fait par nous même à l'aide de python (librairie PyQt) avec des fonctionnalités qui nous semblent pertinentes. On a grâce à cela, un meilleur contrôle sur ce qu'il passe dans les commandes. L'idée étant d'explorer les différentes commandes et de choisir ceux qui seraient les plus pertinents et reconnaissables par notre système.

Pour le logiciel de traitement d'images, nous avons décidé qu'il serait intéressant d'utiliser directement un des logiciels déjà entièrement fait entre krita (qui est open source) et photoshop. Le choix sera fait en fonction des fonctionnalités du logiciel.

Les décisions prisent sur quelles commandes nous semblaient intéressantes à utiliser sur l'assistant sont :

Pour le logiciel de type Word :

- Chercher et remplacer (CTRL + R dans notre logiciel)
- Les commandes de sauts entre les mots (CTRL + -> ou <-)
- Les commandes de sauts vers le début et vers la fin d'une ligne (home ou end sur un clavier qwerty)
- Les commandes de selections de mots (CTRL + SHIFT + <- ou ->) et du document (CTRL + A)
- La commande d'indentation (TAB)

Chacune de ces commandes peuvent être effectuées de façon un peu moins efficace (niveau temps et nombre de commandes)

Pour le Logiciel de type Powerpoint

- Commandes de groupage (CTRL + G) et dégroupage (CTRL + SHIFT + G) + Commandes alignements

Pour photoshop

- blur (gaussien, surface et iris)

Une grande partie du projet de stage a pour objectif d'implémenter les éléments de l'article suivant :

On devra également étudier différentes approches en ce qui concerne le modèle et ses données :

- Des modèles d'apprentissages et des modèles codés en dur.
- Les données seront générées par un outil qui imitera d'une façon l'utilisateur (utilisation aléatoire ou précise des commandes)
- Différents traitements sur les données ou même une façon particulière de les générer (cropping sur les images, normalisation, introduction de données fait à la main)

A partir de tous ces logiciels, le produit final qu'on doit fournir est une vidéo de scénarios pour chaque logiciel qui présente le concept de l'article (DISCO) donné par nos encadrants.

Au fur et à mesure du stage, d'autres points seront développés avec nos encadrants. Les tâches n'étant pas concises dû au fait qu'il faille expérimenter avec les différents logicielles et décider des commandes ainsi que des classifieurs les plus performants pour notre système.