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. Parmi 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 repeindre 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 connaît pas la commande d'alignement procédera 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 distincts. 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 : Alexandre XIA et Nassim AHMED-ALI, et 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 domaines d'amélioration des performances de l'interface

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 :

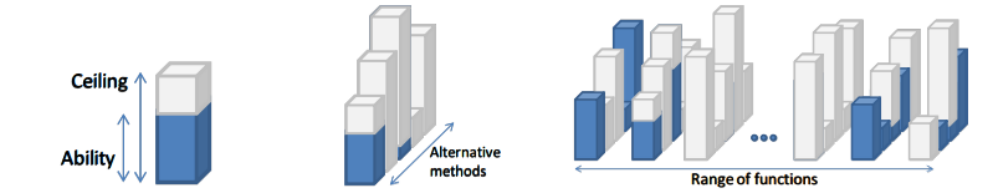
- **L'amélioration intramodale** concerne la vitesse et l'ampleur des performances d'une méthode spécifique du logiciel. Par exemple, aider l'utilisateur à prendre en main un clavier spécifique (ShapeWriter [1])

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

- **L'extension vocabulaire** 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 une fonction plus efficace dans ce cas précis.

- **La planification des tâches** 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 2 Exemples d'approches pour l'extension vocabulaire en 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.

Chapitre 3

Contribution

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

3.1 Général

Puisqu'il s'agit de la suite du projet ANDROÏDE de Master 1, nous allons rapidement résumer ce qui a été fait dans le projet. L'objectif était de concevoir un assistant pour un logiciel simplifié. La figure suivante montre une version de ce qui a été fait :

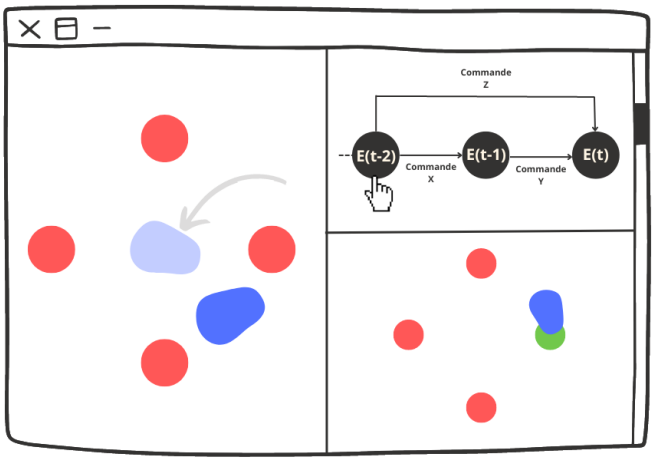


FIGURE 3.1 – Travail du PAndroide

Nous avons codé un programme avec une figure qu'on pouvait déplacer et tourner dont la représentation du vecteur pour l'apprentissage était ses coordonnées (x et y) et sa rotation (à gauche dans la figure). Les mouvements étaient soit un mouvement simple (direction ou rotation d'un sens), soit un mouvement qui fait un plus grand déplacement en une action. Nous avions aussi ajouté une tâche utilisateur (en bas à droite de la figure) qui donne un objectif lors de l'utilisation du logiciel. Le recommandeur (en haut à droite de la figure) suggérait une commande lorsqu'un usager se servait d'une commande qui pouvait être remplacé par une autre qui serait mieux. Un exemple de présentation visible sur

l'image est d'afficher les états et transitions vers les états correspondant aux commandes par l'usager et de montrer l'effet si on utilisait la recommandation à la place.

Pour mieux illustrer, lorsque l'utilisateur déplace la figure en appuyant plusieurs fois sur une touche directionnelle, l'assistant interroge le classifieur entraîné (l'utilisation du Machine Learning était une des tâches dans notre cas) et propose la commande qui fait un plus grand mouvement vers la direction correspondante, ce qui permet de mieux accomplir l'objectif. Le logiciel étant très simple ce stage va nous amener à appliquer ce système vers des trois autres cas :

— un éditeur de texte comme Word
— un logiciel de présentation comme PowerPoint
— un logiciel de retouche comme Photoshop

Le fonctionnement du système pour le stage est représenté dans la suite. Cela est similaire au projet mais sans restriction sur le type de classifieur.

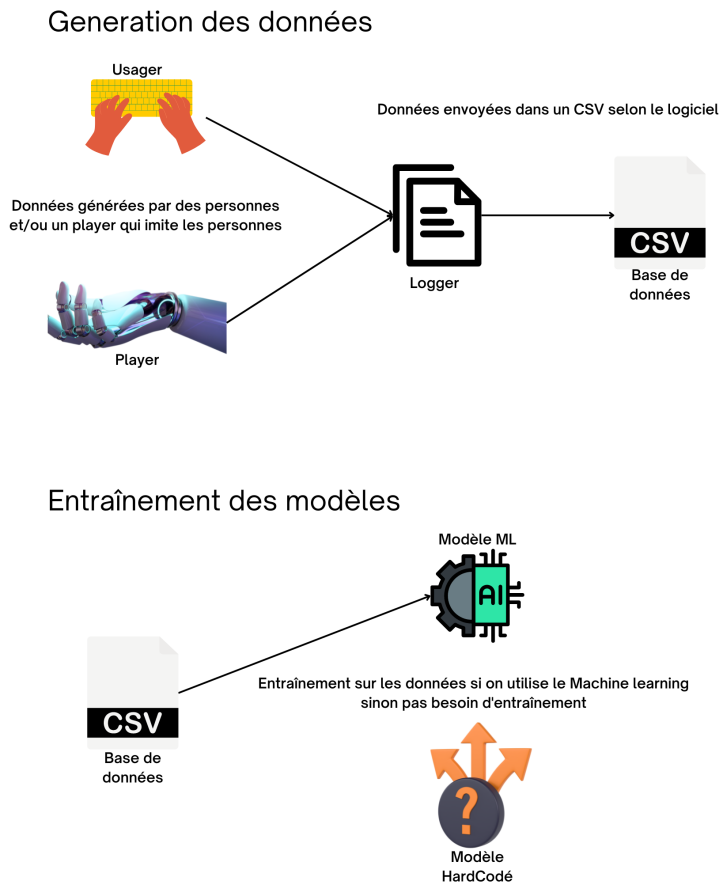


FIGURE 3.2 – Fonctionnement du système hors ligne : Génération des données et entraînement (ou non)

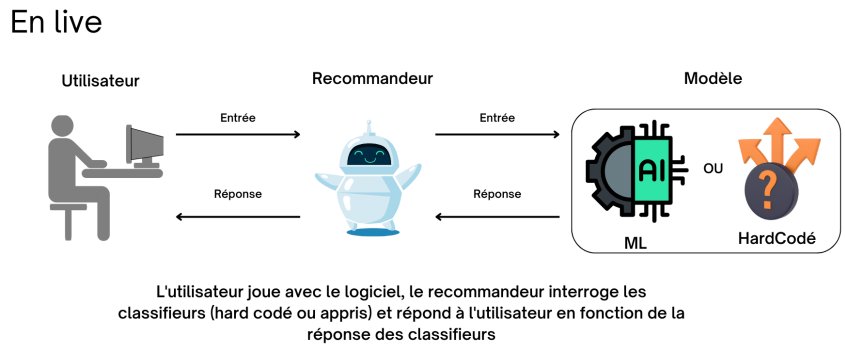


FIGURE 3.3 – Fonctionnement du système en live

Notre système de recommandation aura besoin de 4 principaux composants :

- **Classifieur** : ce composant détermine s'il y a une commande qui permet d'obtenir le même résultat que ce que l'utilisateur vient de faire. Le classifieur peut être basé sur des règles ou entraîné sur une base de données
- **Player** : Ce composant interagit avec l'application et applique des commandes sur des états initiaux aléatoires, afin d'obtenir l'effet des commandes. L'effet d'une commande est déterminé par l'état initiale, l'état résultant de la commande, et la commande. Cela permet de générer une base de données pour l'apprentissage de notre classifieur des effets des commandes. Ce composant est nécessaire car on n'a pas de base de données à disposition.
- **Logger** : il récupère l'état de l'application. Il est important pour la génération des données et la recommandation des commandes.
- **Recommendeur** : le système de recommandation qui récupère les prédictions du Classifieur et décide d'afficher ou non une recommandation à l'utilisateur à l'aide d'un pop-up

3.2 Logiciel de présentation

Pour le logiciel de présentation, on a voulu s'inspirer d'un logiciel connu comme Powerpoint.

Ce qu'on veut s'est de pouvoir récupérer l'état du logiciel et de pouvoir si possible piloter l'application afin de pouvoir générer des données.

Pour le support, on peut avoir 2 approches :

- Utiliser directement Powerpoint. Et chercher des moyens de permettre notre recommendeur d'avoir accès au logiciel comme récupérer l'image du diapositive.
- Développer un programme qui ressemble à Powerpoint, pour avoir un contrôle et un accès total sur le programme. Cependant, cela nécessite de prendre du temps dans le développement du logiciel.

Pour des raisons de simplicité et de contrôle du programme, on a développé nous-même un programme ressemblant à Powerpoint, plutôt que d'utiliser directement Powerpoint.

Par simplification, nous travaillons que sur un seul diapositive. Mais cela ne gêne pas à la généralisation sur plusieurs diapositives, puisque ce serait semblable à avoir plusieurs instance du même logiciel.

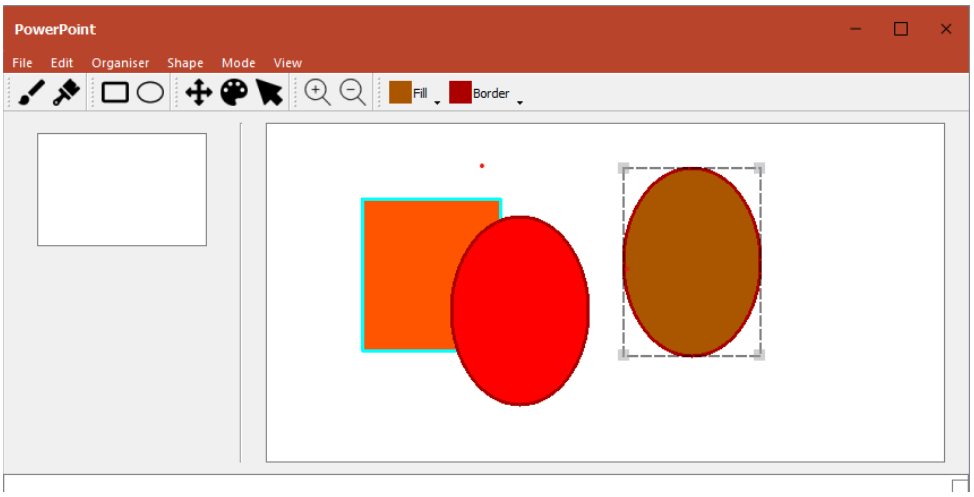


FIGURE 3.4 – Logiciel de présentation similaire à Powerpoint

3.2.1 Commandes implémentés

Pour un logiciel de présentation qui serait semblable à PowerPoint, on choisit de nous concentrés sur un nombre limité de commandes pour des raisons de temps et de simplicité. Dans les choix des commandes, on a également pris en compte les difficultés possibles dans l'implémentation de ceux-ci dans notre logiciel. Voici donc les commandes qui nous ont parue le plus pertinent :

Alignement des objets (transformations spatiales) : permet d'aligner les objets, au lieu de déplacer manuellement les objets pour les aligner. Les alignements possibles sont : en haut, en bas, à gauche et à droite.

l'utilisateur pourra difficilement aligner exactement des objets à la main, alors que la commande le fait parfaitement. Donc ce serait une bonne commande à recommander pour l'utilisateur.

Copier et Aligner : Permet de dupliquer puis faire bouger avec la souris, en gardant d'alignement avec les objets copiés. Cette commande est très similaire à aux commandes d'alignement, mais nous a paru être une commande utile à recommander. Comme pour l'exemple avec la commande d'alignement, l'utilisateur ne va pouvoir aligner parfaitement l'objet copié ou recréer l'objet avec les même dimensions sauf en utilisant la commande 'Dupliquer' ou 'Copier-Coller', puis utiliser une commande d'alignement des objets.

Premier Plan / Arrière-Plan : Permet de changer la profondeur des objets. On l'a choisi car on peut voir un scénario simple à tester : Trois objets A,B,C l'une sur l'autre, l'utilisateur met B et C au premier - plan. On peut faire la même résultat en mettant A en arrière plan. Donc la deuxième méthode serait plus pertinent car il se fait en 1 action, alors que l'autre se fait en 2 actions.

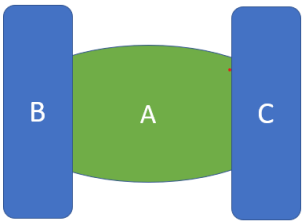


FIGURE 3.5 – Diapositive avec B et C au Premier Plan et A en Arrière-Plan

3.2.2 2 approches pour la prédiction

Pour l'implémentation du modèle à utiliser 2 approches différentes pour prédire les commandes : l'un avec du machine learning (réseau de convolutions), et l'autre avec basé sur des règles qu'on a nous-même établis.

Pour analyser plus précisément, les avantages et désavantages des 2 approches. On va en premier analyser les performances du modèle de machine learning : performance et précision.

Machine Learning	**Rules Based**
Avantages : — plus facile à généraliser sur plus de commandes. On doit juste avoir l'effet de la commande pour l'apprentissage	**Avantages :** — facile à débugger — simplicité et facilité d'implémentation
Désavantages : — besoin d'une base de données — moins interprétable — peut faire des erreurs de prédictions et difficile à debugger	**Désavantages :** — Moins généralisable. Besoin de coder les règles pour chaque commande possibles

TABLE 3.1 – Comparaison entre Machine Learning et Hardcode

3.2.3 État de l'application

En fonction du modèle adopté, certaines façons de représenter l'état de l'application peuvent s'avérer plus appropriées que d'autres. Dans cette section, nous examinerons deux approches pour la représentation de l'état de l'application.

Machine Learning

Pour le modèle de machine learning, l'état du diapositive est l'image du diapositive. Et donc l'entrée du modèle qui prend l'effet de la commande est 2 images, représentant avant et après le lancement de la commande.

Contrairement au modèle basé sur des règles, on ne peut pas prendre une liste avec un nombre d'objets variables, sauf avec les modèle type RNNs et transformers.

L'image est un état fini et décrit visuellement l'état du logiciel, et donc serait une bonne représentation de l'état de l'application.

A base de règles

Pour cette approche, on choisit d'utiliser directement la liste des objets avec leurs propriétés tels que la forme, position, dimensions, les couleurs. L'entrée du modèle prend donc 2 listes d'objets : un avant et l'autre après le déclenchement d'une commande. Il serait pas très pertinent de faire sur une image, ce qui compliquerait uniquement la tâche de prédiction. Il serait donc plus simple d'obtenir directement la liste des objets avec leurs propriétés. Par exemple la position pour l'alignement.

Difficulté, les 2 listes peut être de tailles différentes si on a ajouté ou supprimé des nouveaux objets. Les listes peuvent également avoir des ordres différents, en particulier si on met un objet au premier-plan ou en arrière-plan.

Pour résoudre le problème de la taille des listes, on a choisit de prendre compte uniquement l'état courant et l'état qui le précède. Cela permet de simplifier la prédiction à juste des cas simple, et de obtenir l'intention de l'utilisateur.

Pour le problème, des listes dans un ordre différent. On fait un tri des objets selon leurs propriétés (dimensions, couleurs, ...).

3.2.4 Modele de Machine Learning

Génération des données

Pour notre modèle de machine learning, nous aurons besoin d'une base de données dont laquelle il peut apprendre dessus. cette base de données sera constituée de paires d'images : l'une avant et l'autre après l'exécution des commandes.

Nous générons ces données en insérant des formes aléatoires (rectangles ou ellipse), dotées de dimensions et de positions variées, que nous soumettons ensuite à l'une des commandes choisies. Lors de la création des objets, nous veillons à éviter la superposition d'objets ou l'emboîtement d'un objet dans un autre.

Traitement des données

Pendant le stage, on a effectuer plusieurs et combiner plusieurs traitement possible afin d'améliorer les performances du modèle de machine learning comme la normalisation. On a essayé également d'autres pré-traitement.

- Rogner l'image : On récupère le rectangle englobant de tous les objets. Cela permet de supprimer une partie des pixels inutiles de l'image comme on se concentre sur les objets. Pour cela, on cherche les pixels non-blancs et on récupère le minimum et maximum des x et y pour obtenir le rectangle englobant.
- Resize : on redimensionne l'image en une taille qui convient au modèle par exemple 64x64. On a choisit la taille 64x64 afin de réduire le nombre d'entrée. Cela est possible car on a seulement des commandes simples (alignement, changer la profondeur des objets, copier-aligner).

Architecture du modèle

Pour construire notre architecture de modèle, on s'est inspiré de celle d'un article.

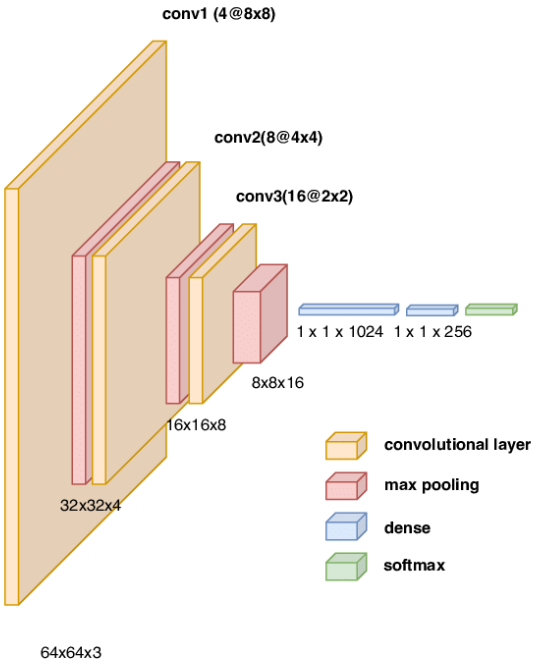


FIGURE 3.6 – Architecture du réseaux de convolution (image de l'article) [2]

Nous avons opté pour l'utilisation du min pooling à la place du max pooling. Cette décision a été prise en considération du fait que notre Powerpoint présente des arrière-plans blancs. Ainsi, effectuer du max pooling avec un fond blanc aurait tendance à privilégier les pixels blancs de valeur (255, 255, 255). De plus, nous avons introduit une couche linéaire avec des paramètres (2048, 1024) dans notre architecture. Cette modification a été apportée parce que nous traitons en entrée deux images de taille 64x64 plutôt qu'une seule image.

3.2.5 Modèle basé sur des règles

Pour cette partie, on va expliquer brièvement le fonctionnement du modèle basé sur des règles. Dans notre modèle basé sur des règles, chaque commande a une fonction de prédiction spécifique. Par exemple, pour les commandes d'alignement, on a une fonction qui détecte et retourne le type d'alignement (haut, bas, gauche, droite), sinon il retourne "Rien".

Cela permet de rendre plus lisible le code, et permet d'ajouter des commandes à prédire plus facilement. Et donc pour la prédiction des commandes en général, on a une liste de fonctions de prédiction spécifique disposée dans un ordre arbitraire. On parcourt la liste de fonction, on retourne le premier résultat non nul.

Dans notre cas, on prédit dans l'ordre suivant :

1. Premier-Plan / Arrière-Plan

2. Duplication Lignes/Colonnes
3. Copier-Aligner
4. Copier-Déplacer
5. Aligner Objets Haut/Bas/Gauche/Droite

Pour les entrées des fonctions de prédictions, c'est 2 liste des objets avec leurs propriétés (avant et après). Les propriétés qu'on prend en compte sont :

— Type de l'objet (rectangle, ellipse)
— Position du rectangle englobant
— Couleurs de remplissage RGB
— Couleur du contour RGB

On va expliquer brièvement l'algorithme pour prédire les différentes commandes qu'on a choisis.

Prédiction Alignement :

1. Si les 2 listes n'ont pas la même taille ou s'il y a aucun changement sur la position des objets, retourne "Rien"
2. Obtenir les objets qui ont changé de position, en comparant avec les objets d'avant.
3. L'objet qui a changé, o, regarde s'il y a un alignement possible (paramètre ϵ détermine la tolérance à l'erreur dans la comparaison) avec un autre objet présent dans la liste.

Prédiction Copier-Aligner :

1. Obtenir l'objet sur lequel on se concentre :
   — Si la taille avant et après est la même, on compare avec les objets d'avant. On devrait pouvoir obtenir seulement l'objet qui a changé.
   — Si la taille de la liste a seulement augmenté de 1, on récupère l'objet récemment créé sur le diapositive.
2. L'objet qu'on a choisi de focaliser, o, est examiné pour voir s'il y a un objet similaire (paramètre ϵ) présent dans la liste et aligné avec un autre objet.

Prédiction Premier-Plan / Arrière-Plan :

1. Si des objets sont déplacés ou s'il y a des changements dans les propriétés (couleurs, etc.), alors la prédiction retourne "Rien".
2. Si les 2 listes n'ont pas la même taille, retourne "Rien"
3. Calculer la matrice des relations avant et après. La matrice des relations est telle que $f(i, j) =$
   — 1 si o_i est devant o_j
   — -1 si o_i est derrière o_j
   — 0 si $o_i = o_j$ ou o_i et o_j ne se chevauchent pas (= pas de relation)
4. Prendre seulement les objets qui ont changé de relations.
5. Prendre celui avec le plus de changements.

3.3 Évaluation modèle

3.3.1 Évaluation statistique

Pour l'évaluation des modèles de machines learning, on débuté sur les commandes d'alignement (haut/bas/gauche/droite), afin de déterminer si on poursuit l'apprentissage des modèles sur les autres commandes.

On a commencé du plus simple (alignement sur tous les objets) au plus compliqué (alignement sur une partie des objets).

— modele 1 : apprentissage sur les images après l'alignement de tous les objets
— modele 2 : apprentissage sur les images avant et après l'alignement de tous objets
— modele 3 : apprentissage sur les images avant et après l'alignement de tous ou une partie des objets

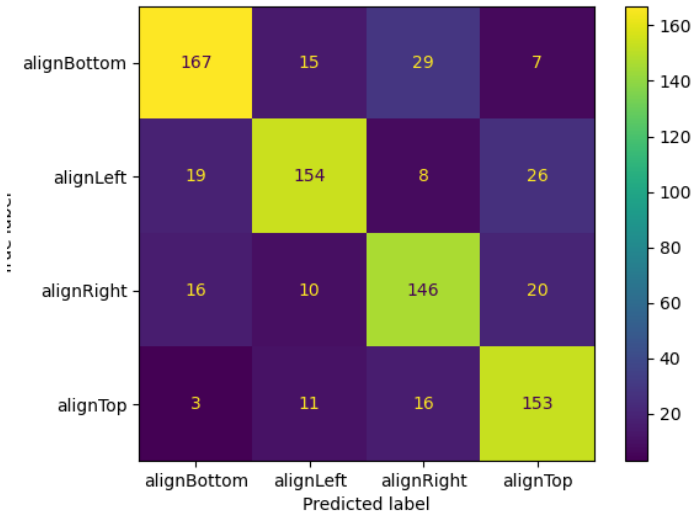


FIGURE 3.7 – Matrice de confusion du modèle 1 sur les commandes d'alignement

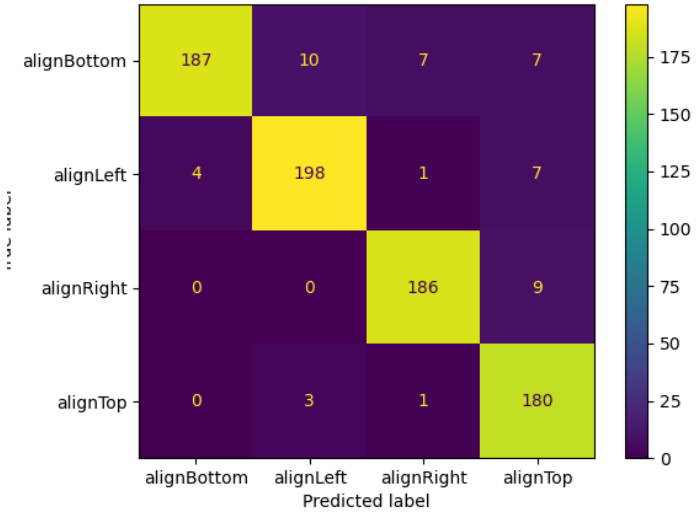


FIGURE 3.8 – Matrice de confusion du modèle 2 sur les commandes d'alignement

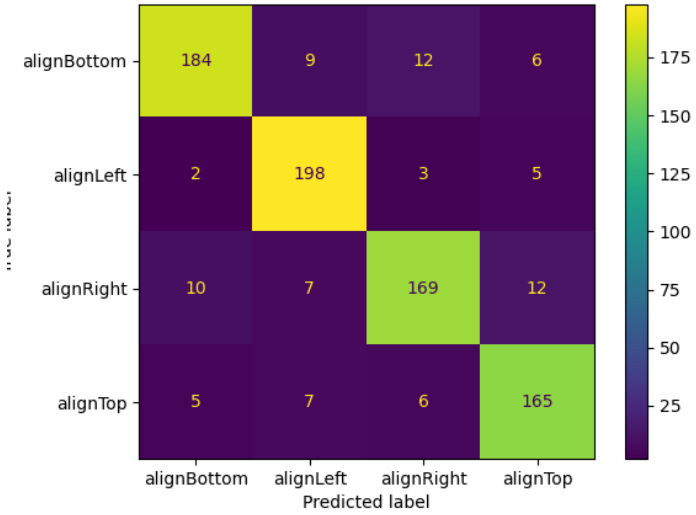


FIGURE 3.9 – Matrice de confusion du modèle 3 sur les commandes d'alignement

	Model 1	Model 2	Model 3
Test Accuracy	0.775	0.939	0.895

TABLE 3.2 – Performances des modèles sur les données de tests

On remarque que la performance du modèle 3 et du modèle 2 sont semblable et est supérieure à celle du modèle 1. Donc il serait préférable d'utiliser le modèle 3 qui est entraîné sur l'alignement de tous ou une partie des objets.

	Modele 1	Modele 2	Modele 3
Alignement en haut	0.743	0.887	0.878
Alignement en bas	0.815	0.979	0.915
Alignement à gauche	0.811	0.938	0.896
Alignement à droite	0.734	0.954	0.889

TABLE 3.3 – Précision des modèles sur les données de tests

3.3.2 Évaluation en conditions pratiques

1. Le modèle 1 détecte efficacement les alignements lorsque ceux-ci sont effectués sur tous les objets présents. Cependant, il éprouve des difficultés lorsqu'il s'agit d'aligner seulement une partie des objets d'une diapositive.
2. Le modèle 2 parvient à repérer les alignements, mais uniquement lorsque le déplacement des objets correspond aux commandes d'alignement prises en compte lors de l'entraînement. Par exemple, des déplacements en diagonale peuvent poser problème car le modèle n'a jamais été exposé à de tels types de données.
3. Le modèle 3 détecte les alignements de manière plus précise que le modèle 2 lorsque l'alignement concerne une partie des objets.

Pour l'évaluation des modèles de machine learning, on a débuté sur les commandes d'alignement des objets (haut / bas / gauche / droite) Les modèles 2 et 3 considèrent les déplacements des objets. Étant formés sur la commande d'alignement, ils semblent accorder plus d'importance au déplacement des objets qu'à leur position finale, ce qui est à la fois valide et incorrect. Par exemple, lorsqu'un alignement à droite est exécuté manuellement en déplaçant un objet vers la gauche, le modèle a tendance à prédire un alignement à gauche. Cette observation peut être expliquée par la nature de la commande d'alignement, car un alignement à droite est systématiquement associé au déplacement d'un objet vers la droite parmi ceux situés le plus à gauche, plutôt qu'aux objets les plus à droite.

Pour notre choix final, on a décider que plutôt de garder un modèle de réseaux de neurones qui pourrait faire des erreurs, on préfère un modèle basé sur des règles qu'on établie nous-même. Et donc on n'a pas continuer à développer les modèles de machine learning sur les autres commandes, et plutôt d'améliorer notre modèle basé sur des règles sur les autres commandes à prédire.

3.4 Photoshop

Comme logiciel de traitement d'image, on a choisit d'utiliser un logiciel existant, comme Photoshop, car contrairement on 2 autres logiciels (Powerpoint et Word), développer les commandes de Photoshop semble difficile.

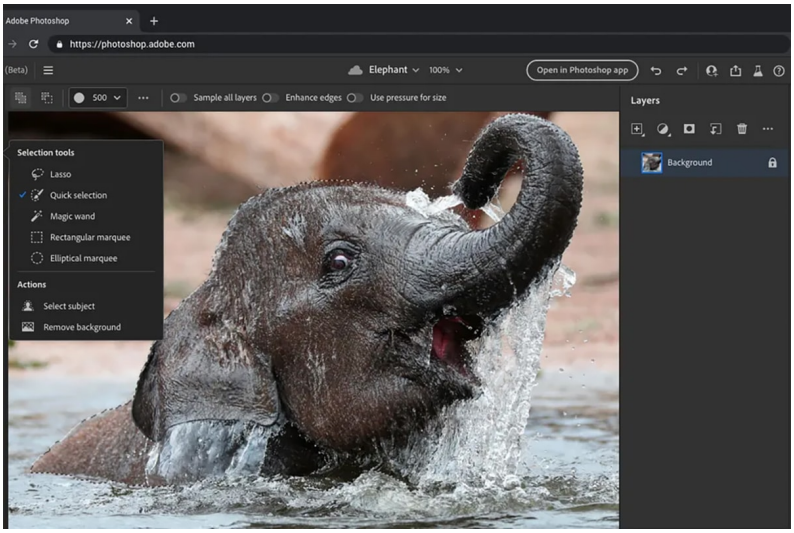


FIGURE 3.10 – Logiciel Photoshop

Pour les commandes qu'on a choisit dans la prédiction notre modèle sont : - filtres qui modifie l'images (gaussian, twirl, pinch, ...). Comme pour Powerpoint, on utilise un modèle de réseau de convolution qui prend en entrée l'image modifié et l'image original. Contrairement à Photoshop, on peut difficilement développer un modèle basé sur des règles, et donc cette partie est plus centré vers le machine learning sur les images.

Les points difficiles, c'est que dans le cas de Photoshop, les images dans Photoshop peuvent être très diverse, on peut utiliser n'importe quelle image sur Photoshop. C'est une partie importante, car c'est un vrai logiciel contrairement au 2 parties.

3.5 Éditeur de texte

Dans cette section, Alexandre a le plus contribuée. On a développer un éditeur de texte similaire à Word. Parmi toutes les commandes de Word, on choisit de prendre ceux qui permet de :

— Modifier le texte. Exemple : "Rechercher et Remplacer"

— Naviguer rapidement dans le texte. Exemple : "Aller au début de la ligne"

— Sélectionner une partie du texte. Exemple : "Sélectionner toute la ligne"

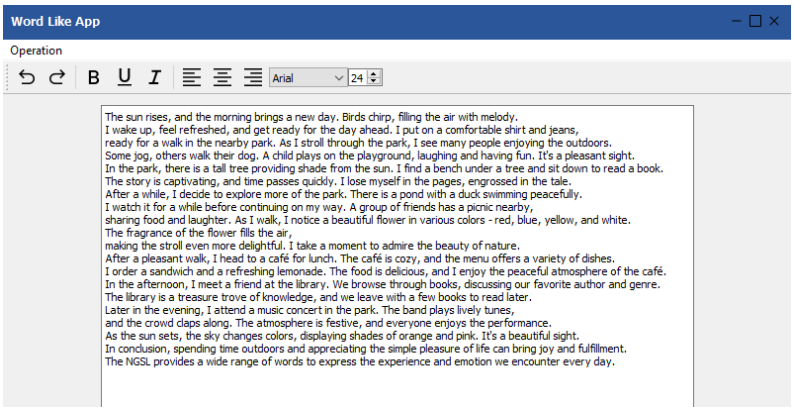


FIGURE 3.11 – Éditeur de texte similaire à Word

L'état d'un éditeur de texte serait le texte entier. Cependant, pour traiter une entrée qui est texte, on aurait besoin d'un type spécifique de modèle de machine learning : réseaux récurrents qui permettent une entrée de taille variable.

On a donc choisi d'un représentation plus simple qui le "Bag of words". Cepedant, cela reste suffisant pour détecter une commande "Rechercher et Remplacer". Dans cette partie, comme pour la partie sur le logiciel de présentation, on a utilisé un modèle basé sur des règles, mais seulement pour les commandes de navigation et de sélection..

Chapitre 4

Répartition des rôles

Majoritairement, nous avons chacun pris chacun un logiciel avec l'éditeur de texte pour moi, le logiciel de présentation (Powerpoint) pour Christian Zhuang et Photoshop pour Nassim Ahmed-Ali. La figure suivante représente la répartition des tâches ; les pourcentages sont assez arbitraires comme le travail était surtout expérimental :

	Alexandre	Nassim	Christian
Photoshop (code/player/images manuelles et données)	10%	80%	10%
Editeur de texte (code/player/données)	80%	10%	10%
Programme de présentation (code/player/données)	10%	10%	80%
Prise en main de Pytorch (du modèle)	50%	25%	25%
Recommandeur	25%	25%	50%
Vidéo	25%	50%	25%

Table : Répartition arbitraire des pourcentages des tâches

— Photoshop, Editeur de texte et Programme de présentation (Powerpoint) : Toutes les valeurs à 10% représentent surtout l'aide au débogage du code de chacun, c'est difficile de vraiment savoir à quel pourcentage cela correspond mais arbitrairement en prenant le temps on a décidé que ce serait à peu près 10%.

— Pour Pytorch, le Recommandeur et les vidéos, il est plus facile de donner des pourcentages :

- Les valeurs à 50% correspondent pour Pytorch à la mise en place du modèle de base ainsi que l'application à l'éditeur de texte, pour le recommandeur c'est le codage de celui-ci et l'application à "Powerpoint" et pour la vidéo c'est surtout le montage en plus des scénarios du logiciel correspondant à Photoshop.

- Les valeurs à 25% sont l'application de Pytorch (classifieur) et recommandeur aux différents logiciels et pour la vidéo aux scénarios (les vidéos sont faites ensemble).

Chapitre 5

Conclusion

En résumé, notre démarche a abouti à la conception d'un système qui présente un preuve de concept pour un recommandeur de commandes contextualisé. Dans la section dédiée au logiciel de présentation, nous avons développé une application similaire à Powerpoint, intégrant l'ensemble des commandes à étudier. Pour la détection des commandes, deux approches distinctes ont été explorées : un modèle de machine learning ainsi qu'un modèle basé sur des règles.

Pour le modèle de machine learning, nous avons procéder à une génération de données via un Player pour l'entraînement de notre modèle de machine learning, suivi d'évaluations statistiques et d'expérimentations en conditions réelles, nous avons pu mettre en évidence les atouts et les limites de notre approche.

Et pour le modèle basé sur des règles. Nous avons réussi à élaborer un modèle basé sur des règles dont l'efficacité s'est avérée notable.

En contrepartie, certains points faibles ont également été identifiés. Notre application reste rudimentaire en comparaison d'un logiciel de présentation complet, qui incorpore des éléments tels que des zones de texte, des images et des fonds. Enfin, compte tenu du nombre limité de commandes examinées, notre succès dans la création d'un modèle basé sur des règles performant pourrait devenir complexe à maintenir et à faire progresser à mesure que l'application s'enrichit et que les interactions entre les objets se complexifient.

En guise d'ouverture, il serait envisageable d'élargir le champ des commandes à prédire ou encore de procéder à des tests sur des logiciels de présentation réels tels que Powerpoint ou Google Slides.

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, 3).

[2] Hung Son Nguyen & Francisco Cruz & Richard Dazeley. « A Broad-persistent Advising Approach for Deep Interactive Reinforcement Learning in Robotic Environments ». In : (2021). URL : https://www.researchgate.net/figure/CNN-architecture-with-64x64-RGB-image-as-input-group-of-three-convolution-layers-three_fig4_355367168 (page 10).

Annexe A

Cahier des charges

A.1 Cahier des charges

A.1.1 Introduction

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) :

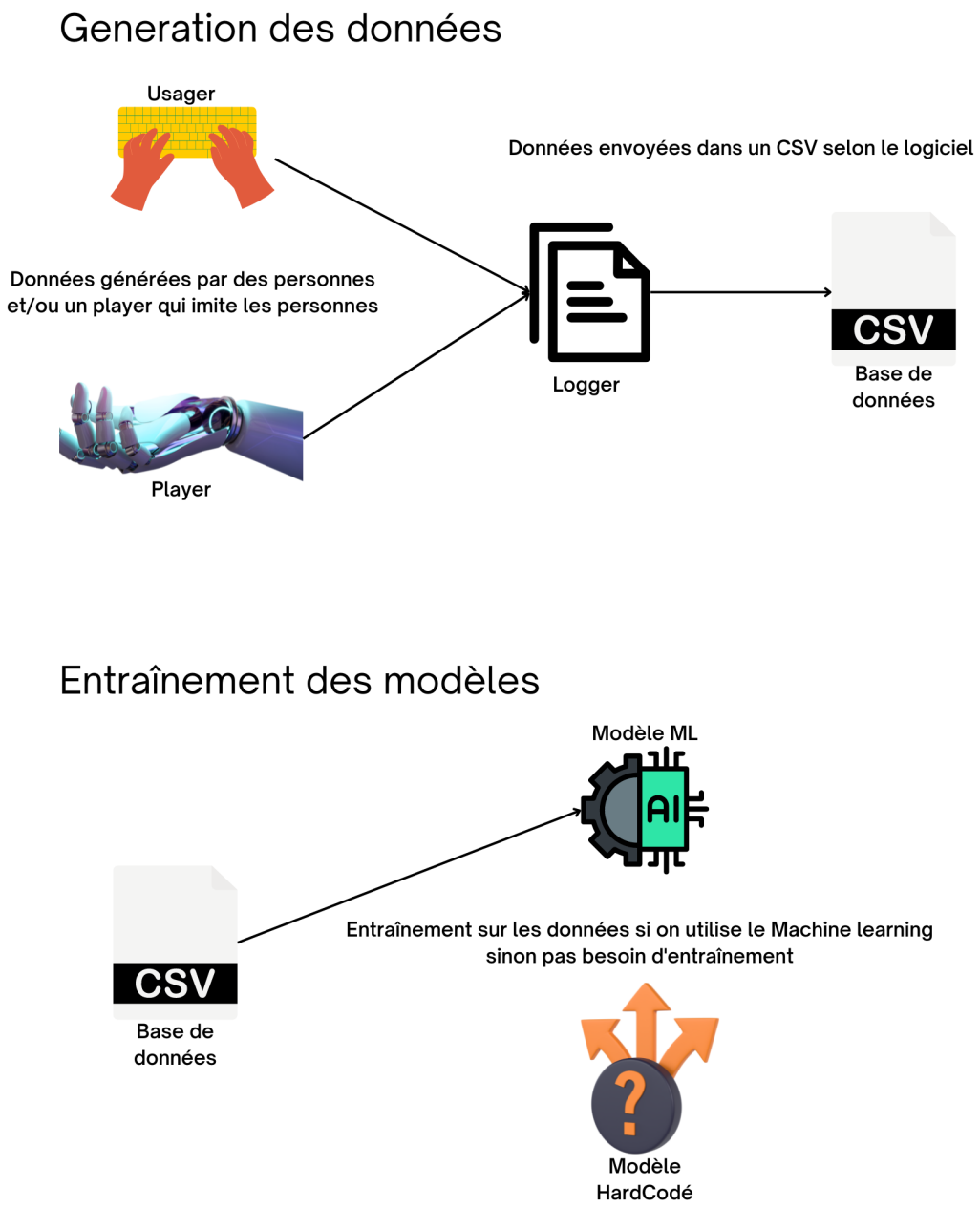


FIGURE A.1 – Partie offline : Génération des données et entaînement

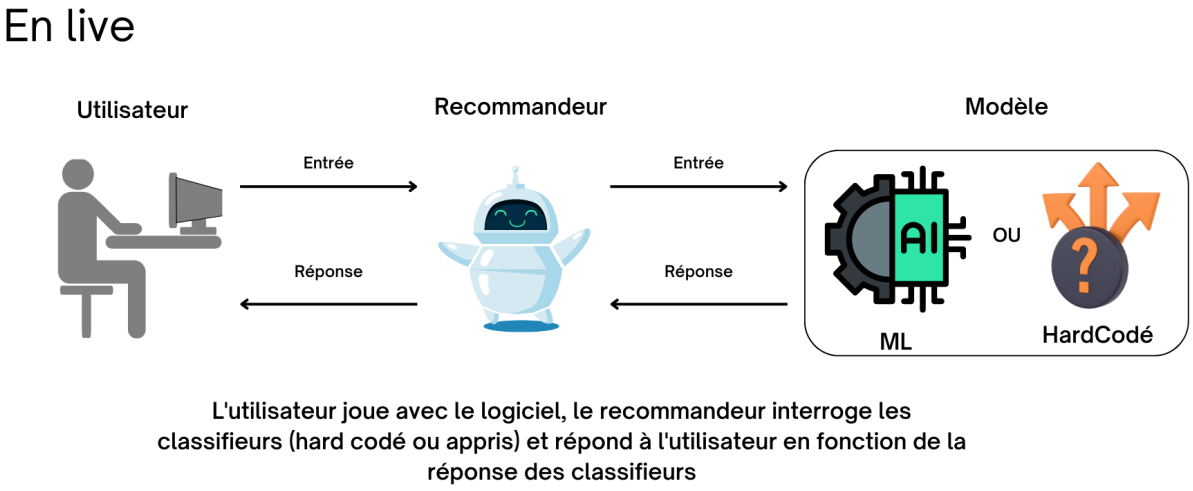


FIGURE A.2 – Utilisation du système en live

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 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.